Universidad Nacional Autónoma de México

Posgrado en Ciencia e Ingeniería de la Computación

Algoritmo subcuadrático para comparación de conjuntos de puntos en $\mathbb{R}^2$

T E S I S

que para optar por el grado de:
Maestro en Ciencias (Computación)

PRESENTA:
Alfonso Francisco García Hernández

director de tesis:
Dr. Jorge Urrutia Galicia
Instituto de Matemáticas

México, D.F. Julio, 2016

Algoritmo subcuadrático para comparación de conjuntos de puntos en $\mathbb{R}^2$

por

Alfonso Francisco García Hernández

Tesis presentada para obtener el grado de

Maestro en Ciencias (Computación)

en el

Posgrado en Ciencia e Ingeniería de la Computación

Universidad Nacional Autónoma de México

México, D.F. Julio, 2016

Una evolución de 13,800 millones
de años que han convertido la materia
y la energía en vida y conciencia.
Carl Sagan, 1934-1996

Agradecimientos

En el principio Dios creó los bits, el resto es trabajo nuestro. Por que sin bits no sería posible realizar mi trabajo, quiero agradecer primeramente a Dios, ya que en mi conjunto de conceptos y creencias, es quien me da la guía y fortaleza espiritual para tomar las mejores desiciones a lo largo de mi vida.

El padre y la madre juegan un papel muy importante en el desarrollo de una persona. Haciendo una analogía de esto, quiero agradecer al Consejo Nacional de Ciencia y Tecnología, que gracias a su Programa Nacional de Becas, provee los recursos necesarios para llevar a cabo estos posgrados. Y a la Universidad Nacional Autónoma de México, que como madre, estuvo a cargo de la educación y el cuidado necesarios para ello.

Un buen profesor es capaz de transmitirle a un alumno conocimientos. Un profesor memorable es aquel que no sólo trasmite conocimientos, proporciona sabiduría. Agradezco infinitamente al Dr. Jorge Urrutia el haberme compartido su sabiduría, que me permite ahora, poder ver los algoritmos mas allá de simples herramientas para solucionar problemas. A todos mis sinodales, la paciencia y aportaciones que me hicieron durante las revisiones de mi trabajo. Y en general a todos mis profesores, que a lo largo de la maestría, no se limitaron a darme conocimientos, sino sabiduría.

Motivación e inspiración son ingredientes necesarios por igual para el logro de los grandes proyectos. Agradezco ahora a mi fuente de motivación, Amadeus y, a mi fuente de inspiración, Laura, por ser fuentes inagotables de esos ingredientes en todos los proyectos importantes de mi vida.

A mis padres Emma y Aniceto, les doy gracias por todas la cosas que me dieron, pero mas les agradezco las cosas que no me dieron, ya que con eso me enseñaron a trabajar para conseguir mis sueños y deseos. Aunado a esto, el haberme inculcado desde siempre valores como honestidad, respeto y responsabilidad, me han permitido ser lo que soy.

De manera especial quiero agradecer a Antonio, Adolfo y Sergio, el haber tenido la confianza de hacer de su empresa mi laboratorio de investigación y experimentación, que gracias a eso puedo ser el profesionista que soy.

En memoria de mi abuelo Miguel, por haberme dado en mi infancia todo ese cariño incondicional que solo un abuelo sabe dar.

A mi hermano Luis Eduardo y a todos mis amigos, para los que no hago una lista por temor de dejar a alguno fuera, pero que sé son incondicionales y siempre están ahí, llenando mi vida de alegría y momentos inolvidables.

A todos, gracias.

Índice general

Índice de figuras

Lista de algoritmos

Índice de tablas

Capítulo 1

Introducción

1.1. Contexto

Desde la antigüedad, el ser humano ha tenido la necesidad de comparar los objetos del mundo que lo rodea. Esta actividad, que básicamente consiste en seleccionar uno o más atributos de los objetos en cuestión, implica la asignación de valores no siempre cuantificables para ellos. Para poder comparar de una forma objetiva e independiente del observador, es necesario contar con un marco de referencia común. Cuando existen múltiples marcos de referencia, comparar implica buscar atributos cuya valoración dependa sólo de los objetos mismos. Sin importar el marco de referencia en el que sean valorados los atributos para una comparación, la elección de los mismos y el procedimiento usado se deben formalizar de modo que la comparación sea universal, independientemente del entorno en donde se efectúe.

En matemáticas, las comparaciones requieren de procedimientos precisos y bien definidos, demostrados y probados sobre una base teórica; esto, con el propósito de que la comparación sea una relación de equivalencia. La creación de algoritmos de comparación sirven de base para otras investigaciones y otras disciplinas como las ciencias de la computación.

Existe una correspondencia biunívoca entre el plano cartesiano y plano geométrico. El plano cartesiano ($\mathbb{R}^2$) es un conjunto formado por pares ordenados de números reales. Estos, miden distancia respecto a dos ejes ortogonales entre sí y que se cruzan en un punto llamado origen. Por otro lado, el plano geométrico es el conjunto de puntos de un espacio euclidiano bidimensional. Aunque se hace dicha correspondencia entre ambos planos, no existe un marco de referencia

universal que indique cuál punto del plano geométrico corresponde al origen del plano cartesiano y tampoco que indique la dirección o el sentido de los ejes, ni la unidad de medida que se tomará para las distancias entre los puntos.

La necesidad de comparar objetos ha evolucionado en la capacidad de identificar objetos y lugares específicos. Al mismo tiempo, esa capacidad no es exclusivamente humana ya que existen varias especies que la poseen. Por ejemplo, las aves migratorias reconocen las rutas y los lugares donde hacer los nidos; en un grupo de pingüinos las madres identifican a sus crías, etc. En el ser humano, a diferencia de otras especies, la capacidad de discernimiento es enriquecida por múltiples factores. Más allá de las capacidades físicas de los sentidos, el cerebro juega un papel muy importante, tanto por los recuerdos, como por su capacidad de análisis y síntesis. Gracias a eso, una persona puede reconocer a un amigo o familiar en un grupo grande, reconocer su canción favorita a pesar del ruido de fondo, o encontrar su nombre en una lista numerosa.

La percepción es fundamental para el desarrollo del ser humano. Es el proceso donde las impresiones sensoriales del mundo quedan integradas en la conciencia. Este proceso se inicia de manera natural al momento de nacer y se mejora en la medida que crecemos y afinamos nuestros sentidos [Percepción, 2001]. Sucede en forma casi instantánea y no nos detenemos a pensar el modo en que ocurre. Es ahí donde se busca crear modelos que reproduzcan o traten de reproducir esos procesos mentales.

Por ejemplo, en la antigüedad los griegos reconocieron 48 constelaciones estelares. Una de las más conocidas es la *Osa Mayor* que se compone de 7 estrellas y cuya disposición se muestra en la figura 1-1.

Figura 1-1: Constelación de la *Osa Mayor* reconocida por los griegos en la antigüedad.

Una constelación, como es observada, se compone de un grupo de varias estrellas. La forma para reconocerla se basa en las posiciones relativas que guardan. Es posible representar la bóveda celeste como un plano y las estrellas como puntos sobre él. La escala, alineación y rotación que tengan los puntos, son irrelevantes debido a que el observador puede encontrarse en un sin

número de lugares y orientaciones; sin embargo, la constelación siempre es la misma, por lo que es reconocible.

La percepción humana juega un papel importante, ya que la constelación se puede identificar sin problemas en el conjunto completo de estrellas observables a simple vista, parecido al mostrado en la figura 1-2.

Figura 1-2: Fragmento de la bóveda celeste observable a simple vista.

Este fragmento que muestra pocas estrellas, pone en evidencia la dificultad de identificar un conjunto de puntos en otro de mayor tamaño. No obstante, nuestra percepción nos permite reconocer en él los 7 puntos que forman la *Osa Mayor*.

Ya que la cantidad de subconjuntos que se pueden obtener de otro, crece de manera exponencial, la dificultad de elegir uno de ellos se incrementa en la misma proporción. En este sentido, comparar uno de ellos es una tarea bastante compleja. Por ejemplo, en la figura 1-2 existen 43 puntos, de los cuales se deben elegir 7 y compararlos en conjunto con los que forman la *Osa Mayor* (figura 1-1).

Existen $\binom{43}{7} = 903$ formas de elegir 7 puntos en un conjunto de 43. Para compararlos hay $7! = 5040$ maneras de ordenarlos y se requieren $7 * 7 = 49$ operaciones individuales para verificar a "fuerza bruta"si corresponden o no. Si una computadora, que no tiene la capacidad de percepción, debe verificar que la *Osa Mayor* se encuentre en ese fragmento de la bóveda celeste, necesita efectuar un total de $903 * 5040 * 49 = 223004880$ operaciones.

Sin importar que las computadoras sean cada vez más rápidas, la cantidad de operaciones

mencionada es bastante alta, por lo que hacer una implementación de esta búsqueda resultaría inútil.

Por otro lado, los problemas evolucionan con el tiempo y lo que fue válido en la antigüedad, no necesariamente lo es en nuestros días. En 1930, la Unión Astronómica Internacional (UIA por sus siglas en inglés), se dio a la tarea de catalogar la bóveda celeste. Para tal efecto cambió el concepto de constelación, de una serie de puntos conectados, al de una región dentro de la misma. El cambio tiene como finalidad que los objetos celestes, sin importar sean visibles o no, se puedan identificar al pertenecer a una constelación particular.

A simple vista es posible observar un cierto número de estrellas dentro de una región. En ese mismo espacio, usando un telescopio la cantidad de estrellas observables es mayor. Esa diferencia se debe a que las estrellas tienen distintas intensidades luminosas (a esto se le conoce como *magnitud estelar*). Posterior al cambio promovido por la UIA, la región de la bóveda celeste que contiene las 7 estrellas de la *Osa Mayor* observadas en la antigüedad, comprende ahora 117 estrellas hasta la magnitud estelar 6, como se aprecia en la figura 1-3.

Figura 1-3: Constelación de la *Osa Mayor* para la Unión Astronómica Internacional.

Reconocer las 117 estrellas de la *Osa Mayor* catalogadas por la UIA (figura 1-3), en el mismo fragmento de bóveda celeste contemplando todas las estrellas hasta la magnitud estelar 6 (figura 1-4) es una labor mucho mas compleja.

Sin un marco de referencia, es difícil dar sentido y orientación a los lugares geométricos. Por el contrario, con un marco de referencia conocido, se visualiza fácilmente lo que de otra forma resulta ininteligible, como se observa en la figura 1-5. Es por ello que se debe tanto acotar el alcance de la búsqueda, como desarrollar nuevos algoritmos, que permitan efectuar

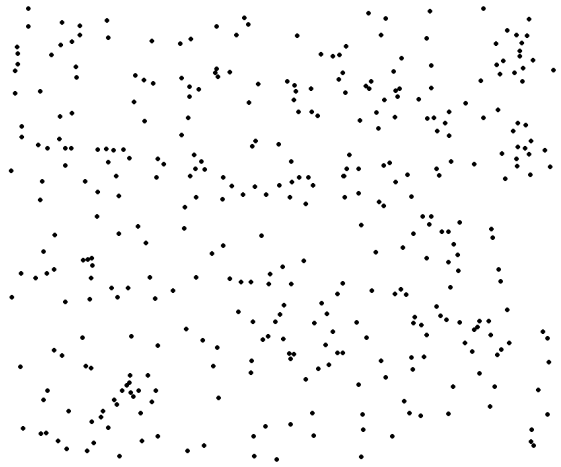


Figura 1-4: Fragmento de la bóveda celeste de la figura 1-2 observado con un telescopio.

las comparaciones de manera eficiente.

