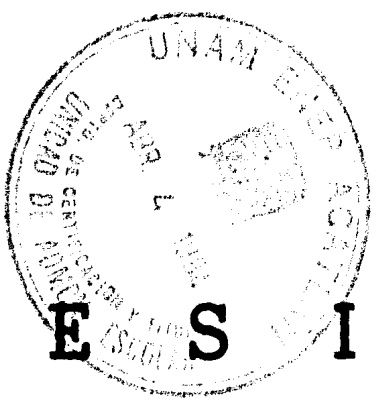
Universidad Nacional Autónoma de México

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES

A C A T L A N

PAQUETE DE PROGRAMAS COMPUTACIONALES PARA APOYO A LA INVESTIGACION DE OPERACIONES



T E S I S

QUE PARA OBTENER EL TITULO DE:

A C T U A R I O

P R E S E N T A :

ENRIQUE KORTRIGHT ORTIZ

MEXICO, D. F. 1984

I N D I C E

I N D I C E:

I N D I C E

I N D I C E

INTRODUCCION

Introducción.

Se ha definido a la Investigación de Operaciones como la mejor manera de llevar a cabo una actividad sin realmente llevarla a cabo. Mediante la construcción de modelos que representan la realidad, es posible repetir indefinidamente experimentos sobre éstos y observar diversas situaciones, las cuales serían imposibles o demasiado costosas de representar sobre la misma realidad. Aún más allá de la simple interpretación y representación del problema real, la Investigación de Operaciones busca incorporar objetivos a perseguir, de tal manera que las técnicas buscan una solución óptima, o por lo menos factible, del problema en términos de esos objetivos.

La Investigación de Operaciones puede ser considerada como el estudio de varios métodos para tomar decisiones en situaciones donde es posible cuantificar la información relevante. Es decir, consiste en plantear la realidad mediante un modelo apropiado y luego aplicar una o varias técnicas para obtener una o varias soluciones al problema. Aun en el caso de que ninguna técnica se adapte al modelo, podemos, a través de éste, observar la realidad y recopilar información para tomar mejores decisiones.

Los orígenes de la Investigación de Operaciones pueden establecerse desde los experimentos de Bernoulli en el siglo XVIII, quien buscaba resolver el problema de comprar o no un seguro para un cargamento de tal manera de optimizar la ganancia esperada. La Investigación de Operaciones no fue reconocida como una ciencia independiente sino hasta la Segunda Guerra Mundial, cuando se buscaban métodos para manejar la logística de las operaciones de los aliados durante la Batalla de Inglaterra y el resto de la guerra. Buscaban una ciencia que les ayudara a destinar recursos escasos a actividades críticas de manera de optimizar un objetivo (minimizar costos, etc.).

Las bases teóricas para la nueva ciencia fueron desarrolladas principalmente en el siglo XX por investigadores como von Newman y Morgernesten con su libro 'Teoría de Juegos y Comportamiento Económico', George Dantzig quien desarrolló el Método Simplex, Ackoff, Churchman, Luce, Raiffa, Bellman y otros. Al mismo tiempo que se desarrolló la teoría, se observó que la mayoría de las técnicas requerían demasiados cálculos aun para problemas de mediana complejidad, y no fue hasta que la computadora electrónica se convirtió en una herramienta efectiva de cálculo, que fue posible aplicar las nuevas técnicas a problemas reales. Esto convirtió a la Investigación de Operaciones en una poderosa herramienta para la toma de decisiones a todos los niveles: estratégico, táctico y operacional. Así mismo de pudo delimitar que problemas tenían una posible solución y cuales, por el volumen de cálculo requerido, deberían esperar mejores técnicas o computadoras mucho más

veloces que las actuales.

Los problemas que resuelve la Investigación de Operaciones se pueden clasificar en las siguientes categorías:

1.- Secuenciación. Secuenciar n trabajos en m máquinas.
2.- Distribución. Destinar recursos a actividades de manera de optimizar alguna medida de efectividad.
3.- Transporte. Determinar la mejor ruta desde un origen hasta un destino, de entre varias rutas posibles.
4.- Reemplazo. Decidir el tiempo óptimo para reemplazar una unidad deteriorada o que falla.
5.- Inventario. Determinar cuándo y cuánto producir o comprar de un producto dado.
6.- Líneas de Espera. Determinar cuántas unidades de servicio tener de manera de encontrar un equilibrio entre costo de servicio y tiempo de espera.
7.- Competencia. Encontrar la mejor estrategia al competir, con uno o más oponentes racionales por la posesión de un recurso escaso.
8.- Búsqueda. Buscar entre un número de alternativas la mejor posible.

El Paquete de Programas de Investigación de Operaciones constituye una herramienta de cálculo para las técnicas más importantes de esta ciencia. Está formado por 31 programas que cubren las siguientes técnicas:

1.- Programación Dinámica.
2.- Programación Lineal.
3.- Programación Entera.
4.- Branch-and-Bound.
5.- Secuenciación.
6.- Teoría de Decisiones.
7.- Teoría de Juegos.
8.- PERT/CPM.
9.- Teoría de Colas.
10.- Simulación.
11.- Teoría de Inventarios.
12.- Cadenas de Markov.

Estas técnicas están orientadas a resolver uno o varios de los ocho tipos de problemas de la Investigación de Operaciones. En especial, el uso de la Simulación constituye un mecanismo de interpretación de la realidad y, a partir de esto, decidir que técnica aplicar en caso de haber una adecuada.

El propósito del paquete es el de apoyar los programas de Investigación de Operaciones en las carreras de Actuaría, Matemáticas e Ingeniería. El alumno podrá aplicar las técnicas recién aprendidas a problemas reales, además de los problemas que

normalmente resuelve en forma manual. Esto le dará una mejor idea de la metodología, tiempo y equipo necesarios para resolver un problema, así como de la posibilidad de alcanzar una solución óptima o por lo menos factible.

Enfatizamos que el uso del paquete requiere del suficiente dominio de cada técnica por parte del alumno y por lo tanto sugerimos al maestro que, al promover su uso, no sustituya esto por su cátedra normal y completa.

Otro de los errores que se deben evitar es el tratar de adaptar el problema a alguna técnica específica por el simple hecho de que conocemos esta última. Es necesario enfocar directamente el problema y de acuerdo a sus verdaderas características, usar o, como en el caso de la Programación dinámica, adaptar una técnica para su solución. De nuevo recomendamos el uso de la Simulación como una primera aproximación.

En el capítulo 3 se describe el diseño de uno de los programas, a fin de mostrar al lector las técnicas de programación utilizadas en la construcción de este paquete. La lectura cuidadosa de esta parte es recomendable a todos aquéllos interesados en el aspecto computacional de este paquete. La notación y la terminología fueron tomadas del libro "Software Development" de Cliff Jones cuya lectura también se sugiere aquí.

La reciente aparición y propagación de las microcomputadoras hizo que este paquete de programas se haya desarrollado para su uso en éstas. Hasta hace poco, la única manera de usar programas de este tipo era a través de una computadora relativamente grande la cual, por su costo y su limitada disponibilidad, disminuía en gran medida la utilidad de este tipo de herramientas. Ahora, gracias al milagro tecnológico que representa la microcomputadora, es posible resolver problemas que involucran un gran número de cálculos a muy bajo costo. De esta forma, el objetivo de esta tesis, una herramienta de cálculo para las materias de Investigación de Operaciones, se alcanza de una manera totalmente accesible tanto al maestro como al alumno.

Enrique Kortright.

INTRODUCCION

DESCRIPCION DEL PAQUETE

1.- Descripción del Paquete.

El equipo necesario para usar el paquete tal y como está escrito es el siguiente:

a) Una microcomputadora APPLE II Plus con 48K de memoria RAM, tarjeta de lenguaje y al menos una unidad de diskette.
b) Sistema operativo UCSD Pascal.

Después de cargar el sistema, la ejecución de cualquiera de los programas consiste en lo siguiente:

a) Oprimir la tecla X.
b) En seguida aparecerá el mensaje:
Execute what file?
c) Contestar este mensaje con el nombre del programa. Por ejemplo:
Execute what file?IO8101.

El resto de la operación depende de cada programa y el usuario deberá referirse a la descripción de cada uno para mayor información.

Es conveniente tener a la mano una copia del manual del sistema operativo UCSD Pascal de APPLE.

En caso de que el paquete, o parte de él, se haya implementado en otro tipo de computadora, la información de operación se encontrará en los manuales correspondientes de la máquina en cuestión.

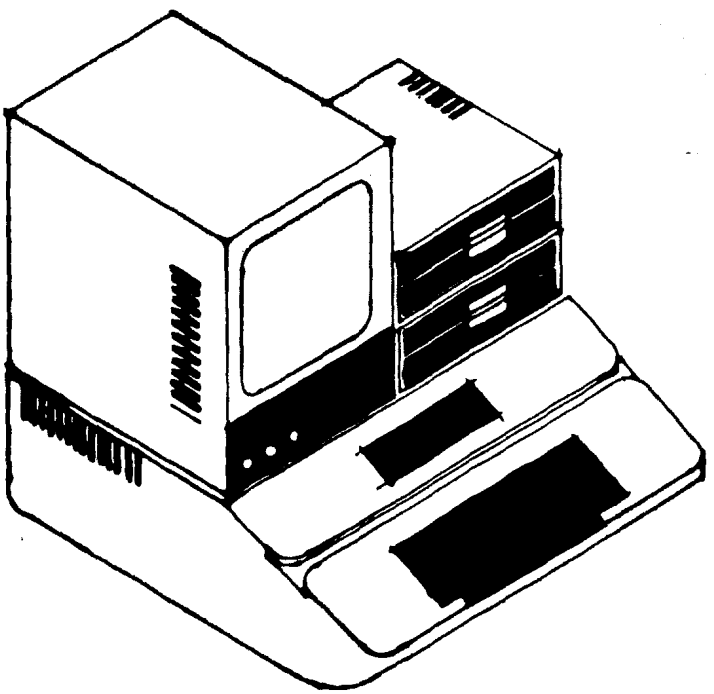


FIGURA I.I. MICROCOMPUTADORA APPLE II PLUS

2.- Descripción General de los Programas.

Los programas de este paquete fueron diseñados e implementados en Pascal bajo el estándar UCSD en una microcomputadora APPLE II.

Se usó la técnica de programación estructurada con el fin de ofrecer programas de mayor calidad que la mayoría de los existentes en esta área. De esta manera, es posible tomarlos como base ya sea para modificarlos y adaptarlos a problemas específicos, o bien para diseñar nuevos programas con características similares.

Cada programa refleja lo mejor posible la técnica de Investigación de Operaciones para la que fue hecho, de manera que, conociendo bien la técnica, es fácil leer el texto del programa.

Se ha hecho lo posible por probarlos con un número apropiado de casos haciendo énfasis en condiciones frontera. Aun así es prácticamente imposible asegurar que están libres de cualquier tipo de error. Es responsabilidad del usuario verificar la validez de los resultados.

En algunos programas aparece la instrucción "uses". En el sistema Pascal APPLE es necesaria esta instrucción para poder usar algunas instrucciones no incluidas en el estándar UCSD. Por ejemplo: se usa la instrucción "uses TRANSCEND" para poder utilizar las funciones trigonométricas en los programas y se usa la instrucción "uses APPLESTUFF" para poder utilizar el generador de números aleatorios (función random).

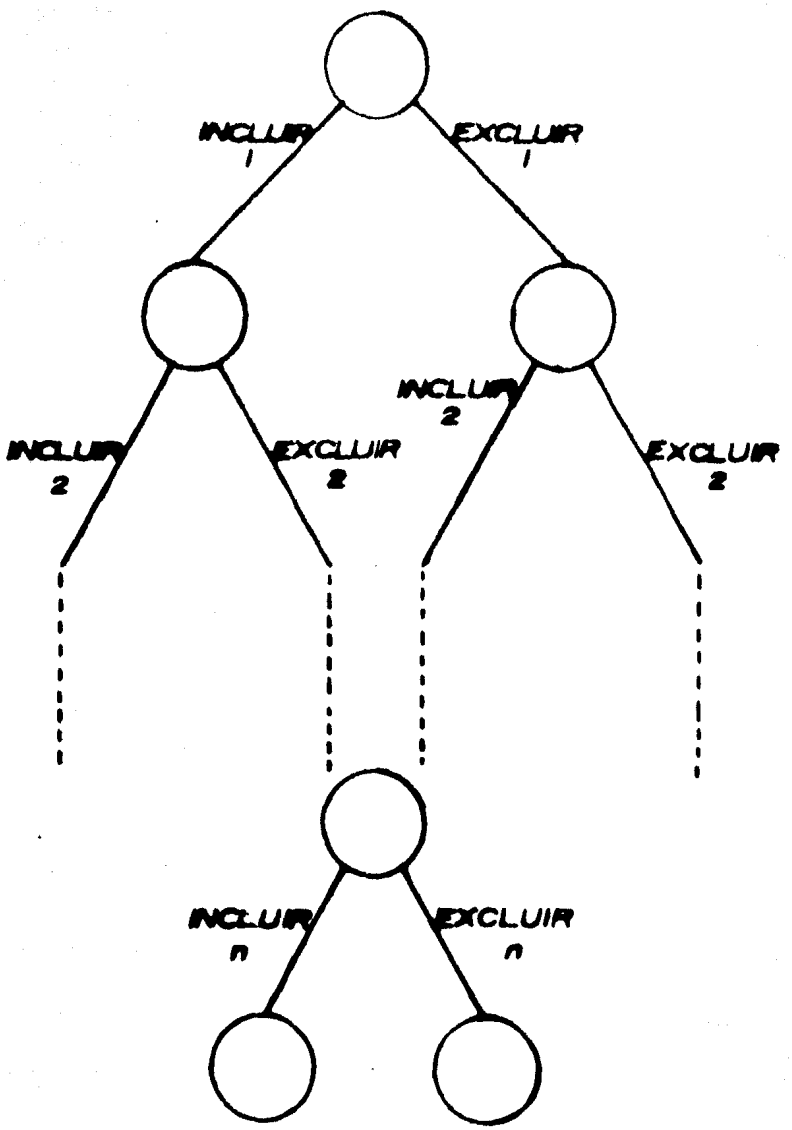


FIGURA 3.1 ARBOL BINARIO

3.- Desarrollo Completo de uno de los Programas.

A continuación se desarrollará totalmente uno de los programas del paquete. Se describirá, analizará y especificará el problema. En seguida se realizará la especificación mediante una serie de pasos justificados informalmente en base a los predicados que componen la mencionada especificación. Por último se darán algunas sugerencias sobre el uso del tipo de programación descrito en este capítulo.

3.1- Descripción del problema.

Suponga que tiene n artículos diferentes y desea buscar una combinación de éstos para formar una carga de valor máximo satisfaciendo una restricción de peso. La información disponible es la siguiente: valor y peso de cada artículo además del peso límite de la carga.

3.2.- Especificación del Problema.

Formalmente, el problema se puede especificar como sigue:

determinar XOpt de tal manera que

(Z=XOpt*V=max(X*V)) y (XOpt*W(=Wl)

donde

N : número de artículos
Z : valor óptimo de la carga
XOpt[i] : decisión de incluir el artículo i en la óptima carga
(1=incluir, 0=excluir)
X[i] : decisión de incluir el artículo i en una carga.

V[i] : valor del artículo i
W[i] : peso del artículo i
Wl : peso límite de la carga
i=1..N

Note que la especificación no indica la manera de realizarla, sólo describe totalmente el problema en términos muy sólidos.

3.3.- Análisis de la especificación.

De lo anterior podemos visualizar que el mecanismo que realice la especificación debe generar combinaciones de artículos y seleccionar, de éstas, la que produzca un valor máximo de la carga satisfaciendo la restricción de peso.

DESARROLLO COMPLETO DE UNO DE LOS PROGRAMAS

Uno de los mecanismos más naturales para generar todas las combinaciones de artículos es un árbol binario (ver figura 3.1). Es decir, una red sin ciclos donde cada nodo, o es una punta, o genera dos ramas:

```
ArbolBin :: <Punta o (ArbolBin,ArbolBin)>
```

3.4.- Especificación del Programa.

En términos de la especificación del problema se escribirá la especificación de un programa que la realice

```
OptimaCarga
sOptimaCarga :: integer,VectReal,VectReal,real,
  real,VectReal
pre-OC(N,V,W,W1,,) = (N>0) y (V[I]>0) y (W[I]>0)
  y (I=1..N)
post-OC(N,V,W,W1,Z',X') =
  (Z'=X'*V=max(X*V)) y (X'*W<=W1)
```

La especificación anterior indica cuál es la relación que deben guardar los valores de las variables antes y después de la ejecución del programa, mediantes la pre y la post condiciones.

3.5.- Realización de la Especificación.

La realización de la especificación se puede hacer como sigue:

```
procedure OptimaCarga(N : integer;
                      V,W : VectReal;
                      W1 : real;
                      var Z : real;
                      var X : VectReal);
begin
  /*
    bOptimaCarga
    sOptimaCarga
    pre-bOC((N,V,W,W1,,)) = (N>0) y (V[I]>0) y (W[I]>0)
      y (I=1..N)
    post-bOC((N,V,W,W1,Z',X') =
      (Z'=X'*V=max(X*V)) y (X'*W<=W1) */

end;
```

La justificación de este paso es inmediata y se basa en la suposición de que la implementación en el compilador de Pascal del paso de parámetros es correcta. Es obvio que un programa cuyo estado final cumpla con la post condición, y pasa correctamente los valores de los parámetros cumple con especificación original.

Antes observamos que el mecanismo que realice estas especificaciones debe generar y evaluar combinaciones de artículos y debe escoger la mejor de las soluciones, por lo que una realización del bloque principal del programa (bOptimaCarga) es:

```
begin (* bOptimaCarga *)
```

```
    Z:=-10E10;
    GeneraComb(N,1,V,W,W1,X,Z,X);

  end;
```

donde

```
  GeneraComb
  sGeneraComb :: integer,integer,VectReal,
    VectReal,real,VectComb,real,VectReal
  pre-OC(<N,Art,V,W,W1,,,,>) = pre-OC
  post-OC(<N,Art,V,W,W1,,,,>,<,,,,,,Z',X'>) =
    (Z'=C"*V=max(C*V)) y (C"*W(=W1)
```

Ya que C es un vector que representa cualquier combinación de artículos, tenemos que max(C*V)=max(X*V) y por lo tanto, al terminar la operación de este procedimiento, Z'=max(C*V)=max(X*V) y como X'=C", entonces X'*W(=W1. Con este se ha probado que la realización propuesta del bloque principal es correcta.

El siguiente paso consiste en la realización de GeneraComb y es aquí donde utilizamos la estructura de árbol binario. Esta realización generará todas las combinaciones de artículos y escogerá la óptima. La estructura recursiva de la realización simulará el árbol binario. La realización es como sigue:

```
procedure GeneraComb(N,Art : integer;
                     V,W : VectReal;
                     W1 : real;
                     C : VectReal;
                     var Z : real;
                     var X : VectReal);
begin
  if Art<N then
    begin
      C[Art]:=1; (* incluye *)
      GeneraComb(N,Art+1,V,W,W1,C,Z,X);
      C[Art]:=0; (* excluye *)
      GeneraComb(N,Art+1,V,W,W1,C,Z,X);
    end
  else
    begin (* punta del arbol *)
      C[Art]:=1;
      (* evalua posible solucion *)
      if (ProdPunto(N,C,V)>Z) and
         (ProdPunto(N,C,W)<=W1 then
        begin
          Z:=ProdPunto(N,C,V);
          X:=C;
        end;
      C[Art]:=0;
```

```
            (* evalua posible solucion *)
            if (ProdPunto(N,C,V)<Z) and
               (ProdPunto(N,C,W)<=W1 then
              begin
                Z:=ProdPunto(N,C,V);
                X:=C;
              end;
          end;

      end;
```

Justificación: si Art<N, el procedimiento genera dos subárboles con las dos únicas combinaciones posibles de ese artículo mediante llamadas recursivas al mismo procedimiento. Cuando Art=N, sólo hay dos combinaciones posibles, las prueba determinando su optimalidad y factibilidad sustituyendo en el caso apropiado.

A continuación se presenta el programa completo.

```
 1:  program pOptimaCarga;
 2:  type
 3:    VectReal = array [1..50] of real;

 4:  var
 5:    N : integer;
 6:    V,W,X : VectReal;
 7:    WL,Z : real;
 8:    DispSal : text;

 9:  procedure IniDispSal;
10:  var
11:    NomDispSal : string;

12:  begin
13:    write('Dispositivo de Salida=');
14:    readln(NomDispSal);
15:    rewrite(DispSal,NomDispSal);

16:  end;

17:  procedure DespVect(N : integer;
18:                     X : VectReal);
19:  var
20:    I : integer;

21:  begin
22:    for I:=1 to N do
23:      begin
24:        writeln(DispSal,'x(',I,')=',X[i]);
25:      end;

26:  end;

27:  function ProdPunto(N : integer;
28:                     X,Y : VectReal) : real;
29:  var
30:    I : integer;
31:    Sum : real;

32:  begin
33:    Sum:=0;
34:    for I:=1 to N do
35:      begin
36:        Sum:=Sum+X[i]*Y[i];
37:      end;
38:    ProdPunto:=Sum;
```

```
39:   end;

40:   procedure LeeVect(N : integer;
41:                     var X : VectReal);
42:   var
43:     I : integer;

44:   begin
45:     for I:=1 to N do
46:       begin
47:         write(DispSal,'X(',I,')=');
48:         readln(X[I]);
49:         writeln(DispSal,X[I]);
50:       end;

51:   end;

52:   procedure Resultados(N : integer;
53:                        V,W : VectReal;
54:                        Wl,Z : real;
55:                        X : VectReal);
56:   begin
57:     writeln(DispSal);
58:     writeln(DispSal,'solucion optima factible:');
59:     writeln(DispSal);
60:     writeln(DispSal,'valor carga=',Z);
61:     writeln(DispSal,'peso carga=',ProdPunto(N,X,W));
62:     writeln(DispSal,'solucion:');
63:     DespVect(N,X);

64:   end;

65:   procedure OptimaCarga(N : integer;
66:                         V,W : VectReal;
67:                         Wl : real;
68:                         var Z : real;
69:                         var X : VectReal);
70:     procedure GeneraCombinacion(N,Art : integer;
71:                                 V,W : VectReal;
72:                                 Wl : real;
73:                                 C : VectReal;
74:                                 var Z : real;
75:                                 var X : VectReal);
76:     var
77:       Cota1,Cota2 : real;

78:     begin
79:       if Art<N then
```

```
 80:         begin
 81:           (* genera dos ramas del arbol *)
 82:           C[art]:=1;
 83:           GeneraCombinacion(N,Art+1,V,W,Wl,
 84:             C,Z,X);
 85:           C[art]:=0;
 86:           GeneraCombinacion(N,art+1,V,W,Wl,
 87:             C,Z,X);
 88:         end
 89:       else
 90:         begin
 91:           (* genera las puntas y checa si hay
 92:              solucion optima factible *)
 93:           C[Art]:=1;
 94:           if (ProdPunto(N,C,V)>Z) and
 95:                (ProdPunto(N,C,W)<=Wl) then
 96:             begin
 97:               Z:=ProdPunto(N,C,V);
 98:               X:=C;
 99:             end;
100:           C[Art]:=0;
101:           if (ProdPunto(N,C,V)>Z) and
102:                (ProdPunto(N,C,W)<=Wl) then
103:             begin
104:               Z:=ProdPunto(N,C,V);
105:               X:=C;
106:             end;
107:         end;

108:     end;

109:   begin
110:     Z:=-10E10;
111:     GeneraComb(N,1,V,W,Wl,X,Z,X);

112:   end;

113:   procedure Lectura(var N : integer;
114:                     var V,W : VectReal;
115:                     var Wl : real);
116:   var
117:     I : integer;

118:   begin
119:     write(DispSal,'numero de articulos=');
120:     readln(N);
121:     writeln(DispSal,N);
122:     writeln(DispSal,'valor de c/art:');
```

```
123:      LeeVect(N,V);
124:      writeln(DispSal,'peso de c/art:');
125:      LeeVect(N,W);
126:      write(DispSal,'peso limite=');
127:      readln(Wl);
128:      writeln(DispSal,Wl);

129:    end;

130:    begin
131:      IniDispSal;
132:      Lectura(N,V,W,Wl);
133:      OptimaCarga(N,V,W,Wl,Z,X);
134:      Resultados(N,V,W,Wl,Z,X);
135:      close(DispSal,lock);

136:    end.
```

3.6.- Evaluación de la Realización.

El problema de esta realización es su alto tiempo de ejecución. Recordemos que su funcionamiento (correcto) depende de generar 2**n combinaciones, lo que indica que si aumentamos un artículo, el tiempo de ejecución se duplica; con cincuenta artículos necesitaría generar 2**50 aprox=1.13E15=1,130,000,000,000,000 combinaciones comparado a 2**7=128 combinaciones con siete artículos.

Una manera de reducir la tarea del programa consiste en no recorrer todo el árbol; si supiéramos de antemano que una rama no tiene posibilidades de mejorar la solución, esto nos evitaría recorrer ese subárbol con el consecuente ahorro en tiempo. Nos proponemos, entonces, a introducir al programa una función que evalue las ramas del árbol y, de acuerdo a este valor, generar o no el subárbol. La función de evaluación que daremos consiste en fijar una cota superior a todas las posibles soluciones del subárbol, de manera que, si la cota es menor que el valor de la última mejor solución, no tiene caso generar ese subárbol. La función que proponemos está dada en el listado del programa final; observe que el uso de esta función para decidir si se genera un subárbol o no, no tiene posibilidades de hacer fallar la realización ya que no genera el subárbol cuando no existen mejores soluciones en ése. La función es muy simple: toma la combinación parcial y la completa incluyendo lo más que se pueda de los artículos restantes incluyendo primero los de mayor valor, esto garantiza que si en el subárbol existiera una solución factible, su valor sería menor o igual que la cota así determinada. Para realizar de una manera simple esta función, se ordenaron los artículos de acuerdo a su razón valor/costo, garantizando esto que los primeros artículos que se incluirán para evaluar la cota son los de mayor valor con respecto a su peso.

El programa final se muestra en a continuación y notará algunas otras afinaciones con el fin de no evaluar demasiadas veces una función con los mismos valores. Este programa es el mismo que el programa I00402.

```
 1:  program pOptimaCarga;
 2:  type
 3:    VectReal = array [1..50] of real;
 4:    VectInt = array [1..50] of integer;

 5:  var
 6:    N : integer;
 7:    V,W,X : VectReal;
 8:    WL,Z : real;
 9:    Ord : VectInt;
10:    DispSal : text;

11:  procedure IniDispSal;
12:  var
13:    NomDispSal : string;

14:  begin
15:    write('Dispositivo de Salida=');
16:    readln(NomDispSal);
17:    rewrite(DispSal,NomDispSal);

18:  end;

19:  procedure DespVect(N : integer;
20:                     X : VectReal;
21:                     Ord : VectInt);
22:  var
23:    I : integer;

24:  begin
25:    (* desplegar vector conservando orden
26:       original Ord *)
27:    for I:=1 to N do
28:      begin
29:        writeln(DispSal,'x(',Ord[I],')=',X[i]);
30:      end;

31:  end;

32:  procedure Sort(N : integer;
33:                 var V,W : VectReal;
34:                 var Ord : VectInt);
35:  var
36:    I,ITemp : integer;
37:    Temp : real;
38:    Fl : boolean;

39:  begin
```

```
40:     For I:=1 to N do
41:       begin
42:         Ord[I]:=I;
43:       end;
44:     repeat
45:       F1:=false;
46:       for I:=2 to N do
47:         begin
48:           if V[I]/W[I]>V[I-1]/W[I-1] then
49:             begin
50:               ITemp:=Ord[I];
51:               Ord[I]:=Ord[I-1];
52:               Ord[I-1]:=ITemp;
53:               Temp:=V[I];
54:               V[I]:=V[I-1];
55:               V[I-1]:=Temp;
56:               Temp:=W[I];
57:               W[I]:=W[I-1];
58:               W[I-1]:=Temp;
59:               F1:=true;
60:             end;
61:         end;
62:       N:=N-1;

63:     until not F1;

64:   end;

65:   function ProdPunto(N : integer;
66:                      X,Y : VectReal) : real;
67:   var
68:     I : integer;
69:     Sum : real;

70:   begin
71:     Sum:=0;
72:     for I:=1 to N do
73:       begin
74:         Sum:=Sum+X[I]*Y[I];
75:       end;
76:     ProdPunto:=Sum;

77:   end;

78:   procedure LecVect(N : integer;
79:                     var X : VectReal);
80:   var
81:     I : integer;
```

```
 82:   begin
 83:     for I:=1 to N do
 84:       begin
 85:         write(DispSal,'x(',I,')=');
 86:         readln(X[I]);
 87:         writeln(DispSal,X[I]);
 88:       end;

 89:   end;

 90:   procedure Resultados(N : integer;
 91:                        V,W : VectReal;
 92:                        Wl,Z : real;
 93:                        X : VectReal;
 94:                        Ord : VectInt);
 95:   begin
 96:     writeln(DispSal);
 97:     writeln(DispSal,'solucion optima factible:');
 98:     writeln(DispSal);
 99:     writeln(DispSal,'valor carga=',Z);
100:     writeln(DispSal,'peso carga=',ProdPunto(N,X,W));
101:     writeln(DispSal,'solucion:');
102:     DespVect(N,X,Ord);

103:   end;

104:   procedure OptimaCarga(N : integer;
105:                         V,W : VectReal;
106:                         Wl : real;
107:                         var Z : real;
108:                         var X : VectReal);
109:     function CotaMax(N,Art : integer;
110:                      Wl : real;
111:                      V,W : VectReal;
112:                      C : VectReal) : real;
113:     var
114:       Sum,Wa : real;

115:     begin
116:       Sum:=ProdPunto(Art,C,V);
117:       (* de los articulos no asignados incluir
118:          los de mayor valor incremental primero
119:          hasta llegar al peso limite.
120:          esto lo garantiza el Sort *)
121:       Wa:=Wl-ProdPunto(Art,C,W);
122:       if Wa>=0 then
123:         begin
```

```
124:            repeat
125:              Art:=Art+1;
126:              if WEArt]<=Wa then
127:                begin
128:                  Wa:=Wa-WEArt];
129:                  Sum:=Sum+VEArt];
130:                end
131:              else
132:                begin
133:                  Sum:=Sum+(VEArt]/WEArt])*Wa;
134:                  Wa:=0;
135:                end;

136:            until (Art=N) or (Wa=0);
137:            CotaMax:=Sum;
138:          end
139:        else (* ninguna solucion factible en subarbol *)
140:          CotaMax:=-10E10;

141:      end;

142:      procedure GeneraCombinacion(N.Art : integer;
143:                                  V.W : VectReal;
144:                                  Wl : real;
145:                                  C : VectReal;
146:                                  var Z : real;
147:                                  var X : VectReal);
148:      var
149:        Cota1.Cota2 : real;

150:      begin
151:        if Art<N then
152:          begin
153:            (* genera dos ramas del arbol *)
154:            CEArt]:=1; (* incluye *)
155:            Cota1:=CotaMax(N.Art.Wl.V.W.C);
156:            CEArt]:=0; (* excluye *)
157:            Cota2:=CotaMax(N.Art.Wl.V.W.C);
158:            if Cota1>Cota2 then
159:              begin
160:                if Cota1>Z then
161:                  begin
162:                    CArt]:=1;
163:                    GeneraCombinacion(N.Art+1.V.W.Wl.
164:                      C.Z.X);
165:                    if Cota2>Z then
166:                      begin
167:                        CArt]:=0;
```

```
168:                        GeneraCombinacion(N,Art+1,
169:                          V,W,Wl,C,Z,X);
170:                      end;
171:                  end;
172:              end
173:            else
174:              begin
175:                if Cota2>Z then
176:                  begin
177:                    C[Art]:=0;
178:                    GeneraCombinacion(N,Art+1,V,W,Wl,
179:                      C,Z,X);
180:                    if Cota1>Z then
181:                      begin
182:                        C[Art]:=1;
183:                        GeneraCombinacion(N,Art+1,
184:                          V,W,Wl,C,Z,X);
185:                      end;
186:                  end;
187:              end;
188:          end
189:        else
190:          begin
191:            (* genera las puntas y checa si hay
192:               solucion optima factible *)
193:            C[Art]:=1;
194:            if (ProdPunto(N,C,V)>Z) and
195:                 (ProdPunto(N,C,W)<=Wl) then
196:              begin
197:                Z:=ProdPunto(N,C,V);
198:                X:=C;
199:              end;
200:            C[Art]:=0;
201:            if (ProdPunto(N,C,V)>Z) and
202:                 (ProdPunto(N,C,W)<=Wl) then
203:              begin
204:                Z:=ProdPunto(N,C,V);
205:                X:=C;
206:              end;
207:          end;

208:      end;

209:    begin
210:      Z:=-10E10;
211:      GeneraComb(N,1,V,W,Wl,X,Z,X);

212:    end;
```

```
213:  procedure Lectura(var N : integer;
214:                      var V,W : VectReal;
215:                      var W1 : real);
216:  var
217:    I : integer;

218:  begin
219:    write(DispSal,'numero de articulos=');
220:    readln(N);
221:    writeln(DispSal,N);
222:    writeln(DispSal,'valor de c/art:');
223:    LecVect(N,V);
224:    writeln(DispSal,'peso de c/art:');
225:    LecVect(N,W);
226:    write(DispSal,'peso limite=');
227:    readln(W1);
228:    writeln(DispSal,W1);

229:  end;

230:  begin
231:    IniDispSal;
232:    Lectura(N,V,W,W1);
233:    Sort(N,V,W,Ord);
234:    OptimaCarga(N,V,W,W1,Z,X);
235:    Resultados(N,V,W,W1,Z,X,Ord);
236:    close(DispSal,lock);

237:  end.
```

3.7.- Conclusiones.

El uso de la programación estructurada es una de las mejores herramientas disponibles al programador. Permite descomponer el problema en subproblemas y realizarlo a través de una jerarquía de módulos. Esto permite que módulos existentes puedan ser usados como parte de la realización de otros. En cada paso se debe verificar que la realización cumpla con la especificación dada. El resultado de esta metodología iterativa, descomposición-especificación-verificación, es la producción de programas de alta calidad.

Además de la programación estructurada, se deben desarrollar o adquirir otras herramientas; un buen editor de textos es esencial. Así mismo, es posible desarrollar bibliotecas de procedimientos y funciones como una manera de extender el lenguaje computacional usado. Al escribir la documentación de los programas es muy útil contar con un buen procesador de textos (la producción de esta tesis se hizo utilizando un programa de este tipo). Además, existen herramientas de programación como: procesadores de tablas de decisiones, generadores y verificadores de programas, etc.

El uso de herramientas computacionales debe constituir una parte integral del proceso de programación.

4.- I001. Descripcion y ejemplo de cada programa.

En esta sección se describe la función de cada programa, así como la manera de usar cada uno de éstos. A manera de facilitar su uso, en cada uno se ilustran uno o varios ejemplos. Es recomendable usarlos para verificar el manejo correcto del módulo.

Cada descripción de programa incluye: breve descripción del tipo de problemas que resuelve, modo de uso, variables que intervienen, un ejemplo listado del programa fuente y una corrida del ejemplo.

Los programas de la sección 4.5, incluyen opciones de corrección como parte de sus rutinas de entrada. El lector podría incluir instrucciones como éstas en los demás programas si así lo juzgara necesario.

4.1.- Programación Dinámica.

No existe un algoritmo general para la Programación Dinámica, así como el Método Simplex para la Programación Lineal. En realidad, se podría decir que la Programación Dinámica es un enfoque en la solución de problemas. Este enfoque consiste en descomponer el problema en un conjunto de subproblemas o etapas y aplicar el siguiente método:

1.- Suponemos que el problema consiste sólo de la última etapa.
2.- Buscamos la solución óptima del problema.
3.- Agregamos la siguiente etapa, obteniendo así un nuevo problema.
4.- Buscamos la solución óptima del problema.
5.- Si todavía no se han agregado todas las etapas, continuar desde el paso 3.
6.- Tenemos la solución optima del problema formado al agregar todas las etapas componentes y esto es equivalente a tener la solución óptima del problema original.

Esta metodología para enfocar y resolver problemas está basada en el principio de optimalidad de Bellman:

- Una política óptima tiene la propiedad de que, cualesquiera que sean el estado y la decisión inicial, las decisiones restantes deben constituir una política optima con respecto al estado resultante de la primera decisión. -

Se ha usado la Programación Dinámica para resolver una gran variedad de clases de problemas, tales como: secuenciación, reemplazo de equipo, programación de actividades, inventarios, etc. Debe señalarse que, en algunos casos, no es el método más eficiente para obtener la solución óptima y es necesario recurrir a otra técnica.

4.1.1.- I00101. Destinar k Unidades de Recurso a n Actividades Maximizando el Beneficio.

Supóngase que se desea destinar una cantidad fija de algún recurso a una serie de actividades a manera de maximizar la ganancia esperada. Para poder aplicar la Programación dinámica, suponemos que:

1.- Las ganancias de cada actividad se miden en las mismas unidades.
2.- La ganancia total es la suma de las ganancias de las actividades individuales.
3.- La ganancia de cada actividad es independiente de la cantidad de recurso asignado a las demás.

El programa I00101 acepta los siguientes datos:

```
n        : número de actividades.(entero)
K        : cantidad de recurso.(entero)
g[i,x]   : ganancia al asignar x unidades de recurso
  a la actividad i.(entero)
```

Y produce las siguientes cantidades:

```
gopt     : ganancia óptima.(entero)
xopt[i] : cantidad de recurso a asignar
  a actividad i.(entero)
  i=1..n, x=0..k.
```

Ejemplo: Suponga que se quiere destinar 5 unidades de recurso a 3 actividades cuyas ganancias están expresadas en la siguiente tabla:

x	0	1	2	3	4	5
g[1,x]	0	5	10	11	23	24
g[2,x]	0	5	9	15	20	25
g[3,x]	0	3	10	16	21	24

La respuesta calculada por el programa es:

```
gopt = 28
xopt = [4 1 0]
```

```
 1:  program IO0101;
 2:  var
 3:    N,K,X,Z,I : integer;
 4:    G,F,D : array [1..5,0..101] of integer;
 5:    XOPT : array [1..5] of integer;
 6:    DispSal : text;

 7:  procedure IniDispSal;
 8:  var
 9:    NomDispSal : string;

10:  begin
11:    write('Dispositivo de Salida=');
12:    readln(NomDispSal);
13:    rewrite(DispSal,NomDispSal);

14:  end;

15:  procedure LEEDATOS;
16:  begin
17:    writeln(DispSal);
18:    writeln(DispSal,'IO0101. DESTINAR K UNIDADES DE',
19:      ' RECURSO');
20:    writeln(DispSal,'A N ACTIVIDADES CON BENEFICIO',
21:      ' MAXIMO');
22:    writeln(DispSal);
23:    write(DispSal,'NUMERO DE ACTIVIDADES?');
24:    readln(N);
25:    writeln(DispSal,N);
26:    write(DispSal,'CANTIDAD DE RECURSO?');
27:    readln(K);
28:    writeln(DispSal,K);
29:    for I:=1 to N do
30:      begin
31:        XOPT[I]:=0;
32:        for X:=0 to K do
33:          begin
34:            write(DispSal,'GANANCIA: ACT ',I,
35:                  ' RECURSO ',X,' =');
36:            readln(G[I,X]);
37:            writeln(DispSal,G[I,X]);
38:            F[I,X]:=0;
39:            D[I,X]:=0;
40:            if I=N then F[N+1,X]:=0
41:          end;
42:      end;

43:  end;
```

```
44:   procedure CALCULO;
45:   begin
46:     for I:=N downto 1 do
47:       for X:=0 to K do
48:         for Z:=0 to X do
49:           begin
50:             if F[I,X]<G[I,Z]+F[I+1,X-Z] then
51:               begin
52:                 F[I,X]:= G[I,Z]+F[I+1,X-Z];
53:                 D[I,X]:=Z
54:               end;
55:           end;

56:   end;

57:   procedure REPORTA;
58:   begin
59:     writeln(DispSal);
60:     writeln(DispSal,'RESULTADOS:');
61:     writeln(DispSal);
62:     writeln(DispSal,'BENEFICIO OPTIMO = ',F[1,K]);
63:     for I:=1 to N do
64:       begin
65:         XOPT[I]:=D[I,K];
66:         writeln(DispSal,'CANTIDAD A DESTINAR A ACTIVIDAD ',
67:                 I,' = ',XOPT[I]);
68:         K:=K-XOPT[I];
69:       end;

70:   end;

71:   begin
72:     IniDispSal;
73:     LEEDATOS;
74:     CALCULO;
75:     REPORTA;
76:     close(DispSal,lock);

77:   end.
```

```
 1:   I00101. DESTINAR K UNIDADES DE RECURSO
 2:   A N ACTIVIDADES CON BENEFICIO MAXIMO

 3:   NUMERO DE ACTIVIDADES?3
 4:   CANTIDAD DE RECURSO?5
 5:   GANANCIA: ACT 1 RECURSO 0 =0
 6:   GANANCIA: ACT 1 RECURSO 1 =5
 7:   GANANCIA: ACT 1 RECURSO 2 =10
 8:   GANANCIA: ACT 1 RECURSO 3 =11
 9:   GANANCIA: ACT 1 RECURSO 4 =23
10:   GANANCIA: ACT 1 RECURSO 5 =24
11:   GANANCIA: ACT 2 RECURSO 0 =0
12:   GANANCIA: ACT 2 RECURSO 1 =5
13:   GANANCIA: ACT 2 RECURSO 2 =9
14:   GANANCIA: ACT 2 RECURSO 3 =15
15:   GANANCIA: ACT 2 RECURSO 4 =20
16:   GANANCIA: ACT 2 RECURSO 5 =25
17:   GANANCIA: ACT 3 RECURSO 0 =0
18:   GANANCIA: ACT 3 RECURSO 1 =3
19:   GANANCIA: ACT 3 RECURSO 2 =10
20:   GANANCIA: ACT 3 RECURSO 3 =16
21:   GANANCIA: ACT 3 RECURSO 4 =21
22:   GANANCIA: ACT 3 RECURSO 5 =24

23:   RESULTADOS:

24:   BENEFICIO OPTIMO = 28
25:   CANTIDAD A DESTINAR A ACTIVIDAD 1 = 4
26:   CANTIDAD A DESTINAR A ACTIVIDAD 2 = 1
27:   CANTIDAD A DESTINAR A ACTIVIDAD 3 = 0
```

```
 1:   IO1204. TIEMPO DE LLEGADA POR PRIMERA VEZ

 2:   NUMERO DE ESTADOS=2
 3:   MATRIZ DE TRANSICION:
 4:   A(11)= 9.00000E-1
 5:   A(12)= 1.00000E-1
 6:   A(21)= 2.00000E-1
 7:   A(22)= 8.00000E-1

 8:   MATRIZ DE TRANSICION ORIGINAL:
 9:   A(11)= 9.00000E-1
10:   A(12)= 1.00000E-1
11:   A(21)= 2.00000E-1
12:   A(22)= 8.00000E-1

13:   MATRIZ DE TIEMPOS DE LLEGADAS POR 1A VEZ:
14:   A(11)= 0.00000
15:   A(12)= 1.00000E1
16:   A(21)= 5.00000
17:   A(22)= 0.00000
```

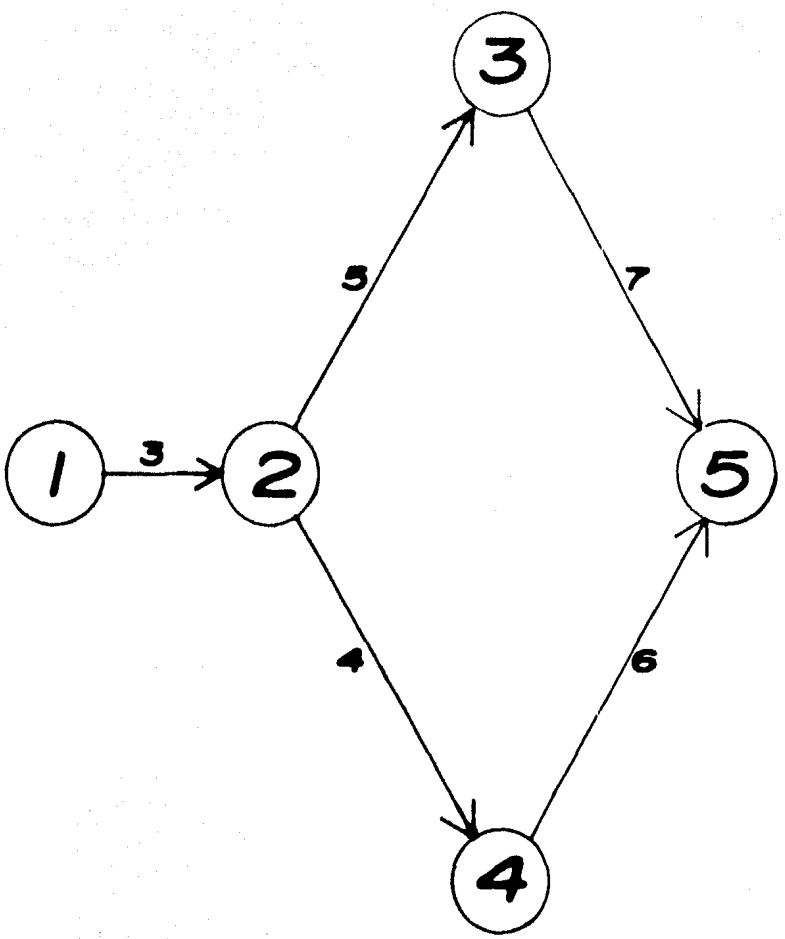


FIGURA 4.1.2.1 RED

4.1.2.- I00102. Determinar la Ruta de Mínimo Costo en una Red.

Considere el problema de encontrar la ruta que minimice el costo al recorrer una red desde el nodo inicial hasta el final.

Para utilizar la Programación Dinámica son necesarias las siguientes suposiciones:

1.- La red puede ser dividida en etapas.
2.- No se puede pasar de un nodo a otro dentro de la misma etapa.
3.- La primera etapa consiste únicamente del nodo 1 y la última consiste del nodo final.

Mediante un ejemplo describimos la forma de utilizar el programa:

1.- Dibujar la red e identificar las etapas.

(ver figura 4.1.2.1)

2.- Construir una cadena usando los nodos de la red seguidos cada uno de los nodos con los cuales están comunicados.