Al comparar conjuntos de puntos, el proceso debe contemplar que no existen marcos de referencia, para que el resultado de la comparación sea válido. Esa eficiencia, es particularmente importante si la comparación ha de aplicarse en conjuntos de miles o cientos de miles de puntos.

1.2. Descripción del problema

Por el alcance y complejidad del problema, el presente trabajo acota el planteamiento a conjuntos del mismo tamaño o bien con diferencia de un punto. Puede describirse en las siguientes dos sentencias:

Problema 1 *Sean S y T dos conjuntos de puntos en posición general en $\mathbb{R}^2$ con la misma cardinalidad. ¿Son semejantes?*

Problema 2 *Sean S y T dos conjuntos de puntos en posición general en $\mathbb{R}^2$, si la diferencia entre S y T es de sólo un punto. ¿Cuál es ese punto?*

La posición de un punto depende del marco de referencia con respecto al cual es posicionado. En $\mathbb{R}^2$ existen infinidad de marcos de referencia en virtud de que, la elección del punto origen, las direcciones y sentidos de los ejes, y la unidad de medida para distancias entre puntos,

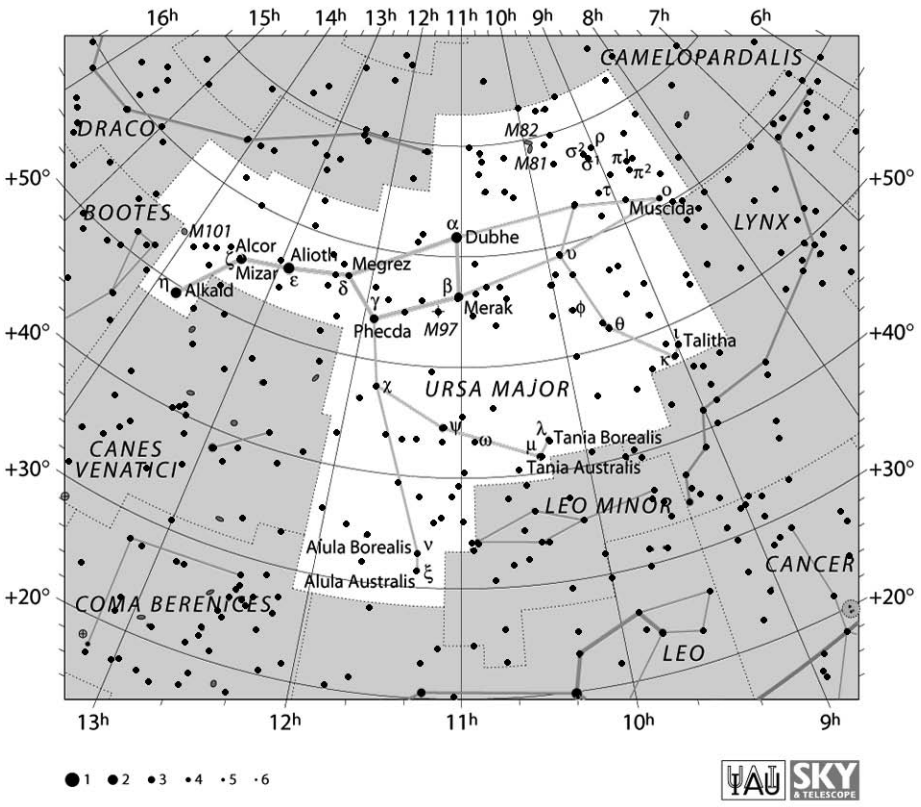


Figura 1-5: Fragmento de la bóveda celeste de la figura 1-4 con marco de referencia.

son completamente arbitrarios. Cuando se comparan dos conjuntos de puntos se debe tomar en cuenta que cada uno tiene su propio marco de referencia y, en general esos marcos son diferentes.

Al no existir un marco de referencia absoluto para posicionar puntos, la única forma de comparar conjuntos de puntos, es mediante las posiciones relativas entre los puntos del mismo conjunto. Por ejemplo, en dos conjuntos de dos puntos distintos cada uno, se puede afirmar que son semejantes ya que en ambos un punto se encuentra *al lado* del otro, al no haber unidades de medida, ni sentido y dirección de los ejes, ni ubicación del origen, como se observa en la figura 1-6.

En otro ejemplo, sean dos conjuntos con tres puntos cada uno, tales que los tres puntos en un mismo conjunto correspondan con los vértices de un triángulo equilátero, al igual que en el ejemplo anterior ambos son semejantes, ya que todos los triángulos equiláteros son semejantes. Es importante observar, que no se trata de hacer una correspondencia entre puntos, sino de las

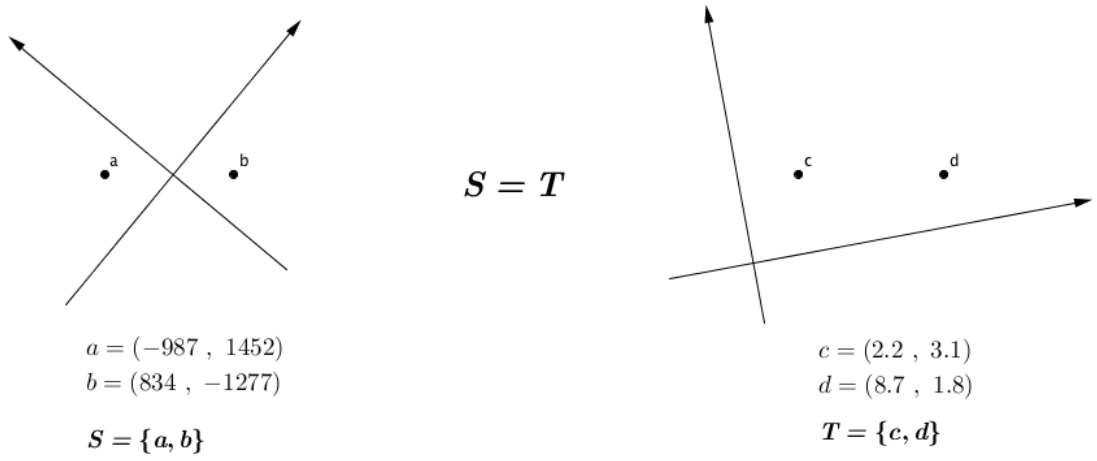


Figura 1-6: Conjuntos de puntos semejantes, sin marco de referencia común.

posiciones relativas entre ellos.

En el presente trabajo se desarrolla un algoritmo determinístico. Utiliza como entrada dos conjuntos de puntos en posición general en $\mathbb{R}^2$, cada uno con su propio marco de referencia. Si son del mismo tamaño o con diferencia de un punto, los compara en tiempo $\mathcal{O}(n \log n)$ indicando si son semejantes o no; si lo son con diferencia de un punto, indica el punto diferencia.

1.3. Trabajos relacionados

Sin una base formal, los conceptos de *orientación* y *posición relativa* entre puntos, pueden resultar ambiguos e incluso subjetivos. Términos como *junto a* o *al lado de* carecen de sentido absoluto y en última instancia, dependen de la observación que se haga para determinarlos.

Estos conceptos han sido estudiados ampliamente. Entre los primeros trabajos realizados en este ámbito, se tienen a Folkman y Lawrence [1978], que formalizan conceptos topológicos asociados a lugares geométricos y definen las propiedades de un *matroide orientado*.

Una forma de definir la orientación de un conjunto de puntos es mediante el *tipo de orden*. Este, permite diferenciar la posición relativa de un punto con respecto a los otros [Goodman y Pollack, 1983]. Dicho concepto tiene varias interpretaciones dependiendo del número de dimensiones en donde se encuentran los puntos, de modo que, para el caso de $\mathbb{R}^2$ sólo es posible colocar un punto a la izquierda o a la derecha, con respecto a la posición de otros dos. En ese

trabajo, los autores definen la $matriz - \lambda$ para codificar el tipo de orden de un conjunto de puntos y a través de ella, es posible efectuar la comparación entre conjuntos de puntos.

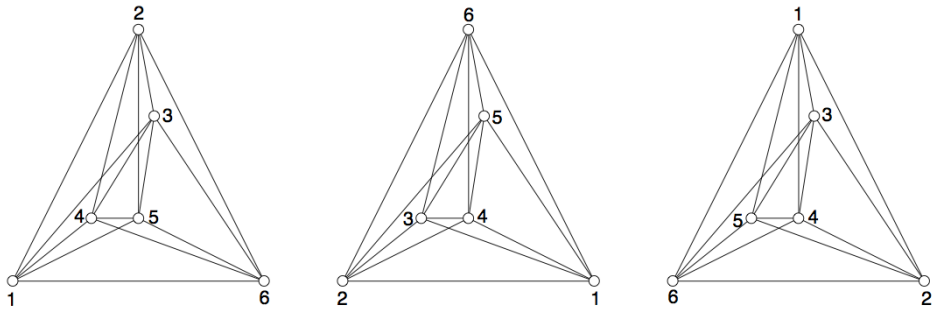
El problema con el tipo de orden es que se trata de una función discreta que numéricamente crece muy rápido, como se muestra en la tabla 1-1. Aunque es útil para otros propósitos, no es práctico tratar de utilizarla como medio de comparación [Aichholzer, Aurenhammer, y Krasser, 2001].

Número de puntos	Número de tipos de orden
3	1
4	2
5	3
6	16
7	135
8	3,315
9	158,817
10	14,309,547
11	2,334,512,907

Tabla 1-1: Número de tipos de orden para n puntos en $\mathbb{R}^2$.

Si se desea utilizar el tipo de orden para efectuar la comparación de conjuntos de puntos, será necesario calcular la $matriz-\lambda$ para cada conjunto al momento de efectuar la comparación. Este cálculo se efectúa en tiempo $\mathcal{O}(n^3)$ y aunque existe un algoritmo que puede efectuarlo en tiempo $\mathcal{O}(n^2)$ es demasiado costoso para conjuntos grandes de puntos [Krasser, 2003] [Aloupis, Iacono, Langerman, y Ozkan, 2013]. En la figura 1-7, se muestra el cálculo de la $matriz - \lambda$ para un conjunto de 6 puntos en sus tres etiquetaciones posibles.

De manera paralela a los trabajos descritos con anterioridad, que son en su totalidad determinísticos, existen otras alternativas basadas principalmente en métodos estadísticos. Estos trabajos son particularmente útiles cuando se trabaja con conjuntos grandes de puntos. De manera especial, es importante mencionar los basados en momentos estadísticos, utilizados ampliamente en procesamiento de imágenes [Gonzalez y Woods, 2008].



–	0	1	2	3	4
4	–	1	3	2	0
3	3	–	2	1	1
2	1	2	–	2	3
1	2	3	2	–	2
0	4	3	2	1	–

–	0	1	2	3	4
4	–	2	3	1	0
3	2	–	2	2	1
2	1	2	–	3	2
1	3	2	1	–	3
0	4	3	2	1	–

–	0	1	2	3	4
4	–	3	2	1	0
3	1	–	1	2	3
2	2	3	–	2	1
1	3	2	2	–	2
0	4	1	3	2	–

Figura 1-7: $Matriz - \lambda$ para un conjunto de 6 puntos.

Capítulo 2

Marco teórico

2.1. Teoría de algoritmos

2.1.1. Algoritmos y programas para computadora

Un *algoritmo* es un procedimiento general de resolución de problemas, mediante la ejecución de una secuencia de pasos elementales, a partir de un conjunto de datos [Algoritmo, 2001].

La primera referencia moderna que se tiene de la palabra *algoritmo*, es en el diccionario Webster hacia el año de 1957, como evolución de *algorismo* que significa, *hacer aritmética usando numerales arábigos*. Su origen es incierto, sin embargo los matemáticos están de acuerdo en que proviene del nombre del matemático árabe *Muhammad Ibn Mûsâ Al-Khwârizmî* [Knuth, 1997]. Al-Khwârizmî nació en el 780 d.C. en Bagdag, su mayor trabajo fue introducir los números indo-arábigos en las matemáticas europeas; vivió en Bagdag durante la época de oro de las ciencias islámicas, el término *álgebra* se debe a su obra *Kitab al-jabr wa al-muqâbalah*, traducida al latín en el siglo XII d.C. en la que se muestra, una compilación de reglas para la solución de ecuaciones lineales y cuadráticas. El término *algorimto* se debe a su obra *Al-goritmi de numero Indorum*, de la que sólo se preserva una traducción en latín [Khwârizmî, 2007].

Desde la educación primaria hemos estudiado algoritmos. Uno de ellos es el utilizado para efectuar multiplicaciones, donde para obtener el producto de dos números enteros de varios dígitos, basta con realizar operaciones elementales como sumas y productos de dos dígitos. Otro ejemplo, lo tenemos en el propuesto por Euclides para obtener el máximo común divisor

de dos números enteros. Estos, son ejemplos de algoritmos correctos cuyas características son mencionadas por Knuth [1997]:

- *Finito.* Debe terminar siempre.
- *Bien definido.* Todos los casos deben estar contemplados.
- *Entrada.* Conjunto de datos sobre los que se va a trabajar.
- *Salida.* Resultado esperado, debe depender única y exclusivamente de los datos de entrada.

Desde el punto de vista de las ciencias de la computación, el estudio de los algoritmos es de vital importancia, ya que la elaboración de programas de computadora para resolver problemas, implica la formulación y especificación detallada del mismo, así como el análisis y diseño del algoritmo para obtener su solución. La especificación formal de un problema, implica la creación de un modelo matemático y posteriormente la creación de un algoritmo [Aho, Hopcroft, y Ullman, 1988].

Es necesario dejar claro que un algoritmo es una herramienta para resolver un problema. La especificación del problema es la relación que guardan los datos de entrada y de salida. El algoritmo define los pasos para encontrar esa relación [Cormen, Leiserson, Rivest, y Stein, 2009]. La especificación del problema describe de manera genérica las características de los datos de entrada. Cuando se tiene un conjunto específico de ellos, se tiene una *instancia* de ese problema. El conjunto de datos de salida puede ser o no único.

El análisis de un algoritmo implica la predicción de los recursos que éste consumirá, a partir de una instancia específica del problema para el que fue diseñado [Cormen *et al.*, 2009]. Existen diversos recursos que pueden ser medidos durante la ejecución de un algoritmo, sin embargo, en general los recursos con los que se determina su eficiencia, son:

- *Tiempo de ejecución.* Es el número de ciclos de computación requeridos para completar su tarea. Muchos autores lo asocian directamente al número de instrucciones ejecutadas, sin embargo, con la evolución que han tenido los lenguajes de programación este enfoque es incorrecto, ya que para algunos lenguajes una instrucción puede implicar un gran número de ciclos de computación.

- *Espacio.* Es la cantidad de almacenamiento normalmente en memoria principal de variables y datos requeridos para completar su tarea.

Las métricas sólo deben depender de la cantidad de datos de entrada y no de las características de los mismos. El término *tamaño de la entrada* se emplea para indicar la cantidad de datos de entrada, donde a partir de él, se determina el espacio y el tiempo de ejecución de un algoritmo.

2.1.2. Notación asintótica

Cuando se tienen varios algoritmos capaces de resolver un problema, aparece la necesidad de elegir alguno en particular. Aunque éstos pueden ser comparados por su tiempo de ejecución y espacio, el método de elección debe ser tal que no dependa de la habilidad de la persona que implemente, ni de las características del lenguaje, ni de la computadora donde se ejecute. Por otro lado, al ser independiente de las características de los datos de entrada y sólo depender de la cantidad de los mismos, es necesario poder predecir dichas métricas para cualquier tamaño de la entrada.

El *orden de crecimiento* es una manera de abstraer la problemática descrita anteriormente. Para ello, sea una función $f : \mathbb{N} \to \mathbb{N}$ donde la variable independiente es el tamaño del problema y la variable dependiente es el tiempo de ejecución o el espacio [Cormen *et al.*, 2009]. Si para un algoritmo en particular, su tiempo de ejecución es $f(n) = 6n^2 + 15n + 8$, se dice que ese algoritmo tiene un tiempo de ejecución del orden $\mathcal{O}(n^2)$.

Formalmente hablando, el orden de crecimiento se refiere a una función asintótica, la cual establece los límites para los que crece o decrece otra función que representa el tiempo de ejecución o el espacio, de un algoritmo en particular. Esta definición requiere de la introducción de los siguientes términos que son independientes del tamaño de la entrada:

- *Peor de los casos.* Implica que, dada las características de los datos de entrada, el algoritmo ejecutará la mayor cantidad de pasos posibles.

- *Mejor de los casos.* Es cuando, por la características de los datos de entrada, el algoritmo ejecutará la menor cantidad de pasos posibles.

- *Caso promedio.* Aquí, se estiman las características de los datos de entrada más probables. Mediante un análisis de valor esperado se determina la cantidad de pasos que ejecutará el algoritmo más frecuentemente.

Una función $g(n)$ es una *cota superior asintótica*, para una familia de funciones $f(n)$, si existen constantes $c, n_0 > 0$ tales que:

$$0 \leq f(n) \leq c \cdot g(n) \ \forall \ n \geq n_0$$

y se denota por $\mathcal{O}(g(n))$, como se muestra en la figura 2-1.

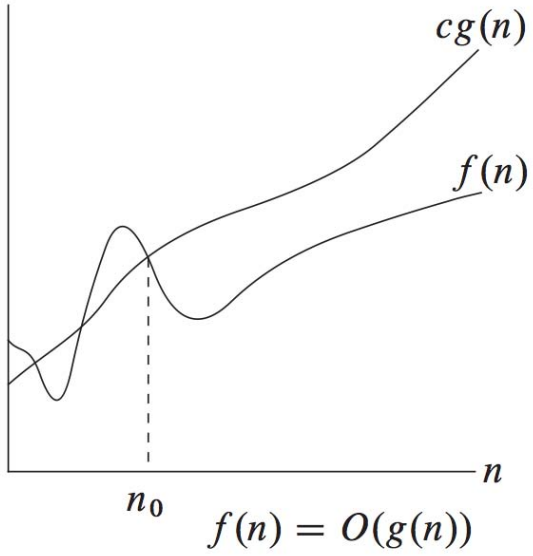


Figura 2-1: Notación $\mathcal{O}$, para la cota superior de una función.

Una función $g(n)$ es una *cota inferior asintótica*, para una familia de funciones $f(n)$, si existen constantes $c, n_0 > 0$ tales que:

$$0 \leq c \cdot g(n) \leq f(n) \ \forall \ n \geq n_0$$

y se denota por $\Omega(g(n))$, como se muestra en la figura 2-2.

Una función $g(n)$ es una *cota justa asintótica*, para una familia de funciones $f(n)$, si existen constantes $c_1, c_2, n_0 > 0$ tales que:

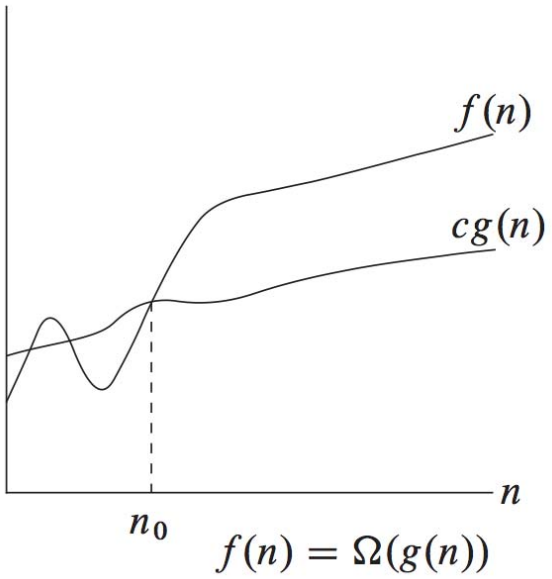


Figura 2-2: Notación Ω, para la cota inferior de una función.

$$c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n) \ \forall \ n \geq n_0$$

y se denota por $\Theta(g(n))$, como se muestra en la figura 2-3.

El establecimiento de estas funciones asintóticas es una base formal para la comparación de algoritmos.

2.1.3. Teoría de la complejidad

Contando con una base formal de comparación para el orden de crecimiento de los algoritmos, surgen otras preguntas:

- ¿Qué algoritmo es más o menos eficiente?
- ¿Qué problema es más o menos complejo?
- ¿Es posible medir la complejidad de un problema a partir del o de los algoritmos diseñados para resolverlo?

Estas preguntas han seguido a matemáticos y científicos de la computación durante décadas, en vista de que, dado un problema en particular con un algoritmo conocido ¿existirá otro

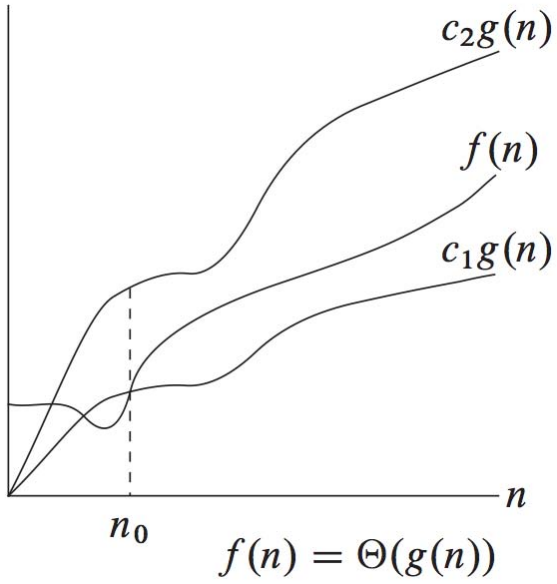


Figura 2-3: Notación Θ, para la cota justa de una función.

algoritmo más rápido? o dicho de otra forma ¿cómo saber que no existe un algoritmo más rápido de los ya conocidos?