1 2 2 3 4 3 5 4 5 5

3.- Para cada elemento de la cadena indicar el costo asociado.

```
1 0
2 3
2 0
3 5
4 4  --> (costo de ir de 2 a 4 = 4)
3 0
5 7
4 0
5 6
5 0
```

4.- Con el problema expresado en esta forma, el programa acepta los siguientes datos:

```
ne     : número de nodos.(entero)
n      : número de etapas.(entero)
t      : tamaño de la cadena.(entero)
m[i]   : número de nodos en la etapa i.(entero)
ca[j]  : cadena.(entero)
co[j]  : costo asociado.(real)
```

Y produce las siguientes cantidades:

```
cm     : costo minimo.(real)
r[i]   : ruta. Nodo a llegar en cada etapa.(entero)
  i=1..n, j=1..t
```

Usando la información del ejemplo:

```
ne = 5
n  = 4
t  = 10
m  = [1 1 2 1]
ca = [1 2 2 3 4 3 5 4 5 5]
co = [0 3 0 5 4 0 7 0 6 0]
```

La solución al problema es:

```
cm = 13
r  = [1 2 4 5]
```

```
 1:  program IO0102;
 2:  var
 3:    I,J,K,N,T,NE,E,A : integer;
 4:    CM : real;
 5:    CO : array [1..200] of real;
 6:    CA : array [1..200] of integer;
 7:    F : array [1..50] of real;
 8:    D : array [1..50] of integer;
 9:    M,R : array [1..20] of integer;
10:    DispSal : text;

11:  procedure IniDispSal;
12:  var
13:    NomDispSal : string;

14:  begin
15:    write('Dispositivo de Salida=');
16:    readln(NomDispSal);
17:    rewrite(DispSal,NomDispSal);

18:  end;

19:  procedure LECTURA;
20:  begin
21:    writeln(DispSal,'IO0102. RUTA MINIMA');
22:    write(DispSal,'NE,N,T = ');
23:    readln(NE,N,T);
24:    writeln(DispSal,' ',NE,' ',N,' ',T);
25:    for I:=1 to N do
26:      begin
27:        write(DispSal,'ETAPA ',I,' # EDOS ?');
28:        readln(M[I]);
29:        writeln(DispSal,M[I]);
30:      end;
31:    writeln(DispSal,'TECLEE CADENA Y COSTOS');
32:    for I:=1to T do
33:      begin
34:        write(DispSal,'EDO,COSTO ?');
35:        readln(CA[I],CO[I]);
36:        writeln(DispSal,CA[I],CO[I]);
37:      end;

38:  end;

39:  procedure INICIALIZA;
40:  begin
41:    A:=T;
42:    E:=NE;
```

```
43:     for I:=1 to NE do
44:       F[I]:=0;

45:   end;

46:   procedure EDOINI;
47:   begin
48:     J:=A;
49:     E:=E-M[I];
50:     while CA[J]<>E do
51:       J:=J-1;
52:     A:=J;

53:   end;

54:   procedure COSMIN;
55:   begin
56:     J:=J+1;
57:     F[K]:=CO[J]+F[CA[J]];
58:     D[K]:=CA[J];
59:     J:=J+1;
60:     while CO[J]<>0 do
61:       begin
62:         if (F[K]>CO[J]+F[CA[J]]) then
63:           begin
64:             F[K]:=CO[J]+F[CA[J]];
65:             D[K]:=CA[J]
66:           end;
67:         J:=J+1;
68:       end;

69:   end;

70:   procedure RUTMIN;
71:   begin
72:     for K:=E to (E+M[I]-1) do
73:       COSMIN;

74:   end;

75:   procedure REPORTA;
76:   begin
77:     writeln(DispSal);
78:     writeln(DispSal,'RESULTADOS:');
79:     writeln(DispSal);
80:     writeln(DispSal,'TIEMPO MINIMO = ',F[1]);
81:     writeln(DispSal,'LA RUTA MAS CORTA ES: ');
82:     write(DispSal,' ',CA[1],' ');
```

```
 83:     for I:=1 to N-1 do
 84:       write(DispSal,' ',R[I],' ');

 85:   end;

 86:   begin
 87:     IniDispSal;
 88:     LECTURA;
 89:     INICIALIZA;
 90:     for I:=N-1 downto 1 do
 91:       begin
 92:         EDOINI;
 93:         RUTMIN
 94:       end;
 95:     R[1]:=D[1];
 96:     for I:=2 to N do
 97:       R[I]:=D[R[I-1]];
 98:     REPORTA;
 99:     close(DispSal,lock);

100:   end.
```

```
 1:   IO0102. RUTA MINIMA
 2:   NE,N,T =  5 4 10
 3:   ETAPA 1 # EDOS ?1
 4:   ETAPA 2 # EDOS ?1
 5:   ETAPA 3 # EDOS ?2
 6:   ETAPA 4 # EDOS ?1
 7:   TECLEE CADENA Y COSTOS
 8:   EDO,COSTO ?1 0.00000
 9:   EDO,COSTO ?2 3.00000
10:   EDO,COSTO ?2 0.00000
11:   EDO,COSTO ?3 5.00000
12:   EDO,COSTO ?4 4.00000
13:   EDO,COSTO ?3 0.00000
14:   EDO,COSTO ?5 7.00000
15:   EDO,COSTO ?4 0.00000
16:   EDO,COSTO ?5 6.00000
17:   EDO,COSTO ?5 0.00000

18:   RESULTADOS:

19:   TIEMPO MINIMO =  1.30000E1
20:   LA RUTA MAS CORTA ES:
21:    1  2  4  5
```

4.1.3.- IO0103. Programación de Producción.

Se desea programar la producción de cierto producto durante n periodos, conociendo la demanda en cada periodo, de manera de minimizar los costos de producción e inventario.

Suponemos:

1.- Los costos de producción e inventario pueden expresarse como funciones lineales del número de unidades del producto.
2.- La cantidad del producto en inventario al principio del primer periodo y al final del último es igual a cero.
3.- La cantidad en inventario a fines del periodo i-1 y la producida durante i está disponible para satisfacer la demanda en cualquier momento de i.

El programa acepta los siguientes datos:

n : número de periodos.(entero)
b1 : intercepción de la función de costo de producción.(entero)
m1 : pendiente de la función de costo de producción.(entero)
b2 : intercepción de la función de costo de inventario.(entero)
m2 : pendiente de la función de costo de inventario.(entero)
d[i] : demanda para el periodo i.(entero)

Y calcula las siguientes cantidades:

cm : costo mínimo de inventario y de producción.(entero)
xopt[i] : cantidad óptima a producir en el periodo i.(entero)

Ejemplo: Suponga que se tiene la siguiente información:

n = 5
b1 = 20
m1 = 5
b2 = 5
m2 = 1
d = [4 4 2 6 8]

La solución óptima es:

cm = 191
xopt = [10 0 0 14 0]

```
 1:  program IO0103;
 2:  var
 3:    I,J,K,N,Z,DT,MAXD,MINP,MAXP : integer;
 4:    CM,TEMP,COSTO : integer;
 5:    B1,M1,B2,M2 : integer;
 6:    D,DE : array [1..25] of integer;
 7:    F,X : array [1..25,0..200] of integer;
 8:    DispSal : text;

 9:  procedure IniDispSal;
10:  var
11:    NomDispSal : string;

12:  begin
13:    write('Dispositivo de Salida=');
14:    readln(NomDispSal);
15:    rewrite(DispSal,NomDispSal);

16:  end;

17:  procedure REPORTA;
18:  begin
19:    CM:=F[1,0];
20:    writeln(DispSal);
21:    writeln(DispSal,'RESULTADOS:');
22:    writeln(DispSal);
23:    writeln(DispSal,'COSTO MINIMO = ',CM);
24:    writeln(DispSal);
25:    for I:=1 to N do
26:      begin
27:        writeln(DispSal,'PERIODO ',I,
28:                  ' PRODUCIR ',DE[I]);
29:      end;

30:  end;

31:  procedure OPTIMOS;
32:  begin
33:    DE[1]:=X[1,0];
34:    TEMP:=DE[1];
35:    for I:=2 to N do
36:      begin
37:        TEMP:=TEMP-D[I-1];
38:        DE[I]:=X[I,TEMP];
39:        if TEMP=0 then
40:          TEMP:=X[I,TEMP];
41:      end;
```

```
42:   end;

43:   procedure DEMANDATOT;
44:   begin
45:     DT:=0;
46:       for J:=I to N do
47:         DT:=DT+D[J];

48:   end;

49:   procedure CALCOSTO;
50:     function PC(Z : integer) : integer;
51:     begin
52:       if Z>0 then
53:         PC:=B1+M1*Z
54:       else
55:         PC:=0;

56:     end;

57:     function EIC(Z : integer) : integer;
58:     begin
59:       if Z>0 then
60:         EIC:=B2+M2*Z
61:       else
62:         EIC:=0;

63:     end;

64:   begin
65:     COSTO:=PC(Z)+EIC(Z+K-D[I])+F[I+1,Z+K-D[I]];

66:   end;

67:   procedure OPTK;
68:   begin
69:     if (D[I]-K)<0 then
70:       MINP:=0
71:     else
72:       MINP:=(D[I]-K);
73:     MAXP:=DT-K;
74:     Z:=MINP;
75:     CALCOSTO;
76:     F[I,K]:=COSTO;
77:     X[I,K]:=Z;
78:     MINP:=MINP+1;
79:     for Z:=MINP to MAXP do
80:       begin
```

```
 81:          CALCOSTO;
 82:          if F[I,K]>COSTO then
 83:            begin
 84:              F[I,K]:=COSTO;
 85:              X[I,K]:=Z
 86:            end;
 87:        end;

 88:    end;

 89:    procedure ETAPA;
 90:    begin
 91:      DEMANDATOT;
 92:      for K:=0 to DT do
 93:        OPTK;

 94:    end;

 95:    procedure INICIALIZA;
 96:    begin
 97:      I:=1;
 98:      DEMANDATOT;
 99:      MAXD:=DT;
100:      I:=N+1;
101:      for K:=0 to DT do
102:        F[I,K]:=0;

103:    end;

104:    procedure LECTURA;
105:    begin
106:      writeln(DispSal,'I00103. PROGRAMACION',
107:        ' DE PRODUCCION');
108:      write(DispSal,'NUMERO DE PERIODOS? ');
109:      readln(N);
110:      writeln(DispSal,N);
111:      writeln(DispSal,
112:         'INTERCEPT Y PENDIENTE DE FNS DE COSTO? ');
113:      readln(B1,M1,B2,M2);
114:      writeln(DispSal,B1,' ',M1,' ',B2,' ',M2);
115:      for I:=1 to N do
116:        begin
117:          write(DispSal,'PERIODO ',I,' DEMANDA? ');
118:          readln(D[I]);
119:          writeln(DispSal,D[I]);
120:        end;

121:    end;
```

```
122:  begin
123:    IniDispSal;
124:    LECTURA;
125:    INICIALIZA;
126:    for I:=N downto 1 do
127:      begin
128:        ETAPA;
129:      end;
130:    OPTIMOS;
131:    REPORTA;
132:    close(DispSal,lock);

133:  end.
```

```
 1:   IO0103. PROGRAMACION DE PRODUCCION
 2:   NUMERO DE PERIODOS? 5
 3:   INTERCEPT Y PENDIENTE DE FNS DE COSTO?
 4:   20 5 5 1
 5:   PERIODO 1 DEMANDA? 4
 6:   PERIODO 2 DEMANDA? 4
 7:   PERIODO 3 DEMANDA? 2
 8:   PERIODO 4 DEMANDA? 6
 9:   PERIODO 5 DEMANDA? 8

10:   RESULTADOS:

11:   COSTO MINIMO = 191

12:   PERIODO 1 PRODUCIR 10
13:   PERIODO 2 PRODUCIR 0
14:   PERIODO 3 PRODUCIR 0
15:   PERIODO 4 PRODUCIR 14
16:   PERIODO 5 PRODUCIR 0
```

4.1.4.- I00104. Reemplazo de Equipo.

Suponga que se desea determinar una política de reemplazo para un equipo a lo largo de n años. Si tenemos estimaciones de la ganancia que produce el equipo, su costo por mantenimiento y su costo de reemplazo, es posible usar la Programación Dinámica para encontrar una política óptima de reemplazo.

El programa acepta los siguientes datos:

n : número de años para la política.(entero)
it : edad inicial del equipo.(entero)
a : factor de descuento.(entero)
r[i,t] : ganancia en el año i de un equipo hecho en el año i-t y que tiene t años al principio del año i.(entero)
u[i,t] : costo por mantenimiento en el año i de un equipo hecho en el año i-t y que tiene t años al principio del año i.(entero)
c[i,t] : costo de reemplazo en el año i de un equipo hecho en el año i-t y que tiene t años en el año i.(entero)

Y produce las siguientes cantidades:

gopt : ganancia óptima.(entero)
x[i] : decisión a tomar en el año i.
1=reemplazo, 0=no reemplazo.(entero).
i=1..n, t=0,1,..,i-1,i+it-1

Ejemplo: Suponga que se quiere determinar la política de reemplazo a lo largo de 3 años para un equipo con 2 años de edad. La ganancia y los costos para este tipo de equipo se encuentran en las siguientes tablas:

EQUIPO HECHO EN 1981 (año -1)

EDAD	2	3	4
GANANCIA	10	8	8
MANTENIMIENTO	3	3	4
REEMPLAZO	25	26	27

EQUIPO HECHO EN 1983 (año 1)

EDAD	0	1	2
GANANCIA	14	16	16
MANTENIMIENTO	1	1	2
REEMPLAZO	20	22	24

EQUIPO HECHO EN 1984 (año 2)

EDAD	0	1
GANANCIA	16	14
MANTENIMIENTO	1	1
REEMPLAZO	20	22

EQUIPO HECHO EN 1985 (año 3)

EDAD	0
GANANCIA	18
MANTENIMIENTO	1
REEMPLAZO	20

Para usar el programa es necesario expresar estos datos en la forma adecuada:

```
n = 3
it = 2
a = 1

r[1 0]=14   r[2 0]=16   r[3 0]=18
u[1 0]=1    u[2 0]=1    u[3 0]=1
c[1 0]=20   c[2 0]=20   c[3 0]=20

r[1 2]=10   r[2 1]=16   r[3 1]=14
u[1 2]=3    u[2 1]=1    u[3 1]=1
c[1 2]=25   c[2 1]=22   c[3 1]=22

            r[2 3]=8    r[3 2]=16
            u[2 3]=3    u[3 2]=2
            c[2 3]=26   c[3 2]=24

                        r[3 4]=8
                        u[3 4]=4
                        c[3 4]=27
```

Obteniendo la solución óptima:

```
gopt = 17
x = [1 0 0]
```

```
 1:  program IO0104;
 2:  var
 3:    COPT,N,IT,A,I,T,CO,NCO : integer;
 4:    R,U,C,D : array [0..20,0..25] of integer;
 5:    F : array [0..21,0..26] of integer;
 6:    X : array [0..20] of integer;
 7:    P : array [1..20] of char;
 8:    DispSal : text;

 9:  procedure IniDispSal;
10:  var
11:    NomDispSal : string;

12:  begin
13:    write('Dispositivo de Salida=');
14:    readln(NomDispSal);
15:    rewrite(DispSal,NomDispSal);

16:  end;

17:  procedure ENTRADA;
18:  begin
19:    writeln(DispSal,'PERIODO ',I);
20:    writeln(DispSal,'HECHO EN ',I-T,' EDAD ',T);
21:    write(DispSal,'R= ');
22:    readln(R[I,T]);
23:    writeln(DispSal,R[I,T]);
24:    write(DispSal,'U= ');
25:    readln(U[I,T]);
26:    writeln(DispSal,U[I,T]);
27:    write(DispSal,'C= ');
28:    readln(C[I,T]);
29:    writeln(DispSal,C[I,T]);

30:  end;

31:  procedure REPORTA;
32:  begin
33:    writeln(DispSal);
34:    writeln(DispSal,'RESULTADOS:');
35:    writeln(DispSal);
36:    COPT:=F[1,IT];
37:    writeln(DispSal,'GANANCIA OPTIMA ',COPT);
38:    writeln(DispSal);
39:    for I:=1 to N do
40:      begin
41:        write(DispSal,'PERIODO ',I);
42:        if X[I]=1 then
```

```
43:           writeln(DispSal,' POLITICA: REEMPLAZAR')
44:         else
45:           writeln(DispSal,' POLITICA: NO REEMPLAZAR');
46:       end;

47:   end;

48:   procedure CALCOPT;
49:   begin
50:     CO:=R[I,0]-U[I,0]-C[I,T]+A*F[I+1,1];
51:     NCO:=R[I,T]-U[I,T]+A*F[I+1,T+1];
52:     if CO>NCO then
53:       begin
54:         F[I,T]:= CO;
55:         D[I,T]:=1;
56:       end
57:     else
58:       begin
59:         F[I,T]:=NCO;
60:         D[I,T]:=0;
61:       end;

62:   end;

63:   procedure INICIAL;
64:   begin
65:     I:=N+1;
66:     for T:=1 to (I+IT-1) do
67:       F[I,T]:=0;

68:   end;

69:   procedure POLITICA;
70:   begin
71:     T:=IT;
72:     for I:=1 to N do
73:       begin
74:         X[I]:=D[I,T];
75:         if X[I]=1 then T:=0
76:         else T:=T+1;
77:       end;

78:   end;

79:   begin
80:     IniDispSal;
81:     writeln(DispSal,'I00104. REEMPLAZO DE',
82:       ' EQUIPO');
```

```
 83:     writeln(DispSal,
 84:        '# PERIODOS, EDAD INICIAL, FACTOR DESC');
 85:     readln(N,IT,A);
 86:     writeln(DispSal,N,' ',IT,' ',A);
 87:     INICIAL;
 88:     for I:= 1 to N do
 89:       begin
 90:         for T:=0 to (I-1) do
 91:           ENTRADA;
 92:         T:=I+IT-1;
 93:         ENTRADA;
 94:       end;
 95:     for I:=N downto 1 do
 96:       begin
 97:         for T:=1 to (I-1) do
 98:           CALCOPT;
 99:         T:=I+IT-1;
100:         CALCOPT;
101:       end;
102:     POLITICA;
103:     REPORTA;
104:     close(DispSal,lock);

105:   END.
```

```
 1:   I00104. REEMPLAZO DE EQUIPO
 2:   # PERIODOS, EDAD INICIAL, FACTOR DESC
 3:   3 2 1
 4:   PERIODO 1
 5:   HECHO EN 1 EDAD 0
 6:   R= 14
 7:   U= 1
 8:   C= 20
 9:   PERIODO 1
10:   HECHO EN -1 EDAD 2
11:   R= 10
12:   U= 3
13:   C= 25
14:   PERIODO 2
15:   HECHO EN 2 EDAD 0
16:   R= 16
17:   U= 1
18:   C= 20
19:   PERIODO 2
20:   HECHO EN 1 EDAD 1
21:   R= 16
22:   U= 1
23:   C= 22
24:   PERIODO 2
25:   HECHO EN -1 EDAD 3
26:   R= 8
27:   U= 3
28:   C= 26
29:   PERIODO 3
30:   HECHO EN 3 EDAD 0
31:   R= 18
32:   U= 1
33:   C= 20
34:   PERIODO 3
35:   HECHO EN 2 EDAD 1
36:   R= 14
37:   U= 1
38:   C= 22
39:   PERIODO 3
40:   HECHO EN 1 EDAD 2
41:   R= 16
42:   U= 2
43:   C= 24
44:   PERIODO 3
45:   HECHO EN -1 EDAD 4
46:   R= 8
47:   U= 4
48:   C= 27
```

```
49:   RESULTADOS:

50:   GANANCIA OPTIMA 17

51:   PERIODO 1 POLITICA: REEMPLAZAR
52:   PERIODO 2 POLITICA: NO REEMPLAZAR
53:   PERIODO 3 POLITICA: NO REEMPLAZAR
```

4.2.- I002. Programación Lineal.

La Programación Lineal es una de las técnicas más utilizadas en la actualidad ya que una vez expresado el problema en términos del modelo general, es posible aplicar un algoritmo también general que resuelva el problema (a diferencia de la Programación Dinámica). El Método Simplex constituye tal algoritmo y ha hecho posible la solución de un gran número de problemas tales como: balanceo de raciones para animales, contratación y entrenamiento de personal, programación de producción, asignación de recursos escasos a diversas actividades, utilización de la tierra para obtener la mejor cosecha, optimizar la mezcla de una gasolina, etc. Aún cuando sí resuelve un gran número de problemas, no debemos olvidar que no resuelve todos los problemas imaginables. En algunos casos, el problema no se comporta linealmente y en otros, es posible usar un algoritmo más eficiente como es el caso del Método Húngaro para el problema de asignación. Aún en el caso de que se decida usar el Método Simplex, es posible que, dadas las características del problema, sea necesario usar alguna de las variaciones del método tales como el Método Simplex Revisado, el Método de las Dos Fases, el Método Dual-Simplex, etc.

4.2.1.- I00201. Método Simplex.

Desarrollado originalmente por George Dantzig, el Método Simplex es un algoritmo general para la solución de problemas de Programación Lineal.

El programa acepta los siguientes datos:

Sean x[j] las variables de decisión del problema. x[j]>=0, j=1..n
n : número de variables.(entero)
m : número de restricciones.(entero)
tipo : tipo de problema. 0=minimización, 1=maximización.(entero)
swimp : switch de impresión. 0=imprime sólo la solución óptima, 1=imprime cada solución parcial.(entero)
c[j] : coeficiente de x[j] en la función objetivo.(real)
a[i,j] : coeficiente de x[j] en la i-ésima restricción.(real)
b[i] : término independiente de la i-ésima restricción.(real)
t[i] : tipo de la i-ésima restricción. 0='<=', 1='>=', 2='=',(entero)
i=1..m, j=1..n

Y calcula las siguientes cantidades:

zopt : valor óptimo de la función objetivo.(real)
xopt[j] : valor óptimo de x[j].(real)
swnfact : indica si exista solución óptima factible. 0=existe, 1=no existe
j=1..p, p : número de variables incluyendo variables de holgura y artificiales.

Ejemplo: Se desea determinar el número de toneladas que se deben producir de 3 tipos de alimentos. La demanda para cada tipo de alimento es de 100, 100 y 50 toneladas por mes respectivamente. Cada mes se dispone de 1000 horas de mano de obra y de $1500.00 para materia prima. Cada alimento requiere de 1, 2.5 y 3 horas de mano de obra y de $3.00, $2.50 y 2.00 para materia prima. La ganancia por tonelada de cada producto es de $5.00, $7.00, y $6.00 respectivamente.

El problema de Programación Lineal a resolver es:

max z = 5.00x[1] + 7.00x[2] + 6.00x[3]
sujeto a

x[1] + 2.5x[2] + 3x[3] <= 1000

$$
\begin{aligned}
3x[1] + 2.5x[2] + 2x[3] &\le 1500 \\
x[1] &\ge 100 \\
x[2] &\ge 100 \\
x[3] &\ge 50 \\
x[j] \ge 0,\ j=1..3 &
\end{aligned}
$$

donde x[j] : cantidad a producir del alimento j.

Expresando la información apropiadamente para el programa:

```
n = 3
m = 5
tipo = 1
swimp = 0
c = [5 7 6]
a[1] = [1 2.5 3]
a[2] = [3 2.5 2]
a[3] = [1 0 0]
a[4] = [0 1 0]
a[5] = [0 0 1]
b = [1000 1500 100 100 50]
t = [0 0 1 1 1]
```

obtenemos la siguiente solución óptima:

```
zopt = 3285
xopt = [275 230 50 0 0 175 0 130 0 0 0]
```

```
 1:  program IO0201;
 2:  var
 3:    N,M,I,J,IDX,R,K,TIPO,SWIMP,
 4:      SWOPT,SWNAC,SWART,ITER : integer;
 5:    MNEG,RMIN,RAZON,PIVOTE,FACTOR,Z,ZOPT : real;
 6:    T,IDB,IART : array [1..25] of integer;
 7:    C,XOPT : array [1..75] of real;
 8:    B : array [0..25] of real;
 9:    XB : array [1..25] of real;
10:    A : array [1..25,1..75] of real;
11:    DispSal : text;

12:  procedure IniDispSal;
13:  var
14:    NomDispSal : string;

15:  begin
16:    write('Dispositivo de Salida=');
17:    readln(NomDispSal);
18:    rewrite(DispSal,NomDispSal);

19:  end;

20:  procedure SOLOPT;
21:  begin
22:    for I:=1 to M do
23:      begin
24:        J:=IDB[I];
25:        XOPT[J]:=XB[I];
26:      end;
27:    ZOPT:=Z;
28:    writeln(DispSal);
29:    writeln(DispSal,'SOLUCION OPTIMA:');
30:    writeln(DispSal);
31:    writeln(DispSal,'ZOPT=',ZOPT);
32:    writeln(DispSal);
33:    for J:=1 to IDX do
34:      begin
35:        writeln(DispSal,'XOPT(',J,')=',XOPT[J]);
36:      end;

37:  end;

38:  procedure REPORTE;
39:  begin
40:    for I:=1 to M do
41:      begin
42:        if IDB[I] = IART[I] then
```

```
43:            if XB[I] <> 0 then
44:              SWART:=1;
45:        end;
46:      if SWNAC=1 then
47:        writeln(DispSal,
48:                'NO HAY SOLUCION FINITA')
49:      else if SWART=1 then
50:        writeln(DispSal,
51:          'NO HAY SOLUCION OPTIMA FACTIBLE')
52:      else
53:        begin
54:          if SWIMP=1 then
55:            writeln(DispSal,
56:              'SOLUCION OPTIMA EN LA ULTIMA ITERACION');
57:          SOLOPT;
58:        end;

59:  end;

60:  procedure SOLUCION;
61:  begin
62:    ITER:=ITER+1;
63:    (* NUEVA SOL *)
64:    for I:=1 to M do
65:      XB[I]:=B[I];
66:    IDB[K]:=R;
67:    Z:=B[0];
68:    if TIPO=0 then
69:      Z:=-Z;
70:    if SWIMP=1 then
71:      begin
72:        writeln(DispSal,
73:          'EN LA ITERACION '.ITER.' LA SOL ES:');
74:        writeln(DispSal);
75:        writeln(DispSal,'Z='.Z);
76:        writeln(DispSal);
77:        for I:=1 to M do
78:          begin
79:            writeln(DispSal,
80:              'X('.IDB[I].')='.XB[I]);
81:          end;
82:      end;

83:  end;

84:  procedure CAMBIOBASE;
85:  begin
86:    (* CAMBIO DE BASE, PIV A(K,R) *)
```

```
 87:      PIVOTE:=A[K,R];
 88:      for J:=1 to IDX do
 89:        A[K,J]:=A[K,J]/PIVOTE;
 90:      B[K]:=B[K]/PIVOTE;
 91:      A[K,R]:=1;
 92:      for I:=1 to M do
 93:        begin
 94:          FACTOR:=A[I,R];
 95:          if I<>K then
 96:            begin
 97:              B[I]:=B[I]-FACTOR*B[K];
 98:              for J:=1 to IDX do
 99:                begin
100:                  A[I,J]:=A[I,J]-FACTOR*A[K,J];
101:                end;
102:            end;

103:        end;
104:      FACTOR:=C[R];
105:      for J:=1 to IDX do
106:        begin
107:          C[J]:=C[J]-FACTOR*A[K,J];
108:        end;
109:      B[0]:=B[0]-FACTOR*B[K];

110:      (* NUEVA SOL *)
111:      SOLUCION;
112:    end;

113:    procedure RSIMPLEX;
114:    begin
115:      K:=0;
116:      RMIN:=10E10;
117:      for I:=1 to M do
118:        begin
119:          if A[I,R]>10E-10 then
120:            begin
121:              RAZON:=B[I]/A[I,R];
122:              if RAZON < RMIN then
123:                begin
124:                  RMIN:=RAZON;
125:                  K:=I;
126:                end;
127:            end;
128:        end;

129:      if K<> 0 then
130:        CAMBIOBASE
```

```
131:      else
132:        begin
133:          SWOPT:=1;
134:          SWNAC:=1;
135:        end;

136:    end;

137:    procedure ITSIMPLEX;
138:    begin
139:      (* BUSCA MAS NEGATIVO EN FN OBJ *)
140:      R:=0;
141:      MNEG:=0;
142:      for J:=1 to IDX do
143:        begin
144:          if (C[J] < 0) and (C[J] < MNEG) then
145:            begin
146:              MNEG:=C[J];
147:              R:=J;
148:            end;
149:        end;
150:      if R<>0 then
151:        RSIMPLEX
152:      else
153:        SWOPT:=1;
154:    end;

155:    procedure ESTANDARIZA;
156:    begin
157:      (* PRIMERO FN OBJ *)
158:      if TIPO = 1 then
159:        for J:= 1 to N do
160:          begin
161:            C[J]:=-C[J];
162:          end;
163:      (* CADA RESTR A IGUALDAD *)
164:      IDX:=N;
165:      for I:=1 to M do
166:        begin
167:          IDX:=IDX+1;
168:          IART[I]:=0;
169:          if T[I]=0 then
170:            begin
171:              A[I,IDX]:=1;
172:              C[IDX]:=0;
173:            end
174:          else if T[I]=1 then
175:            begin
```

```
176:              A[I,IDX]:=-1;
177:              C[IDX]:=0;
178:              IDX:=IDX+1;
179:              A[I,IDX]:=1;
180:              C[IDX]:=100;
181:              IART[I]:=IDX;
182:              for J:=1 to IDX do
183:                C[J]:=C[J]-100*A[I,J];
184:              B[0]:=B[0]-100*B[I];
185:            end
186:          else
187:            begin
188:              A[I,IDX]:=1;
189:              C[IDX]:=100;
190:              IART[I]:=IDX;
191:              for J:=1 to IDX do
192:                C[J]:=C[J]-100*A[I,J];
193:              B[0]:=B[0]-100*B[I];
194:            end;
195:          XB[I]:=B[I];
196:          IDB[I]:=IDX;
197:        end;

198:    end;

199:    procedure LECTURA;
200:    begin
201:      writeln(DispSal,'IO0201. (SIMPLEX)');
202:      writeln(DispSal,'N.M.TIPO.SWIMP');
203:      readln(N,M,TIPO,SWIMP);
204:      writeln(DispSal,N,' ',M,' ',
205:              TIPO,' ',SWIMP);
206:      writeln(DispSal,'FN OBJETIVO');
207:      for J:=1 to N do
208:        begin
209:          write(DispSal,'C(',J,')=');
210:          readln(C[J]);
211:          writeln(DispSal,C[J]);
212:        end;
213:      for I:=1 to M do
214:        begin
215:          write(DispSal,'B(',I,')=');
216:          readln(B[I]);
217:          writeln(DispSal,B[I]);
218:          write(DispSal,'T(',I,')=');
219:          readln(T[I]);
220:          writeln(DispSal,T[I]);
221:          for J:=1 to N do
```

```
222:            begin
223:              write(DispSal,'A(',I,J,')=');
224:              readln(A[I,J]);
225:              writeln(DispSal,A[I,J]);
226:            end;

227:        end;

228:    end;

229:    procedure INICIALIZA;
230:    begin
231:      ITER:=0;
232:      SWOPT:=0;
233:      SWNAC:=0;
234:      SWART:=0;
235:      IDX:=0;
236:      Z:=0;
237:      B[0]:=0;
238:      for J:=1 to 75 do
239:        begin
240:          C[J]:=0;
241:          XOPT[J]:=0;
242:        end;
243:      for I:=1 to 25 do
244:        begin
245:          B[I]:=0;
246:          T[I]:=0;
247:          IART[I]:=0;
248:          for J:=1 to 75 do
249:            begin
250:              A[I,J]:=0;
251:            end;
252:        end;

253:    end;

254:    begin
255:      IniDispSal;
256:      (* FASE 1 *)
257:      INICIALIZA;
258:      LECTURA;
259:      (* FASE 2 *)
260:      ESTANDARIZA;
261:      (* FASE 3 *)
262:      while SWOPT = 0 do
263:        begin
264:          ITSIMPLEX;
```

```
265:        end;
266:      REPORTE:
267:      close(DispSal,lock);

268:    end.
```

```
 1:   I00201. (SIMPLEX)
 2:   N,M,TIPO,SWIMP
 3:   3 5 1 0
 4:   FN OBJETIVO
 5:   C(1)= 5.00000
 6:   C(2)= 7.00000
 7:   C(3)= 6.00000
 8:   B(1)= 1.00000E3
 9:   T(1)=0
10:   A(11)= 1.00000
11:   A(12)= 2.50000
12:   A(13)= 3.00000
13:   B(2)= 1.50000E3
14:   T(2)=0
15:   A(21)= 3.00000
16:   A(22)= 2.50000
17:   A(23)= 2.00000
18:   B(3)= 1.00000E2
19:   T(3)=1
20:   A(31)= 1.00000
21:   A(32)= 0.00000
22:   A(33)= 0.00000
23:   B(4)= 1.00000E2
24:   T(4)=1
25:   A(41)= 0.00000
26:   A(42)= 1.00000
27:   A(43)= 0.00000
28:   B(5)= 5.00000E1
29:   T(5)=1
30:   A(51)= 0.00000
31:   A(52)= 0.00000
32:   A(53)= 1.00000

33:   SOLUCION OPTIMA:

34:   ZOPT= 3.28500E3

35:   XOPT(1)= 2.75000E2
36:   XOPT(2)= 2.30000E2
37:   XOPT(3)= 5.00000E1
38:   XOPT(4)= 0.00000
39:   XOPT(5)= 0.00000
40:   XOPT(6)= 1.75000E2
41:   XOPT(7)= 0.00000
42:   XOPT(8)= 1.30000E2
43:   XOPT(9)= 0.00000
44:   XOPT(10)= 0.00000
45:   XOPT(11)= 0.00000
```

4.2.2.- I00202. Método Simplex Revisado.

El Método Simplex Revisado es un algoritmo más eficiente que el Método Simplex tradicional. La principal diferencia es que en el primero se aplican métodos matriciales que requieren menos cálculos, aumentando así la velocidad del algoritmo y reduciendo el error por truncamiento o redondeo.

Para usar este programa podemos notar que acepta y produce los mismos datos que el programa I00201, por lo que referimos al lector a la documentación y al ejemplo de éste.

```
 1:  program IO0202;
 2:  var
 3:    N,M,I,IK,J,L,IDC,IDNB,IDX,R,K,TIPO : integer;
 4:    SWIMP,SWOPT,SWNAC,SWART,ITER,ECR : integer;
 5:    MNEG,RMIN,RAZON,Z,ZOPT,TEMP : real;
 6:    T,IDB,IAST : array [1..25] of integer;
 7:    IDN : array [1..50] of integer;
 8:    ZJMCJ : array [1..50] of real;
 9:    C,XOPT : array [1..75] of real;
10:    B,ATEMP,BTEMP : array [0..25] of real;
11:    XB,CBPM1,CSI,ALFAJ : array [1..25] of real;
12:    A : array [1..25,1..75] of real;
13:    BM1,BM1NXT,IDENT : array [1..25,1..25] of real;
14:    DispSal : text;

15:  procedure IniDispSal;
16:  var
17:    NomDispSal : string;

18:  begin
19:    write('Dispositivo de Salida=');
20:    readln(NomDispSal);
21:    rewrite(DispSal,NomDispSal);

22:  end;

23:  procedure SOLOPT;
24:  begin
25:    for I:=1 to M do
26:      begin
27:        J:=IDB[I];
28:        XOPT[J]:=XB[I];
29:      end;
30:    ZOPT:=Z;
31:    writeln(DispSal);
32:    writeln(DispSal,'SOLUCION OPTIMA:');
33:    writeln(DispSal);
34:    writeln(DispSal,'ZOPT=',ZOPT);
35:    writeln(DispSal);
36:    for J:=1 to IDX do
37:      begin
38:        writeln(DispSal,'XOPT(',J,')=',XOPT[J]);
39:      end;

40:  end;

41:  procedure REPORTE;
42:  begin
```

```
43:      writeln(DispSal);
44:      for I:=1 to M do
45:        begin
46:          if IDENTI] = IARTICI] then
47:            if YCCI] <> 0 then
48:              SWASI:=1;
49:        end;
50:      if SWNAC=1 then
51:        writeln(DispSal,'NO HAY SOLUCION FINITA')
52:      else if SWAPT=1 then
53:        writeln(DispSal,
54:          'NO HAY SOLUCION OPTIMA FACTIBLE')
55:      else
56:        begin
57:          if SWIMP=1 then
58:            writeln(DispSal,
59:              'SOLUCION OPTIMA EN LA ULTIMA ITERACION');
60:          SOLOPT;
61:        end;

62:    end;

63:    procedure EMINEXT;
64:    begin
65:      for I:=1 to M do
66:        begin
67:          CSICI]:=-ALFAJCI]/ALFAJECR];
69:        end;
69:      CCIECR]:=1/ALFAJECR];
70:      for I:=1 to N do
71:        begin
72:          IDENTCI,ECR]:=CSICI];
73:        end;
74:      (* NUEVA BASE *)
75:      for I:=1 to M do
76:        for K:=1 to M do
77:          begin
78:            TEMP:=0;
79:            for L:=1 to N do
80:              begin
81:                TEMP:=TEMP+IDENTCI,LJ*SMICL,KJ;
82:              end;
83:            EMINXTCI KJ:=TEMP;
84:          end;
85:      for I:=1 to M do
86:        begin
87:          if I <> ECR then
88:            IDENTCI,ECRJ:=1
```

```
 89:          else
 90:            IDENT[I,ECR]:=0;
 91:        end;

 92:    end;

 93:    procedure SOLUCION;
 94:    begin
 95:      if ITER<>0 then
 96:        begin
 97:          (* NUEVA BASE *)
 98:          if SWIMP=1 then
 99:            begin
100:              writeln(DispSal);
101:              writeln(DispSal,'ENTRA X(',J,')',
102:                      ' Y SALE X(',R,')');
103:            end;
104:          BM1NEXT;
105:        end;
106:      for I:=1 to M do
107:        begin
108:          CBBM1[I]:=0;
109:          for K:=1 to M do
110:            begin
111:              IDC:=IDB[K];
112:              BM1[K,I]:=BM1NXT[K,I];
113:              CBBM1[I]:=CBBM1[I]+C[IDC]*BM1[K,I];
114:            end;
115:        end;
116:      Z:=0;
117:      for I:=1 to M do
118:        begin
119:          Z:=Z+CBBM1[I]*B[I];
120:          XB[I]:=0;
121:          for K:=1 to M do
122:            begin
123:              XB[I]:=XB[I]+BM1[I,K]*B[K];
124:            end;
125:        end;
126:      if TIPO=0 then
127:        Z:=-Z;
128:      if SWIMP=1 then
129:        begin
130:          writeln(DispSal);
131:          writeln(DispSal,
132:            'EN LA ITERACION ',ITER,' LA SOL ES:');
133:          writeln(DispSal);
134:          writeln(DispSal,'Z=',Z);
```

```
          writeln(DispSal);
          for I:=1 to M do
            begin
              writeln(DispSal,
                      'X(',IDB[I],')=',XB[I]);
            end;
        end;
      ITER:=ITER+1;

  end;

  procedure SALEPR;
  begin
    R:=0;
    RMIN:=10E6;
    for K:=1 to M do
      begin
        ALFAJ[K]:=0;
        for I:=1 to M do
          begin
            ALFAJ[K]:=ALFAJ[K]+BM1[K,I]*A[I,J];
          end;
        if ALFAJ[K]>10E-6 then
          begin
            RAZON:=XB[K]/ALFAJ[K];
            if RAZON<RMIN then
              begin
                RMIN:=RAZON;
                R:=IDB[K];
                ECR:=K;
              end;
          end;
      end;
      IDB[ECR]:=J;
      IDN[IR]:=R;
      (* SALE PR DE LA BASE*)

  end;

  procedure ENTRAPJ;
  begin
    ATEMP[0]:=1;
    for J:=1 to M do
      ATEMP[J]:=CBBM1[J];
    for J:=1 to (IDNB) do
      begin
        IR:=IDN[J];
        ZJMCJ[J]:=0;
```

```
179:          BTEMP[0]:=-C[1,J];
180:          ZJMCJ[J]:=ZJMCJ[J]+ATEMP[0]*BTEMP[0];
181:          for I:=1 to M do
182:            begin
183:              BTEMP[I]:=A[1,IDC];
184:              ZJMCJ[J]:=ZJMCJ[J]+ATEMP[I]*BTEMP[I];
185:            end;
186:        end;
187:        (* BUSCA MAS NEG DE ZJMCJ *)
188:        J:=0;
189:        MNEG:=0;
190:        for I:=1 to IDNB do
191:          begin
192:            if ZJMCJ[I]<MNEG then
193:              begin
194:                IR:=I;
195:                J:=IDN[I];
196:                MNEG:=ZJMCJ[I];
197:              end;
198:          end;
199:        (* ENTRA PJ A LA BASE *)

200:    end;

201:    procedure ITSIMPLEX;
202:    begin
203:      ENTRAPJ;
204:      if J<>0 then
205:        begin
206:          SALEPR;
207:          if R=0 then
208:            begin
209:              SWOPT:=1;
210:              SWNAC:=1;
211:            end
212:          else
213:            SOLUCION;
214:        end
215:      else SWOPT:=1;

216:    end;

217:    procedure ESTANDARIZA;
218:    begin
219:      J:=0;
220:      R:=0;
221:      (* COEF NO BAS *)
222:      for I:=1 to N do
```

```
223:        IDN[I]:=I;
224:      IDNB:=N;
225:      (* MATRIZ IDENTIDAD Y 1A BASE *)
226:      for I:=1 to M do
227:        for K:=1 to M do
228:          begin
229:            if I<>K then
230:              begin
231:                IDENT[I,K]:=0;
232:                BM1NXT[I,K]:=0;
233:              end
234:            else
235:              begin
236:                IDENT[I,K]:=1;
237:                BM1NXT[I,K]:=1;
238:              end;
239:          end;
240:      (* PRIMERO FN OBJ *)
241:      if TIPO = 0 then
242:        for J:= 1 to N do
243:          begin
244:            C[J]:=-C[J];
245:          end;
246:      (* CADA RESTR A IGUALDAD *)
247:      IDX:=N;
248:      for I:=1 to M do
249:        begin
250:          IDX:=IDX+1;
251:          IART[I]:=0;
252:          if T[I]=0 then
253:            begin
254:              A[I,IDX]:=1;
255:              C[IDX]:=0;
256:            end
257:          else if T[I]=1 then
258:            begin
259:              A[I,IDX]:=-1;
260:              C[IDX]:=0;
261:              IDNB:=IDNB+1;
262:              IDN[IDNB]:=IDX;
263:              IDX:=IDX+1;
264:              A[I,IDX]:=1;
265:              C[IDX]:=-100;
266:              IART[I]:=IDX;
267:            end
268:          else
269:            begin
270:              A[I,IDX]:=1;
```

```
271:              C[J]:=100;
272:              IND[II]:=IDX;
273:            end;
274:          XB[II]:=B[I];
275:          IBS[II]:=IDX;
276:        end;
277:        SOLUCION;

278:    end;

279:    procedure LECTURA;
280:    begin
281:      writeln(DispSal,'IO0702, (SIMPLEX REVISADO)');
282:      writeln(DispSal,'N,M,TIPO,SWIMP');
283:      readln(N,M,TIPO,SWIMP);
284:      writeln(DispSal,N,' ',M,' ',TIPO,' ',SWIMP);
285:      writeln(DispSal,'FN OBJETIVO');
286:      for J:=1 to N do
287:        begin
288:          write(DispSal,'C(',J,')=');
289:          readln(C[J]);
290:          writeln(DispSal,C[J]);
291:        end;
292:      for I:=1 to M do
293:        begin
294:          write(DispSal,'B(',I,')=');
295:          readln(B[I]);
296:          writeln(DispSal,B[I]);
297:          write(DispSal,'T(',I,')=');
298:          readln(T[I]);
299:          writeln(DispSal,T[I]);

300:          for J:=1 to N do
301:            begin
302:              write(DispSal,'A(',I,J,')=');
303:              readln(A[I,J]);
304:              writeln(DispSal,A[I,J]);
305:            end;

306:        end;

307:    end;

308:    procedure INICIALIZA;
309:    begin
310:      ITER:=0;
311:      SWOPT:=0;
312:      SWNAC:=0;
```

```
313:      SWART:=0;
314:      IDX:=0;
315:      Z:=0;
316:      B[0]:=0;
317:      for J:=1 to 75 do
318:        begin
319:          C[J]:=0;
320:          XOPT[J]:=0;
321:        end;
322:      for I:=1 to 25 do
323:        begin
324:          B[I]:=0;
325:          T[I]:=0;
326:          IART[I]:=0;
327:          for J:=1 to 75 do
328:            begin
329:              A[I,J]:=0;
330:            end;
331:        end;

332:    end;

333:    begin
334:      IniDispSal;
335:      (* FASE 1 *)
336:      INICIALIZA;
337:      LECTURA;
338:      (* FASE 2 *)
339:      ESTANDARIZA;
340:      (* FASE 3 *)
341:      while SWOPT = 0 do
342:        begin
343:          ITSIMPLEX;
344:        end;
345:      REPORTE;
346:      close(DispSal,lock);

347:    end.
```

```
 1:   I00202. (SIMPLEX REVISADO)
 2:   N.M.TIPO.SWIMP
 3:   3 5 1 0
 4:   FN OBJETIVO
 5:   C(1)= 5.00000
 6:   C(2)= 7.00000
 7:   C(3)= 6.00000
 8:   B(1)= 1.00000E3
 9:   T(1)=0
10:   A(11)= 1.00000
11:   A(12)= 2.50000
12:   A(13)= 3.00000
13:   B(2)= 1.50000E3
14:   T(2)=0
15:   A(21)= 3.00000
16:   A(22)= 2.50000
17:   A(23)= 2.00000
18:   B(3)= 1.00000E2
19:   T(3)=1
20:   A(31)= 1.00000
21:   A(32)= 0.00000
22:   A(33)= 0.00000
23:   B(4)= 1.00000E2
24:   T(4)=1
25:   A(41)= 0.00000
26:   A(42)= 1.00000
27:   A(43)= 0.00000
28:   B(5)= 5.00000E1
29:   T(5)=1
30:   A(51)= 0.00000
31:   A(52)= 0.00000
32:   A(53)= 1.00000

33:   SOLUCION OPTIMA:

34:   ZOPT= 3.28500E3

35:   XOPT(1)= 2.75000E2
36:   XOPT(2)= 2.30000E2
37:   XOPT(3)= 5.00000E1
38:   XOPT(4)= 0.00000
39:   XOPT(5)= 0.00000
40:   XOPT(6)= 1.75000E2
41:   XOPT(7)= 0.00000
42:   XOPT(8)= 1.30000E2
43:   XOPT(9)= 0.00000
44:   XOPT(10)= 0.00000
```

45: XOPT(11)= 0.00000

4.2.3.- I00203. Problema de Asignación por Método Húngaro.