A medida que las computadoras se vuelven más rápidas y con mayor capacidad, también lo hacen los problemas que resuelven. Por otro lado, el interés humano en que las computadoras resuelvan problemas nuevos, hace necesario el estudio de la complejidad algorítmica.

Se dice que, un algoritmo tiene tiempo de ejecución *polinomial* si su orden de crecimiento $\mathcal{O}(p(n))$, es tal que, $p(n)$ es un polinomio en n. Cualquier otro orden de crecimiento $\mathcal{O}(f(n))$, donde $f(n)$ no sea un polinomio, se dice que tiene tiempo de ejecución *exponencial* [Garey y Johnson, 1979]. La diferenciación en estos dos tipos de algoritmos, a partir de su orden de crecimiento, tiene relevancia cuando se analizan instancias del problema demasiado grandes, como se ejemplifica en la tabla 2-1.

Aún para funciones de complejidad polinomial, el estudio de la complejidad algorítmica es importante. En la tabla 2-2 se muestran para algunas funciones de complejidad subcuadrática, el crecimiento del tiempo de ejecución para tamaños de la entrada grandes, en comparación con la función cuadrática.

En esta última tabla, se aprecia la importancia que tiene el diseño de algoritmos subcuatráticos para procesar volúmenes grandes de información, sobre todo cuando se trata de tamaños

$f(n)$	$n = 10$	$n = 20$	$n = 30$	$n = 40$	$n = 50$	$n = 60$
n	0.00001 segundos	0.00002 segundos	0.00003 segundos	0.00004 segundos	0.00005 segundos	0.00006 segundos
n^2	0.00001 segundos	0.0004 segundos	0.0009 segundos	0.0016 segundos	0.0025 segundos	0.0036 segundos
n^3	0.001 segundos	0.008 segundos	0.027 segundos	0.064 segundos	0.125 segundos	0.216 segundos
n^5	0.1 segundos	3.2 segundos	24.3 segundos	1.7 minutos	5.2 minutos	13.0 minutos
2^n	0.001 segundos	1.0 segundos	17.9 minutos	12.7 días	35.7 años	366 siglos
3^n	0.059 segundos	58 minutos	6.5 años	3855 siglos	$2 * 10^8$ siglos	$1.3 * 10^{13}$ siglos

Tabla 2-1: Comparación de funciones de complejidad polinomial y exponencial.

	$\log n$	n	$n \log n$	n^2
$n = 100$	0.0007 segundos	0.01 segundos	0.0664 segundos	1.0 segundos
$n = 1000$	0.001 segundos	0.1 segundos	0.9966 segundos	1.67 minutos
$n = 10^4$	0013 segundos	1.0 segundos	13.28 segundos	2.78 horas
$n = 10^5$	0.0017 segundos	10 segundos	2.76 minutos	11.57 días
$n = 10^6$	0.002 segundos	1.66 minutos	33.21 minutos	3.17 años

Tabla 2-2: Funciones de complejidad subcuadrática con respecto a la cuadrática.

de la entrada del orden de millones de datos, que se deben procesar de forma casi inmediata y que sin importar la rapidez creciente de la computadoras, sería imposible con algoritmos menos eficientes.

2.2. Geometría

2.2.1. Semejanza

La semejanza geométrica, es el hecho de que dos figuras tengan la misma forma sin importar sus tamaños u orientaciones. Desde la geometría euclidiana clásica se ha estudiado la seme-

janza de triángulos, quedando establecida muy claramente en todos sus casos [Baldor, 1984]. Estas propiedades, que definen cuando dos triángulos son semejantes y que están basadas en el *Teorema de Tales*, pueden ser extendidas a cualquier poligonal simple utilizando los mismos razonamientos.

De esta forma tenemos:

Si $ABC...N$ y $A'B'C'...N'$ son dos poligonales simples y

$$\widehat{A} = \widehat{A'}\ ,\ \widehat{B} = \widehat{B'}\ ,\ \widehat{C} = \widehat{C'}\ ,\ ...\ ,\ \widehat{N} = \widehat{N'}$$

y

$$\frac{\overline{AB}}{\overline{A'B'}} = \frac{\overline{BC}}{\overline{B'C'}} = ... = \frac{\overline{NA}}{\overline{N'A'}}$$

entonces, las poligonales son semejantes.

Para efectos de poder determinar la semejanza de dos poligonales simples, es posible efectuar una normalización en la longitud de sus lados, si consideramos al menor de ellos como unidad, de esta forma tenemos:

Sean $a, b, c, ..., n$ y $a', b', c', ..., n'$ las longitudes de los lados de dos poligonales simples semejantes y, a y a' el menor de sus lados respectivamente, tenemos que:

$$\frac{a}{a'} = \frac{b}{b'} = \frac{c}{c'} = ... = \frac{n}{n'}$$

multiplicando por $\frac{a'}{a}$

$$\frac{aa'}{aa'} = \frac{a'b}{ab'} = \frac{a'c}{ac'} = ... = \frac{a'n}{an'}$$
$$1 = \frac{\frac{b}{a}}{\frac{b'}{a'}} = \frac{\frac{c}{a}}{\frac{c'}{a'}} = ... = \frac{\frac{n}{a}}{\frac{n'}{a'}}$$

por lo tanto, si dividimos las longitudes de todos los lados entre la longitud del lado menor, tenemos:

$$\frac{b}{a} = \frac{b'}{a'} \ , \ \frac{c}{a} = \frac{c'}{a'} \ , \ \ldots \ , \ \frac{n}{a} = \frac{n'}{a'}$$

2.2.2. Ángulo entre dos segmentos

Del cálculo vectorial tenemos, la definición de *producto punto* entre dos vectores en $\mathbb{R}^2$ está dada por:

$$\vec{A} \cdot \vec{B} = a_1 b_1 + a_2 b_2$$

donde $\vec{A} = (a_1, a_2)$ y $\vec{B} = (b_1, b_2)$ [Colley, 2013].

Y la definición de *magnitud*, *norma* o *longitud* de un vector está dada por:

$$\|\vec{A}\| = \sqrt{{a_1}^2 + {a_2}^2}$$

donde $\vec{A} = (a_1, a_2)$.

Mediante estas definiciones se puede demostrar que:

$$\vec{A} \cdot \vec{B} = \|\vec{A}\| \cdot \|\vec{B}\| \cdot \cos\beta$$

donde β es el ángulo menor entre los dos vectores [Colley, 2013].

A partir de este resultado, si tenemos los segmentos $\overline{AB}$ y $\overline{CD}$, donde $A = (a_1, a_2)$, $B = (b_1, b_2)$, $C = (c_1, c_2)$ y $D = (d_1, d_2)$, y queremos determinar el ángulo menor β comprendido entre las rectas en las que se encuentran dichos segmentos, tenemos:

$$\vec{X} = (b_1 - a_1 \ , \ b_2 - a_2)$$

$$\vec{Y} = (d_1 - c_1 \ , \ d_2 - c_2)$$

$$\cos\beta = \frac{\vec{X} \cdot \vec{Y}}{\|\vec{X}\| \cdot \|\vec{Y}\|}$$

donde $0 \leq \beta \leq \pi$, como se muestra en la figura 2-4.

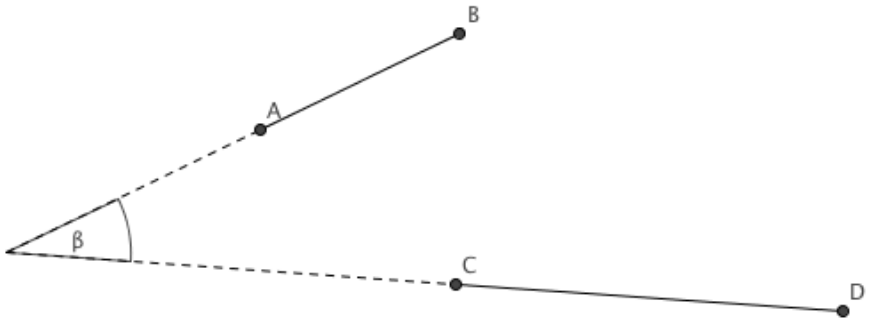


Figura 2-4: Ángulo entre dos segmentos.

2.2.3. Transformaciones

Se define como *transformación* en un modelo geométrico, a aquella operación que modifica alguna característica del objeto, sobre el cuál se efectúa [Foley, Hughes, Van Dam, McGuire, Sklar, Feiner, y Akeley, 2014]. Existen varias operaciones de transformación, para el desarrollo del presente trabajo se usarán las siguientes:

- *Escalamiento.* Modifica las dimensiones del objeto.
- *Rotación.* Modifica la alineación del objeto con respecto al origen del sistema de referencia.
- *Traslación.* Modifica la posición del objeto con respecto al origen del sistema de referencia.

Al expresar las transformaciones de escalamiento, rotación y traslación, mediante operaciones de matrices en $\mathbb{R}^2$ como se muestra a continuación:

$$\begin{bmatrix} k_x & 0 \\ 0 & k_y \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} k_x \cdot x \\ k_y \cdot y \end{bmatrix} = \begin{bmatrix} x' \\ y' \end{bmatrix}$$

$$\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x \cdot \cos\theta + y \cdot \sin\theta \\ -x \cdot \sin\theta + y \cdot \cos\theta \end{bmatrix} = \begin{bmatrix} x' \\ y' \end{bmatrix}$$

y

$$\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} = \begin{bmatrix} x + t_x \\ y + t_y \end{bmatrix} = \begin{bmatrix} x' \\ y' \end{bmatrix}$$

se observa que en el último caso no es posible expresarla como un producto, lo cual impide combinar varias operaciones en la misma expresión sin afectar el resultado.

Para poder expresar las tres transformaciones como productos, es necesario efectuar dichas operaciones de matrices en $\mathbb{R}^3$, asignando $z = 1$ como se muestra en la figura 2-5 [Foley *et al.*, 2014].

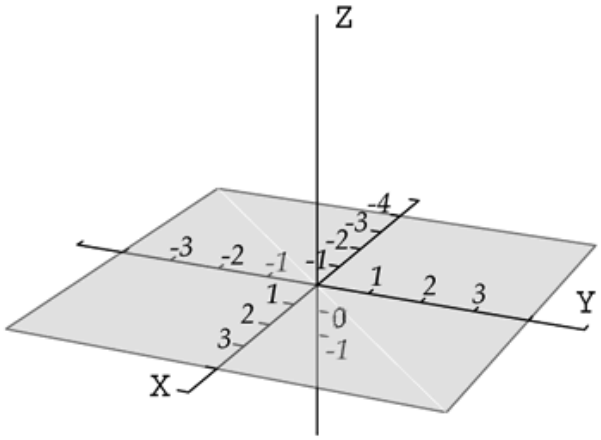


Figura 2-5: Plano $\mathbb{R}^2$ llevado a $\mathbb{R}^3$ en $z = 1$.