Suponga que se desea asignar m personas (o recursos) a m actividades, cada persona puede realizar cualquiera de las m actividades pero posiblemente con diferente efectividad. El problema es asignar cada persona a una sola actividad optimizando la efectividad. El problema de Asignación es un ejemplo de un problema de Programación Lineal para el que existe un algoritmo que aprovecha las características especiales del problema para llegar más rápido a la solución óptima. Tal algoritmo fue desarrollado por D. Konig, matemático húngaro, y recibe el nombre de Método Húngaro.

El problema acepta los siguientes datos:

```
tipo : tipo de problema. 0=minimización,
  1=maximización.(entero)
m : número de recursos.(entero)
n : número de actividades.(entero)
a[i,j] : efectividad al asignar el recurso i
  a la actividad j.(real)
```

Y produce la siguiente solución óptima:

```
zopt : efectividad óptima.(real)
xopt[i] : actividad a asignar el recurso i.(entero)
  i=1..m, j=1..n
```

Ejemplo: Supóngase que se tiene una estimación del tiempo en que 4 obreros realizan 3 trabajos cada uno. El problema es determinar qué obrero asignar a qué trabajo minimizando el tiempo total. la siguiente tabla contiene estas estimaciones:

	trabajo		
obrero	1	2	3
1	14	10	9
2	13	14	16
3	15	10	14
4	15	13	9

Obteniendo la solución óptima:

```
zopt = 32
xopt = [3 1 2 4]
```

Ud. debe notar que la actividad 4 no existe en el problema original y, por lo tanto, esto es equivalente a no asignar actividad alguna al cuarto obrero.

```
 1:  program I00203;
 2:  var
 3:    I,J,K,ZREF,M,N,TIPO,SWASIG,SWNOPT,NASIG : integer;
 4:    ITER,NZEROS,NZERFI,SWCHECK,IREF,JREF  : integer;
 5:    A : array [1..25,1..25] of real;
 6:    XOPT,IASIG,JFLAG,ICHECK,
 7:      JCHECK : array [1..25] of integer;
 8:    IZ,JZ : array [1..25] of real;
 9:    MIN,MAX,Z,ZOPT : real;
10:    DispSal : text;

11:  procedure IniDispSal;
12:  var
13:    NomDispSal : string;

14:  begin
15:    write('Dispositivo de Salida=');
16:    readln(NomDispSal);
17:    rewrite(DispSal,NomDispSal);

18:  end;

19:  procedure RESULTADOS;
20:  begin
21:    writeln(DispSal,'RESULTADOS');
22:    Z:=0;
23:    writeln(DispSal,'SOLUCION OPTIMA EN ',
24:            ITER,' ITERACIONES');
25:    writeln(DispSal);
26:    for I:=1 to M do
27:      begin
28:        XOPT[I]:=IASIG[I];
29:        writeln(DispSal,'ASIGNAR RECURSO ',
30:                I,' A ACTIVIDAD ',IASIG[I]);
31:        ZREF:=IASIG[I];
32:        Z:=Z-IZ[I]-JZ[ZREF];
33:      end;
34:    if TIPO=0 then
35:      ZOPT:=Z
36:    else
37:      ZOPT:=-Z;
38:    writeln(DispSal);
39:    writeln(DispSal,'PARA OPTIMO Z=',ZOPT);

40:  end;

41:  procedure INICIALIZA;
42:  begin
```

```
48:     ITER:=0;
44:     for I:=1 to 25 do
45:       begin
46:         IZ[I]:=0;
47:         JZ[I]:=0;
48:         I[I,I]:=0;
49:         F[A[I]]:=0;
50:         for J:=1 to 25 do
51:           A[I,J]:=0;
52:       end;

53:   end;

54:   procedure LECTURA;
55:   begin
56:     writeln(DispSal,
57:       'IO0203 ASIGNACION. METODO HUNGARO');
58:     writeln(DispSal);
59:     write(DispSal,'TIPO DE PROBLEMA=');
60:     readln(TIPO);
61:     writeln(DispSal,TIPO);
62:     writeln(DispSal);
63:     write(DispSal,'NUMERO DE RECURSOS=');
64:     readln(M);
65:     writeln(DispSal,M);
66:     write(DispSal,'NUMERO DE ACTIVIDADES=');
67:     readln(N);
68:     writeln(DispSal,N);
69:     writeln(DispSal);
70:     writeln(DispSal,'MATRIZ DE EFECTIVIDAD');
71:     for I:=1 to M do
72:       for J:=1 to N do
73:         begin
74:           write(DispSal,'A(',I,J,')=');
75:           readln(A[I,J]);
76:           writeln(DispSal,A[I,J]);
77:           (* CT MAXIMIZACION ENTS A:=-A *)
78:           if TIPO=1 then
79:             A[I,J]:=-A[I,J];
80:         end;
81:     (* CONVIERTE MATRIZ EN CUADRADA *)
82:     if N<M then N:=M;

83:   end;

84:   procedure FASE1;
85:   begin
86:     {--ENCONTRAR MIN ; TENGS CHECK
```

```
 87:        Y [illegible] UE *)
 88:      MIN:=MAXR;
 89:      for I:=1 [illegible] N do
 90:        [illegible]
 91:          [illegible] to M do
 92:            if [illegible]=0 then
 93:              [illegible]A[I,J]<MIN then
 94:                [illegible]N:=A[I,J];
 95:      (* RESTAR EL A MENOS NO CHECK Y SUMAR *)
 96:      (* A [illegible] E JK *)
 97:      for I:=1 to N do
 98:        for J:=1 to M do
 99:          begin
100:            if ICHECK[I]=1 then
101:              A[I,J]:=A[I,J]-MIN;
102:            if JCHECK[J]=1 then
103:              A[I,J]:=A[I,J]+MIN;
104:          end;
105:      (* AFECTAR EN OBJETIVO *)
106:      (* E INIC PARA SIG ITERACION *)
107:      for I:=1 to M do
108:        begin
109:          if ICHLOCK[I]=1 then
110:            IZ[I]:=IZ[I]-MIN;
111:          if JCHECK[I]=1 then
112:            JZ[I]:=JZ[I]+MIN;
113:          IASIG[I]:=0;
114:          JFLAG[I]:=0;
115:        end;

116:    end;

117:    procedure FASEIV;
118:    begin
119:      (* INICIALIZA *)
120:      for I:=1 to M do
121:        begin
122:          ICHECK[I]:=0;
123:          JCHECK[I]:=0;
124:        end;
125:      (* CHECAR RENGLONES NO ASIGNADOS *)
126:      for I:=1 to M do
127:        if IASIG[I]=0 then
128:          [illegible]
129:        [illegible]
130:          [illegible]
131:          [illegible] (O E. MENOS CHECK *)
132:          [illegible]
```

```
133:          if ICHECK[I] = 1 then
134:            for J:=1 to M do
135:              if (A[I,J]=0) and (JCHECK[J]=0) then
136:                begin
137:                  JCHECK[J]:=1;
138:                  SWCHECK:=1;
139:                end;

140:        (* CHECAR RENGS NO CHECK
141:           CON ASIGS EN COLS CHECK *)
142:        for I:=1 to M do
143:          begin
144:            if (IASIG[I]<>0) and (ICHECK[I]=0) then
145:              ZREF:=IASIG[I];
146:              if JCHECK[ZREF]=1 then
147:                begin
148:                  ICHECK[I]:=1;
149:                  SWCHECK:=1;
150:                end;

151:          end;
152:      until SWCHECK = 0;

153:    end;

154:    procedure FASEIII;
155:    begin
156:      (* DETERMINA SI YA ES OPTIMO *)
157:      SWNOPT:=0;
158:      NASIG:=0;
159:      for I:=1 to M do
160:        if IASIG[I]=0 then
161:          SWNOPT:=1
162:        else
163:          NASIG:=NASIG+1;
164:      if SWNOPT=1 then
165:        if (NASIG<>0) and (NZERFI=0) then
166:          begin
167:            FASEIV;
168:            FASEV;
169:          end
170:        else
171:        (* SI NO HAY ASIGNACIONES
172:           SE HACE UNA ARBITRARIA *)
173:          begin
174:            IASIG[IREF]:=JREF;
175:            JFLAG[JREF]:=1;
176:          end;
```

```
177:   end;

178:   procedure FASEII;
179:   begin
180:     ITER:=ITER+1;
       :     (* TODAS LAS ASIGNACIONES POSIBLES *)
182:     repeat
183:       SWASIG:=0;
184:       (* ASIGNACION POR RENGLONES *)
185:       for I:=1 to M do
186:         begin
187:           NZERFI:=0;
188:           if IASIGLI]=0 then
189:             begin
190:               for J:=1 to M do
191:                 if JFLAGLJ]=0 then
192:                   if ALI,J]=0 then
193:                     begin
194:                       IREF:=I;
195:                       JREF:=J;
196:                       NZEROS:=NZEROS+1;
197:                       ZREF:=J;
198:                     end;
199:             end;
200:           if NZEROS=1 then
201:             begin
202:               IASIGLI]:=ZREF;
203:               JFLAGLZREF]:=1;
204:               SWASIG:=1;
205:             end;
206:           if NZERFI=0 then
207:             if NZEROS>1 then
208:               NZERFI:=1;
209:         end;
210:       (* ASIGNACION POR COLUMNAS *)
211:       for J:=1 to M do
212:         begin
213:           NZEROS:=0;
214:           if JFLAGLJ]=0 then
215:             begin
216:               for I:=1 to M do
217:                 if IASIGLI]= 0 then
218:                   if ALI,J]=0 then
219:                     begin
220:                       NZEROS:=NZEROS+1;
221:                       ZREF:=I;
222:
```

```
223:                end;
224:              end;
225:            if NZEROS=1 then
226:              begin
227:                IASIG[IZREF]:=J;
228:                JFLAG[JJ]:=1;
229:                SWASIG:=1;
230:              end;
231:            if NZERFI=0 then
232:              if NZEROS>1 then
233:                NZERFI:=1;
234:          end;
235:        until SWASIG=0;

236:    end;

237:    procedure FASEI;
238:    begin
239:      (*AL MENOS DE 1 0 EN C/REN Y C/COL *)
240:      (* POR RENGLONES *)
241:      MAX:=10E10;
242:      for I:=1 to M do
243:        begin
244:          MIN:=A[I,1];
245:          (*MINIMO*)
246:          for J:=2 to M do
247:            begin
248:              if A[I,J]<MIN then
249:                MIN:=A[I,J];
250:            end;
251:          (* SI NO ES 0 RESTAR A TODO RENG *)
252:          if MIN<>0 then
253:            begin
254:              (* AFECTAR FN OBJETIVO *)
255:              IZ[I]:=IZ[I]-MIN;
256:              for J:=1 to M do
257:                A[I,J]:=A[I,J]-MIN;
258:            end;
259:        end;
260:        (* POR COLUMNAS *)
261:        for J:=1 to M do
262:          begin
263:            MIN:=A[1,J];
264:            (*MINIMO*)
265:            for I:=2 to M do
266:              begin
267:                if A[I,J]<MIN then
268:                  MIN:=A[I,J];
```

```
269:              end;
270:              if MIN<>0 then
271:                begin
272:                  JZEJJ:=JZEJJ-MIN;
273:                  for I:=1 to N do
274:                    AEI,JJ:=AEI,JJ-MIN;
275:                end;
276:          end;

277:    end;

278:    begin
279:      IniDispSal;
280:      INICIALIZA;
281:      LECTURA;
282:      FASEI;
283:      SWNOPT:=1;
284:      while SWNOPT=1 do
285:        begin
286:          FASEII;
287:          FASEIII;
288:        end;
289:      RESULTADOS;
290:      close(DispSal,lock);

291:    end.
```

```
 1:   IO0203 ASIGNACION. METODO HUNGARO

 2:   TIPO DE PROBLEMA=0

 3:   NUMERO DE RECURSOS=4
 4:   NUMERO DE ACTIVIDADES=3

 5:   MATRIZ DE EFECTIVIDAD
 6:   A(11)= 1.40000E1
 7:   A(12)= 1.00000E1
 8:   A(13)= 9.00000
 9:   A(21)= 1.30000E1
10:   A(22)= 1.40000E1
11:   A(23)= 1.60000E1
12:   A(31)= 1.50000E1
13:   A(32)= 1.00000E1
14:   A(33)= 1.40000E1
15:   A(41)= 1.50000E1
16:   A(42)= 1.30000E1
17:   A(43)= 9.00000
18:   RESULTADOS
19:   SOLUCION OPTIMA EN 2 ITERACIONES

20:   ASIGNAR RECURSO 1 A ACTIVIDAD 3
21:   ASIGNAR RECURSO 2 A ACTIVIDAD 1
22:   ASIGNAR RECURSO 3 A ACTIVIDAD 2
23:   ASIGNAR RECURSO 4 A ACTIVIDAD 4

24:   PARA OPTIMO Z= 3.20000E1
```

4.3.- I003. Programación Entera.

Si un problema de Programación Lineal tiene además la restricción de que al menos una variable debe tomar valores enteros no negativos, se dice que es un problema de Programación Entera. De nuevo tenemos un área en donde no existe un algoritmo general para la solución de problemas. En esta sección usamos dos métodos para la Programación Entera: Método de Enumeración Implícita y Método del Plano Cortante de Gomory.

4.3.1.- I00301. Método de Enumeración Implícita.

Supongamos que tenemos un problema de Programación Lineal donde las variables de decisión pueden tomar sólo los valores 0 o 1. El algoritmo de Enumeración Implícita busca la solución óptima al problema probando combinaciones de 1's y 0's en las variables de decisión siempre tratando de mejorar la solución.

El programa acepta los siguientes datos:

n : número de variables.(entero)
m : número de restricciones.(entero)
tipo : tipo de problema. 0=minimización, 1=maximización.(entero)
c[j] : coefeciente de x[j] en la función objetivo.(real)
a[i,j] : coeficiente de x[j] en la i-ésima restricción.(real)
b[i] : término independiente de la i-ésima restricción.(real)
ti[i] : tipo de la i-ésima restricción. 0='<=', 1='>=', 2='='.(entero)

Y produce las siguientes cantidades:

zopt : valor óptimo de la función objetivo.(real)
xopt[j] : valor de x[j].(entero) i=1..m, j=1..n

Ejemplo: Considere el siguiente problema de Programación Lineal:

min: z = 6x[1] + 4x[2] + 2x[3]
sujeto a:

4x[1] - 9x[2] + 5x[3] <= 10
6x[1] + 2x[2] + 4x[3] >= 2
3x[1] + 2x[2] + x[3] >= 1
x[j] = 0,1, j=1..3

Usando la representación adecuada para estos datos:

n = 3
m = 3
tipo = 0
c = [6 4 2]
a[1] = [4 -9 5]
a[2] = [6 2 4]
a[3] = [3 2 1]
b = [10 2 1]
ti = [0 1 1]

Obtenemos la siguiente solución:

```
zopt = 2
xopt = [0 0 1]
```

```
 1:  program IOe3u1;
 2:  var
 3:    M,I,J,K,L,M,N,ITER,TIPO,VENTRA : integer;
 4:    POINTNF,POINTC : integer;
 5:    Q,ZAJ,Z,ZMIN,DMIN,D,DTOT,DTOTF,BOUND : real;
 6:    TI,CAND,RV,X,Y,MERCE,FINE
 7:      : array [1..25] of integer;
 8:    C,B : array [1..25] of real;
 9:    A : array [1..25,1..25] of real;
10:    SWOPT,SWNFACT,SWRV,SWT,SWTV,
11:      SWEQ,SWBKT : integer;
12:    DispSal : text;

13:  procedure IniDispSal;
14:  var
15:    NomDispSal : string;

16:  begin
17:    write('Dispositivo de Salida=');
18:    readln(NomDispSal);
19:    rewrite(DispSal,NomDispSal);

20:  end;

21:  procedure RESULTADOS;
22:  begin
23:    writeln(DispSal);
24:    writeln(DispSal,'RESULTADOS');
25:    if ZMIN=10E10 then
26:      SWNFACT:=1;
27:    ZMIN:=ZMIN+ZAJ;
28:    if SWNFACT=0 then
29:      begin
30:        if TIPO=1 then
31:          ZMIN:=-ZMIN;
32:        writeln(DispSal,'OPTIMO=',ZMIN,
33:          ' EN ',ITER,' ITERACIONES');
34:        writeln(DispSal);
35:        for J:=1 to N do
36:          begin
37:            if Y[J]=1 then
38:              X[J]:=1-X[J];
39:            writeln(DispSal,'X(',J,')=',X[J]);
40:          end;
41:      end
42:    else
43:      begin
44:        writeln(DispSal,'DESPUES DE ',
```

```
45:          ITER.' ITERACIONES.');
46:        writeln(DispSal,
47:          'SE DETERMINO QUE NO HAY SOLUCION FACTIBLE');
48:      end;

49:  end;

50:  procedure MINDIST;
51:  begin
52:    (* SELECCIONAMOS DE CAND LA var CON DIST MIN *)
53:    (* HACIA LA FACTIBILIDAD Y ENTRA A NFREE *)
54:    (* CON VALOR DE 1 *)
55:    DMIN:=10E10;
56:    VENTRA:=0;
57:    DTOTF:=0;
58:    for J:=1 to POINTC do
59:      begin
60:        DTOT:=0;
61:        H:=CAND[J];
62:        for I:=1 to M do
63:          begin
64:            D:=B[I]+A[I,H];
65:            for K:=1 to POINTNF do
66:              begin
67:                if NFREE[K]>0 then
68:                  begin
69:                    L:=NFREE[K];
70:                    D:=D+A[I,L];
71:                  end;
72:              end;
73:            if D<0 then
74:              DTOT:=DTOT+D;
75:          end;
76:          DTOTF:=DTOTF+DTOT;
77:        if ABS(DTOT)<DMIN then
78:          begin
79:            DMIN:=ABS(DTOT);
80:            VENTRA:=H;
81:          end;
82:      end;
83:    if DTOTF<0 then
84:      begin
85:        POINTNF:=POINTNF+1;
86:        NFREE[POINTNF]:=VENTRA;
87:        FREE[VENTRA]:=0;
88:      end
89:    else
90:      (* ENTRA LA QUE TENGA MENOR COEF EN FN OBJ *)
```

```
 91:        begin
 92:          DMIN:=10E10;
 93:          for J:=1 to POINTC do
 94:            begin
 95:              H:=CAND[J];
 96:              if C[H]<DMIN then
 97:                begin
 98:                  DMIN:=C[H];
 99:                  VENTRA:=H;
100:                end;
101:            end;
102:            POINTNF:=POINTNF+1;
103:            NFREE[POINTNF]:=VENTRA;
104:            FREE[VENTRA]:=0;
105:        end;

106:    end;

107:    procedure CHRESTR2;
108:    begin
109:      (* VERIFICAMOS FACTIBILIDAD USANDO NFREE *)
110:      (* Y CAND *);
111:      SWRV:=0;
112:      for I:=1 to M do
113:        begin
114:          if RV[I]=1 then
115:            begin
116:              Q:=B[I];
117:              for J:=1 to POINTC do
118:                begin
119:                  L:=CAND[J];
120:                  if A[I,L]>0 then
121:                    Q:=Q+A[I,L];
122:                end;
123:              for J:=1 to POINTNF do
124:                begin
125:                  if NFREE[J]>0 then
126:                    begin
127:                      L:=NFREE[J];
128:                      Q:=Q+A[I,L];
129:                    end;
130:                end;
131:              if Q<0 then
132:                begin
133:                  RV[I]:=1;
134:                  SWRV:=1;
135:                end;
136:            end;
```

```
137:        end;

138:    end;

139:    procedure CALCCAND;
140:    begin
141:      BOUND:=ZMIN-Z;
142:      POINTC:=0;
143:      SWTV:=0;
144:      for J:=1 to N do
145:        begin
146:          if (C[J]<BOUND) and (FREE[J]=1) then
147:            begin
148:              I:=0;
149:              SWT:=0;
150:              repeat
151:                I:=I+1;
152:                if (A[I,J]>0.) and (RV[I]=1) then
153:                  begin
154:                    POINTC:=POINTC+1;
155:                    L:=POINTC;
156:                    CAND[L]:=J;
157:                    SWT:=1;
158:                  end
159:              until (SWT=1) or (I=M);
160:            end;
161:        end;
162:        if POINTC=0 then
163:          SWTV:=1;

164:    end;

165:    procedure BACKTRACK;
166:    begin
167:      (* CAMBIAMOS DE NODO USANDO LA ULTIMA var *)
168:      (* POSITIVA EN NFREE (I.E.=1). MULTIPLICANDO *)
169:      (* POR -1 (I.E. AHORA=0) Y LIBERANDO TODAS *)
170:      (* LAS VARS A SU DERECHA *)

171:      if POINTNF>0 then
172:        begin
173:          J:=POINTNF+1;
174:          SWBKT:=0;
175:          repeat
176:            J:=J-1;
177:            if NFREE[J]>0 then
178:              begin
179:                SWBKT:=1;
```

```
180:                NFREE[J]:=-NFREE[J];
181:                if J<POINTNF then
182:                  for K:=J+1 to POINTNF do
183:                    begin
184:                      L:=ABS(NFREE[K]);
185:                      FREE[L]:=1;
186:                      NFREE[K]:=0;
187:                    end;
188:                POINTNF:=J;
189:              end;
190:          until (SWBKT=1) or (J=1);
191:          if SWBKT=0 then
192:            (* SI NO HAY POSITIVOS => OPTIMO *)
193:            SWOPT:=1;
194:        end
195:      else
196:        (* NO PUDIMOS ASIGNAR VALORES EN 1ER NODO *)
197:        (* => NO HAY SOL FACTIBLE *)
198:        begin
199:          SWOPT:=1;
200:        end;

201:    end;

202:    procedure NUEVASOL;
203:    begin
204:      if Z<ZMIN then
205:        begin
206:          ZMIN:=Z;
207:          for J:=1 to N do
208:            X[J]:=0;
209:          for J:=1 to POINTNF do
210:            begin
211:              if NFREE[J]>0 then
212:                begin
213:                  L:=NFREE[J];
214:                  X[L]:=1;
215:                end;
216:            end;
217:        end;

218:    end;

219:    procedure CHRESTR1;
220:    begin
221:      (* VERIFICAMOS FACTIBILIDAD USANDO NFREE *)
222:      SWRV:=0;
223:      for I:=1 to M do
```

```
224:        begin
225:          Q:=B[I];
226:          RV[I]:=0;
227:          for J:=1 to POINTNF do
228:            begin
229:              if NFREE[J]>0 then
230:                begin
231:                  L:=NFREE[J];
232:                  Q:=Q+A[I,L];
233:                end;
234:            end;
235:          if Q<0 then
236:            begin
237:              RV[I]:=1;
238:              SWRV:=1;
239:            end;
240:        end;

241:    end;

242:    procedure CALCZ;
243:    begin
244:      (* CALCULAMOS LA FN OBJETIVO USANDO LAS *)
245:      (* VARIABLES QUE YA TIENEN VALOR (NFREE) *)
246:      (* Y LAS DEMAS EN CERO *)
247:      Z:=0;
248:      for J:=1 to POINTNF do
249:        begin
250:          if NFREE[J]>0 then
251:            begin
252:              L:=NFREE[J];
253:              Z:=Z+C[L];
254:            end;
255:        end;

256:    end;

257:    procedure ITERACIONES;
258:    begin
259:      ITER:=ITER+1;
260:      (* RUTINA PRINCIPAL *)

261:      (* CALC FN OBJETIVO *)
262:      CALCZ;

263:      (* VERIFICA FACTIBILIDAD *)
264:      CHRESTR1;
265:      if SWRV=0 then
```

```
266:        begin
267:          (* TENEMOS UNA SOL FACTIBLE *)
268:          NUEVASOL;
269:          (* VEMOS SI SE PUEDE MEJORAR *)
270:          BACKTRACK;
271:        end
272:      else
273:        begin
274:          (* TRATAMOS DE COMPLETAR UNA SOL FACTIBLE *)
275:          (* LO VEMOS QUE VARS PUEDEN ENTRAR A LA SOL *)
276:          CALCCAND;
277:          if SWTV=0 then
278:            begin
279:              (* HAY CANDIDATOS A ENTRAR *)
280:              (* VEAMOS SI PUEDE HABER FACTIBILIDAD *)
281:              CHKESTR2;
282:              if SWRV=0 then
283:                begin
284:                  (* ENTRA A LA SOL LA var CON *)
285:                  (* DISTANCIA MIN HACIA FACTIBILIDAD *)
286:                  MINDIST;
287:                end
288:              else
289:                (* DE ESTE NODO NO SALE
290:                   NINGUNA SOL FACT *)
291:                BACKTRACK;
292:            end
293:          else
294:            (* NO HAY CANDIDATOS QUE MEJOREN LA SOL *)
295:            (*  EN ESTE NODO *)
296:            begin
297:              BACKTRACK;
298:            end;
299:        end;

300:  end;

301:  procedure INICIALIZA;
302:  begin
303:    ITER:=0;
304:    POINTNF:=0;
305:    Z:=0;
306:    ZOLD:=0;
307:    ZPTN:=1.0E10;
308:    SWOPT:=0;
309:    SWNFACT:=0;
310:    SWSU:=0;
311:    CUT:=0;
```

```
312:      SWTV:=0;
313:      SWEQ:=0;
314:      SWBKT:=0;
315:      B[M+1]:=0;
316:      for J:=1 to N do
317:        begin
318:          X[J]:=0;
319:          Y[J]:=0;
320:          NFREE[J]:=0;
321:          FREE[J]:=1;
322:          A[M+1,J]:=0;
323:          RV[J]:=0;
324:        end;
325:    end;

326:    procedure LECTURA;
327:    begin
328:      writeln(DispSal);
329:      writeln(DispSal,
330:        'ID0301. ENUMERACION IMPLICITA');
331:      writeln(DispSal);
332:      writeln(DispSal,'NUMERO DE VARIABLES');
333:      readln(N);
334:      writeln(DispSal,N);
335:      writeln(DispSal,'NUMERO DE RESTRICCIONES');
336:      readln(M);
337:      writeln(DispSal,M);
338:      writeln(DispSal,'TIPO, 0=MIN,1=MAX');
339:      readln(TIPO);
340:      writeln(DispSal,TIPO);
341:      writeln(DispSal);
342:      writeln(DispSal,'FN OBJETIVO');
343:      for J:=1 to N do
344:        begin
345:          write(DispSal,'C(',J,')=');
346:          readln(C[J]);
347:          writeln(DispSal,C[J]);
348:        end;
349:      writeln(DispSal);
350:      writeln(DispSal,'RESTRICCIONES');
351:      for I:=1 to M do
352:        begin
353:          write(DispSal,'B(',I,')=');
354:          readln(B[I]);
355:          writeln(DispSal,B[I]);
356:          write(DispSal,'TI(',I,')=');
357:          readln(TI[I]);
```

```
358:          writeln(DispSal,T[I]);
359:          for J:=1 to N do
360:            begin
361:              write(DispSal,'A(',I,J,')=');
362:              readln(A[I,J]);
363:              writeln(DispSal,A[I,J]);
364:            end;
365:        end;


366:    end;

367:    procedure PREPARACION;
368:    begin
369:      (* DEJAMOS POSITIVOS COEFICIENTES DE FN OBJ *)
370:      (* Y RESTRICCIONES EN LA FORMA: *)
371:      (* B[I]:=-B[I]+SUM A[I,J]*X[J] *)

372:      if TIPO=1 then
373:        for J:=1 to N do
374:          C[J]:=-C[J];
375:      for J:=1 to N do
376:        begin
377:          if C[J]<0 then
378:            begin
379:              ZAJ:=ZAJ+C[J];
380:              C[J]:=-C[J];
381:              Y[J]:=1;
382:              for I:=1 to M do
383:                begin
384:                  B[I]:=B[I]-A[I,J];
385:                  A[I,J]:=-A[I,J];
386:                end;
387:            end;
388:        end;

389:      for I:=1 to M do
390:        begin
391:          if TI[I]=0 then
392:            begin
393:              for J:=1 to N do
394:                A[I,J]:=-A[I,J];
395:            end
396:          else if TI[I]=1 then
397:            begin
398:              B[I]:=B[I];
399:            end
400:          else
```

```
401:            begin
402:              SWEQ:=1;
403:              B[M+1]:=B[M+1]+B[I];
404:              B[I]:=-B[I];
405:              for J:=1 to N do
406:                A[M+1,J]:=A[M+1,J]-A[I,J]
407:            end;
408:        end;

409:      if SWEQ=1 then
410:        begin
411:          M:=M+1;
412:          TI[M]:=1;
413:        end;

414:    end;

415:    begin
416:      IniDispSal;
417:      LECTURA;
418:      INICIALIZA;
419:      PREPARACION;
420:      repeat
421:        ITERACIONES;
422:      until SWOPT=1;
423:      RESULTADOS;
424:      close(DispSal,lock);

425:    end.
```

```
 1:   IO0301. ENUMERACION IMPLICITA

 2:   NUMERO DE VARIABLES
 3:   3
 4:   NUMERO DE RESTRICCIONES
 5:   3
 6:   TIPO, 0=MIN,1=MAX
 7:   0

 8:   FN OBJETIVO
 9:   C(1)= 6.00000
10:   C(2)= 4.00000
11:   C(3)= 2.00000

12:   RESTRICCIONES
13:   B(1)= 1.00000E1
14:   TI(1)=0
15:   A(11)= 4.00000
16:   A(12)=-9.00000
17:   A(13)= 5.00000
18:   B(2)= 2.00000
19:   TI(2)=1
20:   A(21)= 6.00000
21:   A(22)= 2.00000
22:   A(23)= 4.00000
23:   B(3)= 1.00000
24:   TI(3)=1
25:   A(31)= 3.00000
26:   A(32)= 2.00000
27:   A(33)= 1.00000

28:   RESULTADOS
29:   OPTIMO= 2.00000 EN 3 ITERACIONES

30:   X(1)=0
31:   X(2)=0
32:   X(3)=1
```

4.3.2.- I00302. Método del Plano Cortante de Gomory.

La base de este algoritmo es usar el Método Simplex normal sin la restricción de variables enteras. Si la solución no es entera, se introduce una nueva restricción y se resuelve el nuevo problema igual que antes. Se continua así hasta que la solución cumple con la restricción de variables enteras.

El programa acepta los siguientes datos:

n : número de variables.(entero)
m : número de restricciones.(entero)
tipo : tipo de problema. 0=minimización, 1=maximización.(entero)
swimp : 0=imprime sólo la solución óptima, 1=imprime cada solución parcial.(entero).
maxcort : número máximo de cortes.(entero)
c[j] : coefeciente de x[j] en la función objetivo.(real)
a[i,j] : coeficiente de x[j] en la i-ésima restricción.(real)
b[i] : término independiente de la i-ésima restricción.(real)
ti[i] : tipo de la i-ésima restricción. 0='<=', 1='>=', 2='='.(entero)
i=1..m, j=1..n

Y produce las siguientes cantidades:

zopt : valor óptimo de la función objetivo.(real)
xopt[j] : valor de x[j].(entero)
j=1..p, p : número de variables incluyendo las creadas en cada corte

Ejemplo: Suponga el siguiente problema de Programación Lineal:

min: z = x[1] + 2x[2] +3x[3] + 4x[4]
sujeto a:

$$4x[1] + 3x[2] + 2x[3] + x[4] \geq 10$$
$$2x[2] + 4x[4] \geq 5$$
$$x[j] \geq 0,\ x[j] \text{ entero}$$

Usando la representación apropiada para el problema:

```
n = 4
m = 2
tipo = 0
swimp = 0
maxcort = 50
```

```
c = [1 2 3 4]
a[1] = [4 3 2 1]
a[2] = [0 2 0 4]
b = [10 5]
t = [1 1]
```

Obtenemos la siguiente solución óptima:

```
zopt = 7
xopt = [1 3 0 0 3 0 1 0 2 0 1 0 1 0 0 0 0 0]
```

```
 1:  program IO0302;
 2:  var
 3:    MAXCUT,CUT,ICUT,N,M,I,J,IDX,K,K,II : integer;
 4:    ZOPT,SWPIVOT,SWIMP,SWOPT,SWNAC,SWAR : integer;
 5:    MNEG,RMIN,RAZON,PIVOTE,FACTOR,Z : real;
 6:    T,IDB,IART : array [1..25] of integer;
 7:    C : array [1..75] of real;
 8:    XOPT: array [1..75] of integer;
 9:    B : array [0..25] of real;
10:    XB : array [1..25] of real;
11:    A : array [1..25,1..75] of real;
12:    DispSal : text;

13:  procedure IniDispSal;
14:  var
15:    NomDispSal : string;

16:  begin
17:    write('Dispositivo de Salida=');
18:    readln(NomDispSal);
19:    rewrite(DispSal,NomDispSal);

20:  end;

21:  procedure TABLA;
22:  begin
23:    writeln(DispSal,'TABLA');
24:    writeln(DispSal,'B(0)=',B[0]);
25:    for J:=1 to IDX do
26:      begin
27:        writeln(DispSal,'J=',J);
28:        writeln(DispSal,'C=',C[J]);
29:      end;
30:    for I:=1 to M do
31:      begin
32:        writeln(DispSal,'I=',I);
33:        writeln(DispSal,'XB=',XB[I]);
34:        writeln(DispSal,'B=',B[I]);
35:        writeln(DispSal,'T=',T[I]);
36:        writeln(DispSal,'IART=',IART[I]);
37:        for J:=1 to IDX do
38:          begin
39:            writeln(DispSal,'J=',J);
40:            writeln(DispSal,'A=',A[I,J]);
41:          end;
42:      end;

43:  end;
```

```
44:   procedure SOLOPT;
45:   begin
46:     (* EN CASO DE [illegible] SOLUCION
47:        ASIGNAMOS VALORES OPTIMOS *)
48:     for I:=1 to [illegible] do
49:       begin
50:         J:=IDX[I];
51:         XOPT[J]:=ROUND(XB[I]);
52:       end;
53:     ZOPT:=ROUND(Z*
54:     writeln(Disp[illegible]);
55:     writeln(Disp,' SOLUCION OPTIMA:');
56:     writeln(Disp[illegible]);
57:     writeln(DispSal,'ZOPT=',ZOPT);
58:     writeln(DispSal);
59:     for J:=1 to IDX do
60:       begin
61:         writeln(DispSal,'XOPT(',J,')=',XOPT[J]);
62:       end;

63:   end;

64:   (* RUTINAS DE CORTE *)
65:   procedure CORTE;
66:   begin
67:     (* CONSTRUIMOS UNA RESTRICCION >= *)
68:     M:=M+1;
69:     T[M]:=1;
70:     for J:=1 to IDX do
71:       begin
72:         A[M,J]:=A[ICUT,J]-TRUNC(A[ICUT,J]);
73:         if A[M,J]<0 then
74:           A[M,J]:=A[M,J]+1;
75:       end;
76:     B[M]:=B[ICUT]-TRUNC(B[ICUT]);
77:     if B[M]<0 then
78:       B[M]:=B[M]+1;
79:     (* ESTANDARIZAMOS *)
80:     IDX:=IDX+1;
81:     A[M,IDX]:=-1;
82:     C[IDX]:=0;
83:     IDX:=IDX+1;
84:     A[M,IDX]:=1;
85:     IARTL[M]:=IDX;
86:     C[IDX]:=100;
87:     XC[M]:=B[M];
88:     ID[M]:=IDX;
```

```
 89:     (* MULTIPLICAMOS POR 100
 90:        Y RESTAMOS A FN OBJ *)
 91:     for J:=1 to IDX do
 92:       begin
 93:         C[J]:=C[J]-100*A[M,J];
 94:       end;
 95:     B[0]:=B[0]-100*B[M];

 96:   end;

 97:   procedure CHECASOL;
 98:   begin
 99:     I:=0;
100:     SWNINT:=0;
101:     repeat
102:       I:=I+1;
103:       if IDB[I]<>IART[I] then
104:         begin
105:           if XB[I]>(TRUNC(XB[I]+1E-3)+1E-1) then
106:             begin
107:               SWNINT:=1;
108:               ICUT:=I;
109:             end;
110:         end;
111:     until (SWNINT=1) or (I>M);

112:   end;

113:   (* FIN RUTINAS CORTE *)

114:   procedure REPORTE;
115:   begin
116:     for I:=1 to M do
117:       begin
118:         if IDB[I] = IART[I] then
119:           if XB[I] <> 0 then
120:             SWART:=1;
121:       end;
122:     if SWNAC=1 then
123:       writeln(DispSal,
124:         'NO HAY SOLUCION FINITA')
125:     else if SWART=1 then
126:       writeln(DispSal,
127:         'NO HAY SOLUCION OPTIMA FACTIBLE')
128:     else
129:       if SWIMP=1 then
130:         writeln(DispSal,
131:          'SOLUCION OPTIMA EN LA ULTIMA ITERACION');
```

```
132:    end;

133:    procedure SOLUCION;
134:    begin
135:      ITER:=ITER+1;
136:      (* NUEVA SOL *)
137:      for I:=1 to M do
138:        XB[I]:=B[I];
139:      IDB[K]:=R;
140:      Z:=B[0];
141:      if TIPO=0 then
142:        Z:=-Z;
143:      if SWIMP=1 then
144:        begin
145:          writeln(DispSal,'EN LA ITERACION ',ITER,
146:                  ' LA SOL ES:');
147:          writeln(DispSal);
148:          writeln(DispSal,'Z=',Z);
149:          writeln(DispSal);
150:          for I:=1 to M do
151:            begin
152:              writeln(DispSal,'X(',IDB[I],')=',XB[I]);
153:            end;
154:        end;

155:    end;

156:    procedure CAMBIOBASE;
157:    begin
158:      (* CAMBIO DE BASE, PIV A(K,R) *)
159:      PIVOTE:=A[K,R];
160:      for J:=1 to IDX do
161:        A[K,J]:=A[K,J]/PIVOTE;
162:      B[K]:=B[K]/PIVOTE;
163:      A[K,R]:=1;
164:      for I:=1 to M do
165:        begin
166:          FACTOR:=A[I,R];
167:          if I<>K then
168:            begin
169:              B[I]:=B[I]-FACTOR*B[K];
170:              for J:=1 to IDX do
171:                begin
172:                  A[I,J]:=A[I,J]-FACTOR*A[K,J];
173:                end;
174:            end;
175:        end;
176:      FACTOR:=C[R];
```

```
177:      for J:=1 to IDX do
178:        begin
179:          C[J]:=C[J]-FACTOR*A[K,J];
180:        end;
181:      B[0]:=B[0]-FACTOR*B[K];
182:      (* NUEVA SOL *)
183:      SOLUCION;

184:    end;

185:    procedure RSIMPLEX;
186:    begin
187:      K:=0;
188:      RMIN:=10E10;
189:      for I:=1 to M do
190:        begin
191:          if A[I,R]>10E-10 then
192:            begin
193:              RAZON:=B[I]/A[I,R];
194:              if RAZON < RMIN then
195:                begin
196:                  RMIN:=RAZON;
197:                  K:=I;
198:                end;
199:            end;
200:        end;
201:      if K<> 0 then
202:        CAMBIOBASE
203:      else
204:        begin
205:          SWOPT:=1;
206:          SWNAC:=1;
207:        end;

208:    end;

209:    procedure ITSIMPLEX;
210:    begin
211:      (* BUSCA MAS NEGATIVO EN FN OBJ *)
212:      R:=0;
213:      MNEG:=0;
214:      for J:=1 to IDX do
215:        begin
216:          if (C[J] < 0) and (C[J] < MNEG) then
217:            begin
218:              MNEG:=C[J];
219:              R:=J;
220:            end;
```

```
221:        end;
222:      if R<>0 then
223:        RSIMPLEX
224:      else
225:        SWOPT:=1;
226:    end;

227:    procedure ESTANDARIZA;
228:    begin
229:      (* PRIMERO FN OBJ *)
230:      if TIPO = 1 then
231:        for J:= 1 to N do
232:          begin
233:            C[J]:=-C[J];
234:          end;
235:      (* CADA RESTR A IGUALDAD *)
236:      IDX:=N;
237:      for I:=1 to M do
238:        begin
239:          IDX:=IDX+1;
240:          IART[I]:=0;
241:          if T[I]=0 then
242:            begin
243:              A[I,IDX]:=1;
244:              C[IDX]:=0;
245:            end
246:          else if T[I]=1 then
247:            begin
248:              A[I,IDX]:=-1;
249:              C[IDX]:=0;
250:              IDX:=IDX+1;
251:              A[I,IDX]:=1;
252:              C[IDX]:=100;
253:              IART[I]:=IDX;
254:              for J:=1 to IDX do
255:                C[J]:=C[J]-100*A[I,J];
256:              B[0]:=B[0]-100*B[I];
257:            end
258:          else
259:            begin
260:              A[I,IDX]:=1;
261:              C[IDX]:=100;
262:              IART[I]:=IDX;
263:              for J:=1 to IDX do
264:                C[J]:=C[J]-100*A[I,J];
265:              B[0]:=B[0]-100*B[I];
266:            end;
267:          XB[I]:=B[I];
```

```
268:          IDB[I]:=IDX;
269:        end;

270:    end;

271:    procedure LECTURA;
272:    begin
273:      writeln(DispSal,
274:        'I00302. PLANO CORTANTE DE GOMORY');
275:      writeln(DispSal,'N,M,TIPO,SWIMP');
276:      readln(N,M,TIPO,SWIMP);
277:      writeln(DispSal,N,' ',M,' ',TIPO,' ',SWIMP);
278:      writeln(DispSal,'MAXIMO NUMERO DE CORTES');
279:      readln(MAXCUT);
280:      writeln(DispSal,MAXCUT);
281:      writeln(DispSal,'AHORA, FN OBJETIVO');
282:      for J:=1 to N do
283:        begin
284:          write(DispSal,'C(',J,')=');
285:          readln(C[J]);
286:          writeln(DispSal,C[J]);
287:        end;
288:      for I:=1 to M do
289:        for J:=1 to 3*N do
290:          A[I,J]:=0;
291:      for I:=1 to M do
292:        begin
293:          write(DispSal,'B(',I,')=');
294:          readln(B[I]);
295:          writeln(DispSal,B[I]);
296:          write(DispSal,'T(',I,')=');
297:          readln(T[I]);
298:          writeln(DispSal,T[I]);
299:          for J:=1 to N do
300:            begin
301:              write(DispSal,'A(',I,J,')=');
302:              readln(A[I,J]);
303:              writeln(DispSal,A[I,J]);
304:            end;
305:        end;

306:    end;

307:    procedure INICIALIZA;
308:    begin
309:      ITER:=0;
310:      CUT:=0;
311:      ICUT:=0;
```

```
312:      IDX:=0;
313:      Z:=0;
314:      ZOPT:=0;
315:      SWOPT:=0;
316:      SWNAC:=0;
317:      SWNIN:=0;
318:      SWART:=0;
319:      for J:=1 to 75 do
320:        begin
321:          C[J]:=0;
322:          XOPTL[J]:=0;
323:        end;
324:      for I:=1 to 25 do
325:        begin
326:          XB[I]:=0;
327:          B[I]:=0;
328:          T[I]:=0;
329:          IART[I]:=0;
330:          for J:=1 to 75 do
331:            begin
332:              A[I,J]:=0;
333:            end;
334:        end;
335:      B[0]:=0;

336:    end;

337:    begin
338:      IniDispSal;
339:      (* FASE 1 *)
340:      INICIALIZA;
341:      LECTURA;
342:      (* FASE 2 *)
343:      ESTANDARIZA;
344:      (* FASE 3 *)
345:      CUT:=0;
346:      repeat
347:        while SWOPT = 0 do
348:        begin
349:          ITSIMPLEX;
350:        end;
351:        REPORTE;
352:        if (SWNAC<>1) and (SWART<>1) then
353:          (* HAY SOLUCION OPTIMA FINITA *)
354:          begin
355:            (* VERIFICAMOS SON ENTEROS *)
356:            CHECASOL;
357:            if SWNIN=1 then
```

```
358:              begin
359:                (* SI LA SOLUCION NO ES ENTERA *)
360:                (* HACEMOS UN CORTE *)
361:                CUT:=CUT+1;
362:                writeln(DisrSal,1);
363:                writeln(DisrSal,'CORTE # ',CUT);
364:                CORTE;
365:                SWOPT:=0;
366:                SWART:=0;
367:                FUNAC:=0;
368:              end
369:            else
370:              begin
371:                writeln(DisrSal,
372:                  'SOLUCION ENTERA EN ULTIMA ITERACION');
373:                SOLOPT;
374:              end;
375:          end;
376:      until (SWNINT=2) or (CUT=MAXCUT)
377:            or (SWNSC=1) or (SWFST=1);
378:      close(DisrSal,lock);

379:    end.
```

```
 1:   I00302. PLANO CORTANTE DE GOMORY
 2:   N.M.TIPO.SWIMP
 3:   4 2 0 0
 4:   MAXIMO NUMERO DE CORTES
 5:   50
 6:   AHORA. FN OBJETIVO
 7:   C(1)= 1.00000
 8:   C(2)= 2.00000
 9:   C(3)= 3.00000
10:   C(4)= 4.00000
11:   B(1)= 1.00000E1
12:   T(1)=1
13:   A(11)= 4.00000
14:   A(12)= 3.00000
15:   A(13)= 2.00000
16:   A(14)= 1.00000
17:   B(2)= 5.00000
18:   T(2)=1
19:   A(21)= 0.00000
20:   A(22)= 2.00000
21:   A(23)= 0.00000
22:   A(24)= 4.00000

23:   CORTE # 1

24:   CORTE # 2

25:   CORTE # 3

26:   CORTE # 4

27:   CORTE # 5
28:   SOLUCION ENTERA EN ULTIMA ITERACION

29:   SOLUCION OPTIMA:

30:   ZOPT=7

31:   XOPT(1)=1
32:   XOPT(2)=3
33:   XOPT(3)=0
34:   XOPT(4)=0
35:   XOPT(5)=3
36:   XOPT(6)=0
37:   XOPT(7)=1
38:   XOPT(8)=2
39:   XOPT(9)=2
40:   XOPT(10)=0
```

```
41:   XOPT(11)=1
42:   XOPT(12)=0
43:   XOPT(13)=1
44:   XOPT(14)=0
45:   XOPT(15)=0
46:   XOPT(16)=0
47:   XOPT(17)=0
48:   XOPT(10)=0
```

4.4.- I004. Técnica Branch-and-Bound.