Ya en $\mathbb{R}^3$ dichas transformaciones se expresarán mediante operaciones de matrices, de la siguiente forma:

$$\begin{bmatrix} k_x & 0 & 0 \\ 0 & k_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} k_x \cdot x \\ k_y \cdot y \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x \cdot \cos\theta + y \cdot \sin\theta \\ -x \cdot \sin\theta + y \cdot \cos\theta \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

y

$$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + t_x \\ y + t_y \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

en donde para todos los casos son expresadas como productos.

Estas transformaciones pertenecen al grupo de *transformaciones afines* y requieren tiempo $\mathcal{O}(1)$ para ser ejecutadas sobre un punto. Aplicar una transformación a un conjunto de n puntos requiere un tiempo de ejecución $\mathcal{O}(n)$ [Foley *et al.*, 2014].

2.2.4. Orientación

Formalmente la *orientación* de una tripleta de puntos i, j, k en $\mathbb{R}^2$ está dada por el signo del determinante:

$$o(i, j, k) = \det \begin{pmatrix} i_x & j_x & k_x \\ i_y & j_y & k_y \\ 1 & 1 & 1 \end{pmatrix}$$

y puede ser positiva (en sentido contrario a las manecillas del reloj), negativa (en sentido de las manecillas del reloj) o sin orientación (los puntos son colineales), como se muestra en la figura 2-6 [Goodman y Pollack, 1983] [Krasser, 2003].

Para un conjunto S de puntos con $|S| > 3$, la orientación $O(S)$ está dada por:

$$O(S) = \{o(i, j, k) \mid i, j, k \in S \wedge i \neq j \neq k\}$$

por ejemplo:

Sea $A = \{a, b, c, d\}$ un conjunto de puntos en $\mathbb{R}^2$, existen $\frac{4!}{(4-3)!} = 24$ tripletas ordenadas de puntos:

$(a, b, c), (a, c, b), (b, a, c), (b, c, a), (c, a, b), (c, b, a),$

$(a, b, d), (a, d, b), (b, a, d), (b, d, a), (d, a, b), (d, b, a),$

$(a, c, d), (a, d, c), (c, a, d), (c, d, a), (d, a, c), (d, c, a),$

$(b, c, d), (b, d, c), (c, b, d), (c, d, b), (d, b, c), (d, c, b)$

entonces la orientación de A es:

$$o(i,j,k) = \det \begin{pmatrix} i_x & j_x & k_x \\ i_y & j_y & k_y \\ 1 & 1 & 1 \end{pmatrix}$$

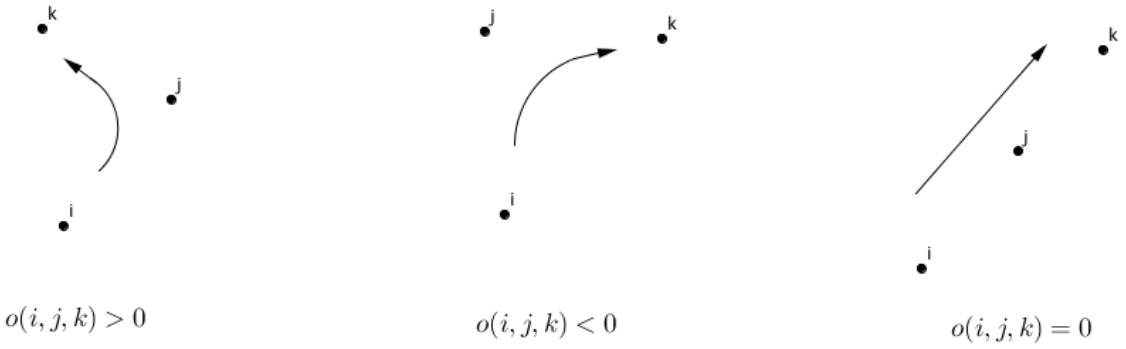


Figura 2-6: Orientación de una tripleta de puntos en $\mathbb{R}^2$

$O(A) = \{$

$o(a,b,c), o(a,c,b), o(b,a,c), o(b,c,a), o(c,a,b), o(c,b,a),$

$o(a,b,d), o(a,d,b), o(b,a,d), o(b,d,a), o(d,a,b), o(d,b,a),$

$o(a,c,d), o(a,d,c), o(c,a,d), o(c,d,a), o(d,a,c), o(d,c,a),$

$o(b,c,d), o(b,d,c), o(c,b,d), o(c,d,b), o(d,b,c), o(d,c,b)\}$

Se dice que dos conjuntos de puntos S y T *tienen la misma orientación* si $O(S) = O(T)$. El cálculo de la orientación de un conjunto de puntos en $\mathbb{R}^2$ requiere de un tiempo $\mathcal{O}(n!)$, de modo que es poco útil. Es mejor utilizar el concepto aplicado solamente a tripletas de puntos.

2.3. Definiciones

2.3.1. Semejanza de conjuntos de puntos en $\mathbb{R}^2$

Es necesario definir *la relación de semejanza* entre dos conjuntos de puntos en $\mathbb{R}^2$, a fin de que tengan sentido las comparaciones objeto de este trabajo.

En un conjunto de puntos en $\mathbb{R}^2$ la distancia $d(i,j)$ entre dos de ellos esta dada por:

$$d(i,j) = \sqrt{(i_x - j_x)^2 + (i_y - j_y)^2}$$

Se le llama conjunto de distancias de S al conjunto:

$$D(S) = \{d(i,j) \mid i,j \in S \wedge i \neq j\}$$

Sean S y T dos conjuntos de puntos en $\mathbb{R}^2$, decimos que *S es semejante a T* si y sólo si:

- $|S| = |T|$ (tienen el mismo número de puntos).
- $O(S) = O(T)$ (tienen la misma orientación).
- $D(S) = \{c \cdot d \mid d \in D(T), c \in \mathbb{R}, c \neq 0\}$ (las distancias entre puntos son proporcionales).

Las transformaciones de traslación, rotación y escalamiento, tienen la característica de conservar la orientación de los puntos sobre los que se aplican. La traslación y la rotación no modifican las distancias entre los puntos. La escalación por su parte puede modificar las distancias, pero lo hace con todas y en la misma proporción. Si se aplica una sucesión de transformaciones a un conjunto de puntos, el conjunto resultante será *semejante* al original bajo esta definición.

2.3.2. Sucesión cíclica

Una sucesión matemática es una función $f : \mathbb{N} \to P$ que define un ordenamiento particular de los elementos del conjunto P, los cuales pueden aparecer más de una vez. La función indica el elemento del conjunto que se debe tomar para una posición específica.

Utilizando aritmética modular, sea $D = \{x \in \mathbb{N} \mid 0 \leq x \leq n-1\}$ donde n es el número de elementos de la sucesión.

Si

$$f : D \to P$$
$$g : \mathbb{N} \to D$$
$$g(x) = x \quad \text{mód } n$$

entonces, la función compuesta

$$f \circ g : \mathbb{N} \to P$$

es una *sucesión cíclica*.

La aritmética modular es especialmente útil para modelar comportamientos de anillo, que de forma similar a un reloj elimina los conceptos de inicio y fin. Por ejemplo, en un reloj después de las 12 horas sigue la una, en lugar de las 13.

De acuerdo a esta definición, la sucesión cíclica siguiente:

$$(p_0,\ p_1,\ p_2,\ p_3,\ p_4)$$

es posible escribirla de las siguientes formas:

$$(p_1,\ p_2,\ p_3,\ p_4,\ p_0)$$
$$(p_2,\ p_3,\ p_4,\ p_0,\ p_1)$$
$$(p_3,\ p_4,\ p_0,\ p_1,\ p_2)$$
$$(p_4,\ p_0,\ p_1,\ p_2,\ p_3)$$

2.3.3. Firma angular cíclica

Sea $P = (p_0,\ p_1,\ ...,\ p_n)$ una sucesión, formada por los elementos de un conjunto de puntos en $\mathbb{R}^2$, en posición convexa ordenados en el sentido de las manecillas del reloj.

Sea r_i la distancia comprendida entre p_i y p_{i+1}, r_n la distancia comprendida entre p_n y p_0, $r_m = min\{r_0,\ r_1,\ ...,\ r_n\}$ y α_i el ángulo interno en p_i.

Sea f_i la pareja ordenada $(r_i/r_m,\ \alpha_i)$.

La *firma angular cíclica de* P, $FAC(P)$, es la sucesión cíclica $(f_0,\ f_1,\ ...,\ f_n)$, como se muestra en la figura 2-7.

El cálculo de α se efectúa de la siguiente manera:

Sea $\vec{A}$ el vector formado por los puntos $\overrightarrow{p_{i-1}p_i}$ y $\vec{B}$ el vector formado por los puntos $\overrightarrow{p_ip_{i+1}}$, tenemos la siguiente relación:

$$\cos\alpha \ = \ \frac{-\vec{A}\cdot\vec{B}}{\|\vec{A}\|\cdot\|\vec{B}\|}$$

como se muestra en la figura 2-8.

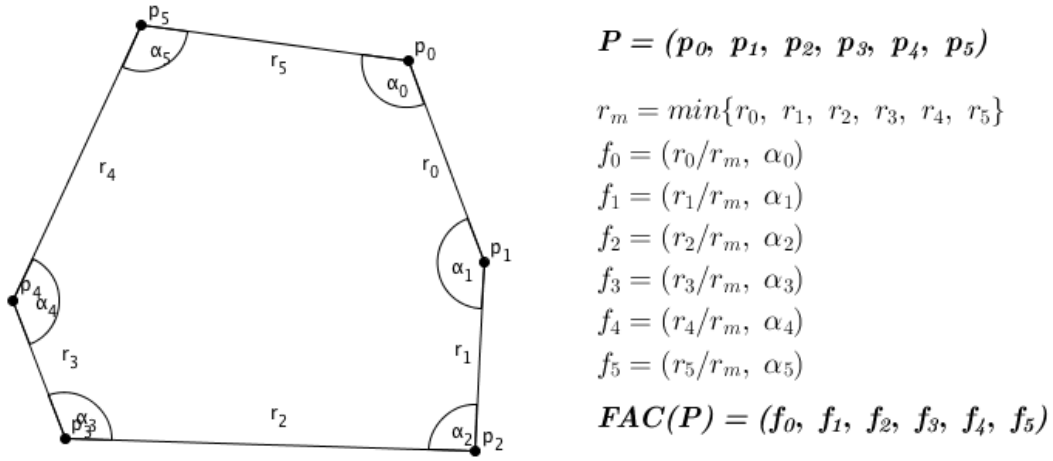


Figura 2-7: Firma angular cíclica.

2.3.4. Firma radial cíclica

Sea S un conjunto de puntos en $\mathbb{R}^2$ y $c \notin S$ otro punto también en $\mathbb{R}^2$.