La técnica Branch-and-Bound consiste en una clase de algoritmos con ciertas características comunes:

Una búsqueda sistemática y bien estructurada sobre el espacio de soluciones factibles de un problema de optimización con un número finito de soluciones factibles. Normalmente se particiona el espacio en subconjuntos más y más chicos (branching) y para cada uno de ellos se calcula una cota (inferior si se trata de minimizar y superior si se trata de maximizar). Después de cada partición se compara la cota con el valor de la última solución factible encontrada y, en el caso de que la cota indique que en ese subespacio es posible encontrar una mejor solución, se particiona éste hasta encontrar tal solución o abandonar la búsqueda aquí. Si la solución actual es mejor que la cota de un subespacio, éste es automáticamente rechazado eliminando así la prueba de muchas posibles soluciones. La búsqueda termina cuando ninguna de las cotas de los subespacios promete una mejor solución que la actual y ésta es, por lo tanto, la solución óptmima del problema.

Presentamos dos algoritmos basados en esta técnica:

1. Algoritmo de Eastman para el problema del vendedor y
2. Algoritmo de Kolesar para el problema de la óptima carga

4.4.1.- I00401. Problema del Vendedor por el Algortimo de Eastman.

Un vendedor quiere visitar cada una de n-1 ciudades una y sólo una vez y regresar a la ciudad desde la cual comenzó. El problema es determinar el orden en que debe visitar las ciudades de tal manera de minimizar la distancia total recorrida.

El problema acepta los siguientes datos:

```
m : numero de nodos (ciudades).(entero)
d[i,j] : distancia desde nodo i a nodo j.(real)
  i=1..m, j=1..m
```

y produce las siguientes cantidades:

```
zopt : distancia mínima.(real)
xopt[i] : ruta óptima.(entero)
  i=1..m+1
```

Ejemplo: Con la siguiente matriz de distancias se desea resolver el problema del vendedor:

	A CIUDAD			
DESDE CIUDAD	1	2	3	4
1	10000	8	2	1
2	4	10000	5	6
3	7	8	10000	7
4	5	4	5	10000

Note que la distancia entre una ciudad y ella misma es un número muy alto, ésto es para evitar la posibilidad de que se quede en una sola ciudad.

Dejando la información anterior en la forma apropiada para el problema:

```
m = 4
d[1] = [10000 8 2 1]
d[2] = [4 10000 5 6]
d[3] = [7 8 10000 7]
d[4] = [5 4 5 10000]
```

Obtenemos la siguiente solución:

```
zopt = 17
xopt = [1 3 4 2 1]
```

```
 1:  program IO0401;
 2:  var
 3:    I,J,K,ZREF,M,N,TIPO,SWASIG,SWNOPT,NASIG : integer;
 4:    ITER,NZEROS,NZERFI,SWCHECK,IREF,JREF : integer;
 5:    A,D : array [1..15,1..15] of real;
 6:    XOPT,IASIG,JFLAG,ICHECK,JCHECK
 7:      : array [1..15] of integer;
 8:    XOPTC,STFLAG : array [1..16] of integer;
 9:    IZ,JZ : array [1..15] of real;
10:    CLUB,CLUBAUX,MIN,MAX,Z,ZOPT,ZOPTC : real;
11:    H,NODO,CNODO,SWOPTBB,L,SWTERM,SWNFAC : integer;
12:    NODOINI,POINTSIG,ARCS,MINARCS,PRINST : integer;
13:    POINTBRANCH,SWNODO,TERM,POINTST
14:      : packed array [0..301] of integer;
15:    BRANCH,ST : packed array [0..301,1..16] of integer;
16:    BOUND : packed array [0..301] of real;
17:    DispSal : text;

18:  procedure IniDispSal;
19:  var
20:    NomDispSal : string;

21:  begin
22:    write('Dispositivo de Salida=');
23:    readln(NomDispSal);
24:    rewrite(DispSal,NomDispSal);

25:  end;

26:  (* SOLUCION DE SUBPROBLEMAS DE
27:     ASIGNACION, M. HUNGARO *)
28:  procedure RESULTADOS;
29:  begin
30:    writeln(DispSal);
31:    writeln(DispSal,
32:      'RESULTADOS DE SUBPROBLEMA EN NODO ',NODO);
33:    Z:=0;
34:    writeln(DispSal,'SOLUCION OPTIMA EN ',
35:            ITER,',ITERACIONES');
36:    writeln(DispSal);
37:    for I:=1 to M do
38:      begin
39:        XOPT[I]:=IASIG[I];
40:        writeln(DispSal,'ASIGNAR RECURSO ',I,
41:                ' A ACTIVIDAD ',IASIG[I]);
42:        ZREF:=IASIG[I];
43:        Z:=Z-IZ[I]-JZ[ZREF];
44:      end;
```

```
45:     ZOPT:=Z;
46:     writeln(DispSal);
47:     writeln(DispSal,'PARA OPTIMO Z=',Z);

48:   end;

49:   procedure INICIALIZA;
50:   begin
51:     ITER:=0;
52:     for I:=1 to 15 do
53:       begin
54:         IZ[I]:=0;
55:         JZ[I]:=0;
56:         IASIG[I]:=0;
57:         JFLAG[I]:=0;
58:       end;

59:   end;

60:   procedure FASE1;
61:   begin
62:     (* ENCONTRAR MIN EN RENGS CHECK
63:        Y COLS NO CHECK *)
64:     MIN:=MAX;
65:     for I:=1 to M do
66:       if ICHECK[I]=1 then
67:         for J:=1 to M do
68:           if JCHECK[J]=0 then
69:             if A[I,J]<MIN then
70:               MIN:=A[I,J];
71:     (* RESTAR MIN A RENGS NO CHECK Y SUMAR *)
72:     (* A COLS CHECK *)
73:     for I:=1 to M do
74:       for J:=1 to M do
75:         begin
76:           if ICHECK[I]=1 then
77:             A[I,J]:=A[I,J]-MIN;
78:           if JCHECK[J]=1 then
79:             A[I,J]:=A[I,J]+MIN;
80:         end;
81:     (* AFECTAR FN OBJETIVO *)
82:     (* E INIC PARA SIG ITERACION *)
83:     for I:=1 to M do
84:       begin
85:         if ICHECK[I]=1 then
86:           IZ[I]:=IZ[I]-MIN;
87:         if JCHECK[I]=1 then
88:           JZ[I]:=JZ[I]+MIN;
```

```
 89:          IASIG[I]:=0;
 90:          JFLAG[I]:=0;
 91:        end;

 92:    end;

 93:    procedure FASEIV;
 94:    begin
 95:      (* INICIALIZA *)
 96:      for I:=1 to M do
 97:        begin
 98:          ICHECK[I]:=0;
 99:          JCHECK[I]:=0;
100:        end;
101:      (* CHECAR RENGLONES NO ASIGNADOS *)
102:      for I:=1 to M do
103:        if IASIG[I]=0 then
104:          ICHECK[I]:=1;
105:      repeat
106:        SWCHECK:=0;
107:        (* CHECAR COLS CON 0S EN RENGS CHECK *)
108:        for I:=1 to M do
109:          if ICHECK[I] = 1 then
110:            for J:=1 to M do
111:              if (A[I,J]=0) and (JCHECK[J]=0) then
112:                begin
113:                  JCHECK[J]:=1;
114:                  SWCHECK:=1;
115:                end;

116:        (* CHECAR RENGS NO CHECK CON
117:           ASIGS EN COLS CHECK *)
118:        for I:=1 to M do
119:          begin
120:            if (IASIG[I]<>0) and (ICHECK[I]=0) then
121:              ZREF:=IASIG[I];
122:              if JCHECK[ZREF]=1 then
123:                begin
124:                  ICHECK[I]:=1;
125:                  SWCHECK:=1;
126:                end;

127:          end;
128:      until SWCHECK = 0;

129:    end;

130:    procedure FASEIII;
```

```
131:   begin
132:     (* DETERMINA SI YA ES OPTIMO *)
133:     SWNOPT:=0;
134:     NASIG:=0;
135:     for I:=1 to M do
136:       if IASIG[I]=0 then
137:         SWNOPT:=1
138:       else
139:         NASIG:=NASIG+1;
140:     if SWNOPT=1 then
141:       if (NASIG<>0) and (NZERFI=0) then
142:         begin
143:           FASEIV;
144:           FASEV;
145:         end
146:       else
147:       (* SI NO HAY ASIGNACIONES SE
148:          HACE UNA ARBITRARIA *)
149:         begin
150:           IASIG[IREF]:=JREF;
151:           JFLAG[JREF]:=1;
152:         end;

153:   end;

154:   procedure FASEII;
155:   begin
156:     ITER:=ITER+1;
157:     (* TODAS LAS ASIGNACIONES POSIBLES *)
158:     repeat
159:       SWASIG:=0;
160:       NZERFI:=0;
161:       (* ASIGNACION POR RENGLONES *)
162:       for I:=1 to M do
163:         begin
164:           NZEROS:=0;
165:           if IASIG[I]=0 then
166:             begin
167:               for J:=1 to M do
168:                 if JFLAG[J]=0 then
169:                   if A[I,J]=0 then
170:                     begin
171:                       IREF:=I;
172:                       JREF:=J;
173:                       NZEROS:=NZEROS+1;
174:                       ZREF:=J;
175:                     end;
176:             end;
```

```
177:            if NZEROS=1 then
178:              begin
179:                IASIG[I]:=ZREF;
180:                JFLAG[ZREF]:=1;
181:                SWASIG:=1;
182:              end;
183:            if NZERFI=0 then
184:              if NZEROS>1 then
185:                NZERFI:=1;
186:          end;
187:        (* ASIGNACION POR COLUMNAS *)
188:        for J:=1 to N do
189:          begin
190:            NZEROS:=0;
191:            if JFLAG[J]=0 then
192:              begin
193:                for I:=1 to N do
194:                  if IASIG[I]= 0 then
195:                    if A[I,J]=0 then
196:                      begin
197:                        NZEROS:=NZEROS+1;
198:                        ZREF:=I;
199:                      end;
200:              end;
201:            if NZEROS=1 then
202:              begin
203:                IASIG[ZREF]:=J;
204:                JFLAG[J]:=1;
205:                SWASIG:=1;
206:              end;
207:            if NZERFI=0 then
208:              if NZEROS>1 then
209:                NZERFI:=1;
210:          end;
211:        until SWASIG=0;

212:    end;

213:    procedure FASEI;
214:    begin
215:      (*AL MENOS DE 1 0 EN C/REN Y C/COL *)
216:      (* POR RENGLONES *)
217:      MAX:=10E10;
218:      for I:=1 to M do
219:        begin
220:          MIN:=A[I,1];
221:          (*MINIMO*)
222:          for J:=2 to M do
```

```
223:            begin
224:              if A[I,J]<MIN then
225:                MIN:=A[I,J];
226:            end;
227:          (* SI NO ES 0 RESTAR A TODO RENG *)
228:          if MIN<>0 then
229:            begin
230:              (* AFECTAR FN OBJETIVO *)
231:              IZ[I]:=IZ[I]-MIN;
232:              for J:=1 to M do
233:                A[I,J]:=A[I,J]-MIN;
234:            end;
235:        end;
236:        (* FOR COLUMNAS *)
237:        for J:=1 to M do
238:          begin
239:            MIN:=A[1,J];
240:            (*MINIMO*)
241:            for I:=2 to M do
242:              begin
243:                if A[I,J]<MIN then
244:                  MIN:=A[I,J];
245:              end;
246:              if MIN<>0 then
247:                begin
248:                  JZ[J]:=JZ[J]-MIN;
249:                  for I:=1 to M do
250:                    A[I,J]:=A[I,J]-MIN;
251:                end;
252:          end;

253:    end;

254:    procedure HUNGARO;
255:    begin
256:      INICIALIZA;
257:      FASEI;
258:      SWNOPT:=1;
259:      while SWNOPT=1 do
260:        begin
261:          FASEII;
262:          FASEIII;
263:        end;
264:      RESULTADOS;

265:    end;
266:    (* FIN SOLUCION PROBLEMA ASIGNACION, M, HUNGARO *)
```

```
267:  procedure SOLFINAL;
268:  begin
269:    (* LA SOLUCION OPTIMA ES LA
270:       ULTIMA SOLUCION FACTIBLE
271:       SI NO SE HA ENCONTRADO, ENTS SE DEJA
272:       LA ULTIMA FACTIBLE *)
273:    ZOPT:=ZUB;
274:    write(DispSal,'Z=');
275:    writeln(DispSal);
276:    writeln(DispSal,Z);
277:    write(DispSal,Z);
278:    writeln(DispSal,'DESPUES DE ',NODO,' NODOS.');
279:    write(DispSal,' TENEMOS SOLUCION ');
280:    if SWOPTIM=1 then
281:      write(DispSal,'OPTIMA ')
282:    else
283:      write(DispSal,'FACTIBLE ');
284:    write(DispSal,'Z=',ZOPT);
285:    writeln(DispSal);
286:    writeln(DispSal,'RUTA');
287:    writeln(DispSal);
288:    for I:=1 to (M+1) do
289:      begin
290:        XOPTCII:=XOPTCII;
291:        write(DispSal,XOPTCII,'=>');
292:      end;

293:  end;

294:  procedure CHECATERMINALES;
295:  begin
296:    (* BUSCA ALGUN NODO QUE
297:       PUDIERA MEJORAR LA SOLUCION *)
298:    SWTERM:=0;
299:    CLUBAUX:=CLUB;
300:    for I:=1 to NODO do
301:      begin
302:        if TERMCII=1 then
303:          if BOUNDCII<CLUBAUX then
304:            begin
305:              CLUBAUX:=BOUNDCII;
306:              SWTERM:=1;
307:              CNODO:=I;
308:            end;
309:      end;

310:  end;
```

```
      ocedure SUBTOUR;
    begin
      (* ASIGNA LA SUBRUTA MAS CORTA AL NODO *)
  1:  POINTSIG:=PRINST;
315:  ST[NODO,1]:=POINTSIG;
316:  H:=1;
317:  repeat
318:    H:=H+1;
319:    POINTSIG:=XOPT[POINTSIG];
320:    ST[NODO,H]:=POINTSIG;
321:  until (POINTSIG=PRINST);
322:  SWNODO[NODO]:=SWNFAC;

323:  end;

324:  procedure NUEVASOL;
325:  begin
326:    (* SI LA SOLUCION DE ALGUN NODO ES FACTIBLE *)
327:    (* SE ASIGNA COMO NUEVA SOLUCION *)
328:    writeln(DispSal);
329:    write(DispSal,chr(7));
330:    writeln(DispSal,
331:      'ENCONTRAMOS UNA SOLUCION FACTIBLE!!!');
332:    writeln(DispSal,' MINIMO =',ZOPT);
333:    writeln(DispSal,'RUTA:');
334:    writeln(DispSal);
335:    for I:=1 to M+1 do
336:      begin
337:        write(DispSal,ST[NODO,I],'=>');
338:      end;
339:    writeln(DispSal);
340:    if ZOPT<CLUB then
341:      begin
342:        writeln(DispSal);
343:        writeln(DispSal,
344:          'Y AHORA ESTA ES LA NUEVA SOLUCION');
345:        CLUB:=ZOPT;
346:        ZOPTC:=ZOPT;
347:        for I:=1 to (M+1) do
348:          begin
349:            XOPTC[I]:=ST[NODO,I];
350:          end;
351:      end
352:    else
353:      if ZOPT=CLUB then
354:        begin
355:          writeln(DispSal);
```

```
311:   procedure SUBTOUR;
312:   begin
313:     (* ASIGNA LA SUBRUTA MAS CORTA AL NODO *)
314:     POINTSIG:=PRINST;
315:     ST[NODO,1]:=POINTSIG;
316:     H:=1;
317:     repeat
318:       H:=H+1;
319:       POINTSIG:=XOPTC[POINTSIG];
320:       ST[NODO,H]:=POINTSIG;
321:     until (POINTSIG=PRINST);
322:     SWNODO[NODO]:=SWNFAC;

323:   end;

324:   procedure NUEVASOL;
325:   begin
326:     (* SI LA SOLUCION DE ALGUN NODO ES FACTIBLE *)
327:     (* SE ASIGNA COMO NUEVA SOLUCION *)
328:     writeln(DispSal);
329:     write(DispSal,chr(7));
330:     writeln(DispSal,
331:       'ENCONTRAMOS UNA SOLUCION FACTIBLE!!!');
332:     writeln(DispSal,' MINIMO =',ZOPT);
333:     writeln(DispSal,'RUTA:');
334:     writeln(DispSal);
335:     for I:=1 to M+1 do
336:       begin
337:         write(DispSal,ST[NODO,I],'=>');
338:       end;
339:     writeln(DispSal);
340:     if ZOPT<CLUB then
341:       begin
342:         writeln(DispSal);
343:         writeln(DispSal,
344:           'Y AHORA ESTA ES LA NUEVA SOLUCION');
345:         CLUB:=ZOPT;
346:         ZOPTC:=ZOPT;
347:         for I:=1 to (M+1) do
348:           begin
349:             XOPTC[I]:=ST[NODO,I];
350:           end;
351:       end
352:     else
353:       if ZOPT=CLUB then
354:         begin
355:           writeln(DispSal);
```

```
354:            write(DispSal,chr(7));
355:            writeln(DispSal,
356:              'VALOR DE ESTA SOLUCION IGUAL AL');
357:            writeln(DispSal,
358:              'MEJOR ENCONTRADO HASTA AHORA');
359:          end;

360:    end;

361:    procedure CHEQUEOSOL;
362:    begin
363:      (* VERIFICA SI TENEMOS UNA SOLUCION FACTIBLE *)
364:      (*  Y SI NO, ENCUENTRA LA MENOR SUBRUTA *)
365:      FIN:=FALSE;
366:      MINARCS:=10000;
367:      PRINST:=1;
368:      for I:=1 to M do
369:        STFLAG[I]:=0;
370:      for I:=1 to M do
371:        begin
372:          if STFLAG[I]=0 then
373:            begin
374:              NODOINI:=I;
375:              H:=2;
376:              ARCS:=2;
377:              POINTSIG:=XOPT[NODOINI];
378:              STFLAG[POINTSIG]:=1;

379:              repeat
380:                H:=H+1;
381:                ARCS:=ARCS+1;
382:                POINTSIG:=XOPT[POINTSIG];
383:                STFLAG[POINTSIG]:=1;
384:              until (POINTSIG=NODOINI);

385:              if H<(M+1) then
386:                begin
387:                  SWNFAC:=1;
388:                  if ARCS<MINARCS then
389:                    begin
390:                      MINARCS:=ARCS;
391:                      PRINST:=NODOINI;
392:                      POINTST[NODO]:=H;
393:                    end
394:                end
395:              else
396:                begin
397:                  POINTST[NODO]:=H;
```

```
400:                    SUBTOUR;
401:                    PUNTSOL;
402:                  end;
403:              end;
404:          end;
405:          if SWNFAC=1 then
406:            begin
407:              SUBTOUR;
408:              writeln(DispSal);
409:              writeln(DispSal,'LA SOLUCION NO ES FACTIBLE');
410:              writeln(DispSal,'SUBRUTA');
411:              for I:=1 to POINTSTENODO] do
412:                begin
413:                  write(DispSal,STENODO,I],'=>');
414:                end;
415:              writeln(DispSal);
416:            end;

417:    end;

418:    procedure BRANCHBOUND;
419:    begin
420:      (* GENERAMOS NUEVOS NODOS Y RESOLVEMOS *)
421:      (* EL SUBPROBLEMA DE CADA NODO *)
422:      NODO:=NODO+1;
423:      writeln(DispSal);
424:      writeln(DispSal);
425:      writeln(DispSal,
426:        'NODO=',NODO,' SALE DE NODO ',CNODO);
427:      writeln(DispSal,
428:        ' DONDE LA COTA INFERIOR ES ',BOUND[CNODO]);
429:      writeln(DispSal);
430:      TERM[NODO]:=1;
431:      (* GUARDAMOS LA INFORMACION DE CADA NODO *)
432:      for I:=1 to POINTBRANCH[CNODO] do
433:        begin
434:          BRANCH[NODO,I]:=BRANCH[CNODO,I];
435:        end;
436:      I:=POINTBRANCH[CNODO];
437:      I:=I+1;
438:      BRANCH[NODO,I]:=K;
439:      I:=I+1;
440:      BRANCH[NODO,I]:=L;
441:      POINTBRANCH[NODO]:=I;
442:      (* *)
443:      for I:=1 to M do
444:        for J:=1 to M do
445:          begin
```

```
                [illegible]: L[illegible].J];
              end;
        (* PARA CADA NODO ASIGNADO LA FLECHA K-L *)
        for I:=1 to POINTERANCH.NODO] do
          begin
            K:=BRANCHNODO.I];
            L:=[illegible].I];
            [illegible].I]:=100;
            writeln(DispSal,
              'FIJAMOS A(',K,L,')=',10000);
          end;
        (* *)
        RUNGRO];
        BOUNDSNODO]:=ZOPT;
        CHECASOL;

      end;

      procedure ITERACIONES;
      begin
        (* SOLUCION DEL PROBLEMA POR BRANCH-and-BOUND *)
        repeat
          TERMINONODO]:=0;
          for N:=2 to POINTSTECNODO] do
            begin
              K:=STECNODO,N-1];
              L:=STECNODO,N];
              BRANCHBOUND;
            end;
          CHECATERMINALES;
          if SWTERM=0 then
            SWOPTBB:=1;

        until (SWOPTBB=1) or (NODO>300);

      end;

      procedure LECTURA;
      begin
        writeln(DispSal,
          'I00401, RUTA MINIMA (BRANCH-and-BOUND');
        writeln(DispSal);
        write(DispSal,'NUMERO DE NODOS=');
        readln(M);
        writeln(DispSal,M);
        writeln(DispSal);
        writeln(DispSal,'MATRIZ DE DISTANCIAS ');
```

```
489:      for I:=1 to M do
490:        for J:=1 to M do
491:          begin
492:            write(DispSal,'D(',I,J,')=');
493:            readln(D[I,J]);
494:            writeln(DispSal,D[I,J]);
495:          end;

496:    end;

497:    procedure BBINICIALIZA;
498:    begin
499:      POINTSTL[0]:=2;
500:      STL[0,1]:=1;
501:      STL[0,2]:=1;
502:      CLUB:=10000;
503:      for I:=1 to 16 do
504:        begin
505:          XOPTC[I]:=0;
506:          STFLAG[I]:=0;
507:        end;
508:      ZOPTC:=0;
509:      CNODO:=0;
510:      SWOPTBB:=0;
511:      NODO:=0;
512:      POINTBRANCH[CNODO]:=0;
513:      BOUND[CNODO]:=0;

514:    end;

515:    begin
516:      IniDispSal;
517:      BBINICIALIZA;
518:      LECTURA;
519:      ITERACIONES;
520:      SOLFINAL;
521:      close(DispSal.lock);

522:    end.
```

```
 1:  IO0401. RUTA MINIMA (BRANCH-and-BOUND

 2:  NUMERO DE NODOS=4

 3:  MATRIZ DE DISTANCIAS
 4:  D(11)= 1.00000E4
 5:  D(12)= 8.00000
 6:  D(13)= 2.00000
 7:  D(14)= 1.00000
 8:  D(21)= 4.00000
 9:  D(22)= 1.00000E4
10:  D(23)= 5.00000
11:  D(24)= 6.00000
12:  D(31)= 7.00000
13:  D(32)= 8.00000
14:  D(33)= 1.00000E4
15:  D(34)= 7.00000
16:  D(41)= 5.00000
17:  D(42)= 4.00000
18:  D(43)= 5.00000
19:  D(44)= 1.00000E4


20:  NODO=1 SALE DE NODO 0
21:   DONDE LA COTA INFERIOR ES  0.00000

22:  FIJAMOS A(11)=10000

23:  RESULTADOS DE SUBPROBLEMA EN NODO 1
24:  SOLUCION OPTIMA EN 2 ITERACIONES

25:  ASIGNAR RECURSO 1 A ACTIVIDAD 3
26:  ASIGNAR RECURSO 2 A ACTIVIDAD 1
27:  ASIGNAR RECURSO 3 A ACTIVIDAD 4
28:  ASIGNAR RECURSO 4 A ACTIVIDAD 2

29:  PARA OPTIMO Z= 1.70000E1

30:  ENCONTRAMOS UNA SOLUCION FACTIBLE!!!
31:   MINIMO = 1.70000E1
32:  RUTA:

33:  1=>3=>4=>2=>1=>

34:  Y AHORA ESTA ES LA NUEVA SOLUCION


35:  DESPUES DE 1 NODOS.
```

```
36:     TENEMOS SOLUCION OPTIMA Z= 1.70000E1
37:    RUTA

38:    1=>3=>4=>2=>1=>
```

4.4.2.- I00402. Problema de la Optima Carga por Algoritmo de Kolesar.

Se desea determinar la óptima carga consistente en incluir uno de cada uno de ciertos artículos de manera de maximizar el valor de la carga al tiempo que se satisface una restricción de peso máximo para la misma.

El programa acepta los siguientes datos:

```
n : número de artículos.(entero)
wl : peso límite.(real)
w[i] : peso del artículo i.(real)
v[i] : valor del artículo i.(real)
  i=1..n
```

Y produce las siguientes cantidades:

```
z : valor óptimo de la carga.(real)
wt : peso total de la carga.(entero)
x[i] : decisión de incluir el artículo i.
  1=incluir, 0=excluir.(entero)
  i=1..n
```

Ejemplo: Suponga que se desea determinar la carga óptima a partir de los siguientes datos:

ARTICULO #	PESO	VALOR
1	8	8
2	10	12
3	4	2
4	2	2
5	4	1
6	8	4
7	6	12

Estructurando los datos adecuadamente para el programa:

```
n = 7
wl = 20
w = [8 10 4 2 4 8 6]
v = [8 12 2 2 1 4 12]
```

obtenemos la siguiente solución óptima:

```
zopt = 26
wt = 18
x = [0 1 0 1 0 0 1]
```

```
 1:  program IO6402;
 2:  type
 3:    VectReal = array [1..50] of real;
 4:    VectInt = array [1..50] of integer;

 5:  var
 6:    N : integer;
 7:    V,W,X : VectReal;
 8:    WL,Z : real;
 9:    Ord : VectInt;
10:    DispSal : text;

11:  procedure IniDispSal;
12:  var
13:    NomDispSal : string;

14:  begin
15:    write('Dispositivo de Salida=');
16:    readln(NomDispSal);
17:    rewrite(DispSal,NomDispSal);

18:  end;

19:  procedure DespVect(N : integer;
20:                     X : VectReal;
21:                     Ord : VectInt);
22:  var
23:    I : integer;

24:  begin
25:    (* desplegar vector conservando orden
26:       original Ord *)
27:    for I:=1 to N do
28:      begin
29:        writeln(DispSal,'X(',Ord[I],')=',X[i]);
30:      end;

31:  end;

32:  procedure Sort(N : integer;
33:                 var V,W : VectReal;
34:                 var Ord : VectInt);
35:  var
36:    I,ITemp : integer;
37:    Temp : real;
38:    C : boolean;

39:  begin
```

```
40:     For I:=1 to N do
41:       begin
42:         Ord[I]:=I;
43:       end;
44:     repeat
45:       F1:=false;
46:       for I:=2 to N do
47:         begin
48:           if V[I]/W[I]>V[I-1]/W[I-1] then
49:             begin
50:               ITemp:=Ord[I];
51:               Ord[I]:=Ord[I-1];
52:               Ord[I-1]:=Itemp;
53:               Temp:=V[I];
54:               V[I]:=V[I-1];
55:               V[I-1]:=Temp;
56:               Temp:=W[I];
57:               W[I]:=W[I-1];
58:               W[I-1]:=Temp;
59:               F1:=true;
60:             end;
61:         end;
62:       N:=N-1;

63:     until not F1;

64:   end;

65:   function ProdPunto(N : integer;
66:                      X,Y : VectReal) : real;
67:   var
68:     I : integer;
69:     Sum : real;

70:   begin
71:     Sum:=0;
72:     for I:=1 to N do
73:       begin
74:         Sum:=Sum+X[i]*Y[i];
75:       end;
76:     ProdPunto:=Sum;

77:   end;

78:   procedure LeeVect(N : integer;
79:                     var X : VectReal);
80:   var
81:     I : integer;
```

```
 82:   begin
 83:     for I:=1 to N do
 84:       begin
 85:         write(DispSal,'x(',I,')=');
 86:         readln(X[I]);
 87:         writeln(DispSal,X[I]);
 88:       end;

 89:   end;

 90:   procedure Resultados(N : integer;
 91:                        V,W : VectReal;
 92:                        W1,Z : real;
 93:                        X : VectReal;
 94:                        Ord : VectInt);
 95:   begin
 96:     writeln(DispSal);
 97:     writeln(DispSal,'resultados:');
 98:     writeln(DispSal);
 99:     writeln(DispSal,'solucion optima factible:');
100:     writeln(DispSal);
101:     writeln(DispSal,'valor carga=',Z);
102:     writeln(DispSal,'peso carga=',ProdPunto(N,X,W));
103:     writeln(DispSal,'solucion:');
104:     DespVect(N,X,Ord);

105:  end;

106:  procedure OptimaCarga(N : integer;
107:                        V,W : VectReal;
108:                        W1 : real;
109:                        var Z : real;
110:                        var X : VectReal);
111:    function CotaMax(N,Art : integer;
112:                     W1 : real;
113:                     V,W : VectReal;
114:                     C : VectReal) : real;
115:    var
116:      Sum,Wa : real;

117:    begin
118:      Sum:=ProdPunto(Art,C,V);
119:      (* de los articulos no asignados incluir
120:         los de mayor valor incremental primero
121:         hasta llegar al peso limite.
122:         esto lo garantiza el Sort *)
123:      Wa:=W1-ProdPunto(Art,C,W);
```

```
124:      if Wa>=0 then
125:        begin
126:          repeat
127:            Art:=Art+1;
128:            if W[Art]<=Wa then
129:              begin
130:                Wa:=Wa-W[Art];
131:                Sum:=Sum+V[Art];
132:              end
133:            else
134:              begin
135:                Sum:=Sum+(V[Art]/W[Art])*Wa;
136:                Wa:=0;
137:              end;

138:          until (Art=N) or (Wa=0);
139:          CotaMax:=Sum;
140:        end
141:      else (* ninguna solucion factible en subarbol *)
142:        CotaMax:=-10E10;

143:    end;

144:    procedure GeneraCombinacion(N,Art : integer;
145:                                V,W : VectReal;
146:                                Wl : real;
147:                                C : VectReal;
148:                                var Z : real;
149:                                var X : VectReal);
150:    var
151:      Cota1,Cota2 : real;

152:    begin
153:      if Art<N then
154:        begin
155:          (* genera dos ramas del arbol *)
156:          C[Art]:=1; (* incluye *)
157:          Cota1:=CotaMax(N,Art,Wl,V,W,C);
158:          C[Art]:=0; (* excluye *)
159:          Cota2:=CotaMax(N,Art,Wl,V,W,C);
160:          if Cota1>Cota2 then
161:            begin
162:              if Cota1>Z then
163:                begin
164:                  C[art]:=1;
165:                  GeneraCombinacion(N,Art+1,V,W,Wl,
166:                    C,Z,X);
167:                  if Cota2>Z then
```

```
148:                        begin
149:                          C[art]:=0;
170:                          GeneraCombinacion(N,art+1,
171:                            V,W,W1,C,Z,X);
172:                        end;
173:                    end;
174:                end
175:              else
176:                begin
177:                  if Cota2>Z then
178:                    begin
179:                      C[art]:=0;
180:                      GeneraCombinacion(N,Art+1,V,W,W1,
181:                        C,Z,X);
182:                      if Cota1>Z then
183:                        begin
184:                          C[art]:=1;
185:                          GeneraCombinacion(N,art+1,
186:                            V,W,W1,C,Z,X);
187:                        end;
188:                    end;
189:                end;
190:            end
191:          else
192:            begin
193:              (* genera las puntas y checa si hay
194:                 solucion optima factible *)
195:              C[Art]:=1;
196:              if (ProdPunto(N,C,V)>Z) and
197:                   (ProdPunto(N,C,W)<=W1) then
198:                begin
199:                  Z:=ProdPunto(N,C,V);
200:                  X:=C;
201:                end;
202:              C[Art]:=0;
203:              if (ProdPunto(N,C,V)>Z) and
204:                   (ProdPunto(N,C,W)<=W1) then
205:                begin
206:                  Z:=ProdPunto(N,C,V);
207:                  X:=C;
208:                end;
209:            end;

210:        end;

211:      begin
212:        Z:=-10E10;
213:        GeneraCombi(N,1,V,W,W1,X,Z,X);
```

```
214:   end;

215:   procedure Lectura(var N : integer;
216:                     var V,W : VectReal;
217:                     var Wl : real);
218:   var
219:     I : integer;

220:   begin
221:     writeln(DispSal,'IO0402. ALGORITMO DE KOLESAR',
222:       ' PARA OPTIMA CARGA');
223:     write(DispSal,'numero de articulos=');
224:     readln(N);
225:     writeln(DispSal,N);
226:     writeln(DispSal,'valor de c/art:');
227:     LeeVect(N,V);
228:     writeln(DispSal,'peso de c/art:');
229:     LeeVect(N,W);
230:     write(DispSal,'peso limite=');
231:     readln(Wl);
232:     writeln(DispSal,Wl);

233:   end;

234:   begin
235:     IniDispSal;
236:     Lectura(N,V,W,Wl);
237:     Sort(N,V,W,Ord);
238:     OptimaCarga(N,V,W,Wl,Z,X);
239:     Resultados(N,V,W,Wl,Z,X,Ord);
240:     close(DispSal,lock);

241:   end.
```

```
 1:   I00402. ALGORITMO DE KOLESAR PARA OPTIMA CARGA
 2:   numero de articulos=7
 3:   valor de c/art:
 4:   x(1)= 8.00000
 5:   x(2)= 1.20000E1
 6:   x(3)= 2.00000
 7:   x(4)= 2.00000
 8:   x(5)= 1.00000
 9:   x(6)= 4.00000
10:   x(7)= 1.20000E1
11:   peso de c/art:
12:   x(1)= 8.00000
13:   x(2)= 1.00000E1
14:   x(3)= 4.00000
15:   x(4)= 2.00000
16:   x(5)= 4.00000
17:   x(6)= 8.00000
18:   x(7)= 6.00000
19:   peso limite= 2.00000E1

20:   resultados:

21:   solucion optima factible:

22:   valor carga= 2.60000E1
23:   peso carga= 1.80000E1
24:   solucion:
25:   x(7)= 1.00000
26:   x(2)= 1.00000
27:   x(1)= 0.00000
28:   x(4)= 1.00000
29:   x(3)= 0.00000
30:   x(6)= 0.00000
31:   x(5)= 0.00000
```

4.5.- IO05. Secuenciación.

En esta sección presentamos dos programas para resolver el problema de cómo secuenciar n trabajos en dos y tres máquinas respectivamente de tal manera de minimizar el tiempo transcurrido desde que se inicia el primer trabajo en la primera máquina hasta que termina el ultimo trabajo en la ultima máquina.

Suponemos que el tiempo de pasar un trabajo en una máquina a otra es insignificante. Al salir de una máquina, los trabajos forman una línea de espera (si es necesario) y entran a la siguiente máquina en ese mismo orden.

4.5.1.- I0050: Secuenciación de n Trabajos en 2 Maquinas.

Se necesario procesar cada uno de n trabajos en dos máquinas se desea determinar la secuencia que minimice el tiempo transcurrido desde que entra el primer trabajo a la primera maquina hasta que sale el n-esimo trabajo de la segunda máquina.

El programa acepta los siguientes datos:

```
n : numero de trabajos,(entero)
ttiempo[i,j] : tiempo que requiere el trabajo i
  en la máquina j,(real)
  i=1,..n, j=1,..2
```

Y produce las siguientes cantidades:

```
secuenc[ i ] : trabajo que entra a proceso en
  el i-ésimo lugar,(entero)
horario[i,1] : hora en que entra el trabajo que ocupa
  el i-ésimo lugar a la maquina 1,(real)
horario[i,2] : hora en que sale el trabajo que ocupa
  el i-ésimo lugar de la maquina 2,(real)
horario[i,3] : hora en que entra el trabajo que ocupa
  el i-ésimo lugar a la maquina 2,(real)
horario[i,4] : hora en que sale el trabajo que ocupa
  el i-ésimo lugar de la maquina 2,(real)
ocio1 : tiempo que permanece inactiva
  la máquina 1,(real)
ocio2 : tiempo que permanece inactiva
  la máquina 2,(real)
  i=1,..n
```

Ejemplo: Se tienen las estimaciones del tiempo que requiere cada uno de 8 trabajos a través de dos máquinas y se desea determinar la secuencia óptima. Tales estimaciones son como sigue:

TRABAJO	TIEMPO EN MAQUINA 1	MAQUINA 2
1	5	3
2	4	5
3	20	20
4	5	2
5	10	11
6	1	1
7	4	5
8	30	10

Estructurando los datos de acuerdo a los requisitos del programa

```
n = 8
ttiempo = [ 8  3]
          [ 4  5]
          [20 20]
          [ 5  2]
          [10 11]
          [ 1  1]
          [ 4  5]
          [30 10]
```

obtenemos la siguiente solución:

```
secuencia = [6 2 7 5 3 8 1 4]
horario = [   0   1   1   2]
          [   1   5   5  10]
          [   5   9  10  15]
          [   9  19  19  30]
          [  19  39  39  59]
          [  39  69  69  79]
          [  69  77  79  82]
          [  77  82  82  84]
ocio1 = 2
ocio2 = 27
```

Note que con la información obtenida podemos estructurar el siguiente horario de operación:

ORDEN	TRABAJO	MAQ 1 ENT	MAQ 1 SAL	MAQ 2 ENT	MAQ 2 SAL
1	6	0	1	1	2
2	2	1	5	5	10
3	7	5	9	10	15
4	5	9	19	19	30
5	3	19	39	39	59
6	8	39	69	69	79
7	1	69	77	79	82
8	4	77	82	82	84

```
 1:  program IO0501;
 2:  type
 3:    MAT2DIM1 = array [1..50,1..2] of real;
 4:    MAT2DIM2 = array [1..50,1..4] of real;
 5:    MAT1DIM  = array [1..50] of integer;

 6:  var
 7:    TTIEMPO : MAT2DIM1;
 8:    OCIO1,OCIO2 : real;
 9:    I,N : integer;
10:    SECUENCIA : MAT1DIM;
11:    HORARIO : MAT2DIM2;
12:    DispSal : text;

13:  procedure IniDispSal;
14:  var
15:    NomDispSal : string;

16:  begin
17:    write('Dispositivo de Salida=');
18:    readln(NomDispSal);
19:    rewrite(DispSal,NomDispSal);

20:  end;

21:  procedure REPORTE(N : integer;
22:                     SECUENCIA : MAT1DIM;
23:                     HORARIO : MAT2DIM2;
24:                     OCIO1,OCIO2 : real);
25:  var
26:    I,J : integer;

27:  begin
28:    writeln(DispSal);
29:    writeln(DispSal,'RESULTADOS:');
30:    writeln(DispSal);
31:    writeln(DispSal,'LA SEQUENCIA OPTIMA ES:');
32:    for I:=1 to N do
33:      begin
34:        write(DispSal,SECUENCIA[I]);
35:        write(DispSal,'=>');
36:      end;
37:    writeln(DispSal);
38:    writeln(DispSal);
39:    writeln(DispSal,'EL HORARIO DE ACTIVIDADES ES:');
40:    writeln(DispSal);
41:    writeln(DispSal,
42:      'TRAB/MAQ 1:ENT/SAL/MAQ 2:ENT/SAL');
```

```
43:     writeln(DispSal);
44:     for I:=1 to N do
45:       begin
46:         write(DispSal,SECUENCIA[I]);
47:         for J:=1 to 4 do
48:           begin
49:             write(DispSal,HORARIO[I,J]);
50:           end;
51:         writeln(DispSal);
52:       end;
53:     writeln(DispSal);
54:     writeln(DispSal,'TIEMPO INACTIVO MAQ 1 =',OCIO1);
55:     writeln(DispSal,'TIEMPO INACTIVO MAQ 2 =',OCIO2);
56:     writeln(DispSal,'TIEMPO TOTAL =',HORARIO[N,4]);

57:   end;

58:   procedure TIEMPOS(N : integer;
59:                     TTIEMPO : MAT2DIM1;
60:                     SECUENCIA : MAT1DIM;
61:                     var HORARIO : MAT2DIM2;
62:                     var OCIO1,OCIO2 : real);
63:   var
64:     I,ISEQ : integer;
65:     DIF : real;

66:   begin
67:     (*HORARIO DE MAQUINAS*)
68:     (* Y TIEMPO INACTIVO DE C/MAQ*)
69:     (* MAQ 1 *)
70:     ISEQ:=SECUENCIA[1];
71:     HORARIO[1,1]:=0;
72:     HORARIO[1,2]:=HORARIO[1,1]+TTIEMPO[ISEQ,1];
73:     OCIO2:=HORARIO[1,2];
74:     for I:=2 to N do
75:       begin
76:         ISEQ:=SECUENCIA[I];
77:         HORARIO[I,1]:=HORARIO[I-1,2];
78:         HORARIO[I,2]:=HORARIO[I,1]+TTIEMPO[ISEQ,1];
79:       end;

80:     (* MAQ 2 *)
81:     ISEQ:=SECUENCIA[1];
82:     HORARIO[1,3]:=HORARIO[1,2];
83:     HORARIO[1,4]:=HORARIO[1,3]+TTIEMPO[ISEQ,2];
84:     for I:=2 to N do
85:       begin
86:         ISEQ:=SECUENCIA[I];
```

```
 87:          DIF:=HORARIO[I,2]-HORARIO[I-1,4];
 88:          if DIF>0 then
 89:            begin
 90:              HORARIO[I,3]:=HORARIO[I,2];
 91:              OCIO2:=OCIO2+DIF;
 92:            end
 93:          else
 94:            begin
 95:              HORARIO[I,3]:=HORARIO[I-1,4];
 96:            end;
 97:          HORARIO[I,4]:=HORARIO[I,3]+TTIEMPO[ISEQ,2];
 98:        end;
 99:      OCIO1:=HORARIO[N,4]-HORARIO[N,2];

100:    end;

101:    procedure SECUENC(N : integer;
102:                      TTIEMPO : MAT2DIM1;
103:                      var SECUENCIA : MAT1DIM);
104:    var
105:      TIMFLAG : MAT1DIM;
106:      I,J,IMIN,JMIN,APPRIN,APFINAL : integer;
107:      MIN : real;

108:    begin
109:      (* SECUENCIA OPTIMA EN LAS 2 MAQUINAS *)
110:      (* INICIALIZACION *)
111:      for I:=1 to N do
112:        begin
113:          TIMFLAG[I]:=0;
114:        end;
115:      APPRIN:=1;
116:      APFINAL:=N;
117:      (* PASO 1: SELECCIONAMOS MENOR ELEM EN TABLA *)
118:      (*  Y LO QUITAMOS DE LA LISTA TIMFLAG=1*)
119:      repeat
120:        MIN:=10510;
121:        IMIN:=0;
122:        JMIN:=0;
123:        for I:=1 to 4 do
124:          for J:=1 to 2 do
125:            begin
126:              if (TTIEMPO[I,J]<MIN)
127:                 and (TIMFLAG[I]=0) then
128:                begin
129:                  MIN:=TTIEMPO[I,J];
130:                  IMIN:=I;
131:                  JMIN:=J;
```

```
132:                end;
133:              end;

134:          (* COLOCAMOS EN SECUENCIA:
135:             SI MAQ 1 =>PRINCIPIO *)
136:          (*  SI MAQ 2 => FINAL *)
137:          if JMIN=1 then
138:            begin
139:              TIMFLAG[IMIN]:=1;
140:              SECUENCIA[APPRIN]:=IMIN;
141:              APPRIN:=APPRIN+1;
142:            end
143:          else if JMIN=2 then
144:            begin
145:              TIMFLAG[IMIN]:=1;
146:              SECUENCIA[APFINAL]:=IMIN;
147:              APFINAL:=APFINAL-1;
148:            end;

149:      until (JMIN=0);(* TODOS EN SECUENCIA *)

150:    end;

151:    procedure LECTURA(var N : integer;
152:                      var TTIEMPO : MAT2DIM1);
153:    var
154:      OPCION,I,J:integer;

155:    begin
156:      writeln(DispSal);
157:      writeln(DispSal,
158:        'IO0501. SECUENCIA DE N TRABAJOS EN 2 MAQUINAS');
159:      writeln(DispSal);
160:      write(DispSal,'NUMERO DE TRABAJOS=');
161:      readln(N);
162:      writeln(DispSal,N);
163:      writeln(DispSal);
164:      writeln(DispSal,'TIEMPOS EN C/MAQUINA');
165:      writeln(DispSal);
166:      for I:=1 to N do
167:        for J:=1 to 2 do
168:          begin
169:            writeln(DispSal,'TRABAJO ',I,
170:              ' EN MAQUINA ',J);
171:            write(DispSal,'TIEMPO=');
172:            readln(TTIEMPO[I,J]);
173:            writeln(DispSal,TTIEMPO[I,J]);
174:          end;
```

```
175:     repeat
176:       writeln(DispSal);
177:       writeln(DispSal,'OPCIONES: 1.- CORREGIR,',
178:         '2.-DESPLEGAR, 9.- CONTINUAR');
179:       readln(OPCION);
180:       writeln(DispSal,OPCION);
181:       if OPCION = 1 then
182:         begin
183:           write(DispSal,'TRABAJO=');
184:           readln(I);
185:           writeln(DispSal,I);
186:           write(DispSal,'MAQUINA=');
187:           readln(J);
188:           writeln(DispSal,J);
189:           write(DispSal,'TIEMPO=');
190:           readln(TTIEMPO[I,J]);
191:           writeln(DispSal,TTIEMPO[I,J]);
192:         end
193:       else if OPCION = 2 then
194:         begin
195:           for I:=1 to N do
196:             for J:=1 to 2 do
197:               begin
198:                  writeln(DispSal,'TIEMPO TRAB ',
199:                    I,' MAQ ',J,' =',TTIEMPO[I,J]);
200:               end;
201:         end;

202:     until OPCION=9;

203:   end;

204:   begin
205:     IniDispSal;
206:     LECTURA(N,TTIEMPO);
207:     SECUENC(N,TTIEMPO,SECUENCIA);
208:     TIEMPOS(N,TTIEMPO,SECUENCIA,HORARIO,OCIO1,OCIO2);
209:     REPORTE(N,SECUENCIA,HORARIO,OCIO1,OCIO2);
210:     close(DispSal,lock);

211:   end.
```

```
 1:   I00501. SECUENCIA DE N TRABAJOS EN 2 MAQUINAS

 2:   NUMERO DE TRABAJOS=8

 3:   TIEMPOS EN C/MAQUINA

 4:   TRABAJO 1 EN MAQUINA 1
 5:   TIEMPO= 8.00000
 6:   TRABAJO 1 EN MAQUINA 2
 7:   TIEMPO= 3.00000
 8:   TRABAJO 2 EN MAQUINA 1
 9:   TIEMPO= 4.00000
10:   TRABAJO 2 EN MAQUINA 2
11:   TIEMPO= 5.00000
12:   TRABAJO 3 EN MAQUINA 1
13:   TIEMPO= 2.00000E1
14:   TRABAJO 3 EN MAQUINA 2
15:   TIEMPO= 2.00000E1
16:   TRABAJO 4 EN MAQUINA 1
17:   TIEMPO= 5.00000
18:   TRABAJO 4 EN MAQUINA 2
19:   TIEMPO= 2.00000
20:   TRABAJO 5 EN MAQUINA 1
21:   TIEMPO= 1.00000E1
22:   TRABAJO 5 EN MAQUINA 2
23:   TIEMPO= 1.10000E1
24:   TRABAJO 6 EN MAQUINA 1
25:   TIEMPO= 1.00000
26:   TRABAJO 6 EN MAQUINA 2
27:   TIEMPO= 1.00000
28:   TRABAJO 7 EN MAQUINA 1
29:   TIEMPO= 4.00000
30:   TRABAJO 7 EN MAQUINA 2
31:   TIEMPO= 5.00000
32:   TRABAJO 8 EN MAQUINA 1
33:   TIEMPO= 3.00000E1
34:   TRABAJO 8 EN MAQUINA 2
35:   TIEMPO= 1.00000E1

36:   OPCIONES: 1.- CORREGIR.2.-DESPLEGAR. 9.- CONTINUAR
37:   9

38:   RESULTADOS:

39:   LA SEQUENCIA OPTIMA ES:
40:   6=>2=>7=>5=>3=>8=>1=>4=>
```

```
41:   EL HORARIO DE ACTIVIDADES ES:

42:   TRAB/MAQ 1:ENT/SAL/MAQ 2:ENT/SAL

43:   6 0.00000 1.00000 1.00000 2.00000
44:   2 1.00000 5.00000 5.00000 1.00000E1
45:   7 5.00000 9.00000 1.00000E1 1.50000E1
46:   5 9.00000 1.90000E1 1.90000E1 3.00000E1
47:   3 1.90000E1 3.90000E1 3.90000E1 5.90000E1
48:   8 3.90000E1 6.90000E1 6.90000E1 7.90000E1
49:   1 6.90000E1 7.70000E1 7.90000E1 8.20000E1
50:   4 7.70000E1 8.20000E1 8.20000E1 8.40000E1

51:   TIEMPO INACTIVO MAQ 1 = 2.00000
52:   TIEMPO INACTIVO MAQ 2 = 2.70000E1
53:   TIEMPO TOTAL = 8.40000E1
```

4.5.2.- I00502. Secuenciacion de n Trabajos en 3 Maquinas.

Es necesario procesar cada uno de n trabajos en tres maquinas y se desea determinar la secuencia que minimice el tiempo transcurrido desde que entra el primer trabajo a la primera maquina hasta que sale el n-ésimo trabajo de la tercera maquina.

Para usar este programa es necesario que se cumpla al menos una de las siguientes condiciones:

1. El minimo tiempo de proceso para la primera máquina es por lo menos de igual magnitud que el máximo tiempo de proceso para la segunda máquina.
2. El minimo tiempo de proceso para la tercera máquina es por lo menos de igual magnitud que el máximo tiempo de proceso para la segunda máquina.

El programa acepta los siguientes datos:

n : número de trabajos.(entero)
ttiempo[i,j] : tiempo que requiere el trabajo i
en la máquina j.(real)
i=1..n, j=1..3

y produce las siguientes cantidades:

secuencia[i] : trabajo que entra a proceso en el
i-ésimo lugar.(entero)
horario[i,1] : hora en que entra el trabajo que ocupa
el i-ésimo lugar a la máquina 1.(real)
horario[i,2] : hora en que sale el trabajo que ocupa
el i-ésimo lugar de la maquina 2.(real)
horario[i,3] : hora en que entra el trabajo que ocupa
el i-ésimo lugar a la máquina 2.(real)
horario[i,4] : hora en que sale el trabajo que ocupa
el i-ésimo lugar de la máquina 3.(real)
horario[i,5] : hora en que entra el trabajo que ocupa
el i-ésimo lugar a la máquina 2.(real)
horario[i,6] : hora en que sale el trabajo que ocupa
el i-ésimo lugar de la máquina 3.(real)
ocio1 : tiempo que permanece inactiva
la máquina 1.(real)
ocio2 : tiempo que permanece inactiva
la máquina 2.(real)
ocio3 : tiempo que permanece inactiva
la máquina 3.(real)
i=1..n

Ejemplo: Se tienen las estimaciones del tiempo que requiere cada uno de 5 trabajos a través de tres maquinas y se desea determinar la secuencia optima. Tales estimaciones son como sigue:

TRABAJO	TIEMPO EN MAQUINA 1	MAQUINA 2	MAQUINA 3
1	8	3	20
2	4	5	21
3	20	20	35
4	5	2	23
5	10	11	22
6	1	1	20
7	4	5	45
8	30	10	20

Estructurando los datos de acuerdo a los requisitos del programa

```
n = 8
ttiempo = [   8   3  20]
          [   4   5  21]
          [  20  20  35]
          [   5   2  23]
          [  10  11  22]
          [   1   1  20]
          [   4   5  45]
          [  30  10  20]
```

obtenemos la siguiente solución:

```
secuencia = [6 4 2 7 1 5 3 8]
horario = [   0   1   1   2   2  22]
          [   1   6   6   8  22  45]
          [   6  10  10  15  45  66]
          [  10  14  15  20  66 111]
          [  14  22  22  25 111 131]
          [  22  32  32  43 131 153]
          [  32  52  52  72 153 188]
          [  52  82  82  92 188 208]
ocio1 = 126
ocio2 = 151
ocio3 =   2
```

Note que con la información obtenida podemos estructurar el siguiente horario de operación:

ORDEN	TRABAJO	MAQ 1 ENT	MAQ 1 SAL	MAQ 2 ENT	MAQ 2 SAL	MAQ 3 ENT	MAQ 3 SAL
1	6	0	1	1	2	2	22
2	4	1	6	6	8	22	45
3	2	6	10	10	15	45	66

4	7	10	14	15	20	66	111
5	1	14	22	22	25	111	131
6	5	22	32	32	43	131	153
7	3	32	52	52	72	153	188
8	8	52	82	82	92	188	206

```
 1:  program IO0502;
 2:  type
 3:    MAT2DIM1 = array [1..50,1..3] of real;
 4:    MAT2DIM2 = array [1..50,1..6] of real;
 5:    MAT1DIM  = array [1..50] of integer;

 6:  var
 7:    TIEMPO : MAT2DIM1;
 8:    OCIO1,OCIO2,OCIO3 : real;
 9:    SWNFAC,I,N : integer;
10:    SECUENCIA : MAT1DIM;
11:    HORARIO : MAT2DIM2;
12:    DispSal : text;

13:  procedure IniDispSal;
14:  var
15:    NomDispSal : string;

16:  begin
17:    write('Dispositivo de Salida=');
18:    readln(NomDispSal);
19:    rewrite(DispSal,NomDispSal);

20:  end;

21:  procedure REPORTE(N : integer;
22:                    SECUENCIA : MAT1DIM;
23:                    HORARIO : MAT2DIM2;
24:                    OCIO1,OCIO2,OCIO3 : real);
25:  var
26:    I,J : integer;

27:  begin
28:    writeln(DispSal);
29:    writeln(DispSal,'RESULTADOS:');
30:    writeln(DispSal);
31:    writeln(DispSal,'LA SECUENCIA OPTIMA ES:');
32:    for I:=1 to N do
33:      begin
34:        write(DispSal,SECUENCIA[I]);
35:        write(DispSal,'=>');
36:      end;
37:    writeln(DispSal);
38:    writeln(DispSal);
39:    writeln(DispSal,'EL HORARIO DE ACTIVIDADES ES:');
40:    writeln(DispSal);
41:    writeln(DispSal,'TRAB/MAQ 1:ENT/SAL   MAQ 2:ENT/SAL
42:    writeln(DispSal,'   MAQ 3:ENT/SAL');
```

```
43:     writeln(DispSal);
44:     for I:=1 to N do
45:       begin
46:         write(DispSal,SECUENCIA[I]);
47:         for J:=1 to 4 do
48:           begin
49:             write(DispSal,HORARIO[I,J]);
50:           end;
51:         writeln(DispSal);
52:         write(DispSal,' ':3);
53:         for J:=5 to 6 do
54:           begin
55:             write(DispSal,HORARIO[I,J]);
56:           end;
57:         writeln(DispSal);
58:       end;
59:     writeln(DispSal);
60:     writeln(DispSal,'TIEMPO INACTIVO MAQ 1 =',OCIO1);
61:     writeln(DispSal,'TIEMPO INACTIVO MAQ 2 =',OCIO2);
62:     writeln(DispSal,'TIMEPO INACTIVO MAQ 3 =',OCIO3);
63:     writeln(DispSal,'TIEMPO TOTAL =',HORARIO[N,6]);

64:   end;

65:   procedure TIEMPOS(N : integer;
66:                     TIEMPO : MAT2DIM1;
67:                     SECUENCIA : MAT1DIM;
68:                     var HORARIO : MAT2DIM2;
69:                     var OCIO1,OCIO2,OCIO3 : real);
70:   var
71:     I,ISEQ : integer;
72:     DIF : real;

73:   begin
74:     (*HORARIO DE MAQUINAS*)
75:     (* Y TIEMPO INACTIVO DE C/MAQ*)
76:     (* MAQ 1 *)
77:     ISEQ:=SECUENCIA[1];
78:     HORARIO[1,1]:=0;
79:     HORARIO[1,2]:=HORARIO[1,1]+TIEMPO[ISEQ,1];
80:     for I:=2 to N do
81:       begin
82:         ISEQ:=SECUENCIA[I];
83:         HORARIO[I,1]:=HORARIO[I-1,2];
84:         HORARIO[I,2]:=HORARIO[I,1]+TIEMPO[ISEQ,1];
85:       end;

86:     (* MAQ 2 *)
```

```
 87:      OCIO2:=HORARIO[1,2];
 88:      ISEQ:=SECUENCIA[1];
 89:      HORARIO[1,3]:=HORARIO[1,2];
 90:      HORARIO[1,4]:=HORARIO[1,3]+TIEMPO[ISEQ,2];
 91:      for I:=2 to N do
 92:        begin
 93:          ISEQ:=SECUENCIA[I];
 94:          DIF:=HORARIO[I,2]-HORARIO[I-1,4];
 95:          if DIF>0 then
 96:            begin
 97:              HORARIO[I,3]:=HORARIO[I,2];
 98:              OCIO2:=OCIO2+DIF;
 99:            end
100:          else
101:            begin
102:              HORARIO[I,3]:=HORARIO[I-1,4];
103:            end;
104:          HORARIO[I,4]:=HORARIO[I,3]+TIEMPO[ISEQ,2];
105:        end;
106:      (* MAQ 3 *)
107:      OCIO3:=HORARIO[1,4];
108:      ISEQ:=SECUENCIA[1];
109:      HORARIO[1,5]:=HORARIO[1,4];
110:      HORARIO[1,6]:=HORARIO[1,5]+TIEMPO[ISEQ,3];
111:      for I:=2 to N do
112:        begin
113:          ISEQ:=SECUENCIA[I];
114:          DIF:=HORARIO[I,4]-HORARIO[I-1,6];
115:          if DIF>0 then
116:            begin
117:              HORARIO[I,5]:=HORARIO[I,4];
118:              OCIO3:=OCIO3+DIF;
119:            end
120:          else
121:            begin
122:              HORARIO[I,5]:=HORARIO[I-1,6];
123:            end;
124:          HORARIO[I,6]:=HORARIO[I,5]+TIEMPO[ISEQ,3];
125:        end;
126:      OCIO1:=HORARIO[N,6]-HORARIO[N,2];
127:      OCIO2:=OCIO2+(HORARIO[N,6]-HORARIO[N,4]);

128:    end;

129:    procedure SECUENC(N : integer;
130:                      TIEMPO : MAT2DIM1;
131:                      var SECUENCIA : MAT1DIM);
132:    var
```

```
133:     TIMFLAG : MAT1DIM;
134:     I,J,IMIN,JMIN,APPRIN,APFINAL : integer;
135:     MIN : real;

136:   begin
137:     (* SECUENC OPTIMA EN LAS 2 MAQUINAS *)
138:     (* INICIALIZACION *)
139:     for I:=1 to N do
140:       begin
141:         TIMFLAG[I]:=0;
142:       end;
143:     APPRIN:=1;
144:     APFINAL:=N;
145:     (* PASO 1: SELECCIONAMOS MENOR ELEM EN TABLA *)
146:     (*  Y LO QUITAMOS DE LA LISTA TIMFLAG=1*)
147:     repeat
148:       MIN:=10E10;
149:       IMIN:=0;
150:       JMIN:=0;
151:       for I:=1 to N do
152:         for J:=1 to 2 do
153:           begin
154:             if ((TIEMPO[I,J]+TIEMPO[I,J+1])<MIN)
155:               and (TIMFLAG[I]=0) then
156:               begin
157:                 MIN:=TIEMPO[I,J]+TIEMPO[I,J+1];
158:                 IMIN:=I;
159:                 JMIN:=J;
160:               end;
161:           end;

162:       (* COLOCAMOS EN SECUENC: SI MAQ 1 =>
163:          PRINCIPIO *)
164:       (*  SI MAQ 2 => FINAL *)
165:       if JMIN=1 then
166:         begin
167:           TIMFLAG[IMIN]:=1;
168:           SECUENCIA[APPRIN]:=IMIN;
169:           APPRIN:=APPRIN+1;
170:         end
171:       else if JMIN=2 then
172:         begin
173:           TIMFLAG[IMIN]:=1;
174:           SECUENCIA[APFINAL]:=IMIN;
175:           APFINAL:=APFINAL-1;
176:         end;

177:     until (JMIN=0);(* TODOS EN SECUENC *)
```

```
178:   end;

179:   procedure FACTIBLE(N : integer;
180:                      var SWNFAC : integer;
181:                      var TIEMPO : MAT2DIM1);
182:   var
183:     OPCION,I,J : integer;
184:     MAX,MIN1,MIN2 : real;

185:   begin
186:     repeat
187:       (* VERIFICA QUE SE PUEDE RESOLVER CON *)
188:       (*  ESTE ALGORITMO *)
189:       SWNFAC:=0;
190:       MIN1:=10E10;
191:       for I:=1 to N do
192:         begin
193:           if TIEMPO[I,1]<MIN1 then
194:             MIN1:=TIEMPO[I,1];
195:         end;
196:       MAX:=0;
197:       for I:=1 to N do
198:         begin
199:           if TIEMPO[I,2]>MAX then
200:             MAX:=TIEMPO[I,2];
201:         end;
202:       MIN2:=10E10;
203:       for I:=1 to N do
204:         begin
205:           if TIEMPO[I,3]<MIN2 then
206:             MIN2:=TIEMPO[I,3];
207:         end;
208:       if (MIN1<MAX) and (MIN2<MAX) then
209:         begin
210:           writeln(DispSal,'NO SE PUEDE RESOLVER');
211:           SWNFAC:=1;
212:         end;
213:       (* OPCION A DESPLEGAR Y CORREGIR *)
214:       writeln(DispSal,'OPCIONES: 1.- CORREGIR,',
215:         ' 2.-DESPLEGAR, 9.- CONTINUAR');
216:       readln(OPCION);
217:       writeln(DispSal,OPCION);
218:       if OPCION = 1 then
219:         begin
220:           write(DispSal,'TRABAJO=');
221:           readln(I);
222:           writeln(DispSal,I);
```

```
223:            write(DispSal,'MAQUINA='):
224:            readln(J):
225:            writeln(DispSal,J):
226:            write(DispSal,'TIEMPO='):
227:            readln(TIEMPO[I,J]):
228:            writeln(DispSal,TIEMPO[I,J]):
229:          end
230:        else if OPCION = 2 then
231:          begin
232:            for I:=1 to N do
233:              for J:=1 to 3 do
234:                begin
235:                  writeln(DispSal,'TIEMPO TRAB ',I,
236:                    ' MAQ ',J,' =',TIEMPO[I,J]):
237:                end:
238:          end:

239:      until OPCION=9:

240:    end:

241:    procedure LECTURA(var N,SWNFAC : integer:
242:                      var TIEMPO : MAT2DIM1):
243:    var
244:      I,J:integer:

245:    begin
246:      writeln(DispSal):
247:      writeln(DispSal,'IO0502. SECUENCIA DE N',
248:        ' TRABAJOS EN 3 MAQUINAS'):
249:      writeln(DispSal):
250:      write(DispSal,'NUMERO DE TRABAJOS='):
251:      readln(N):
252:      writeln(DispSal,N):
253:      writeln(DispSal):
254:      writeln(DispSal,'TIEMPOS EN C/MAQUINA'):
255:      writeln(DispSal):
256:      for I:=1 to N do
257:        for J:=1 to 3 do
258:          begin
259:            writeln(DispSal,'TRABAJO ',I,
260:              ' EN MAQUINA ',J):
261:            write(DispSal,'TIEMPO='):
262:            readln(TIEMPO[I,J]):
263:            writeln(DispSal,TIEMPO[I,J]):
264:          end:
265:        writeln(DispSal):
266:        FACTIBLE(N,SWNFAC,TIEMPO):
```

```
267:   end;

268:   procedure INICIALIZA(var N : integer;
269:                        var TIEMPO : MAT2DIM1;
270:                        var SECUENCIA : MAT1DIM;
271:                        var HORARIO : MAT2DIM2);

272:   var
273:     I,J : integer;

274:   begin
275:     for I:=1 to 50 do
276:       begin
277:         SECUENCIA[I]:=0;
278:         for J:=1 to 3 do
279:           begin
280:             TIEMPO[I,J]:=0;
281:           end;
282:         for J:=1 to 6 do
283:           begin
284:             HORARIO[I,J]:=0;
285:           end;
286:       end;
287:     OCIO1:=0;
288:     OCIO2:=0;
289:     OCIO3:=0;
290:     N:=0;

291:   end;

292:   begin
293:     IniDispSal;
294:     INICIALIZA(N,TIEMPO,SECUENCIA,HORARIO);
295:     LECTURA(N,SWNFAC,TIEMPO);
296:     if (SWNFAC<>1) then
297:       begin
298:         SECUENC(N,TIEMPO,SECUENCIA);
299:         TIEMPOS(N,TIEMPO,SECUENCIA,HORARIO,
300:           OCIO1,OCIO2,OCIO3);
301:         REPORTE(N,SECUENCIA,HORARIO,
302:           OCIO1,OCIO2,OCIO3);
303:       end;
304:     close(DispSal,lock);

305:   end.
```

```
 1:   IO0502. SECUENCIA DE N TRABAJOS EN 3 MAQUINAS

 2:   NUMERO DE TRABAJOS=8

 3:   TIEMPOS EN C/MAQUINA

 4:   TRABAJO 1 EN MAQUINA 1
 5:   TIEMPO= 8.00000
 6:   TRABAJO 1 EN MAQUINA 2
 7:   TIEMPO= 3.00000
 8:   TRABAJO 1 EN MAQUINA 3
 9:   TIEMPO= 2.00000E1
10:   TRABAJO 2 EN MAQUINA 1
11:   TIEMPO= 4.00000
12:   TRABAJO 2 EN MAQUINA 2
13:   TIEMPO= 5.00000
14:   TRABAJO 2 EN MAQUINA 3
15:   TIEMPO= 2.10000E1
16:   TRABAJO 3 EN MAQUINA 1
17:   TIEMPO= 2.00000E1
18:   TRABAJO 3 EN MAQUINA 2
19:   TIEMPO= 2.00000E1
20:   TRABAJO 3 EN MAQUINA 3
21:   TIEMPO= 3.50000E1
22:   TRABAJO 4 EN MAQUINA 1
23:   TIEMPO= 5.00000
24:   TRABAJO 4 EN MAQUINA 2
25:   TIEMPO= 2.00000
26:   TRABAJO 4 EN MAQUINA 3
27:   TIEMPO= 2.30000E1
28:   TRABAJO 5 EN MAQUINA 1
29:   TIEMPO= 1.00000E1
30:   TRABAJO 5 EN MAQUINA 2
31:   TIEMPO= 1.10000E1
32:   TRABAJO 5 EN MAQUINA 3
33:   TIEMPO= 2.20000E1
34:   TRABAJO 6 EN MAQUINA 1
35:   TIEMPO= 1.00000
36:   TRABAJO 6 EN MAQUINA 2
37:   TIEMPO= 1.00000
38:   TRABAJO 6 EN MAQUINA 3
39:   TIEMPO= 2.00000E1
40:   TRABAJO 7 EN MAQUINA 1
41:   TIEMPO= 4.00000
42:   TRABAJO 7 EN MAQUINA 2
43:   TIEMPO= 5.00000
44:   TRABAJO 7 EN MAQUINA 3
```

```
45:   TIEMPO= 4.50000E1
46:   TRABAJO 8 EN MAQUINA 1
47:   TIEMPO= 3.00000E1
48:   TRABAJO 8 EN MAQUINA 2
49:   TIEMPO= 1.00000E1
50:   TRABAJO 8 EN MAQUINA 3
51:   TIEMPO= 2.00000E1

52:   OPCIONES: 1.- CORREGIR, 2.-DESPLEGAR, 9.- CONTINUAR
53:   9

54:   RESULTADOS:

55:   LA SECUENCIA OPTIMA ES:
56:   6=>4=>2=>7=>1=>5=>3=>8=>

57:   EL HORARIO DE ACTIVIDADES ES:

58:   TRAB/MAQ 1:ENT/SAL    MAQ 2:ENT/SAL
59:      MAQ 3:ENT/SAL

60:   6 0.00000 1.00000 1.00000 2.00000
61:       2.00000 2.20000E1
62:   4 1.00000 6.00000 6.00000 8.00000
63:       2.20000E1 4.50000E1
64:   2 6.00000 1.00000E1 1.00000E1 1.50000E1
65:       4.50000E1 6.60000E1
66:   7 1.00000E1 1.40000E1 1.50000E1 2.00000E1
67:       6.60000E1 1.11000E2
68:   1 1.40000E1 2.20000E1 2.20000E1 2.50000E1
69:       1.11000E2 1.31000E2
70:   5 2.20000E1 3.20000E1 3.20000E1 4.30000E1
71:       1.31000E2 1.53000E2
72:   3 3.20000E1 5.20000E1 5.20000E1 7.20000E1
73:       1.53000E2 1.88000E2
74:   8 5.20000E1 8.20000E1 8.20000E1 9.20000E1
75:       1.88000E2 2.08000E2

76:   TIEMPO INACTIVO MAQ 1 = 1.26000E2
77:   TIEMPO INACTIVO MAQ 2 = 1.51000E2
78:   TIMEPO INACTIVO MAQ 3 = 2.00000
79:   TIEMPO TOTAL = 2.08000E2
```

4.6.- IO06. Teoría de Decisiones.

La Teoría de Decisiones es una de las herramientas estocásticas de la Investigación de Operaciones. Su finalidad es auxiliar en la toma de decisiones en casos de incertidumbre. Para usar estos programas es necesario suponer la siguiente situación:

1. El tomador de decisiones puede escoger entre un número finito de acciones posibles.
2. Para cada acción existe un número finito de posibles estados de la naturaleza.
3. A cada combinación de acción y estado se le asocia un beneficio.

Ya que no se sabe con certeza cuál será el estado de la naturaleza, el objeto de estos programas es determinar la acción que minimice la pérdida suponiendo que este tipo de decisión se toma repetidas veces.

4.6.1.- I00601. Decisiones usando criterio de Bayes.

Supongamos que conocemos la probabilidad con que cada estado puede ocurrir. Este programa determina la acción con máximo valor esperado de acuerdo a las probabilidades dadas.

El programa acepta los siguientes datos:

```
n : número de acciones posibles.(entero)
m : número de estados posibles.(entero)
p[j] : probabilidad de que ocurra el estado j.(real)
benef[i,j] : beneficio al tomar la acción i cuando
  el estado resultante es j.(real)
   i=1..n, j=1..m
```

Y produce las siguientes cantidades:

```
z : beneficio máximo esperado.(real)
decision : acción a tomar.(entero)
```

Ejemplo: Se desea tomar la acción que maximice el beneficio esperado teniendo la siguiente información:

	estados		
acciones	1	2	3
1	5	6	1
2	2	2	10
3	7	4	4

Las probabilidades son .25, .6 y .15 para cada estado.

Expresando los datos de manera apropiada para el programa:

```
n = 3
m = 3
p = [.25 .6 .15]
benef = [5  6  1]
        [2  2 10]
        [7  4  4]
```

La solución buscada es:

```
z = 5
decision = 1
```

```
 1:  program IO0601;
 2:  type
 3:    MAT1DIM = array [1..50] of real;
 4:    MAT2DIM = array [1..50,1..50] of real;

 5:  var
 6:    N,M,DECISION : integer;
 7:    Z : real;
 8:    P,BENACC : MAT1DIM;
 9:    BENEF : MAT2DIM;
10:    DispSal : text;

11:  procedure IniDispSal;
12:  var
13:    NomDispSal : string;

14:  begin
15:    write('Dispositivo de Salida=');
16:    readln(NomDispSal);
17:    rewrite(DispSal,NomDispSal);

18:  end;

19:  procedure RESULTADOS(DECISION : integer;
20:                        Z : real);
21:  begin
22:    writeln(DispSal);
23:    writeln(DispSal,'RESULTADOS:');
24:    writeln(DispSal);
25:    writeln(DispSal,'BENEFICIO MAXIMO ESPERADO=',Z);
26:    writeln(DispSal,' AL TOMAR LA ACCION ',DECISION);
27:    writeln(DispSal);

28:  end;

29:  procedure POLITICA(N : integer;
30:                     BENACC : MAT1DIM;
31:                     var DECISION : integer;
32:                     var Z : real);
33:  var
34:    I : integer;

35:  begin
36:    (* DETERMINA ACCION CON MAXIMO BENEF ESPERADO *)
37:    Z:=-10E10;
38:    for I:=1 to N do
39:      begin
40:        if BENACC[I]>Z then
```

```
41:            begin
42:              Z:=BENACC[I];
43:              DECISION:=I;
44:            end;
45:        end;

46:    end;

47:    procedure ESPBENEF(N,M : integer;
48:                       P : MAT1DIM;
49:                       BENEF : MAT2DIM;
50:                       var BENACC : MAT1DIM);
51:    var
52:      I,J : integer;

53:    begin
54:      (* CALCULAR BENEF ESPERADO DE C/ACCION *)
55:      for I:=1 to N do
56:        begin
57:          BENACC[I]:=0;
58:          for J:=1 to M do
59:            begin
60:              BENACC[I]:=BENACC[I]+P[J]*BENEF[I,J];
61:            end;
62:        end;

63:    end;

64:    procedure LECTURA(var N,M : integer;
65:                      var P : MAT1DIM;
66:                      var BENEF : MAT2DIM);
67:    var
68:      I,J : integer;

69:    begin
70:      writeln(DispSal);
71:      writeln(DispSal,'IO0601, DECISIONES ',
72:        '(PROCEDIMIENTO DE BAYES)');
73:      writeln(DispSal);
74:      write(DispSal,'# ACCIONES POSIBLES=');
75:      readln(N);
76:      writeln(DispSal,N);
77:      write(DispSal,'# RESULTADOS POSIBLES=');
78:      readln(M);
79:      writeln(DispSal,M);
80:      for I:=1 to N do
81:        begin
82:          write(DispSal,'PROBABILIDAD A ',
```

```
 83:            'PRIORI DEL EDO ',I,' =');
 84:          readln(P[I]);
 85:          writeln(DispSal,P[I]);
 86:        end;
 87:      writeln(DispSal);
 88:      writeln(DispSal,'TABLA DE BENEFICIOS:');
 89:      writeln(DispSal);
 90:      for I:=1 to N do
 91:        for J:=1 to M do
 92:          begin
 93:            writeln(DispSal,'BENEFICIO SI ACCION ',
 94:              I,' Y RESULTADO ',J);
 95:            readln(BENEF[I,J]);
 96:            writeln(DispSal,BENEF[I,J]);
 97:          end;

 98:    end;

 99:    procedure INICIALIZA(var N,M : integer;
100:                         var P : MAT1DIM;
101:                         var BENEF : MAT2DIM;
102:                         var BENACC : MAT1DIM;
103:                         var DECISION : integer;
104:                         var Z : real);
105:    var
106:      I,J : integer;

107:    begin
108:      N:=0;
109:      M:=0;
110:      DECISION:=0;
111:      Z:=0;
112:      for I:=1 to 50 do
113:        begin
114:          P[I]:=0;
115:          for J:=1 to 50 do
116:            begin
117:              BENEF[I,J]:=0;
118:            end;
119:        end;

120:    end;

121:    begin
122:      IniDispSal;
123:      INICIALIZA(N,M,P,BENEF,BENACC,DECISION,Z);
124:      LECTURA(N,M,P,BENEF);
125:      ESPERENZA(N,M,P,BENEF,BENACC);
```

```
126:      POLITICA(N,GENACC,DECISION,Z);
127:      RESULTADOS(DECISION,Z);
128:      close(DispSal,lock);

129:    end.
```

```
 1:   IO0601. DECISIONES (PROCEDIMIENTO DE BAYES)

 2:   # ACCIONES POSIBLES=3
 3:   # RESULTADOS POSIBLES=3
 4:   PROBABILIDAD A PRIORI DEL EDO 1 = 2.50000E-1
 5:   PROBABILIDAD A PRIORI DEL EDO 2 = 6.00000E-1
 6:   PROBABILIDAD A PRIORI DEL EDO 3 = 1.50000E-1

 7:   TABLA DE BENEFICIOS:

 8:   BENEFICIO SI ACCION 1 Y RESULTADO 1
 9:    5.00000
10:   BENEFICIO SI ACCION 1 Y RESULTADO 2
11:    6.00000
12:   BENEFICIO SI ACCION 1 Y RESULTADO 3
13:    1.00000
14:   BENEFICIO SI ACCION 2 Y RESULTADO 1
15:    2.00000
16:   BENEFICIO SI ACCION 2 Y RESULTADO 2
17:    2.00000
18:   BENEFICIO SI ACCION 2 Y RESULTADO 3
19:    1.00000E1
20:   BENEFICIO SI ACCION 3 Y RESULTADO 1
21:    7.00000
22:   BENEFICIO SI ACCION 3 Y RESULTADO 2
23:    4.00000
24:   BENEFICIO SI ACCION 3 Y RESULTADO 3
25:    4.00000

26:   RESULTADO3:

27:   BENEFICIO MAXIMO ESPERADO= 5.00000
28:    AL TOMAR LA ACCION 1
```

4.6.2.- I00602. Decisiones Usando Criterio de Bayes con Información Adicional.

Supóngase que puede obtenerse información adicional sobre los estados de la naturaleza. En este caso es posible usar esta información para obtener una estimación más confiable de las probabilidades de los estados.

Por ejemplo: Podríamos usar información sobre una variable aleatoria dependiente del estado de la naturaleza y si es posible cuantificar éstos usando la fórmula de Bayes podemos determinar las probabilidades posteriores de los estados.

El programa acepta los siguientes datos:

```
n : número de acciones posibles.(entero)
m : número de estados posibles.(entero)
p[j] : probabilidad de que ocurra el estado j.(real)
benef[i,j] : beneficio al tomar la acción i
  cuando el estado de la naturaleza es j.(real)
thetha[] : valor del estado j.(real)
xstar : valor muestrado de la variable aleatoria x
  asociada a los estados.(real)
sigma2 : varianza de la distribución normal
  de x.(real)
  i=1..n,  j=1..m
```

Y produce las siguientes cantidades:

```
a) Política sin usar la información adicional:
     benacc1[i] : beneficio esperado el tomar
       la acción i.(real)
     z1         : beneficio máximo esperado.(real)
     decisio1   : acción a tomar.(entero)

b) Política usando la información adicional:
     ppost[i]   : probabilidad posterior
       del estado j.(real)
     benacc2[i] : beneficio esperado al tomar
       la acción i.(real)
     z2         : beneficio máximo esperado.(real)
     decisio2   : acción a tomar.(entero)
         i=1..n,  j=1..m
```

Ejemplo. Supóngase que se tiene que tomar la siguiente decisión:

	estado		
	1	2	3
valor estado:	4	8	12

acción			
1	5	6	1
2	2	2	10
3	7	4	4

p = [.25 .6 .15]
x se distribuye con n(thetha , sigma2) *
sigma2 = 10 *
xstar = 9 *
*informacion adicional

Expresando los datos de manera apropiada para el programa:

```
n = 3
m = 3
p = [.25 .6 .15]
benef = [5  6  1]
        [2  2 10]
        [7  4  4]
thetha = [4, 8, 12]
xstar = 9
sigma2 = 10
```

Obtenemos la siguiente solución:

a) Política sin usar la información adicional:

```
benacc1  = [5 3.2 4.75]
           z1       = 5
           decisio1 = 1
```

b) Política cuando la información adicional:

```
ppost    = [.09705 .77335 .12960]
benacc2  = [5.26 3.04 4.29]
z2       = 5.26
decisio2 = 1
```

```
 1:  program IO0602;
 2:  uses TRANSCEND;

 3:  type
 4:    MAT1DIM = array [1..50] of real;
 5:    MAT2DIM = array [1..50,1..50] of real;

 6:  var
 7:    N,M,DECISION1,DECISION2 : integer;
 8:    Z1,Z2,XSTAR,SIGMA2 : real;
 9:    THETHA,P,PPOST,BENACC1,BENACC2 : MAT1DIM;
10:    BENEF : MAT2DIM;
11:    DispSal : text;

12:  procedure IniDispSal;
13:  var
14:    NomDispSal : string;

15:  begin
16:    write('Dispositivo de Salida=');
17:    readln(NomDispSal);
18:    rewrite(DispSal,NomDispSal);

19:  end;

20:  procedure RESULTADOS;
21:  var
22:    I,J : integer;

23:  begin
24:    writeln(DispSal);
25:    writeln(DispSal,'RESULTADOS:');
26:    writeln(DispSal);
27:    writeln(DispSal,'POLITICA USANDO ',
28:      'PROBABILIDADES A PRIORI:');
29:    writeln(DispSal);
30:    for I:=1 to M do
31:      begin
32:        writeln(DispSal,'PROB EDO ',I,' =',P[I]);
33:      end;
34:    writeln(DispSal);
35:    for I:=1 to N do
36:      begin
37:        writeln(DispSal,'BENEF ESP AL TOMAR ACCION ',
38:          I,' =',PPNACC1[I]);
39:      end;
40:    writeln(DispSal);
41:    writeln(DispSal,'DECISION OPTIMA: ',
```

```
42:       'TOMAR ACCION ',DECISIO1);
43:     writeln(DispSal,' PARA UN BENEFICIO ',
44:       'ESPERADO DE ',Z1);
45:     writeln(DispSal);
46:     writeln(DispSal,'POLITICA USANDO ',
47:       'PROBABILIDADES POSTERIORES:');
48:     writeln(DispSal);
49:     for I:=1 to M do
50:       begin
51:         writeln(DispSal,'PROB EDO ',I,
52:           ' =',PPOST[I]);
53:       end;
54:     writeln(DispSal);
55:     writeln(DispSal,'VALOR ',
56:       'DE ESTADO ESPERADO =',XSTAR);
57:     writeln(DispSal);
58:     for I:=1 to N do
59:       begin
60:         writeln(DispSal,'BENEF ESP AL TOMAR ',
61:           'ACCION ',I,' =',BENACC2[I]);
62:       end;
63:     writeln(DispSal);
64:     writeln(DispSal,'DECISION OPTIMA: ',
65:       'TOMAR ACCION ',DECISIO2);
66:     writeln(DispSal,' PARA UN BENEFICIO ',
67:       'ESPERADO DE ',Z2);
68:     writeln(DispSal);

69:   end;

70:   procedure POLITICA(N : integer;
71:                      BENACC : MAT1DIM;
72:                      var DECISION : integer;
73:                      var Z : real);
74:   var
75:     I : integer;

76:   begin
77:     (* DETERMINA ACCION CON MAXIMO BENEF ESPERADO *)
78:     Z:=-10E10;
79:     for I:=1 to N do
80:       begin
81:         if BENACC[I]>Z then
82:           begin
83:             Z:=BENACC[I];
84:             DECISION:=I;
85:           end;
86:       end;
```

```
 87:   end;

 88:   procedure ESPBENE2(N,M : integer;
 89:                      P : MAT1DIM;
 90:                      BENEF : MAT2DIM;
 91:                      var BENACC,PPOST : MAT1DIM;
 92:                      THETHA : MAT1DIM;
 93:                      XSTAR,SIGMA2 : real);
 94:   var
 95:     MED,XTEMP1,XTEMP2 : real;
 96:     I,J : integer;

 97:     function NORMAL(XSTAR,MED,SIGMA2 : real) : real;
 98:     var
 99:       PI,XTEMP1,XTEMP2 : real;

100:     begin
101:       PI:=3.14159;
102:       XTEMP1:=1/(SQR(2*PI*SIGMA2));
103:       XTEMP2:=(XSTAR-MED)*(XSTAR-MED);
104:       XTEMP2:=-XTEMP2/(2*SIGMA2);
105:       XTEMP2:=EXP(XTEMP2);
106:       NORMAL:=XTEMP1*XTEMP2;

107:     end;

108:   begin
109:     (* CALCULA PROB POSTERIOR DE C/EDO *)
110:     (*  USANDO BAYES *)
111:     for I:=1 to M do
112:       begin
113:         MED:=THETHA[I];
114:         XTEMP1:=NORMAL(XSTAR,MED,SIGMA2)*P[I];
115:         XTEMP2:=0;
116:         for J:=1 to M do
117:           begin
118:             MED:=THETHA[J];
119:             XTEMP2:=XTEMP2+
120:               NORMAL(XSTAR,MED,SIGMA2)*P[J];
121:           end;
122:         PPOST[I]:=XTEMP1/XTEMP2;
123:       end;

124:     (* CALCULA VALOR ESPERADO DE C/ACCION *)
125:     for I:=1 to N do
126:       begin
127:         BENACC[I]:=0;
```

```
128:          for J:=1 to M do
129:            begin
130:              BENACC[I]:=BENACC[I]+
131:                BENEF[I,J]*P[POST[J]];
132:            end;
133:        end;

134:    end;

135:    procedure ESPBENE1(N,M : integer;
136:                       P : MAT1DIM;
137:                       BENEF : MAT2DIM;
138:                       var BENACC : MAT1DIM);
139:    var
140:      I,J : integer;

141:    begin
142:      (* CALCULAR BENEF ESPERADO DE C/ACCION *)
143:      for I:=1 to N do
144:        begin
145:          BENACC[I]:=0;
146:          for J:=1 to M do
147:            begin
148:              BENACC[I]:=BENACC[I]+P[J]*BENEF[I,J];
149:            end;
150:        end;

151:    end;

152:    procedure LECTURA(var N,M : integer;
153:                      var P : MAT1DIM;
154:                      var BENEF : MAT2DIM;
155:                      var THETHA : MAT1DIM;
156:                      var XSTAR,SIGMA2 : real);
157:    var
158:      I,J : integer;

159:    begin
160:      writeln(DispSal);
161:      writeln(DispSal,'IO0602. DECISIONES ',
162:        '(PROCEDIMIENTO DE BAYES)');
163:      writeln(DispSal);
164:      write(DispSal,'# ACCIONES POSIBLES=');
165:      readln(N);
166:      writeln(DispSal,N);
167:      write(DispSal,' # RESULTADOS POSIBLES=');
168:      readln(M);
169:      writeln(DispSal,M);
```

```
170:      for I:=1 to N do
171:        begin
172:          write(DispSal,'PROBABILIDAD A ',
173:            'PRIORI DEL EDO ',I,' =');
174:          readln(P[I]);
175:          writeln(DispSal,P[I]);
176:        end;
177:      writeln(DispSal);
178:      writeln(DispSal,'TABLA DE BENEFICIOS:');
179:      writeln(DispSal);
180:      for I:=1 to N do
181:        for J:=1 to M do
182:          begin
183:            writeln(DispSal,'BENEFICIO SI ACCION ',I,
184:              ' Y RESULTADO ',J);
185:            readln(BENEFI[I,J]);
186:            writeln(DispSal,BENEFI[I,J]);
187:          end;
188:      writeln(DispSal);
189:      writeln(DispSal,'DATOS PARA CALCULAR PROBABILIDADES');
190:      writeln(DispSal,'POSTERIORES:');
191:      writeln(DispSal);
192:      for I:=1 to M do
193:        begin
194:          write(DispSal,'VALOR DEL EDO ',I,' =');
195:          readln(THETHA[I]);
196:          writeln(DispSal,THETHA[I]);
197:        end;
198:      write(DispSal,'VALOR DEL EDO ESPERADO=');
199:      readln(XSTAR);
200:      writeln(DispSal,XSTAR);
201:      write(DispSal,'VARIANZA DISTRIBUCION NORMAL=');
202:      readln(SIGMA2);
203:      writeln(DispSal,SIGMA2);

204:    end;

205:    procedure INICIALIZA;
206:    var
207:      I,J : integer;

208:    begin
209:      N:=0;
210:      M:=0;
211:      DECISION1:=0;
212:      Z1:=0;
213:      DECISION2:=0;
214:      Z2:=0;
```

```
215:      XSTAR:=0;
216:      SIGMA2:=0;
217:      for I:=1 to 50 do
218:        begin
219:          P[I]:=0;
220:          PPOST[I]:=0;
221:          THETHA[I]:=0;
222:          BENACC1[I]:=0;
223:          BENACC2[I]:=0;
224:          for J:=1 to 50 do
225:            begin
226:              BENEF[I,J]:=0;
227:            end;
228:        end;

229:    end;

230:    begin
231:      IniDispSal;
232:      INICIALIZA;
233:      LECTURA(N,M,P,BENEF,THETHA,XSTAR,SIGMA2);
234:      ESPBENE1(N,M,P,BENEF,BENACC1);
235:      POLITICA(N,BENACC1,DECISIO1,Z1);
236:      ESPBENE2(N,M,P,BENEF,BENACC2,PPOST,
237:        THETHA,XSTAR,SIGMA2);
238:      POLITICA(N,BENACC2,DECISIO2,Z2);
239:      RESULTADOS;
240:      close(DispSal,lock);

241:    end.
```

```
 1:   I00602. DECISIONES (PROCEDIMIENTO DE BAYES)

 2:   # ACCIONES POSIBLES=3
 3:   # RESULTADOS POSIBLES=3
 4:   PROBABILIDAD A PRIORI DEL EDO 1 = 2.50000E-1
 5:   PROBABILIDAD A PRIORI DEL EDO 2 = 6.00000E-1
 6:   PROBABILIDAD A PRIORI DEL EDO 3 = 1.50000E-1

 7:   TABLA DE BENEFICIOS:

 8:   BENEFICIO SI ACCION 1 Y RESULTADO 1
 9:    5.00000
10:   BENEFICIO SI ACCION 1 Y RESULTADO 2
11:    6.00000
12:   BENEFICIO SI ACCION 1 Y RESULTADO 3
13:    1.00000
14:   BENEFICIO SI ACCION 2 Y RESULTADO 1
15:    2.00000
16:   BENEFICIO SI ACCION 2 Y RESULTADO 2
17:    2.00000
18:   BENEFICIO SI ACCION 2 Y RESULTADO 3
19:    1.00000E1
20:   BENEFICIO SI ACCION 3 Y RESULTADO 1
21:    7.00000
22:   BENEFICIO SI ACCION 3 Y RESULTADO 2
23:    4.00000
24:   BENEFICIO SI ACCION 3 Y RESULTADO 3
25:    4.00000

26:   DATOS PARA CALCULAR PROBABILIDADES
27:   POSTERIORES:

28:   VALOR DEL EDO 1 = 4.00000
29:   VALOR DEL EDO 2 = 8.00000
30:   VALOR DEL EDO 3 = 1.20000E1
31:   VALOR DEL EDO ESPERADO= 9.00000
32:   VARIANZA DISTRIBUCION NORMAL= 1.00000E1

33:   RESULTADOS:

34:   POLITICA USANDO PROBABILIDADES A PRIORI:

35:   PROB EDO 1 = 2.50000E-1
36:   PROB EDO 2 = 6.00000E-1
37:   PROB EDO 3 = 1.50000E-1

38:   BENEF ESP AL TOMAR ACCION 1 = 5.00000
```

```
39:   BENEF ESP AL TOMAR ACCION 2 = 3.20000
40:   BENEF ESP AL TOMAR ACCION 3 = 4.75000

41:   DECISION OPTIMA: TOMAR ACCION 1
42:    PARA UN BENEFICIO ESPERADO DE  5.00000

43:   POLITICA USANDO PROBABILIDADES POSTERIORES:

44:   PROB EDO 1 = 9.70534E-2
45:   PROB EDO 2 = 7.73349E-1
46:   PROB EDO 3 = 1.29590E-1

47:   VALOR DE ESTADO ESPERADO = 9.00000

48:   BENEF ESP AL TOMAR ACCION 1 = 5.25496
49:   BENEF ESP AL TOMAR ACCION 2 = 3.03678
50:   BENEF ESP AL TOMAR ACCION 3 = 4.29116

51:   DECISION OPTIMA: TOMAR ACCION 1
52:    PARA UN BENEFICIO ESPERADO DE  5.25496
```

4.7.1007.- Teoría de juegos.