Sea $P = (p_0,\ p_1,\ ...,\ p_n)$ la sucesión cíclica formada por los puntos de S ordenados radialmente en torno a c.

Sea r_i la distancia entre c y p_i, $r_m = min\{r_0,\ r_1,\ ...,\ r_n\}$ y α_i el ángulo con vértice en c, comprendido entre dos puntos consecutivos p_i y p_{i+1}.

Sea f_i la pareja ordenada $(r_i/r_m,\ \alpha_i)$.

La *firma radial cíclica de P con respecto a c*, $FRC(P, c)$, es la sucesión cíclica $(f_0,\ f_1,\ ...,\ f_n)$, como se muestra en la figura 2-9.

No hay restricción en que los puntos de $S \cup \{c\}$ se encuentren en posición general, por lo que es posible, c se encuentre alineado con dos puntos de S. Al ordenarlos radialmente en torno a c quedarán ambos en la misma posición dentro de la secuencia, al formar la FRC estos empates se romperán utilizando su valor r_i, tomando en primer lugar el punto más cercano a c.

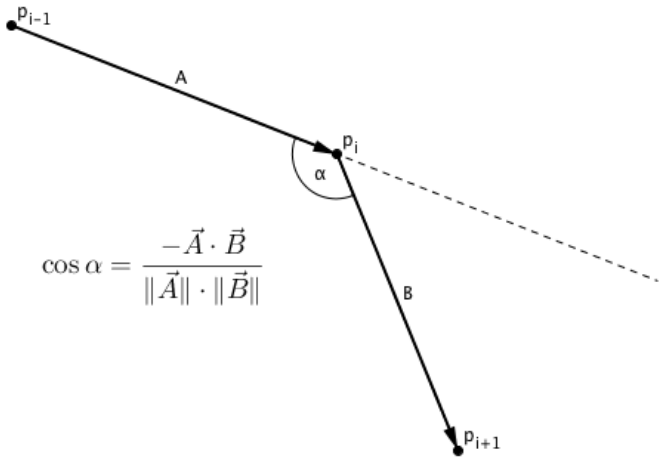


Figura 2-8: Cálculo de α para la firma angular cíclica.

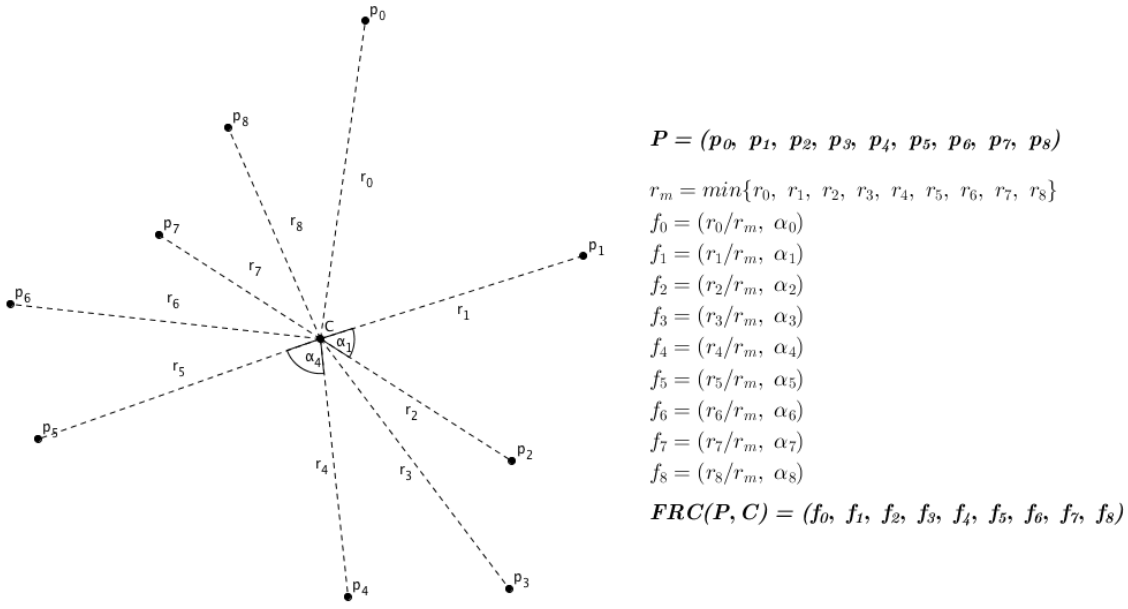


Figura 2-9: Firma radial cíclica.

Capítulo 3

Desarrollo del algoritmo

3.1. Planteamiento

La estrategia del algoritmo es obtener las capas convexas de ambos conjuntos de puntos y compararlas sucesivamente hasta que sean diferentes, encontrar el punto diferencia si es que hay, o bien concluir que son semejantes. Tiene como entrada dos conjuntos de puntos en $\mathbb{R}^2$ en posición general y obtiene como salida:

- *Los conjuntos son semejantes.*
- *Los conjuntos son diferentes.*
- *Los conjuntos son diferentes y el punto diferencia es x.*

Se hace uso de los siguientes conceptos:

- *Punto diferencia.* Es aquel que resulta de la comparación de dos conjuntos tal que al eliminarlo los conjuntos son semejantes.
- *Capa convexa.* Es el cierre convexo de un subconjunto de puntos.
- *Capa convexa consecutiva.* Es el cierre convexo del subconjunto formado por los puntos que no pertenecen al cierre convexo del conjunto original.

La extracción de las capas convexas consecutivas está basada en el algoritmo diseñado por Bernard Chazelle en 1985. Este, obtiene todas las capas convexas consecutivas de un conjunto

de puntos en tiempo $\mathcal{O}(n \log n)$. Construye una estructura de datos a partir de la cual obtiene el cierre convexo del conjunto de puntos. Elimina de la estructura de datos los puntos que se encuentran en dicho cierre convexo en tiempo constante para cada punto, de modo que se puede obtener la siguiente capa convexa consecutiva. La construcción inicial de la estructura de datos se efectúa en tiempo $\mathcal{O}(n \log n)$ y la eliminación posterior de todos los puntos en tiempo $\mathcal{O}(n)$, por lo que el tiempo total es $\mathcal{O}(n \log n)$ [Chazelle, 1985]. En el apéndice A se da una descripción detallada del funcionamiento de dicho algoritmo.

La comparación de las capas convexas se efectúa en dos pasos:

1. Se analiza la correspondencia de los vértices, de los polígonos descritos por las capas convexas y se localiza el posible punto diferencia.

2. Si las capas convexas en el mismo nivel fueron semejantes, se analiza la correspondencia entre capas convexas consecutivas.

Estos análisis de las capas convexas se basan en el algoritmo KMP diseñado por Knuth, Morris y Pratt en 1977 para la búsqueda de subcadenas. En este caso el algoritmo es modificado para comparar las firmas cíclicas obtenidas de las capas convexas y, poder compararlas en tiempo lineal [Knuth, Morris, y Pratt, 1977]. En el apéndice B se da una descripción detallada del funcionamiento de dicho algoritmo.

3.2. Algoritmo principal

El algoritmo 1 se encarga de comparar dos conjuntos de puntos en $\mathbb{R}^2$, de la misma cardinalidad o con diferencia de uno, cada cual con su propio marco de referencia. Al finalizar indica si los conjuntos son semejantes, si son diferentes, o si son diferentes en un punto lo identifica como el punto diferencia.

Las entradas son:

- Conjunto S.
- Conjunto T.

Las posibles salidas son:

- **(0)** si los conjuntos son semejantes. Este caso implica que los conjuntos S y T, tienen el mismo número de puntos y fue posible construir una correspondencia uno a uno entre ellos, por lo que son semejantes.
- **(1)** si los conjuntos son diferentes. Este caso implica que no fue posible construir una correspondencia uno a uno entre los puntos de los conjuntos S y T, por lo que son diferentes.
- **(2, $\boldsymbol{x}$)** si los conjuntos son diferentes y fue identificado el punto diferencia x. Este caso implica que uno de los dos conjuntos tiene un punto más que el otro, tal que si lo eliminamos, sería posible construir una correspondencia uno a uno entre los puntos restantes.

Algoritmo 1 Comparación de dos conjuntos de puntos en $\mathbb{R}^2$

Entrada: Conjuntos de puntos $\boldsymbol{S}$ y $\boldsymbol{T}$

Salida: **(0)** si $S = T$, **(1)** si $S \neq T$ y **(2, $\boldsymbol{x}$)** si $S \neq T$ pero $S \setminus \{x\} = T \setminus \{x\}$

1: **si** $abs(|\boldsymbol{S}| - |\boldsymbol{T}|) = 1$ **entonces**
2: $\boldsymbol{hayPuntoDiferencia}$ = **cierto**
3: **si no**
4: **si** $|\boldsymbol{S}| \neq |\boldsymbol{T}|$ **entonces**
5: **devolver (1)**
6: **si no**
7: $\boldsymbol{hayPuntoDiferencia}$ = **falso**
8: **fin si**
9: **fin si**
10: usar el algoritmo de Bernard Chazelle (apéndice A) con $\boldsymbol{S}$ y $\boldsymbol{T}$ y dejar las capas convexas en los arreglos $\boldsymbol{CHS}$ y $\boldsymbol{CHT}$
11: **para** $\boldsymbol{i}$ de **1** al tamaño de $\boldsymbol{CHS}$ o al tamaño de $\boldsymbol{CHT}$, el que resulte menor **hacer**
12: usar el algoritmo 3 con $\boldsymbol{i}$, $\boldsymbol{CHS}$, $\boldsymbol{CHT}$ y $\boldsymbol{hayPuntoDiferencia}$ y dejar la salida en $(\boldsymbol{compF}, \boldsymbol{v})$
13: **si** $\boldsymbol{compF} = \mathbf{2}$ y $\boldsymbol{hayPuntoDiferencia}$ = **cierto entonces**
14: marcar punto diferencia $\boldsymbol{u} = \boldsymbol{v}$ y salir del ciclo
15: **si no**
16: **si** $\boldsymbol{compF} \neq \mathbf{0}$ **entonces**

```
17:          devolver (1)
18:        fin si
19:      fin si
20:  fin para
21:  si hay punto marcado u entonces
22:    usar el algoritmo 2 con S, T y u y dejar la salida en fase2
23:    si fase2 = cierto entonces
24:      x = u
25:      devolver (2, x)
26:    si no
27:      devolver (1)
28:    fin si
29:  si no
30:    si tamaño de CHS ≠ tamaño de CHT entonces
31:      si hayPuntoDiferencia = cierto entonces
32:        x = punto en la capa no procesada de CHS o CHT
33:        devolver (2, x)
34:      si no
35:        devolver (1)
36:      fin si
37:    si no
38:      devolver (0)
39:    fin si
40:  fin si
```

3.3. Algoritmo de verificación del punto diferencia

El algoritmo 2 se encarga de la verificación del punto diferencia. Para ello, ejecuta nuevamente la comparación de las capas convexas pero no considera el punto que fue marcado en el algoritmo 1.

Esta fase sólo se ejecuta si, la diferencia en las cardinalidades de los conjuntos es igual a uno

y el punto marcado pertenece al conjunto con más puntos. Al no considerar el punto marcado, ambos conjuntos pueden ser semejantes.

Las entradas son:

- Conjunto S.
- Conjunto T.
- Punto marcado u.

Las posibles salidas son:

- **cierto** si $S \setminus \{u\} = T \setminus \{u\}$.
- **falso** si $S \setminus \{u\} \neq T \setminus \{u\}$.

Algoritmo 2 Comparación de dos conjuntos de puntos en $\mathbb{R}^2$, sin considerar el posible punto diferencia

---

Entrada: Conjuntos de puntos $\boldsymbol{S}$ y $\boldsymbol{T}$ y el posible punto diferencia $\boldsymbol{u}$

Salida: **cierto** si $S \setminus \{u\} = T \setminus \{u\}$ y **falso** si $S \setminus \{u\} \neq T \setminus \{u\}$

1: usar el algoritmo de Bernard Chazelle (apéndice A) con $\boldsymbol{S} \setminus \{\boldsymbol{u}\}$ y $\boldsymbol{T} \setminus \{\boldsymbol{u}\}$ y dejar las capas convexas en los arreglos $\boldsymbol{CHS}$ y $\boldsymbol{CHT}$
2: **si** tamaño de $\boldsymbol{CHS} \neq$ tamaño de $\boldsymbol{CHT}$ **entonces**
3: **devolver falso**
4: **fin si**
5: **para** $\boldsymbol{i}$ de **1** al tamaño de $\boldsymbol{CHS}$ **hacer**
6: usar el algoritmo 4 con $\boldsymbol{i}$, $\boldsymbol{CHS}$ y $\boldsymbol{CHT}$ y dejar la salida en ***compQ***
7: **si** ***compQ*** $=$ **falso entonces**
8: **devolver falso**
9: **fin si**
10: **fin para**
11: **devolver cierto**

3.4. Algoritmo de comparación de capas convexas buscando el punto diferencia

El algoritmo 3 es invocado por el algoritmo 1 para cada nivel de capas convexas calculadas. Su función es comparar las capas convexas de ambos conjuntos que se encuentran al mismo nivel, iniciando con la más externa. La comparación es en base a la correspondencia que tienen entre sí los puntos de la propia capa, así como la correspondencia que tienen con los puntos de la capa convexa contigua exterior, en caso de no ser la más externa. Durante el análisis de correspondencia de los puntos entre sí se identifica el punto diferencia.

Las entradas son:

- $\boldsymbol{i}$, nivel de las capas convexas a comparar, la más externa es la número 1.
- $\boldsymbol{CHS}$, arreglo de capas convexas del conjunto S.
- $\boldsymbol{CHT}$, arreglo de capas convexas del conjunto T.
- $\boldsymbol{hayPuntoDiferencia}$, indicador para considerar la posible existencia del punto diferencia.

Las posibles salidas son:

- **(0)** si $CHS_i = CHT_i$. Las capas convexas de ambos conjuntos al mismo nivel, son semejantes.
- **(1)** si $CHS_i \neq CHT_i$. Las capas convexas de ambos conjuntos al mismo nivel, son diferentes en más de un punto.
- **(2, $\boldsymbol{v}$)** si $CHS_i \neq CHT_i$. Las capas convexas de ambos conjuntos al mismo nivel, son diferentes y el punto diferencia es $\boldsymbol{v}$.

Algoritmo 3 Comparación de capas convexas buscando el punto diferencia

Entrada: Nivel al comparar $\boldsymbol{i}$, arreglos de capas convexas $\boldsymbol{CHS}$ y $\boldsymbol{CHT}$ e indicador para el posible punto diferencia $\boldsymbol{hayPuntoDiferencia}$

Salida: **(0)** si $CHS_i = CHT_i$, **(1)** si $CHS_i \neq CHT_i$ en más de un punto, **(2, $\boldsymbol{v}$)** si $CHS_i \neq CHT_i$ y el punto diferencia es $\boldsymbol{v}$

1: calcular $\boldsymbol{facS} = FAC(\boldsymbol{CHS}_i)$ y $\boldsymbol{facT} = FAC(\boldsymbol{CHT}_i)$
2: usar el algoritmo 5 con $\boldsymbol{facS}$ y $\boldsymbol{facT}$ y dejar la salida en $(\boldsymbol{compFAC}, \boldsymbol{w})$
3: **si** $\boldsymbol{compFAC} = \mathbf{2}$ **entonces**
4: **si** $\boldsymbol{hayPuntoDiferencia} =$ **cierto entonces**
5: marcar punto diferencia $\boldsymbol{v} = \boldsymbol{w}$
6: **devolver (2,** $\boldsymbol{v}$**)**
7: **si no**
8: **devolver (1)**
9: **fin si**
10: **si no**
11: **si** $\boldsymbol{compFAC} = \mathbf{1}$ **entonces**
12: **devolver (1)**
13: **fin si**
14: **fin si**
15: **si** $\boldsymbol{i} > 1$ **entonces**
16: calcular los centroides $\boldsymbol{c_S}$ y $\boldsymbol{c_T}$ de $\boldsymbol{CHS}_i$ y $\boldsymbol{CHT}_i$ respectivamente
17: calcular $\boldsymbol{frcS} = FRC(\boldsymbol{CHS}_i \cup \boldsymbol{CHS}_{i-1}, \boldsymbol{c_S})$ y $\boldsymbol{frcT} = FRC(\boldsymbol{CHT}_i \cup \boldsymbol{CHT}_{i-1}, \boldsymbol{c_T})$
18: usar el algoritmo 6 con $\boldsymbol{frcS}$ y $\boldsymbol{frcT}$ y dejar la salida en $\boldsymbol{compFRC}$
19: **si** $\boldsymbol{compFRC} =$ **cierto entonces**
20: **devolver (0)**
21: **si no**
22: **devolver (1)**
23: **fin si**
24: **si no**
25: **devolver (0)**
26: **fin si**

3.5. Algoritmo de comparación de capas convexas sin buscar el punto diferencia

El algoritmo 4 es invocado por el algoritmo 2 para cada nivel de capas convexas calculadas. Su función es comparar las capas convexas de ambos conjuntos que se encuentran al mismo nivel, iniciando con la más externa. La comparación es en base a la correspondencia que tienen entre sí los puntos de la propia capa, así como la correspondencia que tienen con los puntos de la capa convexa contigua exterior, en caso de no ser la más externa.

Las entradas son:

- $\boldsymbol{i}$, nivel de las capas convexas a comparar, la más externa es la número 1.
- $\boldsymbol{CHS}$, arreglo de capas convexas del conjunto S.
- $\boldsymbol{CHT}$, arreglo de capas convexas del conjunto T.

Las posibles salidas son:

- **cierto** si $CHS_i = CHT_i$. Las capas convexas de ambos conjuntos al mismo nivel, son semejantes.
- **falso** si $CHS_i \neq CHT_i$. Las capas convexas de ambos conjuntos al mismo nivel, son diferentes.

Algoritmo 4 Comparación de capas convexas sin buscar el punto diferencia

Entrada: Nivel al comparar $\boldsymbol{i}$ y arreglos de capas convexas $\boldsymbol{CHS}$ y $\boldsymbol{CHT}$

Salida: **cierto** si $CHS_i = CHT_i$, **falso** si $CHS_i \neq CHT_i$

1: calcular $\boldsymbol{facS} = FAC(\boldsymbol{CHS}_i)$ y $\boldsymbol{facT} = FAC(\boldsymbol{CHT}_i)$
2: usar el algoritmo 5 con $\boldsymbol{facS}$ y $\boldsymbol{facT}$ y dejar la salida en $(\boldsymbol{compFAC}, \boldsymbol{w})$
3: **si** $\boldsymbol{compFAC} = \boldsymbol{0}$ **entonces**
4: **si** $\boldsymbol{i} > 1$ **entonces**
5: calcular los centroides $\boldsymbol{c_S}$ y $\boldsymbol{c_T}$ de $\boldsymbol{CHS}_i$ y $\boldsymbol{CHT}_i$ respectivamente
6: calcular $\boldsymbol{frcS} = FRC(\boldsymbol{CHS}_i \cup \boldsymbol{CHS}_{i-1}, \boldsymbol{c_S})$ y $\boldsymbol{frcT} = FRC(\boldsymbol{CHT}_i \cup \boldsymbol{CHT}_{i-1}, \boldsymbol{c_T})$
7: usar el algoritmo 6 con $\boldsymbol{frcS}$ y $\boldsymbol{frcT}$ y dejar la salida en $\boldsymbol{compFRC}$
8: **si** $\boldsymbol{compFRC}$ = **cierto** **entonces**

```
 9:          devolver cierto
10:       si no
11:          devolver falso
12:       fin si
13:    si no
14:       devolver cierto
15:    fin si
16: si no
17:    devolver falso
18: fin si
```

3.6. Algoritmo de comparación de firmas angulares cíclicas

El algoritmo 5 se encarga de comparar las FAC de dos capas convexas, basándose en el algoritmo KMP descrito en el apéndice B. Para la implementación del algoritmo KMP con firmas cíclicas en lugar de cadenas, es necesario hacer algunas modificaciones que no alteran la complejidad del mismo.

El algoritmo KMP se divide en dos fases, en la primera se calcula la matriz de prefijos de la cadena a ser buscada, en la segunda fase se recorre la cadena donde se busca. Una cadena tiene inicio y fin, situación que no ocurre con una firma cíclica, por lo tanto es necesario elegir un punto dentro de la misma como inicio, el fin queda determinado al dar una vuelta y regresar a ese punto. Así mismo, el comportamiento cíclico de la firma permite recorrerla en ambos sentidos, para lo cual se considera sentido positivo si el indice de las posiciones se incrementa, se considera sentido negativo, si el indice de las posiciones se decrementa. Adicional a lo anterior, es necesario registrar el tamaño de la subcadena común más larga, así como su posición final en ambas firmas.

Las entradas son:

- $\boldsymbol{F_S}$, FAC de la capa convexa del conjunto S.
- $\boldsymbol{F_T}$, FAC de la capa convexa del conjunto T.

Las posibles salidas son:

- **(0)** si $F_S = F_T$. Las FAC son iguales.
- **(1)** si $F_S \neq F_T$. Las FAC son diferentes en más de un punto.
- **(2, w)** si $F_S \neq F_T$. Las FAC son diferentes y el punto diferencia es w.