Si consideramos que el tomador de decisiones no se enfrenta a la naturaleza sino a un oponente racional y activo, tenemos un problema de teoría de juegos. Cada una de los oponentes intentará maximizar su ganancia personal y por lo tanto deberán usar las estrategias más conservadoras que pueda.

En esta sección presentamos un programa que determina una estrategia de acción para cada uno de los jugadores. Suponemos que la pérdida de uno de uno es la ganancia del otro. A este tipo de juegos de le denomina juegos de suma cero.

4.7.1.- [10070]. Algoritmo de Brown.

Se desea determinar la estrategia optima de juegos para dos jugadores racionales.

El programa acepta los siguientes datos:
- m : número de acciones disponibles al jugador I.(entero)
- n : numero de acciones disponibles al jugador II.(entero)
- a[i,j] : beneficio para el jugador I si si escoge la acción i cuando el el jugador elige la acción j.(real)

Y produce las siguientes cantidades:
- upval : cota superior para el valor del juego para I.(real)
- lowval : cota inferior para el valor del juego.(real)
- xopt[i] : proporción de las veces que I debe escoger la acción i.(real)
- yopt[i] : proporción de las veces que II debe escoger la acción i.(real)

Ejemplo: Dos asesores industriales quieren determinar a qué tipo de industria orientar sus servicios. Cada uno tiene la opción de orientar sus servicios a la pequeña, mediana y gran industria.

Se estima que las ganancias para el asesor I se distribuyen de acuerdo a la siguiente tabla:

	Acciones asesor II		
Acciones asesor I	1	2	3
1	3	-1	-3
2	-3	3	-1
3	-4	-3	3

Expresando la información en forma apropiada para el programa:

```
m = 3
n = 3
a = [ 3 -1 -3]
    [-3  3 -1]
    [-4 -3  3]
```

Obtenemos la siguiente solución después de 1000 iteraciones y comenzando I con su primera estrategia:

```
upval  = -.62
lowval = -.652
xopt = [.442 .248 .310]
yopt = [.306 .248 .446]
```

Lo que quiere decir que si I sigue su estrategia óptima, en promedio tendrá un beneficio entre -.652 y -.62.

La solución exacta obtenida por Programación Lineal es:

```
valor del juego = -29/45
xopt = (1/45)*[20 11 14]
yopt = (1/45)*[14 11 20]
```

```
 1:  program I00731;
 2:  type
 3:    MAT1DIM = array [1..50] of real;
 4:    MAT2DIM = array [1..50,1..50] of real;

 5:  var
 6:    N,M,NUMIT,RENGINI : integer;
 7:    VALINF,VALSUP : real;
 8:    PAGOS : MAT2DIM;
 9:    ESTRAT1,ESTRAT2 : MAT1DIM;
10:    DispSal : text;

11:  procedure IniDispSal;
12:  var
13:    NomDispSal : string;

14:  begin
15:    write('Dispositivo de Salida=');
16:    readln(NomDispSal);
17:    rewrite(DispSal,NomDispSal);

18:  end;

19:  procedure RESULTADOS(N,M,NUMIT,RENGINI : integer;
20:                       ESTRAT1,ESTRAT2 : MAT1DIM;
21:                       VALINF,VALSUP : real);
22:  var
23:    I,J : integer;

24:  begin
25:    writeln(DispSal);
26:    writeln(DispSal,'RESULTADOS:');
27:    writeln(DispSal);
28:    writeln(DispSal,'DESPUES DE ',NUMIT,
29:      ' ITERACIONES Y');
30:    writeln(DispSal,' USANDO 1 LA ESTRATEGIA ',
31:      'INICIAL ',RENGINI,',');
32:    writeln(DispSal,
33:      'TENEMOS LOS SIGUIENTES RESULTADOS:');
34:    writeln(DispSal);
35:    writeln(DispSal,'VALOR DEL JUEGO:');
36:    writeln(DispSal,VALINF,
37:      ' < VALOR JUEGO < ',VALSUP);
38:    writeln(DispSal);
39:    writeln(DispSal,'ESTRATEGIA PARA I:');
40:    for I:=1 to N do
41:      begin
42:        writeln(DispSal,'USAR ESTRATEGIA ',I,
```

```
43:            ' '.ESTRAT1[I],' DEL TIEMPO');
44:        end;
45:      writeln(DispSal);
46:      writeln(DispSal,'ESTRATEGIA PARA II');
47:      for J:=1 to M do
48:        begin
49:          writeln(DispSal,'USAR ESTRATEGIA ',J,
50:            ' '.ESTRAT2[J],' DEL TIEMPO');
51:        end;

52:    end;

53:    procedure IMPRIMIRESULTADOS(N,M,NUMIT,RENGINI : integer;
54:                                PAGOS : MAT2DIM;
55:                                var VALINF,VALSUP : real;
56:                                var ESTRAT1,ESTRAT2 : MAT1DIM);
57:    var
58:      IRENG,JCOL,ITER,I,J : integer;
59:      TOTRENG,TOTCOL,JUEGRENG,JUEGCOL: MAT1DIM;
60:      MIN,MAX : real;

61:      procedure JUEGAII;
62:      begin
63:        (* II SUMA LOS RENGLONES QUE HA JUGADO I *)
64:        (*  Y JUEGA LA COLUMNA QUE CORRESPONDA
65:            AL MINIMO ELEMENTO *)
66:        MIN:=10E10;
67:        for J:=1 to M do
68:          begin
69:            TOTRENG[J]:=TOTRENG[J]+PAGOS[IRENG,J];
70:            if TOTRENG[J]<MIN then
71:              begin
72:                MIN:=TOTRENG[J];
73:                JCOL:=J;
74:              end;
75:          end;
76:        JUEGCOL[JCOL]:=JUEGCOL[JCOL]+1;

77:      end;

78:      procedure JUEGAI;
79:      begin
80:        (* I JUEGA EL RENGLON QUE CORRESPONDE
81:           AL MAX ELEMENTO *)
82:        (* DE LA SUMA DE LAS COLS QUE HA JUGADO II *)
83:        MAX:=-10E10;
84:        for I:=1 to N do
85:          begin
```

```
 86:             TOTCOL[I]:=TOTCOL[I]+PAGOS[I,JCOL];
 87:             if TOTCOL[I]>MAX then
 88:               begin
 89:                 MAX:=TOTCOL[I];
 90:                 IRENG:=I;
 91:               end;
 92:           end;
 93:         JUEGRENG[IRENG]:=JUEGRENG[IRENG]+1;

 94:       end;

 95:       procedure INICIALIZA;
 96:       begin
 97:         for I:=1 to N do
 98:           begin
 99:             TOTRENG[I]:=0;
100:             JUEGRENG[I]:=0;
101:           end;
102:         for J:=1 to M do
103:           begin
104:             TOTCOL[J]:=0;
105:             JUEGCOL[J]:=0;
106:           end;

107:       end;

108:     begin
109:       (* APROXIMA VALOR DEL JUEGO
110:          VALINF<VALOR<VALSUP *)
111:       (*  Y ESTRATEGIAS PARA I Y II *)
112:       INICIALIZA;
113:       IRENG:=RENGINI;
114:       for ITER:=1 to NUMIT do
115:         begin
116:           JUEGAII;
117:           JUEGAI;
118:         end;
119:       (* CALCULA COTAS INF Y SUP DEL VALOR DEL JUEGO *)
120:       VALINF:=MIN/NUMIT;
121:       VALSUP:=MAX/NUMIT;
122:       (* CALCULA ESTRATEGIAS APROX PARA I Y II *)
123:       for I:=1 to N do
124:         begin
125:           ESTRATI[I]:=JUEGRENG[I]/NUMIT;
126:         end;
127:       for J:=1 to M do
128:         begin
129:           ESTRATII[J]:=JUEGCOL[J]/NUMIT;
```

```
      end;

  end;

  procedure LECTURA(var N,M,NUMIT,RENGINI : integer;
                    var PAGOS : MAT2DIM);
  var
    I,J : integer;

  begin
    writeln(DispSal);
    writeln(DispSal,'IO0701. ALGORITMO DE BROWN PARA');
    writeln(DispSal,'JUEGOS DE M X N');
    writeln(DispSal);
    write(DispSal,'# ESTRATEGIAS PARA JUGADOR I=');
    readln(N);
    writeln(DispSal,N);
    write(DispSal,'# ESTRATEGIAS PARA JUGADOR II=');
    readln(M);
    writeln(DispSal,M);
    write(DispSal,'# ITERACIONES=');
    readln(NUMIT);
    writeln(DispSal,NUMIT);
    write(DispSal,'ESTRATEGIA INICIAL PARA I=');
    readln(RENGINI);
    writeln(DispSal,RENGINI);
    writeln(DispSal);
    writeln(DispSal,'MATRIZ DE PAGOS');
    writeln(DispSal);
    for I:=1 to N do
      begin
        writeln(DispSal,'I ESCOGE ESTRATEGIA ',I,':');
        for J:=1 to M do
          begin
            write(DispSal,' SI II ESCOGE ',
              'ESTRATEGIA ',J,' I GANA=');
            readln(PAGOS[I,J]);
            writeln(DispSal,PAGOS[I,J]);
          end;
      end;

  end;

  procedure INICIALIZA;
  var
    I,J : integer;

  begin
```

```
170:      (* INICIALIZA TODAS LAS VARIABLES GLOBALES *)
171:      I:=0;
172:      K:=0;
173:      NUMIT:=0;
174:      RENGINI:=0;
175:      VALINF:=0;
176:      VALSUP:=0;
177:      for I:=1 to 50 do
178:        begin
179:          ESTRATIII:=0;
180:          ESTRAT2II:=0;
181:          for J:=1 to 50 do
182:            begin
183:              PAGOSII,JJ:=0;
184:            end;
185:        end;

186:    end;

187:    begin
188:      InitDispGral;
189:      InitTotal_ZP;
190:      LEESTRATI(N,M,NUMIT,RENGINI,PAGOS);
191:      ITERACIONES(N,M,NUMIT,RENGINI,
192:        PAGOS,VALINF,VALSUP,ESTRAT1,ESTRAT2);
193:      ESTULTPAGOS(N,M,NUMIT,RENGINI,
194:        ESTRAT1,ESTRAT2,VALINF,VALSUP);
195:      ClearDevice;SetGraphMode(DetectGral,Lmode);

196:    end.
```

```
 1:   IC9701. ALGORITMO DE BROWN PARA
 2:   JUEGOS DE M X N

 3:   # ESTRATEGIAS PARA JUGADOR I=3
 4:   # ESTRATEGIAS PARA JUGADOR II=3
 5:   # ITERACIONES=1000
 6:   ESTRATEGIA INICIAL PARA I=1

 7:   MATRIZ DE PAGOS

 8:   I ESCOGE ESTRATEGIA 1:
 9:    SI II ESCOGE ESTRATEGIA 1 I GANA= 3.00000
10:    SI II ESCOGE ESTRATEGIA 2 I GANA=-1.00000
11:    SI II ESCOGE ESTRATEGIA 3 I GANA=-3.00000
12:   I ESCOGE ESTRATEGIA 2:
13:    SI II ESCOGE ESTRATEGIA 1 I GANA=-3.00000
14:    SI II ESCOGE ESTRATEGIA 2 I GANA= 3.00000
15:    SI II ESCOGE ESTRATEGIA 3 I GANA=-1.00000
16:   I ESCOGE ESTRATEGIA 3:
17:    SI II ESCOGE ESTRATEGIA 1 I GANA -4.00000
       SI II ESCOGE ESTRATEGIA 2 I GANA=-3.00000
       SI II ESCOGE ESTRATEGIA 3 I GANA= 3.00000

20:   RESULTADOS:

21:   DESPUES DE 1000 ITERACIONES Y
22:    USANDO I LA ESTRATEGIA INICIAL 1.
23:   TENEMOS LOS SIGUIENTES RESULTADOS:

24:   VALOR DEL JUEGO:
25:   -6.52000E-1 < VALOR JUEGO < -6.20000E-1

26:   ESTRATEGIA PARA I:
27:   USAR ESTRATEGIA 1  4.42000E-1 DEL TIEMPO
28:   USAR ESTRATEGIA 2  2.48000E-1 DEL TIEMPO
29:   USAR ESTRATEGIA 3  3.10000E-1 DEL TIEMPO

30:   ESTRATEGIA PARA II
31:   USAR ESTRATEGIA 1  3.08000E-1 DEL TIEMPO
32:   USAR ESTRATEGIA 2  2.48000E-1 DEL TIEMPO
33:   USAR ESTRATEGIA 3  4.44000E-1 DEL TIEMPO
```

4.8.- I008. PERT/CPM.

Las técnicas PERT y CPM fueron desarrolladas para control y planeación de proyectos. PERT (Program Evaluation and Review Technique) fue desarrollado por la Marina de los Estados Unidos en 1958 para el control del proyecto de desarrollo del misil Polaris. CPM (Critical Path Method) fue desarrollado por la Corporación Sperry-Rand para control de proyectos en la industria de la construcción. Las dos técnicas se han fusionado ahora en una sola.

El problema de control de proyectos es uno de los más estudiados en el área de Investigación de Operaciones y podemos decir que en cualquier proyecto de gran tamaño es una herramienta indispensable.

4.8.1.- I00801. PERT/CPM.

Es posible definir un proyecto como: un conjunto de actividades a ejecutar en determinado orden.

El primer paso para usar la técnica PERT/CPM consiste en diseñar la red de actividades del proyecto. Esencialmente, esta red determina la relación de precedencia entre actividades.

Además de la precedencia u orden entre actividades, es necesario estimar la duración de cada actividad. La técnica PERT/CPM está diseñada para manejar tres estimaciones de tiempo: el más optimista, el más probable y el más pesimista.

Teniendo el orden y la duración de las actividades, el siguiente paso es dar nombre a cada una de éstas. Para esto, definimos un evento como el principio o fin de un conjunto de actividades. Estos eventos están representados en la red como nodos y las actividades como flechas. Así, podemos hablar de que la actividad (1,2) comienza cuando termina el evento 1 y el evento 2 no puede terminar antes de que (1,2) sea completada.

El programa acepta los siguientes datos:

```
nact    : número de actividades.(entero)
nev     : número de eventos.(entero)
ini[i]  : evento donde comienza la actividad i.(entero)
fin[i]  : evento donde termina la actividad i.(entero)
a[i]    : tiempo optimista para terminar
  la actividad i.(real)
m[i]    : tiempo probable para terminar
  la actividad i.(real)
b[i]    : tiempo pesimista para terminar
  la actividad i.(real)
sd[j]   : tiempo programado para completar
  el evento j.(real)
  i=1..nact,   j=1..nev
```

Y produce las siguientes cantidades:

```
et[i]     : tiempo estimado para terminar
  la actividad i.(real)
siget[i]  : varianza del tiempo estimado
  para terminar la actividad i.(real)
cp[i]     : bandera que indica si la actividad i
  es crítica.(real)
te[j]     : tiempo más cercano para completar
  el evento j.(real)
tl[j]     : tiempo más lejano para completar
  el evento j.(real)
se[j]     : tiempo de holgura para completar
```

```
    el evento j.(real)
  sigte[j] : varianza del tiempo estimado para
    completar el evento j.(real)
  pr[j]    : punto para obtener de la distribución
    normal N(0,1) la probabilidad de completar
    el evento j antes de sd[j] unidades de tiempo.
    Pr(t[j]<sd[j])=Pr(z<pr[j]) donde z--N(0,1) y
    t[j] es el tiempo para comletar el evento j.(real)
    i=1..nact.   j=1..nev
```

Ejemplo: Supóngase que se desea hacer una tesis. Se ha determinado la siguiente lista básica de actividades:

ACTIVIDAD	TIEMPO OPT	PROB	PES (días)
1) Selección de tema	1	3	7
2) Investigación bibliográfica	7	15	30
3) Estructuración de temas	1	3	8
4) Introducción	4	5	10
5) Desarrollo	15	35	60
6) Conclusiones	4	7	15
7) Aprobación	7	15	30

La relación de precedencia que guarda una actividad con otra es:

a) La actividad 1 precede a todas las demás.
b) Las actividades 2, 3 y 4 se pueden realizar al mismo tiempo.
c) Las actividades 2 y 3 preceden a la actividad 5.
d) La actividad 5 precede a la actividad 6.
e) Las actividades 4 y 6 preceden a la actividad 7.
f) La actividad 7 es la última actividad.

Utilizando la información anterior, podemos formular la red del proyecto como en la figura 4.8.1.1.

Expresando los datos tal y como los requiere el programa:

```
nact = 8
nev  = 7
ini = [1 2 3 2 4 5 6 3]
fin = [2 4 3 6 5 6 7 4]
a   = [ 1  7  1  4 15  4  7  0]
m   = [ 3 15  3  5 35  7 15  0]
b   = [ 7 30  8 10 60 15 30  0]
sd  = [ 0  4  7 25 60 70 90]
```

Obtenemos la siguiente solución:

```
et     = [3.33 16.2 3.5 5.7 35.8 7.8 16.2 0]
s.get  = [1 14.7 1.36 1 5e.3 3.36 14.7 0]
cp     = [1 1 0 0 1 1 1 0]
te     = [0 3.33 e.83 19.5 55.3 e3.17 77.33]
tl     = [0 3.33 19.5 19.5 55.3 e3.17 77.33]
se     = [0 0 12.67 0 0 0 0 0]
sigte  = [0 1 2.36 19.e? 7..9 75.31 90]
pr     = [0 .e7 .103 .139 .55 .787 1.12]
```

```
 1:  program IO0601;
 2:  uses TRANSCEND;
 3:  type
 4:    MAT1DIMR = array [1..50] of real;
 5:    MAT1DIMI = array [1..50] of integer;

 6:  var
 7:    NEV,NACT : integer;
 8:    CP,SD,PR,A,M,B,ET,TE,TL,SE,SIGET,SIGTE : MAT1DIMR;
 9:    INI,FIN : MAT1DIMI;
10:    DispSal : text;

11:  procedure IniDispSal;
12:  var
13:    NomDispSal : string;

14:  begin
15:    write('Dispositivo de Salida=');
16:    readln(NomDispSal);
17:    rewrite(DispSal,NomDispSal);

18:  end;

19:  procedure RESULTADOS(NACT,NEV : integer;
20:                       INI,FIN : MAT1DIMI;
21:                       A,M,B,ET,SIGET : MAT1DIMR;
22:                       TE,TL,SE,SD,CP,
23:                         SIGTE,PR : MAT1DIMR);
24:  var
25:    I : integer;

26:  begin
27:    writeln(DispSal);
28:    writeln(DispSal,'RESULTADOS');
29:    writeln(DispSal);
30:    writeln(DispSal,'#ACT/INI-FIN/A/M/B/');
31:    writeln(DispSal,' ':8,'ET/SIGET');
32:    writeln(DispSal);
33:    for I:=1 to NACT do
34:      begin
35:        writeln(DispSal,I,' .',INI[I],'-',FIN[I],
36:          ',',A[I],',',B[I],',',M[I],',');
37:        write(DispSal,' ':8,ET[I],' .',SIGET[I]);
38:        if CP[I]=1 then
39:          write(DispSal,',','CRITICA');
40:        writeln(DispSal);
41:      end;
42:    writeln(DispSal);
```

```
43:     writeln(DispSal,'#EVENT/TE/TL/SE');
44:     writeln(DispSal);
45:     for I:=1 to NEV do
46:       begin
47:         writeln(DispSal,I,',',TE[I],',',
48:           TL[I],',',SE[I]);
49:       end;
50:     writeln(DispSal);
51:     writeln(DispSal,'#EVENT/TE/SD/SIGTE,PR');
52:     writeln(DispSal);
53:     for I:=2 to NEV do
54:       begin
55:         writeln(DispSal,I,',',TE[I],',',
56:           SD[I],',',SIGTE[I],',',PR[I]);
57:       end;

58:   end;

59:   procedure PROBSD(NEV : integer;
60:                    TE,SD,SIGET,SIGTE : MAT1DIMR;
61:                    var PR : MAT1DIMR);
62:   var
63:     I : integer;

64:   begin
65:     SD[1]:=0;
66:     for I:=2 to NEV do
67:       begin
68:         PR[I]:=(SD[I]-TE[I])/SQRT(SIGTE[I]);
69:       end;

70:   end;

71:   procedure CRITPAT(NACT : integer;
72:                     TE,TL,SE,ET : MAT1DIMR;
73:                     var CP : MAT1DIMR);
74:   var
75:     I,INIC,FINA : integer;

76:   begin
77:     for I:=1 to NACT do
78:       begin
79:         CP[I]:=0;
80:         INIC:=INI[I];
81:         FINA:=FIN[I];
82:         if (ABS(TL[INIC]-TE[INIC]-SE[NEV])<1E-5) then
83:          if (ABS(TL[FINA]-TE[FINA]-SE[NEV])<1E-5) then
84:           if (ABS(TE[FINA]-TE[INIC]-ET[I])<1E-5) then
```

```
 85:             if (ABS(TL[FINA]-TL[INIC]-ET[I])<1E-5) then
 86:               begin
 87:                 CP[I]:=1;
 88:               end;
 89:       end;

 90:   end;

 91:   procedure TIMSLC(NEV : integer;
 92:                    TE.TL : MAT1DIMR;
 93:                    var SE : MAT1DIMR);
 94:   var
 95:     I : integer;

 96:   begin
 97:     for I:=1 to NEV do
 98:       begin
 99:         SE[I]:=TL[I]-TE[I];
100:       end;

101:   end;

102:   procedure TIMLAT(NACT.NEV : integer;
103:                    INI.FIN : MAT1DIMI;
104:                    ET.TE : MAT1DIMR;
105:                    var TL : MAT1DIMR);
106:   var
107:     I.J.FINA : integer;
108:     MIN : real;

109:   begin
110:     TL[NEV]:=TE[NEV];
111:     for I:=NEV-1 downto 1 do
112:       begin
113:         MIN:=10E10;
114:         for J:=1 to NACT do
115:           begin
116:             if INI[J]=I then
117:               begin
118:                 FINA:=FIN[J];
119:                 if (TL[FINA]-ET[J])<MIN then
120:                   begin
121:                     MIN:=TL[FINA]-ET[J];
122:                   end;
123:               end;
124:           end;
125:         TL[I]:=MIN;
126:       end;
```

```
127:   end;

128:   procedure TIMEAR(NACT,NEV : integer;
129:                    INI,FIN : MAT1DIMI;
130:                    ET : MAT1DIMR;
131:                    var TE : MAT1DIMR);
132:   var
133:     I,J,INIC : integer;
134:     MAX : real;

135:   begin
136:     TE[1]:=0;
137:     SIGTE[1]:=0;
138:     for I:=2 to NEV do
139:       begin
140:         MAX:=-10E10;
141:         for J:=1 to NACT do
142:           begin
143:             if FIN[J]=I then
144:               begin
145:                 INIC:=INI[J];
146:                 if (TE[INIC]+ET[J])>MAX then
147:                   begin
148:                     MAX:=TE[INIC]+ET[J];
149:                     SIGTE[I]:=SIGTE[INIC]+SIGET[J];
150:                   end;
151:               end;
152:           end;
153:         TE[I]:=MAX;
154:       end;

155:   end;

156:   procedure TIMEXP(NACT : integer;
157:                    A,M,B : MAT1DIMR;
158:                    var ET,SIGET : MAT1DIMR);
159:   var
160:     I : integer;

161:   begin
162:     for I:=1 to NACT do
163:       begin
164:         ET[I]:=(A[I]+4*M[I]+B[I])/6;
165:         SIGET[I]:=((B[I]-A[I])/6);
166:         SIGET[I]:=SIGET[I]*SIGET[I];
167:       end;
```

```
211:    end;

212:    begin
213:      IniDispSal;
214:      LECTURA(NACT,NEV,INI,FIN,A,M,B,SD);
215:      TIMEXP(NACT,A,M,B,ET,SIGET);
216:      TIMEAR(NACT,NEV,INI,FIN,ET,TE);
217:      TIMLAT(NACT,NEV,INI,FIN,ET,TE,TL);
218:      TIMSLC(NEV,TE,TL,SE);
219:      CRITPA(NACT,TE,TL,SE,ET,CP);
220:      PROBSD(NEV,TE,SD,SIGET,SIGTE,PR);
221:      RESULTADOS(NACT,NEV,INI,FIN,A,M,B,ET,SIGET
222:        ,TE,TL,SE,SD,CP,SIGTE,PR);
223:      close(DispSal,lock);

224:    end.
```

```
168:   end;

169:   procedure LECTURA(var NACT,NEV : integer;
170:                     var INI,FIN : MAT1DIMI;
171:                     var A,M,B,SD : MAT1DIMR);

172:   var
173:     I : integer;

174:   begin
175:     writeln(DispSal,
176:       'IO0801. PERT/CPM PROBABILISTICO');
177:     writeln(DispSal);
178:     write(DispSal,'NUMERO DE ACTIVIDADES=');
179:     readln(NACT);
180:     writeln(DispSal,NACT);
181:     write(DispSal,'NUMERO DE EVENTOS=');
182:     readln(NEV);
183:     writeln(DispSal,NEV);
184:     for I:=1 to NACT do
185:       begin
186:         writeln(DispSal,'ACTIVIDAD ',I);
187:         write(DispSal,'INI=');
188:         readln(INI[I]);
189:         writeln(DispSal,INI[I]);
190:         write(DispSal,'FIN=');
191:         readln(FIN[I]);
192:         writeln(DispSal,FIN[I]);
193:         write(DispSal,'A=');
194:         readln(A[I]);
195:         writeln(DispSal,A[I]);
196:         write(DispSal,'M=');
197:         readln(M[I]);
198:         writeln(DispSal,M[I]);
199:         write(DispSal,'B=');
200:         readln(B[I]);
201:         writeln(DispSal,B[I]);
202:       end;
203:     writeln(DispSal);
204:     writeln(DispSal,'TIEMPO PROGRAMADO PARA COMPLETAR ');
205:     for I:=2 to NEV do
206:       begin
207:         write(DispSal,'EVENTO ',I,' =');
208:         readln(SD[I]);
209:         writeln(DispSal,SD[I]);
210:       end;
```

```
 1:   I00801. PERT/CPM PROBABILISTICO

 2:   NUMERO DE ACTIVIDADES=8
 3:   NUMERO DE EVENTOS=7
 4:   ACTIVIDAD 1
 5:   INI=1
 6:   FIN=2
 7:   A= 1.00000
 8:   M= 3.00000
 9:   B= 7.00000
10:   ACTIVIDAD 2
11:   INI=2
12:   FIN=4
13:   A= 7.00000
14:   M= 1.50000E1
15:   B= 3.00000E1
16:   ACTIVIDAD 3
17:   INI=2
18:   FIN=3
19:   A= 1.00000
20:   M= 3.00000
21:   B= 8.00000
22:   ACTIVIDAD 4
23:   INI=2
24:   FIN=6
25:   A= 4.00000
26:   M= 5.00000
27:   B= 1.00000E1
28:   ACTIVIDAD 5
29:   INI=4
30:   FIN=5
31:   A= 1.50000E1
32:   M= 3.50000E1
33:   B= 6.00000E1
34:   ACTIVIDAD 6
35:   INI=5
36:   FIN=6
37:   A= 4.00000
38:   M= 7.00000
39:   B= 1.50000E1
40:   ACTIVIDAD 7
41:   INI=6
42:   FIN=7
43:   A= 7.00000
44:   M= 1.50000E1
45:   B= 3.00000E1
46:   ACTIVIDAD 8
47:   INI=3
```

```
48:   FIN=4
49:   A= 0.00000
50:   M= 0.00000
51:   B= 0.00000

52:   TIEMPO PROGRAMADO PARA COMPLETAR
53:   EVENTO 2 = 4.00000
54:   EVENTO 3 = 7.00000
55:   EVENTO 4 = 2.50000E1
56:   EVENTO 5 = 6.00000E1
57:   EVENTO 6 = 7.00000E1
58:   EVENTO 7 = 9.00000E1

59:   RESULTADOS

60:   #ACT/INI-FIN/A/M/B/
61:             ET/SIGET

62:   1 .1-2. 1.00000. 7.00000. 3.00000.
63:            3.33333. 1.00000.CRITICA
64:   2 .2-4. 7.00000. 3.00000E1. 1.50000E1.
65:            1.61667E1. 1.46944E1.CRITICA
66:   3 .2-3. 1.00000. 8.00000. 3.00000.
67:            3.50000. 1.36111
68:   4 .2-6. 4.00000. 1.00000E1. 5.00000.
69:            5.66667. 1.00000
70:   5 .4-5. 1.50000E1. 6.00000E1. 3.50000E1.
71:            3.58333E1. 5.62500E1.CRITICA
72:   6 .5-6. 4.00000. 1.50000E1. 7.00000.
73:            7.83333. 3.36111.CRITICA
74:   7 .6-7. 7.00000. 3.00000E1. 1.50000E1.
75:            1.61667E1. 1.46944E1.CRITICA
76:   8 .3-4. 0.00000. 0.00000. 0.00000.
77:            0.00000. 0.00000

78:   #EVENT/TE/TL/SE

79:   1. 0.00000. 7.15256E-7. 7.15256E-7
80:   2. 3.33333. 3.33333. 7.15256E-7
81:   3. 6.83333. 1.95000E1. 1.26667E1
82:   4. 1.95000E1. 1.95000E1. 0.00000
83:   5. 5.53333E1. 5.53333E1. 0.00000
84:   6. 6.31667E1. 6.31667E1. 0.00000
85:   7. 7.93333E1. 7.93333E1. 0.00000

86:   #EVENT/TE/SD/SIGTE.PR

87:   2. 3.33333. 4.00000. 1.00000. 6.66667E-1
```

```
88:    3. 6.83333. 7.00000. 2.36111. 1.08465E-1
89:    4. 1.95000E1. 2.50000E1. 1.56944E1. 1.38832
90:    5. 5.53333E1. 6.00000E1. 7.19444E1. 5.50184E-1
91:    6. 6.31667E1. 7.00000E1. 7.53056E1. 7.87443E-1
92:    7. 7.93333E1. 9.00000E1. 9.00000E1. 1.12437
```

4.9.- I009. Teoría de Colas.

En frecuentes ocasiones, un administrador se enfrenta a la necesidad de tomar decisiones sobre problemas que involucran líneas de espera (colas). Es decir, problemas donde algunas personas esperan a que se les dé uno o varios servicios.

La decisión que se debe tomar es la siguente: Cuántos servicios tener disponibles de tal manera que los clientes no esperen demasiado ni los servicios permanezcan demasiado tiempo ociosos.

Como habrá observado el lector, el problema es bastante complicado dadas las combinaciones que pueden presentarse. Existen soluciones analíticas para cierto número de casos; estas soluciones nos indican cantidades como: número promedio de clientes en el sistema (en servicio o en línea de espera), tiempo promedio en el sistema, probabilidad de esperar menos de t unidades de tiempo en la línea de espera, etc. Sin embargo, para desarrollar este tipo de análisis, es necesario restringir los casos para que sean manejables, probablemente dejándolos tan simples que su utilidad sea mínima. En esta sección se presentarán algunos casos y su solución analítica, así como lo que debe cumplir el sistema para que la solución sea "válida".

En la siguiente sección se presentará la técnica de simulación y se darán algunas razones del por qué de su gran utilidad al analizar sistemas de líneas de espera y, en general, cualquier sistema.

4.9.1.- IO0901. Cola Infinita, Fuente Infinita, s Servicios. Llegadas Poisson, Servicio Exponencial.

Para usar este programa suponemos:

a) Llegadas al sistema en forma aleatoria.
b) Las llegadas forman una sola línea de espera (cola).
c) La disciplina de la línea de espera es FIFO (first in-first out), que indica que el primero en llegar es el primero en obtener servicio y por lo tanto en salir.
d) Salidas del sistema en forma aleatoria (es decir, que el tiempo de servicio es una variable aleatoria).
e) La probabilidad de una llegada en el intervalo (t,t+Dt), para Dt suficientemente pequeño, es lambdaDt.
f) La probabilidad de una salida en el intervalo (t,t+Dt), para Dt sufucientemente pequeño, es muDt.
g) La probabilidad de una o más llegadas y/o salidas en el intervalo (t,t+Dt) es negligible.
h) lambda<smu. Indica que no se forman líneas de espera infinitas.
i) El sistema está en estado estable. Es decir, ni la tasa de llegadas, ni la de servicio cambian con el tiempo.
donde:

lambda : tasa promedio de llegadas.
mu : tasa promedio de servicio.
s : número de servicios.

El programa acepta los siguientes datos:

s : número de servicios.(entero)
n : número máximo de clientes en el sistema.(entero)
t : tiempo crítico.(real);
lambda : tasa promedio de llegadas.(real)
mu : tasa promedio de servicio.(real)

Y produce las siguientes cantidades:

p[i] : probabilidad de que haya i clientes
en el sistema en cualquier momento.(real)
pzero : probabilidad de que no haya clientes
en el sistema en cualquier momento.(real)
pt : probabilidad de que el tiempo que pasa un cliente
en el sistema sea mayor al tiempo crítico t.(real);
ps : probabilidad de que haya al menos s clientes
en el sistema (es decir,que al llegar un cliente
lo encuentre ocupado) en cualquier momento.(real)
l : número promedio de clientes en el sistema.(real)
lq : número promedio de clientes en la
línea de espera.(real)
w : tiempo promedio en el sistema.(real)

wq : tiempo promedio en la línea de espera.(real)
i=1..n

Ejemplo: Supóngase que se desea analizar una gasolinera que tiene tres bombas (servicios). Los vehículos llegan a cargar gasolina con una tasa de 7 por hora y cada bomba tiene la capacidad de dar servicio en promedio a 9 vehículos por hora. El dueño indica que nunca hay más de 10 vehículos al mismo tiempo y quiere saber qué tan probable es que un cliente pase más de un cuarto de hora en la gasolinera.

Expresando los datos en la forma adecuada para el programa:

```
s = 3
n = 10
t = .25
lambda = 7
mu = 9
```

Obtenemos la siguiente solución:

```
p = [.36 .14 .036 .0093 .0024 .00063 .00016
     .000042 .000011 .0000028]
pzero = .46
pt = .11
ps = .05
l = .79
lq = .017
w = .114
wq = .0024
```

```
 1:  program I00901;
 2:  uses TRANSCEND;

 3:  type
 4:    MAT1DIM = array [1..100] of real;

 5:  var
 6:    N,S : integer;
 7:    LAMBDA,MU,T,PZERO,PS,PT,L,LQ,W,WQ: real;
 8:    P : MAT1DIM;
 9:    DispSal : text;

10:  procedure IniDispSal;
11:  var
12:    NomDispSal : string;

13:  begin
14:    write('Dispositivo de Salida=');
15:    readln(NomDispSal);
16:    rewrite(DispSal,NomDispSal);

17:  end;

18:  function POTENCIA(X : real;
19:                    N : integer) : real;
20:  var
21:    I : integer;
22:    XN : real;

23:  begin
24:    XN:=1;
25:    for I:=1 to N do
26:      begin
27:        XN:=XN*X;
28:      end;
29:    POTENCIA:=XN;

30:  end;

31:  function FACTORIAL(N : integer) : real;
32:  var
33:    I : integer;
34:    NFACT : real;

35:  begin
36:    NFACT:=1;
37:    for I:=2 to N do
38:      begin
```

```
39:         NFACT:=NFACT*I;
40:       end;
41:     FACTORIAL:=NFACT;

42:   end;

43:   procedure RESULTADOS(N,S : integer;
44:                        LAMBDA,MU,PZERO,PS,
45:                          PT,L,LQ,W,WQ : real;
46:                        P : MATIDIM);
47:   VAR
48:     I : integer;

49:   begin
50:     writeln(DispSal);
51:     writeln(DispSal,'RESULTADOS');
52:     writeln(DispSal);
53:     writeln(DispSal,'TASA DE LLEGADAS=',LAMBDA);
54:     writeln(DispSal,'TASA DE SERVICIO=',MU);
55:     writeln(DispSal,'PROB DE QUE HAYA');
56:     for I:=1 to N do
57:       begin
58:         writeln(DispSal,I,
59:           ' CLIENTES EN EL SIST=',P[I]);
60:       end;
61:     writeln(DispSal,'PROB DE 0 CLIENTES ',
62:       'EN SIST=',PZERO);
63:     writeln(DispSal,'PROB DE TIEMPO > CRITICO=',PT);
64:     writeln(DispSal,'PROB DE AL MENOS ',S,
65:       ' CLIENTES EN SIST=',PS);
66:     writeln(DispSal,'# PROM DE CLIENTES EN SIST=',L);
67:     writeln(DispSal,'# PROM DE CLIENTES EN COLA=',LQ);
68:     writeln(DispSal,'TIEMPO PROMEDIO EN SIST=',W);
69:     writeln(DispSal,'TIEMPO PROMEDIO EN COLA=',WQ);

70:   end;

71:   procedure CALCPARS(S : integer;
72:                      LAMBDA,MU,PZERO : real;
73:                      var PS,PT,L,LQ,W,WQ : real);
74:   var
75:     LMS,SFACT : real;

76:   begin
77:     LMS:=POTENCIA(LAMBDA/MU,S);
78:     SFACT:=FACTORIAL(S);
79:     PS:=LMS*PZERO/(SFACT*(1-LAMBDA/(MU*S)));
80:     PT:=LMS*PZERO*(1-EXP(-MU*T*(S-1-LAMBDA/MU)));
```

```
 81:     PT:=PT/(SQRT*(1-LAMBDA/(MU*S))*(S-1-LAMBDA/MU));
 82:     PT:=EXP(-MU*T)*(1+PT);
 83:     LQ:=LAMS*(LAMBDA/MU)*PZERO;
 84:     LQ:=LQ/(S*FACT*SQR(1-LAMBDA/(MU*S)));
 85:     L:=LQ+LAMBDA/MU;
 86:     W:=L/LAMBDA;
 87:     WQ:=LQ/LAMBDA;

 88:   end;

 89:   procedure CALCP(N,S : integer;
 90:                   PZERO : real;
 91:                   var P : MATIDIM);
 92:   var
 93:     I : integer;
 94:     FACTOR : real;

 95:   begin
 96:     for I:=1 to N do
 97:       begin
 98:         if I<S then
 99:           FACTOR:=FACTORIAL(I)
100:         else
101:           FACTOR:=FACTORIAL(S)*POTENCIA(S,I-S);
102:         P[I]:=(1/FACTOR)*POTENCIA(LAMBDA/MU,I)*PZERO;
103:       end;

104:   end;

105:   procedure CALCPZERO(S : integer;
106:                       LAMBDA,MU : real;
107:                       var PZERO : real);
108:   var
109:     I : integer;
110:     SUM : real;

111:   begin
112:     SUM:=0;
113:     for I:=0 to S-1 do
114:       begin
115:         SUM:=SUM+(1/FACTORIAL(I))*
116:           (POTENCIA(LAMBDA/MU,I));
117:       end;
118:     SUM:=SUM+(1/(FACTORIAL(S)*
119:       (1-LAMBDA/(MU*S)))*POTENCIA(LAMBDA/MU,S));
120:     PZERO:=1/SUM;

121:   end;
```

```
122:   procedure LECTURA(var N,S : integer;
123:                        var T,LAMBDA,MU : real);

124:   begin
125:     writeln(DispSal);
126:     writeln(DispSal,'IU0901. COLAS');
127:     writeln(DispSal);
128:     write(DispSal,'NUMERO DE SERVICIOS=');
129:     readln(S);
130:     writeln(DispSal,S);
131:     write(DispSal,'# MAX DE CLIENTES=');
132:     readln(N);
133:     writeln(DispSal,N);
134:     write(DispSal,'TIEMPO CRITICO=');
135:     readln(T);
136:     writeln(DispSal,T);
137:     write(DispSal,'TASA DE LLEGADAS=');
138:     readln(LAMBDA);
139:     writeln(DispSal,LAMBDA);
140:     write(DispSal,'TASA DE SERVICIO=');
141:     readln(MU);
142:     writeln(DispSal,MU);

143:   end;

144:   begin
145:     IniDispSal;
146:     LECTURA(N,S,T,LAMBDA,MU);
147:     CALCPZERO(S,LAMBDA,MU,PZERO);
148:     CALCP(N,S,PZERO,P);
149:     CALCPARS(S,LAMBDA,MU,PZERO,PS,PT,L,LQ,W,WQ);
150:     RESULTADOS(N,S,LAMBDA,MU,PZERO,PS,PT,L,LQ,W,WQ,P);
151:     close(DispSal,lock);

152:   end.
```

```
 1:   I00901. COLAS

 2:   NUMERO DE SERVICIOS=3
 3:   # MAX DE CLIENTES=10
 4:   TIEMPO CRITICO= 2.50000E-1
 5:   TASA DE LLEGADAS= 7.00000
 6:   TASA DE SERVICIO= 9.00000

 7:   RESULTADOS

 8:   TASA DE LLEGADAS= 7.00000
 9:   TASA DE SERVICIO= 9.00000
10:   PROB DE QUE HAYA
11:   1 CLIENTES EN EL SIST= 3.55781E-1
12:   2 CLIENTES EN EL SIST= 1.38359E-1
13:   3 CLIENTES EN EL SIST= 3.58710E-2
14:   4 CLIENTES EN EL SIST= 9.29988E-3
15:   5 CLIENTES EN EL SIST= 2.41108E-3
16:   6 CLIENTES EN EL SIST= 6.25095E-4
17:   7 CLIENTES EN EL SIST= 1.62062E-4
18:   8 CLIENTES EN EL SIST= 4.20160E-5
19:   9 CLIENTES EN EL SIST= 1.08930E-5
20:   10 CLIENTES EN EL SIST= 2.82412E-6
21:   PROB DE 0 CLIENTES EN SIST= 4.57433E-1
22:   PROB DE TIEMPO > CRITICO= 1.09308E-1
23:   PROB DE AL MENOS 3 CLIENTES EN SIST= 4.84258E-2
24:   # PROM DE CLIENTES EN SIST= 7.94727E-1
25:   # PROM DE CLIENTES EN COLA= 1.69490E-2
26:   TIEMPO PROMEDIO EN SIST= 1.13532E-1
27:   TIEMPO PROMEDIO EN COLA= 2.42129E-3
```

4.9.2.- I00902. Cola Finita, Fuente Infinita, s Servicios. Llegadas Poisson, Servicio Exponencial.

En este programa suponemos los mismo que en el programa anterior. La situación aquí representada es aquella donde el espacio disponible para que los clientes esperen es limitado.

El programa acepta los siguientes datos:

```
s : número de servicios.(entero)
m : máximo lugar disponible en el sistema.(entero)
lambda : tasa de llegadas.(real)
mu : tasa de servicio.(real)
```

Y produce las siguientes cantidades:

```
p[i] : probabilidad de que haya i clientes
  en el sistema en cualquier momento.(real)
lmbeff : tasa efectiva de llegadas.(real)
l : número promedio de clientes en el sistema.(real)
lq : número promedio de clientes en línea de espera.
w : tiempo promedio en el sistema.(real)
wq : tiempo promedio en la línea de espera.(real)
```

Ejemplo: Tomando los mismos datos que el ejemplo del programa anterior, pero suponiendo que el lugar disponible en el sistema es para un máximo de 10 clientes:

```
s = 3
m = 10
lambda = 7
mu = 9
```

obtenemos la siguiente solución:

```
p = [.46 .36 .14 .036 .0093 .0024 .00063
     .00016 .000042 .000011 .00000281
lmbeff = 6.99998
l = .79
lq = .017
w = .114
wq = .0024
```

```
 1:  program I0896Z;

 2:  type
 3:    MATIDIM = array [0..50] of real;

 4:  var
 5:    M,C : integer;
 6:    LAMBDA,MF,LASEFF,L,LQ,W,WQ,PTERO : real;
 7:    P : MATIDIM;
 8:    DispSal : text;

 9:  procedure IniDispSal;
10:  var
11:    NomDispSal : string;

12:  begin
13:    write('Dispositivo de Salida=');
14:    readln(NomDispSal);
15:    rewrite(DispSal,NomDispSal);

16:  end;

17:  function POTENCIA(X : real;
18:                    N : integer) : real;
19:  var
20:    I : integer;
21:    XN : real;

22:  begin
23:    XN:=1;
24:    for I:=1 to N do
25:      begin
26:        XN:=XN*X;
27:      end;
28:    POTENCIA:=XN;

29:  end;

30:  function FACTORIAL(N : integer) : real;
31:  var
32:    I : integer;
33:    NFACT : real;

34:  begin
35:    NFACT:=1;
36:    for I:=1 to N do
37:      begin
38:        NFACT:=NFACT*I;
```

```
39:       end;
40:     FACTORIAL:=NFACT;

41:   end;

42:   procedure RESULTADOS(M,S : integer;
43:                        LAMBDA,MU,PZERO,L,
44:                          LQ,LMBEFF,W,WQ : real;
45:                        P : MATIDIM);
46:   var
47:     I : integer;

48:   begin
49:     writeln(DispSal);
50:     writeln(DispSal,'RESULTADOS');
51:     writeln(DispSal);
52:     writeln(DispSal,'# SERVICIOS      =',S);
53:     writeln(DispSal,'CAPACIDAD EN SIST=',M);
54:     writeln(DispSal,'TASA DE LLEGADAS =',LAMBDA);
55:     writeln(DispSal,'TASA DE SERVICIO =',MU);
56:     writeln(DispSal,'PROBABILIDAD DE ',
57:       'QUE EN EL SIST HAYA');
58:     for I:=0 to M do
59:       begin
60:         writeln(DispSal,I,' CLIENTES      =',P[I]);
61:       end;
62:     writeln(DispSal,'TASA EFECTIVA DE ',
63:       'LLEGADAS   =',LMBEFF);
64:     writeln(DispSal,'PROMEDIO DE CLIENTES ',
65:       'EN SIST=',L);
66:     writeln(DispSal,'PROMEDIO DE CLIENTES',
67:       ' EN COLA=',LQ);
68:     writeln(DispSal,'TIEMPO PROMEDIO ',
69:       'EN SIST     =',W);
70:     writeln(DispSal,'TIEMPO PROMEDIO ',
71:       'EN COLA     =',WQ);

72:   end;

73:   procedure CALCPARS(S : integer;
74:                      LQ : real;
75:                      P : MATIDIM;
76:                      var L,LMBEFF,W,WQ : real);
77:   var
78:     I : integer;
79:     SUM : real;

80:   begin
```

```
 81:      SUM:=0;
 82:      for I:=0 to S-1 do
 83:        begin
 84:          SUM:=SUM+(S-I)*P[I];
 85:        end;
 86:      L:=LQ+S-SUM;
 87:      LAMBEFF:=MU*(S-SUM);
 88:      WQ:=LQ/LAMBEFF;
 89:      W:=L/LAMBEFF;

 90:    end;

 91:    procedure CALCLQ(M,S : integer;
 92:                     LAMBDA,MU,PZERO : real;
 93:                     var LQ : real);

 94:    begin
 95:      LQ:=(M-S)*POTENCIA(LAMBDA/
 96:        (MU*S),M-S)*(1-LAMBDA/(MU*S));
 97:      LQ:=1-POTENCIA(LAMBDA/(MU*S),M-S)-LQ;
 98:      LQ:=LQ*PZERO*POTENCIA(LAMBDA/MU,S)*
 99:        LAMBDA/(MU*S);
100:      LQ:=LQ/(FACTORIAL(S)*SQR(1-LAMBDA/(MU*S)));

101:   end;

102:   procedure CALCP(M,S : integer;
103:                   PZERO,LAMBDA,MU : real;
104:                   var P : MAT1DIM);
105:   var
106:     I : integer;

107:   begin
108:     P[0]:=PZERO;
109:     for I:=1 to S do
110:       begin
111:         P[I]:=1/FACTORIAL(I)*
112:           POTENCIA(LAMBDA/MU,I)*PZERO;
113:       end;
114:     for I:=S+1 to M do
115:       begin
116:         P[I]:=1/(FACTORIAL(S)*POTENCIA(S,I-S))*
117:           POTENCIA(LAMBDA/MU,I)*PZERO;
118:       end;

119:   end;

120:   procedure CALCPZERO(M,S : integer;
```

```
121:                          LAMBDA,MU : real;
122:                          var PZERO : real);
123:    var
124:      I : integer;
125:      SUM1,SUM2 : real;

126:    begin
127:      SUM1:=0;
128:      for I:=0 to S do
129:        begin
130:          SUM1:=SUM1+(1/FACTORIAL(I))*
131:            POTENCIA(LAMBDA/MU,I);
132:        end;
133:      SUM2:=0;
134:      for I:=S+1 to M do
135:        begin
136:          SUM2:=SUM2+POTENCIA(LAMBDA/(MU*S),I-S);
137:        end;
138:      SUM2:=SUM2*(1/FACTORIAL(S))*
139:        POTENCIA(LAMBDA/MU,S);
140:      PZERO:=1/(SUM1+SUM2);

141:    end;

142:    procedure LECTURA(var M,S : integer;
143:                      var LAMBDA,MU : real);

144:    begin
145:      writeln(DispSal);
146:      writeln(DispSal,'IO0902, COLAS');
147:      writeln(DispSal);
148:      write(DispSal,'NUMERO DE SERVICIOS=');
149:      readln(S);
150:      writeln(DispSal,S);
151:      write(DispSal,'LUGAR MAX EN SIST  =');
152:      readln(M);
153:      writeln(DispSal,M);
154:      write(DispSal,'TASA DE LLEGADAS   =');
155:      readln(LAMBDA);
156:      writeln(DispSal,LAMBDA);
157:      write(DispSal,'TASA DE SERVICIO   =');
158:      readln(MU);
159:      writeln(DispSal,MU);

160:    end;

161:    begin
162:      Initialize;
```

```
163:      LECTURA(M,S,LAMBDA,MU);
164:      CALCPZERO(M,S,LAMBDA,MU,PZERO);
165:      CALCP(M,S,PZERO,LAMBDA,MU,P);
166:      CALCLQ(M,S,LAMBDA,MU,PZERO,LQ);
167:      CALCPARS(S,LQ,P,L,LMBEFF,W,WQ);
168:      RESULTADOS(M,S,LAMBDA,MU,PZERO,
169:        L,LQ,LMBEFF,W,WQ,P);
170:      close(DispSal,lock);

171:    end.
```

```
 1:   IO0902. COLAS

 2:   NUMERO DE SERVICIOS=3
 3:   LUGAR MAX EN SIST  =10
 4:   TASA DE LLEGADAS   = 7.00000
 5:   TASA DE SERVICIO   = 9.00000

 6:   RESULTADOS

 7:   # SERVICIOS       =3
 8:   CAPACIDAD EN SIST=10
 9:   TASA DE LLEGADAS = 7.00000
10:   TASA DE SERVICIO = 9.00000
11:   PROBABILIDAD DE QUE EN EL SIST HAYA
12:   0 CLIENTES       = 4.57434E-1
13:   1 CLIENTES       = 3.55702E-1
14:   2 CLIENTES       = 1.38360E-1
15:   3 CLIENTES       = 3.58710E-2
16:   4 CLIENTES       = 9.29909E-3
17:   5 CLIENTES       = 2.41108E-3
18:   6 CLIENTES       = 6.25096E-4
19:   7 CLIENTES       = 1.62062E-4
20:   8 CLIENTES       = 4.20160E-5
21:   9 CLIENTES       = 1.08930E-5
22:   10 CLIENTES       = 2.82412E-6
23:   TASA EFECTIVA DE LLEGADAS   = 6.99998
24:   PROMEDIO DE CLIENTES EN SIST= 7.74717E-1
25:   PROMEDIO DE CLIENTES EN COLA= 1.59408E-2
26:   TIEMPO PROMEDIO EN SIST     = 1.1353IE-1
27:   TIEMPO PROMEDIO EN COLA     = 2.42012E-3
```

4.9.3.- I00903. Cola Infinita, Fuente Infinita, 1 Servicio. Llegadas Poisson, Servicio Arbitrario.

Para usar este programa debemos suponer:

a) Llegadas el sistema en forma aleatoria.
b) Las llegadas forman una sola línea de espera.
c) Disciplina FIFO.
d) El tiempo de servicio T tiene una distribución arbitraria con media E(T) y varianza var(T).
e) El tiempo promedio entre llegadas es mayor que el tiempo esperado de servicio: 1/lambda>E(T).

El programa acepta los siguientes datos:

lambda : tasa de llegadas.(real)
timexp : tiempo esperado de servicio.(real)
vartex : varianza del tiempo esperado de servicio.(real)

Y produce las siguientes cantidades:

pzero : probabilidad de que el sistema esté vacío en cualquier momento.(real)
l : número promedio de clientes en el sistema.(real)
lq : número promedio de clientes en la línea de espera.(real)
w : tiempo promedio en el sistema.(real)
wq : tiempo promedio en la línea de espera.(real)

Ejemplo: Una peluquería con un solo peluquero recibe clientes con una tasa de 7 por hora. El tiempo promedio de servicio por cliente es de .11111 hrs con una varianza de .07.

Expresando los datos de manera adecuada para el programa:

lambda = 7
E(T) = .11111
vartex(T) = .07

obtenemos la siguiente solución:

pzero = .22
l = 9.86
lq = 9.08
w = 1.41
wq = 1.3

```
 1:  program IQ0903;
 2:  var
 3:    LAMBDA,TIMEXP,VARTEX,PZERO,L,LQ,W,WQ : real;
 4:    DispSal : text;

 5:  procedure IniDispSal;
 6:  var
 7:    NomDispSal : string;

 8:  begin
 9:    write('Dispositivo de Salida=');
10:    readln(NomDispSal);
11:    rewrite(DispSal,NomDispSal);

12:  end;

13:  procedure RESULTADOS(LAMBDA,TIMEXP,VARTEX,
14:                           PZERO,L,LQ,W,WQ : real);
15:  begin
16:    writeln(DispSal);
17:    writeln(DispSal,'RESULTADOS');
18:    writeln(DispSal);
19:    writeln(DispSal,
20:      'TASA DE LLEGADAS              =',LAMBDA);
21:    writeln(DispSal,
22:      'TIEMPO ESPERADO DE SERVICIO   =',TIMEXP);
23:    writeln(DispSal,
24:      'VARIANZA DE TIEMPO ESPERADO   =',VARTEX);
25:    writeln(DispSal);
26:    writeln(DispSal,
27:      'PROB DE CERO CLIENTES EN SIST=',PZERO);
28:    writeln(DispSal,
29:      '# PROM DE CLIENTES EN SIST    =',L);
30:    writeln(DispSal,
31:      '# PROM DE CLIENTES EN COLA    =',LQ);
32:    writeln(DispSal,
33:      'TIEMPO PROM EN SIST           =',W);
34:    writeln(DispSal,
35:      'TIEMPO PROM EN COLA           =',WQ);

36:  end;

37:  procedure CALCPARS(LAMBDA,TIMEXP,VARTEX : real;
38:                       var PZERO,L,LQ,W,WQ : real);
39:  begin
40:    PZERO:=1-LAMBDA*TIMEXP;
41:    L:=SQR(LAMBDA)*VARTEX+SQR(LAMBDA*TIMEXP);
42:    L:=L/(2*(1-LAMBDA*TIMEXP));
```

```
46:     L:=LAMBDA*TIMEXP+LQ;
47:     LQ:=L-LAMBDA*TIMEXP;
48:     W:=L/LAMBDA;
49:     WQ:=LQ/LAMBDA;

50:   end;

51:   procedure LECTURA(var LAMBDA,TIMEXP,VARTEX : real);
52:   begin
53:     write(DispSal,'');
54:     writeln(DispSal,'IO0903. COLAS');
55:     writeln(DispSal);
56:     write(DispSal,'TASA DE LLEGADAS=');
57:     readln(LAMBDA);
58:     writeln(DispSal,LAMBDA);
59:     writeln(DispSal,'DISTRIBUCION DE ',
60:       'TIEMPO DE SERVICIO:');
61:     write(DispSal,' MEDIA=');
62:     readln(TIMEXP);
63:     writeln(DispSal,TIMEXP);
64:     write(DispSal,' VARIANZA=');
65:     readln(VARTEX);
66:     writeln(DispSal,VARTEX);

67:   end;

68:   procedure INICIALIZA;
69:   begin
70:     PZERO:=0;
71:     L:=0;
72:     LQ:=0;
73:     W:=0;
74:     WQ:=0;

75:   end;

76:   begin
77:     AnaDispSal;
78:     INICIALIZA;
79:     LECTURA(LAMBDA,TIMEXP,VARTEX);
80:     IF (LAMBDA*TIMEXP) <1
81:       THEN
82:         BEGIN
83:           CALCPROB(LAMBDA,TIMEXP,VARTEX,PZERO,L,LQ,W,WQ);
84:           IMPRESULT(LAMBDA,TIMEXP,VARTEX,PZERO,L,LQ,W,WQ);
85:         END
86:       ELSE
87:         writeln(DispSal,'TIEMPO DE SERVICIO DE SERVICIO ',
88:           'MAYOR QUE LA LAMBDA');
```

```
85:     close(DiskCalcBook);

86:   end.
```

```
 1:   100703. COLAS

 2:   TASA DE LLEGADAS= 7.00000
 3:   DISTRIBUCION DE TIEMPO DE SERVICIO:
 4:    MEDIA= 1.11110E-1
 5:    VARIANZA= 7.00000E-2

 6:   RESULTADOS

 7:   TASA DE LLEGADAS              = 7.00000
 8:   TIEMPO ESPERADO DE SERVICIO   = 1.11110E-1
 9:   VARIANZA DE TIEMPO ESPERADO   = 7.00000E-2

10:   PROB DE CERO CLIENTES EN SIST= 2.22230E-1
11:   # PROM DE CLIENTES EN SIST    = 9.85604
12:   # PROM DE CLIENTES EN COLA    = 9.07827
13:   TIEMPO PROM EN SIST           = 1.40801
14:   TIEMPO PROM EN COLA           = 1.29690
```

4.10.- IO10. Simulación.

La simulación es una de las técnicas que han tenido más éxito entre las herramientas para la toma de decisiones. Así mismo, es quizá la técnica mas prometedora y con más futuro en la solución de problemas cada vez más complicados.

El poder de la simulación radica en su misma esencia. Las principales características de esta técnica son:

a) Para aplicar la simulación es necesario construir un modelo del problema. Esto marca una diferencia muy grande entre la simulación y las demás técnicas, ya que no se expresa el problema en términos de la técnica que se va a utilizar, sino que el modelo se construye de manera de conservar las características relevantes de la situación problemática.

b) La simulación no trata de optimizar alguna medida. El analista construye un modelo con un propósito y al simular, observará el comportamiento de éste, esperando que la realidad se comporte de manera similar. De esta forma, tanto la conducción de la simulación como las conclusiones que de ésta se deriven corren por cuenta del analista. Es tan sólo una herramienta que el tomador de decisiones usará para complementar su intuición, buen juicio y experiencia. Aquí radica su poder y, por lo tanto su dificultad. Un modelo de simulación construído sin un propósito o manejarlo sin inteligencia será de muy poca o ninguna utilidad en el análisis de una situación problemática. Los objetivos se manejan fuera del modelo y no lo caracterizan y lo restringen, dándole mayor libertad de representar situaciones problemáticas más realistas.

En esta sección se presentará un modelo de simulación de líneas de espera y varios programas que generan variables aleatorias de las distribuciones de probabilidad más importantes.

4.10.1.- I01001. Simulación de una Línea de Espera. Cola Infinita, Fuente Infinita, s Servicios, Llegadas Poisson, Servicio Exponencial.

Se refiere al lector a la descripción del programa I00901 para la solucion analitica de este modelo.

El programa acepta los siguientes datos:

maxt : tiempo de la simulacion.(real)
muit : tiempo promedio entre llegadas.(real)
must : tiempo promedio de servicio.(real)
s : número de servicios.(entero)
swmon : switch del monitor (0=muestra sólo resultados finales. 1=muestra resultados cada vez que hay un evento).(entero)
swran : switch para secuencia de números aleatorios. (0=utiliza la misma secuencia cada corrida, 1=diferente secuencia cada corrida).(entero)

Y produce las siguientes cantidades:

clock : último tiempo en la simulación.(real)
totarr : total de llegadas.(entero)
totq : total de clientes que entraron a la línea de espera.(entero)
maxq : longitud máxima de la línea de espera.(entero)
tis : tiempo total en el sistema.(real)
tisf : tiempo total en servicio(s).(real)
twt : tiempo total en espera.(real)
atis : tiempo promedio en el sistema.(real)
atisf : tiempo promedio en servicio.(real)
atiq : tiempo promedio en línea de espera.(real)
atneq : tiempo promedio en línea de espera para los clientes que entraron a ésta.(real)
idle[i] : tiempo ocioso del servicio i.(real)
totidle : tiempo ocioso total del sistema.(real)
potb : porcentaje de tiempo que el servicio estuvo activo.(real)

Ejemplo: Utilizando los datos del ejemplo de la sección 4.9.1, tenemos lo siguiente:

lambda = 7
mu = 9
s = 3

Expresando los datos de manera adecuada para el programa:

maxt = 100 (hrs)
muit = .1429 (=1/7)

```
  must = .1111 (=1/9)
  s = 3
  swmon = 0
  swran = 0
```

obtenemos la siguiente solución:

```
  clock = 100.39
  totarr = 715
  totq = 43
  maxq = 4
  tis = 82.1262
  tisf = 79.3613
  twt = 2.76498
  atis = .114862
  atisf = .110995
  atiq = .00387
  atneq = .0643
  idle = [55.79 75.17 89.8]
  totidle = 220.757
  potb = 26.44%
```

```
 1:  program Lu1001;
 2:  uses APPLESTUFF,TRANSCEND;

 3:  type
 4:    MAT1DIM=array [1..10] of real;

 5:  var
 6:    MAXT,MUIT,MUST,CLOCK,TWT : real;
 7:    TOTIDLE,TIS,TISF,ATIS,ATISF,
 8:      ATIQ,ATNEQ,POTB : real;
 9:    IDLE : MAT1DIM;
10:    S,SWHON,SWRAN,TOTARR,TOTQ,MAXQ : integer;
11:    DispSal : text;

12:  procedure IniDispSal;
13:  var
14:    NomDispSal : string;

15:  begin
16:    write('Dispositivo de Salida=');
17:    readln(NomDispSal);
18:    rewrite(DispSal,NomDispSal);

19:  end;

20:  procedure RESULTADOS(CLOCK,MUIT,MUST,TWT : real;
21:                        S,TOTARR,TOTQ,MAXQ : integer;
22:                        IDLE : MAT1DIM;
23:                        TOTIDLE,TIS,TISF,ATIS,
24:                          ATISF,ATIQ,ATNEQ,POTB : real);
25:  var
26:    I : integer;

27:  begin
28:    writeln(DispSal);
29:    writeln(DispSal,'RESULTADOS:');
30:    writeln(DispSal);
31:    writeln(DispSal,
32:      'TIEMPO DE SIMULACION          =',CLOCK);
33:    writeln(DispSal,
34:      'TIEMPO PROMEDIO ENTRE LLEGADAS=',MUIT);
35:    writeln(DispSal,
36:      'TIEMPO PROMEDIO DE SERVICIO   =',MUST);
37:    writeln(DispSal,
38:      'NUMERO TOTAL DE LLEGADAS      =',TOTARR);
39:    writeln(DispSal,
40:      'NUM DE CLIENTES QUE ESPERARON =',TOTQ);
41:    writeln(DispSal,
```

```
42:       'LONGITUD MAXIMA LINEA DE ESP  =',MAXQ);
43:     writeln(DispSal,
44:       'TIEMPO TOTAL EN EL SISTEMA    =',TIS);
45:     writeln(DispSal,
46:       'TIEMPO TOTAL EN SERVICIO      =',TISF);
47:     writeln(DispSal,
48:       'TIEMPO TOTAL EN LINEA DE ESP  =',TWT);
49:     writeln(DispSal,
50:       'TIEMPO PROMEDIO EN SISTEMA    =',ATIS);
51:     writeln(DispSal,
52:       'TIEMPO PROMEDIO EN SERVICIO   =',ATISF);
53:     writeln(DispSal,
54:       'TIEMPO PROMEDIO EN LINEA      =',ATIQ);
55:     writeln(DispSal,
56:       'T.P.E.L PARA LOS QUE SE FORMAN=',ATNEQ);
57:     for I:=1 to S do
58:       begin
59:         writeln(DispSal,'TIEMPO OCIOSO EN ',
60:           'SERVICIO ',I,'=',IDLE[I]);
61:       end;
62:     writeln(DispSal,
63:       'TIEMPO TOTAL SISTEMA OCIOSO   =',TOTIDLE);
64:     writeln(DispSal,
65:       'PCT DE TIEMPO SERVICIO ACTIVO =',POTB);

66:   end;

67:   procedure CALCPARS(CLOCK,TWT : real;
68:                      S,TOTARR,TOTQ,MAXQ : integer;
69:                      IDLE : MAT1DIM;
70:                      var TOTIDLE,TIS,TISF,ATIS
71:                        ,ATISF,ATIQ,ATNEQ,POTB : real);
72:   var
73:     I : integer;

74:   begin
75:     TISF:=S*CLOCK;
76:     for I:=1 to S do
77:       begin
78:         TOTIDLE:=TOTIDLE+IDLE[I];
79:         TISF:=TISF-IDLE[I];
80:       end;
81:     TIS:=TISF+TWT;
82:     ATIS:=TIS/TOTARR;
83:     ATISF:=TISF/TOTARR;
84:     ATIQ:=TWT/TOTARR;
85:     ATNEQ:=TWT/TOTQ;
86:     POTB:=TISF/(S*CLOCK);
```

```
 87:   end;

 88:   procedure SIMULA(MAXT,MU11,MUST : real;
 89:                    S,SWMON : integer;
 90:                    var TOTARR,TOTQ,MAXQ : integer;
 91:                    var CLOCK,TWT : real;
 92:                    var IDLE : MAT1DIM);
 93:   var
 94:     I,SWDESC,SWNSERV,QUEUE : integer;
 95:     CLOCKANT : real;
 96:     SALTIM : MAT1DIM;

 97:     function GENTIM(MU : real) : real;
 98:       function RANDM : real;
 99:       var
100:         RAND,MX,C,D : integer;

101:       begin
102:         C:=10000;
103:         MX:=(MAXINT-9999) div C + 1;
104:         MX:=MX*(9999)+(MX-1);
105:         repeat
106:           D:=RANDOM;
107:         until D<=MX;
108:         RAND:=1+D mod C;
109:         RANDM:=RAND/10001;

110:       end;

111:     begin
112:       GENTIM:=(-MU)*LN(RANDM);

113:     end;

114:     procedure MONITR;
115:     var
116:       I : integer;

117:     begin
118:       writeln(DispSal);
119:       writeln(DispSal,'**************************');
120:       writeln(DispSal,'*** MONITOR DEL SISTEMA ***');
121:       writeln(DispSal);
122:       writeln(DispSal,'RELOJ=',CLOCK);
123:       writeln(DispSal,'RELOJ ANT=',CLOCKANT);
124:       writeln(DispSal,
125:         'TOTAL DE LLEGADAS=',TOTARR);
```

```
127:        writeln(DispSal,
128:          'TOTAL DE LLEGADAS QUE ESPERA N');
129:        writeln(DispSal,' A LA LINEA =',TOTQ);
130:        writeln(DispSal,
131:          'EN LINEA DE ESPERA=',QUEUE);
132:        writeln(DispSal,
133:          'LONGITUD MAXIMA DE L.E.=',MAXQ);
134:        writeln(DispSal,
135:          'TOTAL TIEMPO DE ESPERA=',TWT);
136:        for I:=1 to S do
137:          begin
138:            writeln(DispSal,
139:              'HORA EN QUE SE DESOCUPARA O SE');
140:            writeln(DispSal,
141:              'DESOCUPO EL SERVICIO ',I,' =',SALTIM[I]);
142:            writeln(DispSal,'TIEMPO OCIOSO DEL ',
143:              'SERVICIO ',I,' =',IDLE[I]);
144:          end;

145:      end;

146:      procedure ENTLINEA(CLOCK,CLOCKANT : real;
147:                         var QUEUE,TOTQ,MAXQ : integer;
148:                         var TWT : real);

149:      begin
150:        TWT:=TWT+(CLOCK-CLOCKANT)*QUEUE;
151:        TOTQ:=TOTQ+1;
152:        QUEUE:=QUEUE+1;
153:        if QUEUE>MAXQ then
154:          MAXQ:=QUEUE;

155:      end;

156:      procedure ENTSERV(CLOCK : real;
157:                        var SALTIM,IDLE : MATIDIM;
158:                        var OWNSERV : integer);
159:      var
160:        I : integer;

161:      begin
162:        OWNSERV:=1;
163:        I:=0;
164:        repeat
165:          I:=I+1;
166:          if SALTIM[I]<=CLOCK then
167:            begin
168:              OWNSERV:=0;
```

```
168:              IDLE[I]:=IDLE[I]+(CLOCK-SALTIM[I]);
169:              SALTIM[I]:=CLOCK+GENTIM(MUST);
170:            end;
171:        until (I=S) or (SWNSERV=0);

172:      end;

173:      procedure DESCARGA(CLOCK,CLOCKANT : real;
174:                         var S,SWMON,QUEUE : integer;
175:                         var SALTIM : MAT1DIM
176:                         var TWT : real);
177:      var
178:        I,SWDESC : integer;

179:      begin
180:        repeat
181:          SWDESC:=0;
182:          I:=0;
183:          while (QUEUE<>0) and (I<S) do
184:            begin
185:              I:=I+1;
186:              if SALTIM[I]<CLOCK then
187:                begin
188:                  TWT:=TWT+(SALTIM[I]-CLOCKANT);
189:                  QUEUE:=QUEUE-1;
190:                  SALTIM[I]:=SALTIM[I]+GENTIM(MUST);
191:                  SWDESC:=1;
192:                end;
193:              if SWMON=1 then
194:                MONITR;
195:            end;
196:        until SWDESC=0;

197:      end;

198:      procedure SLTMINI(S : integer;
199:                        var SALTIM : MAT1DIM);
200:      var
201:        I : integer;

202:      begin
203:        for I:=1 to S do
204:          begin
205:            SALTIM[I]:=0;
206:          end;

207:      end;
```

```
208:   begin
209:     QUEUE:=0;
210:     SLTMINI(S,SALTIM);
211:     repeat
212:       CLOCKANT:=CLOCK;
213:       CLOCK:=GENTIM(MUIT)+CLOCK;
214:       TOTARR:=TOTARR+1;
215:       SWDESC:=0;
216:       DESCARGA(CLOCK,CLOCKANT,S,SWMON,
217:         QUEUE,SALTIM,TWT);
218:       if QUEUE=0 then
219:         begin
220:           ENTSERV(CLOCK,SALTIM,IDLE,SWNSERV);
221:           if SWNSERV=1 then
222:             begin
223:               ENTLINEA(CLOCK,CLOCKANT,
224:                 QUEUE,TOTQ,MAXQ,TWT);
225:             end;
226:         end
227:       else
228:         begin
229:           ENTLINEA(CLOCK,CLOCKANT,
230:             QUEUE,TOTQ,MAXQ,TWT);
231:         end;
232:       if SWMON=1 then
233:         MONITR;

234:     until CLOCK>MAXT;

235:   end;

236:   procedure LECTURA(var MAXT,MUIT,MUST : real;
237:                     var S,SWMON,SWRAN : integer);
238:   begin
239:     writeln(DispSal);
240:     writeln(DispSal,'IO1001. SIMULACION ',
241:       'DE 1 LINEA DE ESPERA');
242:     writeln(DispSal,' S SERVICIOS. TIEMPOS ',
243:       'EXPONENCIALES DE LLEGADAS');
244:     writeln(DispSal,' Y DE SERVICIO');
245:     writeln(DispSal);
246:     write(DispSal,'TIEMPO DE LA SIMULACION=');
247:     readln(MAXT);
248:     writeln(DispSal,MAXT);
249:     write(DispSal,'TIEMPO PROMEDIO ENTRE LLEGADAS=');
250:     readln(MUIT);
251:     writeln(DispSal,MUIT);
252:     write(DispSal,'TIEMPO PROMEDIO DE SERVICIO   =');
```

```
253:      readln(MUST);
254:      writeln(DispSal,MUST);
255:      write(DispSal,'NUMERO DE SERVICIOS ACTIVOS   =');
256:      readln(S);
257:      writeln(DispSal,S);
258:      write(DispSal,'MONITOR (1/0)=');
259:      readln(SWMON);
260:      writeln(DispSal,SWMON);
261:      write(DispSal,'RANDOMIZE (1/0)=');
262:      readln(SWRAN);
263:      writeln(DispSal,SWRAN);

264:    end;

265:    procedure INICIALIZA;
266:    var
267:      I : integer;

268:    begin
269:      CLOCK:=0;
270:      for I:=1 to 10 do
271:        begin
272:          IDLE[I]:=0;
273:        end;
274:      TOTARR:=0;
275:      TOTQ:=0;
276:      TWT:=0;
277:      TOTIDLE:=0;
278:      S:=0;
279:      MAXQ:=0;
280:      SWMON:=0;

281:    end;

282:    begin
283:      IniDispSal;
284:      INICIALIZA;
285:      LECTURA(MAXT,MUIT,MUST,S,SWMON,SWRAN);
286:      if SWRAN=1 then
287:        RANDOMIZE;
288:      SIMULA(MAXT,MUIT,MUST,S,SWMON,TOTARR,
289:        TOTQ,MAXQ,CLOCK,TWT,IDLE);
290:      CALCPARS(CLOCK,TWT,S,TOTARR,TOTQ,MAXQ,IDLE,
291:        TOTIDLE,TIS,TISF,ATIS,ATISF,ATIQ,ATNEQ,POTB);
292:      RESULTADOS(CLOCK,MUIT,MUST,TWT,S,TOTARR,TOTQ,
293:        MAXQ,IDLE,TOTIDLE,TIS,TISF,ATIS,ATISF,ATIQ,
294:        ATNEQ,POTB);
295:      close(DispSal,lock);
```

296: end.

```
 1:   I01001. SIMULACION DE 1 LINEA DE ESPERA
 2:    S SERVICIOS. TIEMPOS EXPONENCIALES DE LLEGADAS
 3:    Y DE SERVICIO

 4:   TIEMPO DE LA SIMULACION= 1.00000E2
 5:   TIEMPO PROMEDIO ENTRE LLEGADAS= 1.42900E-1
 6:   TIEMPO PROMEDIO DE SERVICIO   = 1.11100E-1
 7:   NUMERO DE SERVICIOS ACTIVOS   =3
 8:   MONITOR (1/0)=0
 9:   RANDOMIZE (1/0)=0

10:   RESULTADOS:

11:   TIEMPO DE SIMULACION          = 1.00039E2
12:   TIEMPO PROMEDIO ENTRE LLEGADAS= 1.42900E-1
13:   TIEMPO PROMEDIO DE SERVICIO   = 1.11100E-1
14:   NUMERO TOTAL DE LLEGADAS      =715
15:   NUM DE CLIENTES QUE ESPERARON =43
16:   LONGITUD MAXIMA LINEA DE ESP  =4
17:   TIEMPO TOTAL EN EL SISTEMA    = 8.21262E1
18:   TIEMPO TOTAL EN SERVICIO      = 7.93613E1
19:   TIEMPO TOTAL EN LINEA DE ESP  = 2.76498
20:   TIEMPO PROMEDIO EN SISTEMA    = 1.14862E-1
21:   TIEMPO PROMEDIO EN SERVICIO   = 1.10995E-1
22:   TIEMPO PROMEDIO EN LINEA      = 3.86711E-3
23:   T.P.E.L PARA LOS QUE SE FORMAN= 6.43020E-2
24:   TIEMPO OCIOSO EN SERVICIO 1= 5.57919E1
25:   TIEMPO OCIOSO EN SERVICIO 2= 7.51671E1
26:   TIEMPO OCIOSO EN SERVICIO 3= 8.97976E1
27:   TIEMPO TOTAL SISTEMA OCIOSO   = 2.20757E2
28:   PCT DE TIEMPO SERVICIO ACTIVO = 2.64434E-1
```

4.10.2.- IQ1002. Variables Aleatorias de una Distribución Exponencial.

Este programa genera valores de una variable aleatoria cuya función de distribución es Exponencial:

$$f(x) = (1/mu)*exp(-(x/mu)), \quad x>0.$$

El programa acepta los siguientes datos:

n : número de valores.(entero)
mu : media de la distribución.(real)

Y produce las siguientes cantidades:

x : valor muestral de la variable aleatoria.(real)

Ejemplo: Si mu = 9, obtener 10 valores de la variable aleatoria.

Expresando los datos de manera adecuada para el programa:

mu = 9

obtenemos la siguiente solución:

x = .14179
x = .04155
x = .23780
x = .12527
x = .00341
x = .00982
x = .14058
x = .05193
x = .00337
x = .18720

```
 1:  program IO1002;
 2:  uses APPLESTUFF,TRANSCEND;

 3:  var
 4:    I,N : integer;
 5:    MU,X : real;
 6:    DispSal : text;

 7:  procedure IniDispSal;
 8:  var
 9:    NomDispSal : string;

10:  begin
11:    write('Dispositivo de Salida=');
12:    readln(NomDispSal);
13:    rewrite(DispSal,NomDispSal);

14:  end;

15:  function RANDM : real;
16:  var
17:    RND : real;

18:  begin
19:    RND:=RANDOM;
20:    RANDM:=(RND*3.0-3.0)/100000.0;

21:  end;

22:  function FEXP(MU : real) : real;
23:  begin
24:    FEXP:=-1/MU*LN(RANDM);

25:  end;

26:  begin
27:    IniDispSal;
28:    writeln(DispSal);
29:    writeln(DispSal,'IO1002. VARIABLES ',
30:      'ALEATORIAS EXPONENCIALES');
31:    writeln(DispSal);
32:    write(DispSal,'NUMERO DE VALORES=');
33:    readln(N);
34:    writeln(DispSal,N);
35:    write(DispSal,'MEDIA=');
36:    readln(MU);
37:    writeln(DispSal,MU);
38:    writeln(DispSal);
```

```
39:     writeln(DispSal,'VALORES:');
40:     for I:=1 to N do
41:       begin
42:         X:=FEXP(MU);
43:         writeln(DispSal,X);
44:       end;
45:     close(DispSal,lock);

46:   end.
```

```
 1:   IO1002. VARIABLES ALEATORIAS EXPONENCIALES

 2:   NUMERO DE VALORES=10
 3:   MEDIA= 9.00000

 4:   VALORES:
 5:    1.41790E-1
 6:    4.15534E-2
 7:    2.37801E-1
 8:    1.25275E-1
 9:    3.40956E-3
10:    9.81488E-2
11:    1.40579E-1
12:    5.19261E-2
13:    3.37176E-3
14:    1.87195E-1
```

4.10.3.- IQ1003. Variables Aleatorias de una Distribución Uniforme.

Este programa genera valores de una variable aleatoria cuya función de distribucion es Uniforme:

f(x) = 1/(b - a), a <= x <= b.

El programa acepta los siguientes datos:

n : número de valores.(entero)
a : valor izquierdo del intervalo [a,b].(real)
b : valor derecho del intervalo [a,b].real)

Y produce las siguientes cantidades:

x : valor muestral de la variable aleatoria.(real)

Ejemplo: Si a=0 y b=1, obtener 10 valores de la variable aleatoria.

Expresando los datos de manera adecuada para el programa:

a = 0
b = 1

obtenemos la siguiente solución:

x = .27912
x = .68799
x = .11763
x = .32385
x = .96978
x = .41340
x = .28218
x = .62667
x = .97011
x = .18549

```
 1:  program [illegible];
 2:  uses [illegible]RANGELIND;

 3:  var
 4:    [illegible] : integer;
 5:    [illegible] : real;
 6:    DispSal : text;

 7:  procedure IniDispSal;
 8:  var
 9:    NomDispSal : string;

10:  begin
11:    write('Dispositivo de Salida=');
12:    readln(NomDispSal);
13:    rewrite(DispSal,NomDispSal);

14:  end;

15:  function RANDM : real;
16:  var
17:    RND : real;

18:  begin
19:    RND:=RANDOM;
20:    RANDM:=(RND*3.0-3.0)/100000.0;

21:  end;

22:  function UNIFORME(A,B : real) : real;
23:  begin
24:    UNIFORME:=[illegible](B-A)*RANDM;

25:  end;

26:  begin
27:    IniDispSal;
28:    writeln(D[illegible]);
29:    writeln(DispSal,'IOI003. VARIABLES ',
30:      'ALEATORIAS UNIFORMES');
31:    writeln(D[illegible]);
32:    write(DispSal,' NUMERO DE VALORES=');
33:    [illegible](N);
34:    [illegible];
35:    [illegible];
36:    [illegible];
37:    [illegible];
38:    [illegible];
```

```
39:     readln(B);
40:     writeln(DispSal,B);
41:     writeln(DispSal);
42:     writeln(DispSal,'VALORES:');
43:     for I:=1 to N do
44:       begin
45:         X:=UNIFORME(A,B);
46:         writeln(DispSal,X);
47:       end;
48:     close(DispSal,lock);

49:   end.
```

```
 1:   IO1003. VARIABLES ALEATORIAS UNIFORMES

 2:   NUMERO DE VALORES=10
 3:   A= 0.00000
 4:   B= 1.00000

 5:   VALORES:
 6:    2.79120E-1
 7:    6.87990E-1
 8:    1.17630E-1
 9:    3.23850E-1
10:    9.69780E-1
11:    4.13400E-1
12:    2.82180E-1
13:    6.26670E-1
14:    9.70110E-1
15:    1.85490E-1
```

4.10.4.- IO1004. Variables Aleatorias de una Distribución Erlang.

Este programa genera valores de una variable aleatoria cuya función de distribución es Erlang (suma de k variables aleatorias Exponenciales con media mu):

$$f(x) = \frac{(k*mu)^k * x^{k-1} * \exp(-k*mu*x)}{(k-1)!}$$

El programa acepta los siguientes datos:

n : número de valores.(entero)
k : número de variables aleatorias independientes.(entero)
mu : media de cada variable aleatoria.(real)

Y produce las siguientes cantidades:

x : valor muestral de la variable aleatoria.(real)

Ejemplo: Si k=2 y mu=9, obtener 10 valores de la variable aleatoria.

Expresando los datos de manera adecuada para el programa:

k = 2
mu = 9

obtenemos la siguiente solución:

x = .09167
x = .18154
x = .05078
x = .09625
x = .09528
x = .19571
x = .09497
x = .05819
x = .05373
x = .03815

```
 1:  program TD10B3;
 2:  uses APPLESTUFF,TRANSCEND;

 3:  var
 4:    N,I,K : integer;
 5:    MU,X : real;
 6:    DispSal : text;

 7:  procedure IniDispSal;
 8:  var
 9:    NomDispSal : string;

10:  begin
11:    write('Dispositivo de Salida=');
12:    readln(NomDispSal);
13:    rewrite(DispSal,NomDispSal);

14:  end;

15:  function RANDM : real;
16:  var
17:    RND : real;

18:  begin
19:    RND:=RANDOM;
20:    RANDM:=(RND*3.0-3.0)/100000.0;

21:  end;

22:  function ERLANG(K : integer;
23:                    MU : real) : real;
24:  var
25:    I : integer;
26:    PROD : real;

27:  begin
28:      PROD:=1;
29:      for I:=1 to K do
30:        begin
31:          PROD:=PROD*RANDM;
32:        end;
33:      ERLANG:=-1.0/(MU*K)*LN(PROD);
34:  end;

35:  begin
36:    IniDispSal;
37:    writeln(DispSal);
38:    writeln(DispSal,'TD10B3, VARIABLES ',
```

```
39:        'ALEATORIAS ERLANG');
40:      write(DispSal,'NUMERO DE VALORES=');
41:      readln(N);
42:      writeln(DispSal,N);
43:      writeln(DispSal);
44:      write(DispSal,'K=');
45:      readln(K);
46:      writeln(DispSal,K);
47:      write(DispSal,'MU=');
48:      readln(MU);
49:      writeln(DispSal,MU);
50:      writeln(DispSal);
51:      writeln(DispSal,'VALORES:');
52:      for I:=1 to N do
53:        begin
54:          X:=ERLANG(K,MU);
55:          writeln(DispSal,X);
56:        end;
57:      close(DispSal,lock);

58:    end.
```

```
 1:    I01004. VARIABLES ALEATORIAS ERLANG
 2:    NUMERO DE VALORES=10

 3:    K=2
 4:    MU= 9.00000

 5:    VALORES:
 6:     9.16719E-2
 7:     1.81538E-1
 8:     5.07792E-2
 9:     9.62525E-2
10:     9.52833E-2
11:     1.95705E-1
12:     9.49748E-2
13:     5.81861E-2
14:     5.37349E-2
15:     3.81493E-2
```

4.10.5.- I01005. Variables aleatorias de una distribucion Binomial.

Este programa genera valores de una variable aleatoria cuya función de distribución es Binomial:

$$f(x) = (n\ x) * p^{**}x * (1 - p)^{**}(n-x),\ x=0,1..n,\ 0<p<1,$$

(n x) significa combinaciones de n en x.

El programa acepta los siguientes datos:

m : número de valores.(entero)
n : número de experimentos.(entero)
p : probabilidad de un éxito..(real)

Y produce las siguientes cantidades:

x : valor muestral de la variable aleatoria.(real)

Ejemplo: Si n=12 y p=.3, obtener 10 valores de la variable aleatoria.

Expresando los datos de manera adecuada para el programa:

n = 12
p = .3

obtenemos la siguiente solución:

x = 5
x = 3
x = 4
x = 5
x = 2
x = 3
x = 2
x = 6
x = 7
x = 2

```
 1:  Program [illegible];
 2:  uses [illegible] CRT;

 3:  var
 4:    [illegible]
 5:    [illegible] : integer;
 6:    [illegible] : text;

 7:  procedure InDispSal;
 8:  var
 9:    NombreSal : string;

10:  begin
11:    write('Dispositivo de Salida=');
12:    readln(NombreSal);
13:    rewrite([illegible]NombreSal);

14:  end;

15:  function RANDOM : real;
16:  var
17:    RND : real;

18:  begin
19:    RND:=RANDOM;
20:    RANDOM:=[illegible]*2 - 1)/100000.0;

21:  end;

22:  function [illegible]NOMINA(d : integer;
23:                           P :real) : integer;
24:  var
25:    [illegible] : integer;

26:  begin
27:    [illegible];
28:    for I:=1 to [illegible]
29:      begin
30:        if RANDOM < [illegible]
31:          [illegible]
32:      end;
33:    [illegible]

34:  end;

35:  [illegible]
36:    [illegible];
37:    [illegible]
```

```
38:     writeln(DispSal,'IO1005. VARIABLES '.
39:       'ALEATORIAS BINOMIALES');
40:     writeln(DispSal);
41:     write(DispSal,'NUMERO DE VALORES=');
42:     readln(M);
43:     writeln(DispSal,M);
44:     write(DispSal,'P=');
45:     readln(P);
46:     writeln(DispSal,P);
47:     write(DispSal,'N=');
48:     readln(N);
49:     writeln(DispSal,N);
50:     writeln(DispSal);
51:     writeln(DispSal,'VALORES:');
52:     for I:=1 to M do
53:       begin
54:         X:=BINOMIAL(N,P);
55:         writeln(DispSal,X);
56:       end;
57:     close(DispSal,lock);

58:   end.
```

```
 1:   I01005. VARIABLES ALEATORIAS BINOMIALES

 2:   NUMERO DE VALORES=10
 3:   P= 3.00000E-1
 4:   N=12

 5:   VALORES:
 6:   5
 7:   3
 8:   4
 9:   5
10:   2
11:   3
12:   2
13:   6
14:   7
15:   2
```

4.10.6.- IQ1006. Variables aleatorias de una distribución Normal.

Este programa genera valores de una variable aleatoria cuya función de distribución es Normal:

$$f(x) = (1/(\sigma\sqrt{2\pi})) * \exp(-(1/2)*[(x-\mu)/\sigma]^2).$$

El programa acepta los siguientes datos:

n : número de valores.(entero)
mu : media.(real)
sigma : desviación estándar.(real)
Y produce las siguientes cantidades:

x : valor muestral de la variable aleatoria.(real)

Ejemplo: Si mu=10 y sigma=5, obtener 10 valores de la variable aleatoria.

Expresando los datos de manera adecuada para el programa:

mu = 10
sigma = 5

obtenemos la siguiente solución:

x = 7.4697
x = 14.4650
x = 9.0580
x = 5.0062
x = 9.0652
x = 12.5104
x = 12.6509
x = -.1340
x = .5131
x = 18.4060

```
 1:  program IO1006;
 2:  uses APPLESTUFF,TRANSCEND;

 3:  var
 4:    MU,SIGMA,X : real;
 5:    I,N : integer;
 6:    DispSal : text;

 7:  procedure IniDispSal;
 8:  var
 9:    NomDispSal : string;

10:  begin
11:    write('Dispositivo de Salida=');
12:    readln(NomDispSal);
13:    rewrite(DispSal,NomDispSal);

14:  end;

15:  function RANDM : real;
16:  var
17:    RND : real;

18:  begin
19:    RND:=RANDOM;
20:    RANDM:=(RND*3.0-3.0)/100000.0;

21:  end;

22:  function NORMAL(MU,SIGMA : real) : real;
23:  var
24:    I : integer;
25:    SUM : real;

26:  begin
27:    SUM:=0;
28:    for I:=1 to 12 do
29:      begin
30:        SUM:=SUM+RANDM;
31:      end;
32:    NORMAL:=MU+SIGMA*(SUM-6);

33:  end;

34:  begin
35:    IniDispSal;
36:    writeln(DispSal);
37:    writeln(DispSal,'IO1006, VARIABLES ',
```

```
38:         'ALEATORIAS NORMALES');
39:       writeln(DispSal);
40:       write(DispSal,'CUANTOS NUMEROS=');
41:       readln(N);
42:       writeln(DispSal,N);
43:       write(DispSal,'MU=');
44:       readln(MU);
45:       writeln(DispSal,MU);
46:       write(DispSal,'SIGMA=');
47:       readln(SIGMA);
48:       writeln(DispSal,SIGMA);
49:       writeln(DispSal);
50:       writeln(DispSal,'VALORES:');
51:       for I:=1 to N do
52:         begin
53:           X:=NORMAL(MU,SIGMA);
54:           writeln(DispSal,'X = ',X);
55:         end;
56:       close(DispSal,lock);

57:   end.
```

```
 1:    I01006. VARIABLES ALEATORIAS NORMALES

 2:    CUANTOS NUMEROS=10
 3:    MU= 1.00000E1
 4:    SIGMA= 5.00000

 5:    VALORES:
 6:    X =  7.46965
 7:    X =  1.44655E1
 8:    X =  9.05800
 9:    X =  5.00620
10:    X =  9.06520
11:    X =  1.25104E1
12:    X =  1.26509E1
13:    X = -1.34001E-1
14:    X =  5.13101E-1
15:    X =  1.84060E1
```

TEORIA DE INVENTARIOS

4.11.- I011. Teoría de Inventarios.

Es posible definir un inventario como un nivel de existencia que independiza la producción o compra de un determinado producto de su correspondiente demanda. La decisión que se debe tomar con respecto al inventario de un producto involucra la minimización de todos los costos relevantes: costos de compra o de producir un lote del inventario, costo de hacer un pedido o de producir un producto, costo por faltantes (costo incurrido al no poder vender el producto por no haber existencia). Las variables de decisión consisten en determinar cuándo hacer un pedido para elevar el nivel de inventario y cuánto pedir del producto de tal manera de minimizar el costo total. Estas cantidades dependen de la demanda que, en el caso se presentará, es una variable aleatoria y esta dependencia hace que las mencionadas cantidades sean a su vez variables aleatorias.

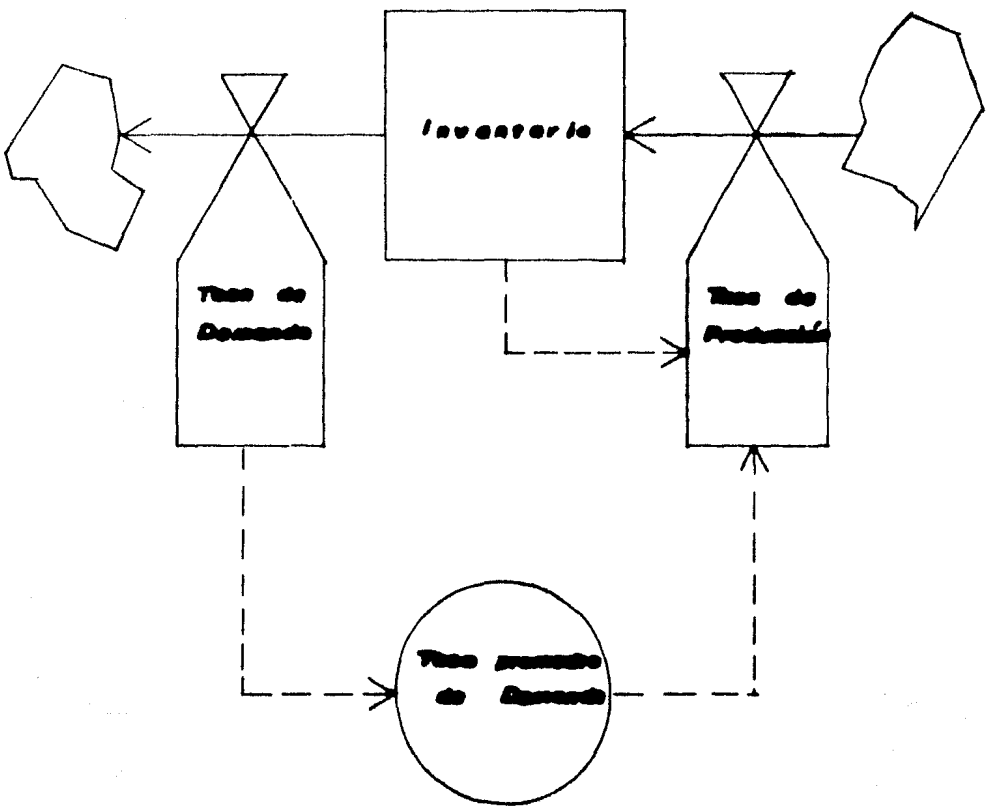


FIGURA 4.12 Nivel de Inventario para Independizar la Demanda de la Producción

4.11.1.- IQ101. 1 Periodo, Punto Fijo de Reorden, Cantidad Fija a Reordenar.

Considere la situación en la que se desea determinar el nivel óptimo de inventario de un producto para una sola temporada (ejemplo: juguetes en Navidad). Suponemos que:

a) No es posible llenar el inventario durante la temporada.
b) Una vez que hacemos el pedido, el inventario llega al nivel deseado inmediatamente.
c) La demanda es una variable aleatoria con función de probabilidad conocida.

El nivel óptimo de inventario será el que minimice el costo total. Es decir, el que busque el equilibrio entre los costos de tener demasiado inventario (y muy poca demanda) o tener muy poco (y demasiada demanda).

El programa acepta los siguientes datos:

```
ioh : inventario inicial.(entero)
mind : mínima demanda.(entero)
maxd : máxima demanda.(entero)
fdp[i] : probabilidad de que la demanda
  sea igual a i.(real)
c0 : costo de hacer un pedido(.real)
c1 : costo de inventario (por unidad).(real)
c3 : costo de compra (por unidad).(real)
c4 : costo por faltantes (por unidad).(real)
  i=mind..maxd.
```

Y produce las siguientes cantidades:

```
dil : nivel óptimo de inventario.(entero)
a : valor crítico para decidir si ordenar o no
  (ordenar si ioh<a).(entero)
o : cantidad a ordenar.(entero)
costo : costo total esperado.(real)
```

Ejemplo: Supóngase que se desea saber si llenar un inventario o no para un periodo. Se han determinado que los costos son: c0=10, c1=1, c3=5, c4=10 y se tiene un inventario inicial de 10 unidades. Asimismo, se ha estimado que la demanda sigue la siguiente distribución:

x	11	12	13	14	15	16	17	18	19	20
f(x)	.10	.05	.05	.20	.15	.05	.10	.10	.10	.10

Expresando los datos en forma adecuada para el programa:

```
ioh = 10
mind = 11
maxd = 20

fdp = [.10 .05 .05 .20 .15
       .05 .10 .10 .10 .10]
c0 = 10
c1 = 1
c3 = 5
c4 = 10
```

obtenemos la siguiente solución:

```
dil = 15
a = 11
q = 5 (ya que a>ioh)
costo = 50.35
```

```
 1:  program IO1101;
 2:  type
 3:    MAT1DIM = array [1..100] of real;

 4:  var
 5:    IOH,MIND,MAXD,DIL,A,Q : integer;
 6:    C0,C1,C3,C4,COSTO : real;
 7:    FDP,FCP : MAT1DIM;
 8:    DispSal : text;

 9:  procedure IniDispSal;
10:  var
11:    NomDispSal : string;

12:  begin
13:    write('Dispositivo de Salida=');
14:    readln(NomDispSal);
15:    rewrite(DispSal,NomDispSal);

16:  end;

17:  procedure RESULTADOS(A,DIL,Q : integer;
18:                       COSTO : real);

19:  begin
20:    writeln(DispSal);
21:    writeln(DispSal,'RESULTADOS:');
22:    writeln(DispSal);
23:    writeln(DispSal,'NIVEL DE INVENTARIO '.
24:      'OPTIMO=',DIL);
25:    writeln(DispSal,'VALOR CRITICO PARA '.
26:      'ORDENAR O NO=',A);
27:    writeln(DispSal,'CANTIDAD A REORDENAR=',Q);
28:    writeln(DispSal,'COSTO ESPERADO=',COSTO);

29:  end;

30:  procedure CALCOPT(IOH,MIND,MAXD : integer;
31:                     FDP,FCP : MAT1DIM;
32:                     C0,C1,C3,C4 : real;
33:                     var COSTO : real;
34:                     var A,DIL,Q : integer);

35:    procedure CALCA(C0,C1,C3,C4,COSTO : real;
36:                     FDP : MAT1DIM;
37:                     DIL : integer;
38:                     var A : integer);
39:    var
```

```
40:       I,J : integer;
41:       MUA,SUM : real;

42:     begin
43:       A:=DIL;
44:       repeat
45:         A:=A-1;
46:         MUA:=C3*(A-IOH);
47:         SUM:=0;
48:         for I:=MIND to A do
49:           begin
50:             J:=I-MIND+1;
51:             SUM:=SUM+(A-I)*FDP[J];
52:           end;
53:         MUA:=MUA+C1*SUM;
54:         SUM:=0;
55:         for I:=A to MAXD do
56:           begin
57:             J:=I-MIND+1;
58:             SUM:=SUM+(I-A)*FDP[J];
59:           end;
60:         MUA:=MUA+C4*SUM;
61:       until MUA>COSTO;

62:     end;

63:     procedure CALCCOS(C0,C1,C3,C4 : real;
64:                       FDP : MAT1DIM;
65:                       DIL,IOH : integer;
66:                       var COSTO : real);
67:     var
68:       I,J : integer;
69:       SUM : real;

70:     begin
71:       COSTO:=C0+C3*(DIL-IOH);
72:       SUM:=0;
73:       for I:=MIND to DIL do
74:         begin
75:           J:=I-MIND+1;
76:           SUM:=SUM+(DIL-I)*FDP[J];
77:         end;
78:       COSTO:=COSTO+C1*SUM;
79:       SUM:=0;
80:       for I:=DIL to MAXD do
81:         begin
82:           J:=I-MIND+1;
83:           SUM:=SUM+(I-DIL)*FDP[J];
```

```
 84:         end;
 85:       COSTO:=COSTO+C4*SUM;

 86:     end;

 87:     procedure CALCDIL(MIND,MAXD : integer;
 88:                       C1,C3,C4 : real;
 89:                       FCP : MAT1DIM;
 90:                       var DIL : integer);
 91:     var
 92:       I,J : integer;
 93:       C,DILR : real;

 94:     begin
 95:       C:=(C4-C3)/(C1+C4);
 96:       DILR:=0;
 97:       I:=MIND;
 98:       J:=I-MIND+1;
 99:       repeat
100:         if FCP[J]>C then
101:           DILR:=I;
102:         I:=I+1;
103:         J:=J+1;
104:       until (DILR<>0) or (I>MAXD);
105:       DIL:=ROUND(DILR);

106:     end;

107:   begin
108:     CALCDIL(MIND,MAXD,C1,C3,C4,FCP,DIL);
109:     CALCCOS(C0,C1,C3,C4,FDP,DIL,IOH,COSTO);
110:     if C0<>0 then
111:       CALCA(C0,C1,C3,C4,COSTO,FDP,DIL,A)
112:     else
113:       A:=DIL;
114:     if IOH<=A then
115:       begin
116:         Q:=DIL-IOH;
117:       end
118:     else
119:       begin
120:         Q:=0;
121:       end;

122:   end;

123:   procedure LECTURA(var IOH,MIND,MAXD : integer;
```

```
124:                        var FDP,FCP : MATIDIM;
125:                        var C0,C1,C3,C4 : real);
126:   var
127:     I,J : integer;

128:   begin
129:     writeln(DispSal);
130:     writeln(DispSal,'IO1101, INVENTARIOS: ',
131:       '1 PERIODO,PUNTO FIJO DE');
132:     writeln(DispSal,'  REORDEN,CANTIDAD ',
133:       'FIJA DE REORDEN');
134:     writeln(DispSal);
135:     write(DispSal,'INVENTARIO INICIAL=');
136:     readln(IOH);
137:     writeln(DispSal,IOH);
138:     write(DispSal,'MINIMA DEMANDA=');
139:     readln(MIND);
140:     writeln(DispSal,MIND);
141:     write(DispSal,'MAXIMA DEMANDA=');
142:     readln(MAXD);
143:     writeln(DispSal,MAXD);
144:     writeln(DispSal);
145:     writeln(DispSal,'FN DE DENSIDAD DE ',
146:       'PROBABILIDAD:');
147:     writeln(DispSal);
148:     for I:=MIND to MAXD do
149:       begin
150:         J:=I-MIND+1;
151:         write(DispSal,'FDP(',I,')=');
152:         readln(FDP[J]);
153:         writeln(DispSal,FDP[J]);
154:       end;
155:     FCP[1]:=FDP[1];
156:     for I:=MIND+1 to MAXD do
157:       begin
158:         J:=I-MIND+1;
159:         FCP[J]:=FCP[J-1]+FDP[J];
160:       end;
161:     writeln(DispSal);
162:     writeln(DispSal,'COSTOS:');
163:     writeln(DispSal);
164:     write(DispSal,'COSTO DE HACER UN PEDIDO=');
165:     readln(C0);
166:     writeln(DispSal,C0);
167:     write(DispSal,'COSTO DE QUEDARSE CON 1 UNIDAD=');
168:     readln(C1);
169:     writeln(DispSal,C1);
170:     write(DispSal,'COSTO DE PRODUCIR 1 UNIDAD=');
```

```
171:      readln(C3);
172:      writeln(DispSal,C3);
173:      write(DispSal,'COSTO PERDER LA VENTA ',
174:        'DE 1 UNIDAD=');
175:      readln(C4);
176:      writeln(DispSal,C4);

177:    end;

178:    procedure INICIALIZA;

179:    begin
180:      DIL:=0;
181:      A:=0;
182:      Q:=0;

183:    end;

184:    begin
185:      IniDispSal;
186:      INICIALIZA;
187:      LECTURA(IOH,MIND,MAXD,FDP,FCP,C0,C1,C3,C4);
188:      CALCOPT(IOH,MIND,MAXD,FDP,FCP,C0,C1,C3,C4,COSTO,A,DIL,Q);
189:      RESULTADOS(A,DIL,Q,COSTO);
190:      close(DispSal,lock);

191:    end.
```

```
 1:   101101. INVENTARIOS: 1 PERIODO,PUNTO FIJO DE
 2:     REORDEN,CANTIDAD FIJA DE REORDEN

 3:   INVENTARIO INICIAL=10
 4:   MINIMA DEMANDA=11
 5:   MAXIMA DEMANDA=20

 6:   FN DE DENSIDAD DE PROBABILIDAD:

 7:   FDP(11)= 1.00000E-1
 8:   FDP(12)= 5.00000E-2
 9:   FDP(13)= 5.00000E-2
10:   FDP(14)= 2.00000E-1
11:   FDP(15)= 1.50000E-1
12:   FDP(16)= 5.00000E-2
13:   FDP(17)= 1.00000E-1
14:   FDP(18)= 1.00000E-1
15:   FDP(19)= 1.00000E-1
16:   FDP(20)= 1.00000E-1

17:   COSTOS:

18:   COSTO DE HACER UN PEDIDO= 1.00000E1
19:   COSTO DE QUEDARSE CON 1 UNIDAD= 1.00000
20:   COSTO DE PRODUCIR 1 UNIDAD= 5.00000
21:   COSTO PERDER LA VENTA DE 1 UNIDAD= 1.00000E1

22:   RESULTADOS:

23:   NIVEL DE INVENTARIO OPTIMO=15
24:   VALOR CRITICO PARA ORDENAR O NO=11
25:   CANTIDAD A REORDENAR=5
26:   COSTO ESPERADO= 5.03500E1
```

4.12.- I012. Cadenas de Markov.

Suponga que se tiene un sistema que sólo puede estar en uno de un número finito de estados a la vez, que se conocen las probabilidades de transición de un estado a otro y que el estado actual del sistema solamente depende del estado del sistema en el instante anterior. Decimos que este sistema constituye un proceso markoviano de primer orden.

Los programas que se presentan en esta sección analizan el comportamiento de un sistema de este tipo.

4.12.1.- IO1201. N Potencias de la Matriz de Transición y Proporciones Después de n Pasos.

El programa acepta los siguientes datos:

ned : número de estados.(entero)
npt : potencia.(entero)
a[i,j] : probabilidad de pasar del estado i al estado j.(real)
x[j] : proporción inicial del estado j.(real)
i=1..ned, j=1..ned

Y produce las siguientes cantidades:

anpt[i,j] : probabilidad de pasar del estado i al estado j después de npt pasos.(real)
xnpt[j] : proporción del estado j después de npt pasos.(real)
i=1..ned, j=1..ned

Ejemplo: El productor A retiene el 90% de sus clientes y el resto le compra al productor B en el siguiente periodo. El productor B retiene el 80% y pierde el 20% de sus clientes en cada periodo. Si A controla actualmente el 60% del mercado, cual será la proporción de mercado controlada por cada productor después de 15 periodos.

Expresando los datos de manera apropiada para el programa:

```
ned = 2
npt = 15
a = [.9 .1]
    [.2 .8]
x = [.6,.4]
```

obtenemos la siguiente solución:

```
anpt = [.668 .332]
       [.664 .336]
x = [.666 .334]
```

```
 1:  program IDI2001;
 2:  type
 3:    MAT2DIM = array [1..20,1..20] of real;

 4:  var
 5:    NED,MAT : integer;
 6:    A,BMAT : MAT2DIM;
 7:    X,CMAT : MAT2DIM;
 8:    DispSal : text;

 9:  procedure IniDispSal;
10:  var
11:    NomDispSal : string;

12:  begin
13:    write('Dispositivo de Salida=');
14:    readln(NomDispSal);
15:    rewrite(DispSal,NomDispSal);

16:  end;

17:  procedure DESPMAT(N,M : integer;
18:                    A : MAT2DIM);
19:  var
20:    I,J : integer;

21:  begin
22:    for I:=1 to N do
23:      for J:=1 to M do
24:        begin
25:          writeln(DispSal,'A(',I,J,')=',A[I,J]);
26:        end;

27:  end;

28:  procedure MULMAT(N,M,L : integer;
29:                    A,B : MAT2DIM;
30:                    var C : MAT2DIM);
31:  var
32:    I,J,K : integer;

33:  begin
34:    for I:=1 to N do
35:      for J:=1 to L do
36:        begin
37:          C[I,J]:=0;
38:          for K:=1 to M do
39:            begin
```

```
40:                C[I,J]:=C[I,J]+A[I,K]*B[K,J];
41:              end;
42:          end;
43:    end;

44:    procedure POTMAT(N,PO : integer;
45:                     A : MAT2DIM;
46:                     var APO : MAT2DIM);
47:    var
48:      I,J,L : integer;

49:    begin
50:      for I:=1 to N do
51:        for J:=1 to N do
52:          begin
53:            if I<>J then
54:              APO[I,J]:=0
55:            else
56:              APO[I,J]:=1;
57:          end;
58:      for L:=1 to PO do
59:        begin
60:          MULMAT(N,N,N,A,APO,APO);
61:        end;

62:    end;

63:    procedure RESULTADOS(NED,NPT : integer;
64:                         ANPT : MAT2DIM;
65:                         XNPT : MAT2DIM);
66:    begin
67:      writeln(DispSal);
68:      writeln(DispSal,'MATRIZ DE TRANSICION',
69:        ' ELEVADA A LA ',NPT);
70:      DESPMAT(NED,NED,ANPT);
71:      writeln(DispSal);
72:      writeln(DispSal,'PROPORCIONES DESPUES DE ',
73:        NPT,' PASOS');
74:      DESPMAT(1,NED,XNPT);

75:    end;

76:    procedure CALCPO(NED,NPT : integer;
77:                     A : MAT2DIM;
78:                     X : MAT2DIM;
79:                     [illegible] : MAT2DIM;
80:                     [illegible] : MAT2DIM);
```

```
 81:  begin
 82:    POTMAT(NED,N,I,A,ANPT);
 83:    MULMAT(1,NED,NED,X,ANPT,XNPT);

 84:  end;

 85:  procedure LECTURA(var NED,NPT : integer;
 86:                    var A : MATDIM;
 87:                    var X : MATDIM);
 88:  var
 89:    I,J : integer;

 90:  begin
 91:    writeln(DispSal,'IO1201. N POTENCIAS ',
 92:      'DE MAT DE TRANSICION');
 93:    writeln(DispSal,' Y PROPORCIONES ',
 94:      'DESPUES DE N PASOS');
 95:    writeln(DispSal);
 96:    write(DispSal,'NUMERO DE ESTADOS=');
 97:    readln(NED);
 98:    writeln(DispSal,NED);
 99:    write(DispSal,'POTENCIA=');
100:    readln(NPT);
101:    writeln(DispSal,NPT);
102:    writeln(DispSal,'MATRIZ DE TRANSICION:');
103:    for I:=1 to NED do
104:      for J:=1 to NED do
105:        begin
106:          write(DispSal,'A(',I,J,')=');
107:          readln(A[I,J]);
108:          writeln(DispSal,A[I,J]);
109:        end;
110:    writeln(DispSal,'PROPORCIONES INICIALES');
111:    for J:=1 to NED do
112:      begin
113:        write(DispSal,'X(',J,')=');
114:        readln(X[1,J]);
115:        writeln(DispSal,X[1,J]);
116:      end;

117:  end;

118:  procedure INICIALIZA;
119:  var
120:    I,J : integer;

121:  begin
122:    for I:=1 to 25 do
```

```
123:        begin
124:          for J:=1 to 25 do
125:            begin
126:              XNPTII.JJ:=0;
127:              ANPTII.JJ:=0;
128:            end;
129:        end;

130:    end;

131:    begin
132:      IniDispSal;
133:      INICIALIZA;
134:      LECTURA(NED,NPT,A,X);
135:      CALCFOT(NED,NPT,A,X,ANPT,XNPT);
136:      RESULTADOS(NED,NPT,ANPT,XNPT);
137:      close(DispSal,lock);

138:    end.
```

```
 1:   IC1201. N POTENCIAS DE MAT DE TRANSICION
 2:     Y PROPORCIONES DESPUES DE N PASOS

 3:   NUMERO DE ESTADOS=2
 4:   POTENCIA:15
 5:   MATRIZ DE TRANSICION:
 6:   A(11)= 9.00000E-1
 7:   A(12)= 1.00000E-1
 8:   A(21)= 2.00000E-1
 9:   A(22)= 8.00000E-1
10:   PROPORCIONES INICIALES
11:   X(1)= 6.00000E-1
12:   X(2)= 4.00000E-1

13:   MATRIZ DE TRANSICION ELEVADA A LA 15
14:   A(11)= 6.68249E-1
15:   A(12)= 3.31751E-1
16:   A(21)= 6.63502E-1
17:   A(22)= 3.36498E-1

18:   PROPORCIONES DESPUES DE 15 PASOS
19:   A(11)= 6.66350E-1
20:   A(12)= 3.33650E-1
```

4.12.2.- IQ1202. Estado Estable, Tiempo de Recurrencia.

El programa acepta los siguientes datos:

```
ned : número de estados.(entero)
a[i,j] : probabilidad de pasar del estado i
  al estado j.(real)
  i=1..ned,  j=1..ned
```

Y produce las siguientes cantidades:

```
xe[j] : proporción del estado j en
  estado estable.(real)
mu[i,j] : tiempo de recurrencia i=j.(real)
  i=1..ned,  i=1..ned
```

Ejemplo: Mismo ejemplo que 4.12.1.

Expresando los datos en forma adecuada para el programa:

```
ned = 2
a = [.9 .1]
    [.2 .8]
```

obtenemos la siguiente solución:

```
xe = [.667 .333]
mu = [1.5 0.0]
     [0.0 3.0]
```

```
 1:  program IOI202;
 2:  type
 3:    MAT2DIM = array [1..25,1..25] of real;

 4:  var
 5:    NED,SWERROR : integer;
 6:    A,XE,MU : MAT2DIM;
 7:    DispSal : text;

 8:  procedure IniDispSal;
 9:  var
10:    NomDispSal : string;

11:  begin
12:    write('Dispositivo de Salida=');
13:    readln(NomDispSal);
14:    rewrite(DispSal,NomDispSal);

15:  end;

16:  procedure DESPMAT(N,M : integer;
17:                    A : MAT2DIM);
18:  var
19:    I,J : integer;

20:  begin
21:    for I:=1 to N do
22:      for J:=1 to M do
23:        begin
24:          writeln(DispSal,'A(',I,J,')=',A[I,J]);
25:        end;

26:  end;

27:  procedure GSSJRD(N,M : integer;
28:                  A : MAT2DIM;
29:                  var B : MAT2DIM);
30:  var
31:    I,J,K,L,IPIV,JPIV : integer;
32:    FACTOR,MAX : real;

33:    procedure BUSCAMAX(N,M,IACT : integer;
34:                       A : MAT2DIM;
35:                       var MAX : real;
36:                       var IPIV,JPIV : integer);
37:    var
38:      I,J : integer;
```

```
39:     begin
40:       for I:=IACT to N do
41:         for J:=1 to M do
42:           begin
43:             if (J<=N) then
44:               if ABS(A[I,J])>MAX then
45:                 begin
46:                   MAX:=ABS(A[I,J]);
47:                   IPIV:=I;
48:                   JPIV:=J;
49:                 end;
50:           end;

51:     end;

52:     procedure CAMBRENG(N,I,IPIV : integer;
53:                        var A : MAT2DIM);
54:     var
55:       J : integer;
56:       FACTOR : real;

57:     begin
58:       for J:=1 to M do
59:         begin
60:           FACTOR:=A[IPIV,J];
61:           A[IPIV,J]:=A[I,J];
62:           A[I,J]:=FACTOR;
63:         end;

64:     end;

65:   begin
66:     for I:=1 to N do
67:       for J:=1 to M do
68:         B[I,J]:=A[I,J];
69:     for I:=1 to N do
70:       begin
71:         MAX:=0;
72:         IPIV:=0;
73:         JPIV:=0;
74:         BUSCAMAX(N,M,I,B,MAX,IPIV,JPIV);
75:         if (IPIV<>I) and (IPIV<>0) then
76:           CAMBRENG(N,I,IPIV,B);
77:         if JPIV<>0 then
78:           begin
79:             FACTOR:=B[I,JPIV];
80:             for J:=1 to M do
81:               begin
```

```
 82:                 B[I,J]:=B[I,J]/FACTOR;
 83:               end;
 84:             B[I,JPIV]:=1;
 85:             for K:=1 to N do
 86:               begin
 87:                 if K<>I then
 88:                   begin
 89:                     FACTOR:=B[K,JPIV];
 90:                     for L:=1 to n do
 91:                       begin
 92:                         B[K,L]:=B[K,L]-FACTOR*B[I,L];
 93:                       end;
 94:                   end;
 95:               end;
 96:           end;
 97:       end;

 98:   end;

 99:   procedure RESULTADOS(NED : integer;
100:                        A,XE,MU : MAT2DIM;
101:                        SWERROR : integer);
102:   begin
103:     writeln(DispSal);
104:     writeln(DispSal,'MATRIZ DE ',
105:       'TRANSICION ORIGINAL:');
106:     DESPMAT(NED,NED,A);
107:     writeln(DispSal);
108:     writeln(DispSal,'PROPORCIONES EN ',
109:       'EL ESTADO ESTABLE:');
110:     DESPMAT(1,NED,XE);
111:     writeln(DispSal);
112:     writeln(DispSal,'TIEMPOS ESPERADOS ',
113:       'DE RECURRENCIA:');
114:     DESPMAT(NED,NED,MU);
115:     if SWERROR=1 then
116:       writeln(DispSal,'***** NINGUNA O ',
117:         'INFINITAS SOLUCIONES ***');

118:   end;

119:   procedure CALCTMR(NED : integer;
120:                     XE : MAT2DIM;
121:                     var MU : MAT2DIM);
122:   var
123:     I,J : integer;

124:   begin
```

```
125:      for I:=1 to NED do
126:        begin
127:          for J:=1 to NED do
128:            begin
129:              MU[I,J]:=0;
130:            end;
131:          if XE[1,I]>0 then
132:            MU[I,I]:=1/XE[1,I];
133:        end;

134:    end;

135:    procedure CALCEDE(NED : integer;
136:                      B : MAT2DIM;
137:                      var XE : MAT2DIM;
138:                      var SWERROR : integer);
139:    var
140:      I,J,SWASIG : integer;
141:      A: MAT2DIM;

142:    begin
143:      SWERROR:=0;
144:      (* TRASPONE B *)
145:      for I:=1 to NED do
146:        for J:=1 to NED do
147:          begin
148:            A[I,J]:=B[J,I];
149:          end;
150:      (**)
151:      for I:=1 to NED do
152:        begin
153:          A[I,I]:=A[I,I]-1;
154:          A[I,NED+1]:=0;
155:        end;
156:      for J:=1 to NED+1 do
157:        begin
158:          A[1,J]:=1;
159:        end;
160:      GSSJRD(NED,NED+1,A,A);
161:      for I:=1 to NED do
162:        begin
163:          SWASIG:=0;
164:          for J:=1 to NED do
165:            begin
166:              if A[I,J]=1 then
167:                begin
168:                  SWASIG:=1;
169:                  XE[J,J]:=A[I,NED+1];
```

```
170:                end;
171:            end;
172:          if SWASIG=0 then
173:            begin
174:              SWERROR:=1;
175:              for I:=1 to NED do
176:                XL[1,I]:=0;
177:            end;
178:        end;

179:    end;

180:    procedure LECTURA(var NED : integer;
181:                      var A : MAT2DIM);
182:    var
183:      I,J : integer;

184:    begin
185:      writeln(DispSal,'IO1202. ESTADO ESTABLE');
186:      writeln(DispSal);
187:      write(DispSal,'NUMERO DE ESTADOS=');
188:      readln(NED);
189:      writeln(DispSal,NED);
190:      writeln(DispSal,'MATRIZ DE TRANSICION:');
191:      for I:=1 to NED do
192:        for J:=1 to NED do
193:          begin
194:            write(DispSal,'A(',I,J,')=');
195:            readln(A[I,J]);
196:            writeln(DispSal,A[I,J]);
197:          end;

198:    end;

199:    begin
200:      IniDispSal;
201:      LECTURA(NED,A);
202:      CALCEDE(NED,A,XE,SWERROR);
203:      CALCTMR(NED,XE,MU);
204:      RESULTADOS(NED,A,XE,MU,SWERROR);
205:      close(DispSal,lock);

206:    end.
```

```
 1:          E. ESTADO ESTABLE

 2:    NO. DE ESTADOS= 2
 3:    MATRIZ DE TRANSICION:
 4:    A(11)= 9.000000E-1
 5:    A(12)= 1.000000E-1
 6:    A(21)= 2.000000E-1
 7:    A(22)= 8.000000E-1

 8:    MATRIZ DE TRANSICION ORIGINAL:
 9:    A(11)= 9.000000E-1
10:    A(12)= 1.000000E-1
11:    A(21)= 2.000000E-1
12:    A(22)= 8.000000E-1

13:    PROPORCIONES EN EL ESTADO ESTABLE:
14:    A(11)= 6.666667E-1
15:    A(12)= 3.333333E-1

16:    TIEMPOS ESPERADOS DE RECURRENCIA:
17:    A(11)= 1.50000
18:    A(12)= 0.00000
19:    A(21)= 0.00000
20:    A(22)= 3.00000
```

4.12.3.- [01203]. Probabilidad de Pasar del Estado r al Estado s por Primera Vez en n Pasos.

El programa acepta los siguientes datos:

```
ned : numero de estados.(entero)
r : estado inicial.(entero)
s : estado final.(entero)
n : número de pasos.(entero)
a[i,j] : probabilidad de pasar del estado i
  al estado j.(real)
  i=1..ned,  j=1..ned
```

Y produce las siguientes cantidades:

```
prsn : probabilidad de pasar del estado r
  al estado s por primera vez en n pasos.(real)
```

Ejemplo: Mismos datos que 4.12.1.

Expresando los datos en forma apropiada para el programa:

```
ned = 2
r = 2
s = 1
n = 5
a = [.9 .1]
    [.2 .8]
```

obtenemos la siguiente solución:

```
prsn = .08192
```

```
 1:  program IO1203;
 2:  type
 3:    MAT2DIM = array [1..25,1..25] of real;

 4:  var
 5:    NED,K,S,N : integer;
 6:    A : MAT2DIM;
 7:    PRSN : real;
 8:    DispSal : text;

 9:  procedure IniDispSal;
10:  var
11:    NomDispSal : string;

12:  begin
13:    write('Dispositivo de Salida=');
14:    readln(NomDispSal);
15:    rewrite(DispSal,NomDispSal);

16:  end;

17:  procedure DESPMAT(N,M : integer;
18:                    A : MAT2DIM);
19:  var
20:    I,J : integer;

21:  begin
22:    for I:=1 to N do
23:      for J:=1 to M do
24:        begin
25:          writeln(DispSal,'A(',I,J,')=',A[I,JJ);
26:        end;

27:  end;

28:  procedure MULMAT(N,M,L : integer;
29:                    A,B : MAT2DIM;
30:                    var C : MAT2DIM);
31:  var
32:    I,J,K : integer;

33:  begin
34:    for I:=1 to N do
35:      for J:=1 to L do
36:        begin
37:          C[I,J]:=0;
38:          for K:=1 to M do
39:            begin
```

```
40:                C[I,J]:=C[I,J]+A[I,K]*B[K,J];
41:              end;
42:          end; {FOR I:=1 TO N}
43:    end;

44:    procedure LEEMAT(N,M : integer;
45:                     var A : MAT2DIM);
46:    var
47:      I,J : integer;

48:    begin
49:      for I:=1 to N do
50:        for J:=1 to M do
51:          begin
52:            write(DispSal,'A(',I,J,')=');
53:            readln(A[I,J]);
54:            writeln(DispSal,A[I,J]);
55:          end;

56:    end;

57:    procedure RESULTADOS(NED,R,S,N : integer;
58:                         A : MAT2DIM;
59:                         PRSN : real);
60:    begin
61:      writeln(DispSal);
62:      writeln(DispSal,'RESULTADOS:');
63:      writeln(DispSal);
64:      writeln(DispSal,'MATRIZ DE ',
65:        'TRANSICION ORIGINAL:');
66:      DESPMAT(NED,NED,A);
67:      writeln(DispSal);
68:      writeln(DispSal,'PROBABILIDAD DE IR DE EDO ',
69:        R,' A EDO ',S);
70:      writeln(DispSal,' EN ',N,' PASOS = ',PRSN);

71:    end;

72:    function G(NED,R,S,N : integer;
73:               A : MAT2DIM) : real;
74:    var
75:      I,J : integer;
76:      B : MAT2DIM;
77:      SUM : real;

78:    begin
79:      for I:=1 to NED do
80:        for J:=1 to NED do
```

```
 81:          B[I,J]:=A[I,J];
 82:      if N>1 then
 83:        begin
 84:          SUM:=0;
 85:          for I:=N-1 downto 1 do
 86:            begin
 87:              SUM:=SUM+G(NED,R,S,I,A)*B[S,J];
 88:              MULMAT(NED,NED,NED,A,B,B);
 89:            end;
 90:          G:=B[R,J]-SUM;
 91:        end
 92:      else
 93:        G:=B[R,J];

 94:    end;

 95:    procedure LECTURA(var NED,R,S,N : integer;
 96:                      var A : MAT2DIM);
 97:    begin
 98:      writeln(DispSal);
 99:      writeln(DispSal,'ID1203. PROB DE IR DE R A S');
100:      writeln(DispSal,'POR PRIMERA VEZ EN N PASOS');
101:      writeln(DispSal);
102:      write(DispSal,'NUMERO DE ESTADOS=');
103:      readln(NED);
104:      writeln(DispSal,NED);
105:      write(DispSal,'R=');
106:      readln(R);
107:      writeln(DispSal,R);
108:      write(DispSal,'S=');
109:      readln(S);
110:      writeln(DispSal,S);
111:      write(DispSal,'N=');
112:      readln(N);
113:      writeln(DispSal,N);
114:      writeln(DispSal,'MATRIZ DE TRANSICION:');
115:      LEEMAT(NED,NED,A);

116:    end;

117:    procedure INICIALIZA;

118:    begin
119:      PRSN:=0;

120:    end;

121:    begin
```

```
122:      InitDispCol;
123:      INICIALIZA;
124:      LECTURA(MED,R,S,N,A);
125:      PRSN:=G(MED,R,S,N,A);
126:      RESULTADO(MED,R,S,N,A,PRSN);
127:      close(g_bal,lock);

170:   end.
```

```
 1:   ID1203. PROB DE IR DE R A S
 2:   POR PRIMERA VEZ EN N PASOS

 3:   NUMERO DE ESTADOS=2
 4:   R=2
 5:   S=1
 6:   N=5
 7:   MATRIZ DE TRANSICION:
 8:   A(11)= 9.00000E-1
 9:   A(12)= 1.00000E-1
10:   A(21)= 2.00000E-1
11:   A(22)= 8.00000E-1

12:   RESULTADOS:

13:   MATRIZ DE TRANSICION ORIGINAL:
14:   A(11)= 9.00000E-1
15:   A(12)= 1.00000E-1
16:   A(21)= 2.00000E-1
17:   A(22)= 8.00000E-1

18:   PROBABILIDAD DE IR DE EDO 2 A EDO 1
19:    EN 5 PASOS =  8.19200E-2
```

4.12.4.- I01204. Tiempo de Llegada por Primera Vez del Estado i al Estado j. i<>j.

El programa acepta los siguientes datos:

ned : número de estados.(entero)
a[i,j] : probabilidad de pasar del estado i al estado j.(real)
i=1..ned, j=1..ned

Y produce las siguientes cantidades:

mu[i,j] : tiempo de llegada por primera vez desde el estado i al estado j i<>j (véase tiempos de recurrencia).(real)
i=1..ned, j=1..ned

Ejemplo: Mismos datos que 4.12.1.

Expresando los datos en forma apropiada para el programa:

```
ned = 2
a = [.9 .1]
    [.2 .8]
```

obtenemos la siguiente solución:

```
mu = [0 1]
     [5 10]
```

```
 1:  program [illegible];
 2:  type
 3:    MAT2DIM = array [1..25,1..25] of real;

 4:  var
 5:    N,M,[illegible] : integer;
 6:    A,MI : MAT2DIM;
 7:    Disp[illegible] : text;

 8:  procedure IniDispSal;
 9:  var
10:    NomD[illegible] : string;

11:  begin
12:    write('Dispositivo de Salida=');
13:    readln(NomDispSal);
14:    rewrite(DispSal,NomDispSal);

15:  end;

16:  procedure DESPMAT(N,M : integer;
17:                    A : MAT2DIM);
18:  var
19:    I,J : integer;

20:  begin
21:    for I:=1 to N do
22:      for J:=1 to M do
23:        begin
24:          writeln(DispSal,'A(',I,J,')=',A[I,J]);
25:        end;

26:  end;

27:  procedure GSSJRD(N,M : integer;
28:                   A : MAT2DIM;
29:                   var B : MAT2DIM);
30:  var
31:    I,J,K,L,IPIV,JPIV : integer;
32:    FACTOR,MAX : real;

33:    procedure BUSCAMAX(N,M,IACT : integer;
34:                       A : MAT2DIM;
35:                       var MAX : real;
36:                       var JPIV,JPIV : integer);
37:    var
38:      I,J : integer;
```

```
39:     begin
40:       for I:=IACT to N do
41:         for J:=1 to M do
42:           begin
43:             if (J<=N) then
44:               if ABS(A[I,J])>MAX then
45:                 begin
46:                   MAX:=ABS(A[I,J]);
47:                   IPIV:=I;
48:                   JPIV:=J;
49:                 end;
50:           end;

51:     end;

52:     procedure CAMBRENG(M,I,IPIV : integer;
53:                        var A : MAT2DIM);
54:     var
55:       J : integer;
56:       FACTOR : real;

57:     begin
58:       for J:=1 to M do
59:         begin
60:           FACTOR:=A[IPIV,J];
61:           A[IPIV,J]:=A[I,J];
62:           A[I,J]:=FACTOR;
63:         end;

64:     end;

65:   begin
66:     for I:=1 to N do
67:       for J:=1 to M do
68:         B[I,J]:=A[I,J];
69:     for I:=1 to N do
70:       begin
71:         MAX:=0;
72:         IPIV:=0;
73:         JPIV:=0;
74:         BUSCAMAX(N,M,I,B,MAX,IPIV,JPIV);
75:         if (IPIV<>I) and (IPIV<>0) then
76:           CAMBRENG(M,I,IPIV,B);
77:         if JPIV<>0 then
78:           begin
79:             FACTOR:=B[I,JPIV];
80:             for J:=1 to M do
81:               begin
```

```
 82:                 B[I,J]:=B[I,J]/FACTOR;
 83:               end;
 84:             B[I,JPIV]:=1;
 85:             for K:=1 to N do
 86:               begin
 87:                 if K<>I then
 88:                   begin
 89:                     FACTOR:=B[K,JPIV];
 90:                     for L:=1 to M do
 91:                       begin
 92:                         B[K,L]:=B[K,L]-FACTOR*B[I,L];
 93:                       end;
 94:                   end;
 95:               end;
 96:           end;
 97:       end;

 98:   end;

 99:   procedure LEEMAT(N,M : integer;
100:                    var A : MAT2DIM);
101:   var
102:     I,J : integer;

103:   begin
104:     for I:=1 to N do
105:       for J:=1 to M do
106:         begin
107:           write(DispSal,'A(',I,J,')=');
108:           readln(A[I,J]);
109:           writeln(DispSal,A[I,J]);
110:         end;

111:   end;

112:   procedure RESULTADOS(NED : integer;
113:                        A : MAT2DIM;
114:                        MU : MAT2DIM;
115:                        SWERROR : integer);

116:   begin
117:     writeln(DispSal);
118:     writeln(DispSal,'MATRIZ DE ',
119:       'TRANSICION ORIGINAL:');
120:     DESPMAT(NED,NED,A);
121:     writeln(DispSal);
122:     writeln(DispSal,'MATRIZ DE TIEMPOS ',
123:       'DE LLEGADAS POR 1A VEZ:');
```

```
124:      DESPMAT(NED,NED,MU);
125:      if SWERROR =1 then
126:        writeln(DispSal,'**** NO HAY SOLUCION '+
127:          'O SOLUCION INFINITA ***');

128:    end;

129:    procedure CALCTIM(NED : integer;
130:                      A : MAT2DIM;
131:                      var MU : MAT2DIM;
132:                      var SWERROR : integer);
133:    var
134:      J : integer;

135:      procedure CALCMU(NED,S : integer;
136:                       A : MAT2DIM;
137:                       var MU : MAT2DIM;
138:                       var SWERROR : integer);
139:      var
140:        I,J,SWASIG : integer;
141:        B : MAT2DIM;

142:      begin
143:        for I:=1 to NED do
144:          begin
145:            for J:=1 to NED do
146:              begin
147:                B[I,J]:=A[I,J];
148:                if I=J then
149:                  B[I,J]:=B[I,J]-1;
150:                if J=S then
151:                  B[I,J]:=0;
152:              end;
153:            B[I,NED+1]:=-1;
154:            if I=S then
155:              begin
156:                for J:=1 to NED+1 do
157:                  B[I,J]:=0;
158:                B[S,S]:=1;
159:              end;
160:          end;
161:        GOSJRD(NED,NED+1,B,B);
162:        for I:=1 to NED do
163:          begin
164:            SWASIG:=0;
165:            for J:=1 to NED do
166:              begin
167:                if B[I,J]=1 then
```

```
160:                    begin
169:                      MU[J,S]:=G[I,NED+1];
170:                      SWAGIG:=1;
171:                    end;
172:               end;
173:             if SWAGIG=0 then
174:               SWERROR:=1;
175:           end;

176:     end;

177:   begin
178:     SWERROR:=0;
179:     for J:=1 to NED do
180:       begin
181:         CALCMU(NED,J,A,MU,SWERROR);
182:       end;

183:   end;

184:   procedure LECTURA(var NED : integer;
185:                     var A : MAT2DIM);
186:   begin
187:     writeln(DispSal);
188:     writeln(DispSal,'IO:20%, TIEMPO DE ',
189:       'LLEGADA POR PRIMERA VEZ');
190:     writeln(DispSal);
191:     write(DispSal,'NUMERO DE ESTADOS=');
192:     readln(NED);
193:     writeln(DispSal,NED);
194:     writeln(DispSal,'MATRIZ DE TRANSICION:');
195:     LEEMAT(NED,NED,A);

196:   end;

197:   begin
198:     IniDispSal;
199:     LECTURA(NED,A);
200:     CALCTIM(NED,A,MU,SWERROR);
201:     RESULTADOS(NED,A,MU,SWERROR);
202:     close(DispSal,lock);

203:   end.
```

Conclusiones:

El objetivo fundamental de esta tesis es proporcionar una herramienta de cálculo para usarse como complemento en las clases de Investigación de Operaciones. A través de treinta y un programas Pascal, se espera haber alcanzado ese objetivo. Sin embargo, ahora el resto queda a cargo del alumno o maestro que tenga interés en usar este paquete de programas. Es decir, que queda pendiente la parte más importante: su uso. Si a través de la utilización de los programas aquí presentados, el maestro o alumno alcanza los objetivos de la materia de una manera más completa, se considerará que este trabajo logró sus máximos propósitos.

Aparte de su utilización, este paquete representa un material, a partir del cual se pueden diseñar nuevos paquetes. El actual trabajo, por razones de su propósito, de ninguna manera es exhaustivo y, por lo mismo, no cubre los detalles particulares de cada técnica. Se hace una invitación para que en futuros trabajos se desarrollen paquetes computacionales para apoyar alguna área en particular (ejemplo: rutinas de utilería para apoyar la construcción de programas en el área de Cadenas de Markov, etc.).

Otro propósito que se pretende haber alcanzado es el de mostrar las ventajas de usar un lenguaje computacional "estructurado" como lo es Pascal en oposición al lenguaje tradicional para este tipo de aplicaciones: FORTRAN. Los productos finales son más intelegibles y, por lo tanto, menos propensos a errores lógicos. Esto hace que se pueda hablar de una mayor calidad en la programación aumentando la confiabilidad de los programas. Se hace un exhortación a sustituir los programas FORTRAN por la programación con lenguajes más poderosos y apropiados como Pascal, ALGOL, APL, LISP, etc.

La disponibilidad y bajo costo de las microcomputadoras, así como su gran poder, hacen que herramientas como es este paquete de programas estén al alcance de cualquier persona, lo cual hace sólo algunos años era prácticamente imposible a menos que se tuviera acceso a un centro de cálculo.

Se espera que el uso de este trabajo sea de tanto provecho para el lector como lo fue para el autor. Una gran parte de los programas de este paquete fueron seleccionados para apoyar el curso de Modelos Computacionales I, impartido a la carrera de Ingeniería de Sistemas en el Instituto Tecnológico y de Estudios Superiores de Monterrey Unidad Querétaro durante el semestre enero-mayo de 1983. En este curso, los alumnos desarrollaron, además de los programas básicos (expuestos en clase), paquetes completos para aplicaciones en áreas como PERT/CPM, Simulación, Teoría de Juegos y Cadenas de Markov. La existencia de esta tesis hizo posible tener una base para desarrollar el material de este

curso. Los programas de Programación Lineal han sido utilizados por alumnos de diferentes áreas en trabajos de tesis. Así mismo las técnicas de programación utilizadas en el diseño de este trabajo se han impartido en los cursos de Computación Electrónica y Sistemas Computacionales del instituto. Por último, existen planes para desarrollar sistemas computacionales usando algunos de los programas de este trabajo como subprogramas.

BIBLIOGRAFIA

Bibliografia:

L. Atkinson,
Pascal Programming,
1980,
John Wiley and Sons.

C.W. Churchman, R.L. Ackoff y E.L. Arnoff,
Introduction to Operations Research,
1957,
John Wiley and Sons.

E.V. Denardo,
Dynamic Programming - Models and Applications,
1982,
Prentice-Hall.

E.W. Dijkstra,
A Discipline of Programming,
1976,
Prentice-hall.

L.C.W. Dixon,
Nonlinear Optimisation,
1972,
Crane, Russak and Company.

C. Gane y T. Sarson,
Structured Systems Analysis, Tools and Techniques,
1975,
Improved System Technologies.

B.E. Gillett,
Introduction to Operations Research:
A Computer-Oriented Algorithmic Approach,
1976,
McGraw-Hill.

W.J. Graybeal y U.D. Pooch,
Simulation: Principles and Methods,
1980,
Winthrop Publishers.

F.S. Hillier y G.J. Lieberman,
Introduction to Operations Research,
1980,
Holden-Day.

M. Jackson,
System Development,

1983,
Prentice-Hall International.

K. Jensen y N. Wirth,
Pascal - User Manual and Report,
1974,
Springer-Verlag.

C.B. Jones,
Software Development - A Rigorous Approach,
1980,
Prentice-Hall International.

D. Knuth,
The Art of Computer Programming,
Vol. 1, "Fundamental Algorithms",
1968,
Addison-Wesley.

H. Ledgard,
Programming Proverbs,
1975,
Hayden.

B. Noble y S.W. Daniel,
Applied Linear Algebra,
1979,
Prentice-Hall.

J.W. Robbin,
Mathematical Logic,
1969,
W.A. Benjamin.

G.M. Schneider, S.W. Weingert y D.M. Perlman,
An Introduction to Programming and
Problem Solving with Pascal,
1978,
John Wiley and Sons.

H.A. Taha,
Operations Research, An Introduction,
1971,
Macmillan.

N. Wirth,
Algorithms + Data Structures = Programs,
1976,
Prentice-hall.