Algoritmo 5 Comparación de firmas angulares cíclicas

Entrada: FAC de las capa convexas $\boldsymbol{F_S}$ y $\boldsymbol{F_T}$

Salida: **0** si $F_S = F_T$, **1** si $F_S \neq F_T$ en más de un punto, **(2, w)** si $F_S \neq F_T$ y el punto diferencia es $\boldsymbol{w}$

si $|\boldsymbol{F_S}| = |\boldsymbol{F_T}|$ **entonces**
  elegir de forma aleatoria $1 \leq \boldsymbol{s_i} \leq |\boldsymbol{F_S}|$ como inicio de $\boldsymbol{F_S}$
  calcular la matriz de prefijos $\boldsymbol{MP_S}$ de $\boldsymbol{F_S}$ en sentido positivo
  elegir de forma aleatoria $1 \leq \boldsymbol{t_i} \leq |\boldsymbol{F_T}|$ como inicio de $\boldsymbol{F_T}$
  usar el algoritmo *KMP* en sentido positivo, con $\boldsymbol{F_S}$, $\boldsymbol{F_T}$ y $\boldsymbol{MP_S}$ y dejar la salida en $\boldsymbol{firmasIguales}$, $\boldsymbol{longSCML}$, $\boldsymbol{s_m}$ y $\boldsymbol{t_m}$
  si $\boldsymbol{firmasIguales}$ = **cierto entonces**
    devolver (0)
  si no
    elegir $\boldsymbol{s_m}$ como inicio de $\boldsymbol{F_S}$
    calcular la matriz de prefijos $\boldsymbol{MP_S}$ de $\boldsymbol{F_S}$ en sentido negativo
    usar el algoritmo *KMP* en sentido negativo, con $\boldsymbol{F_S}$, $\boldsymbol{F_T}$ y $\boldsymbol{MP_S}$ y dejar la salida en $\boldsymbol{firmasIguales}$, $\boldsymbol{longSCML}$, $\boldsymbol{s_m}$ y $\boldsymbol{t_m}$
    si $\boldsymbol{firmaIguales}$ = **cierto entonces**
      devolver (0)
    si no
      si $\boldsymbol{longSCML} = |\boldsymbol{F_S}| - 3$ **entonces**
        $\boldsymbol{w}$ = punto de $\boldsymbol{F_S}$ en la posición $\boldsymbol{s_m} - 2$
        devolver (2, w)
      si no
        devolver (1)
      fin si
    fin si

22: **fin si**

23: **si no**

24: $\boldsymbol{A}$ = firma más larga entre $\boldsymbol{F_S}$ y $\boldsymbol{F_T}$

25: $\boldsymbol{B}$ = firma más corta entre $\boldsymbol{F_S}$ y $\boldsymbol{F_T}$

26: elegir de forma aleatoria $1 \leq \boldsymbol{a_i} \leq |\boldsymbol{A}|$ como inicio de $\boldsymbol{A}$

27: calcular la matriz de prefijos $\boldsymbol{MP_A}$ de $\boldsymbol{A}$ en sentido positivo

28: elegir de forma aleatoria $1 \leq \boldsymbol{b_i} \leq |\boldsymbol{B}|$ como inicio de $\boldsymbol{B}$

29: usar el algoritmo *KMP* en sentido positivo, con $\boldsymbol{A}$, $\boldsymbol{B}$ y $\boldsymbol{MP_A}$ y dejar la salida en $\boldsymbol{firmasIguales}$, $\boldsymbol{longSCML}$, $\boldsymbol{a_m}$ y $\boldsymbol{b_m}$

30: **si** $|\boldsymbol{A}| = |\boldsymbol{B}| - 1$ **entonces**

31: **si** $\boldsymbol{longSCML} = |\boldsymbol{B}| - 3$ **entonces**

32: $\boldsymbol{w}$ = punto de $\boldsymbol{B}$ en la posición $\boldsymbol{b_m} + 2$

33: **devolver (2,** $\boldsymbol{w}$**)**

34: **fin si**

35: **fin si**

36: **si** $|\boldsymbol{A}| \leq |\boldsymbol{B}| - 2$ **entonces**

37: **si** $\boldsymbol{longSCML} = |\boldsymbol{A}| - 3$ **entonces**

38: $\boldsymbol{w}$ = punto de $\boldsymbol{A}$ en la posición $\boldsymbol{a_m} + 2$

39: **devolver (2,** $\boldsymbol{w}$**)**

40: **fin si**

41: **fin si**

42: elegir $\boldsymbol{a_m}$ como inicio de $\boldsymbol{A}$

43: calcular la matriz de prefijos $\boldsymbol{MP_A}$ de $\boldsymbol{A}$ en sentido negativo

44: elegir $\boldsymbol{b_m}$ como inicio de $\boldsymbol{B}$

45: usar el algoritmo *KMP* en sentido negativo, con $\boldsymbol{A}$, $\boldsymbol{B}$ y $\boldsymbol{MP_A}$ y dejar la salida en $\boldsymbol{firmasIguales}$, $\boldsymbol{longSCML}$, $\boldsymbol{a_m}$ y $\boldsymbol{b_m}$

46: **si** $|\boldsymbol{A}| = |\boldsymbol{B}| - 1$ **entonces**

47: **si** $\boldsymbol{longSCML} = |\boldsymbol{B}| - 3$ **entonces**

48: $\boldsymbol{w}$ = punto de $\boldsymbol{B}$ en la posición $\boldsymbol{b_m} - 2$

49: **devolver (2,** $\boldsymbol{w}$**)**

50: **fin si**

51: **fin si**

52: **si** $|\boldsymbol{A}| \leq |\boldsymbol{B}| - 2$ **entonces**

53: **si** $\boldsymbol{longSCML} = |\boldsymbol{A}| - 3$ **entonces**

54: $\boldsymbol{w}$ = punto de $\boldsymbol{A}$ en la posición $\boldsymbol{a_m} - 2$

55: **devolver (2,** $\boldsymbol{w}$**)**

56: **fin si**

57: **fin si**

58: **devolver (1)**

59: **fin si**

3.7. Algoritmo de comparación de firmas radiales cíclicas

El algoritmo 6 se encarga de comparar las *FRC* de dos conjuntos de puntos, basándose en el algoritmo *KMP* descrito en el apéndice B. Es necesario considerar las mismas modificaciones descritas en el algoritmo de comparación de FACs.

En este caso, los conjuntos de puntos son el resultado de la unión de dos capas convexas consecutivas, de modo que las FRC calculadas contienen la relación de posición que guardan dichas capas entre sí.

Las entradas son:

- $\boldsymbol{F_S}$, FRC del subconjunto de S.
- $\boldsymbol{F_T}$, FRC del subconjunto de T.

Las posibles salidas son:

- **cierto** si $F_S = F_T$. Las FRC son iguales.
- **falso** si $F_S \neq F_T$. Las FRC son diferentes.

Algoritmo 6 Comparación de firmas radiales cíclicas

Entrada: FRC de los subconjuntos de S y T, $\boldsymbol{F_S}$ y $\boldsymbol{F_T}$

Salida: **cierto** si $F_S = F_T$, **falso** si $F_S \neq F_T$

1: elegir de forma aleatoria $1 \leq \boldsymbol{s_i} \leq |\boldsymbol{F_S}|$ como inicio de $\boldsymbol{F_S}$

2: calcular la matriz de prefijos $\boldsymbol{MP_S}$ de $\boldsymbol{F_S}$ en sentido positivo

3: elegir de forma aleatoria $1 \leq \boldsymbol{t_i} \leq |\boldsymbol{F_T}|$ como inicio de $\boldsymbol{F_T}$

4: usar el algoritmo *KMP* en sentido positivo, con $\boldsymbol{F_S}$, $\boldsymbol{F_T}$ y $\boldsymbol{MP_S}$ y dejar la salida en $\boldsymbol{firmasIguales}$, $\boldsymbol{longSCML}$, $\boldsymbol{s_m}$ y $\boldsymbol{t_m}$

5: **si** $\boldsymbol{firmasIguales}$ = **cierto entonces**

6: **devolver cierto**

7: **si no**

8: **devolver falso**

9: **fin si**

Capítulo 4

Fundamentación teórica

4.1. Base de la comparación

La comparación de los conjuntos de puntos en $\mathbb{R}^2$ se basa en la semejanza geométrica, resultante de los polígonos formados por la unión de los puntos mediante segmentos. De esta forma, se establece la relación entre el aspecto geométrico y el analítico del problema. Si bien es cierto que se pueden obtener múltiples polígonos a partir de un conjunto de puntos, lo contrario no es cierto, ya que de un polígono dado, el conjunto de puntos obtenido a partir de sus vértices es único. Esto lo formalizamos con el siguiente lema.

Lema 3 *Sean P y Q dos polígonos, si P y Q son semejantes geométricamente, entonces los conjuntos de puntos en $\mathbb{R}^2$ formados por los vértices de P y Q, son semejantes.*

Prueba. Si P y Q son semejantes, entonces P puede ser superpuesto sobre Q mediante transformaciones afines (rotación, traslación y escalado), de modo que los vértices de P corresponderán uno a uno con los vértices de Q. ■

Más específicamente nos interesa comparar:

- La firma angular cíclica que corresponde a una poligonal convexa.
- La firma radial cíclica que corresponde a una poligonal estrellada.

La comparación de las poligonales es posible efectuarla a partir de los vértices de las mismas, sin embargo, dichas poligonales tienen marcos de referencia distintos, así que dicha comparación

no debe depender del marco de referencia. El uso de las firmas cíclicas es adecuado para la comparación, ya que no dependen del cambio de marco de referencia como se formaliza con el siguiente lema.

Lema 4 *La FAC y la FRC no dependen del cambio de marco de referencia.*

Prueba. Los términos individuales de la FAC y la FRC son parejas ordenadas de una distancia y un ángulo. En el caso de las distancias todas son divididas entre la menor de ellas, de modo que quedan normalizadas con respecto a la menor sin importar el valor numérico de la misma. En el caso de los ángulos, son medidos de manera relativa a las posiciones de los puntos y no con respecto a los ejes del sistema de referencia. ■

Dado que existe una relación uno a uno entre los vértices de una poligonal y los términos de sus firmas cíclicas, el tiempo para calcularlas es lineal en función del número de vértices, como se formaliza con el siguiente lema.

Lema 5 *La FAC y la FRC se calculan en tiempo* $\mathcal{O}(n)$.

Prueba. El valor de la distancia entre dos puntos:

$$r \;=\; \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2}$$

se calcula en tiempo constante, de modo que para n puntos el cálculo se efectúa en tiempo $\mathcal{O}(n)$.

La medida del ángulo entre dos vectores y el valor de la pendiente entre dos puntos:

$$m \;=\; \frac{y_a - y_b}{x_a - x_b}$$

se calculan en tiempo constante, de modo que para n puntos el cálculo se efectúa en tiempo $\mathcal{O}(n)$.

Si el valor de los dos componentes del término de la sucesión cíclica se efectúan en tiempo constante, el cálculo total de la sucesión tomará $\mathcal{O}(n)$. ■

El algoritmo propuesto busca la correspondencia uno a uno de los puntos mediante capas convexas, así que se debe verificar que todos los puntos son considerados. El siguiente lema demuestra que todos los puntos de un conjunto pertenecen a alguna capa convexa.

Lema 6 *Si P es un conjunto de puntos en posición general en $\mathbb{R}^2$, entonces $\forall p \in P$ se cumple que $p \in \mathcal{CH}_i(P)$, donde $\mathcal{CH}_i(P)$ es alguna capa convexa de P.*

Prueba. Si P es un conjunto de puntos en posición general en $\mathbb{R}^2$, y $\mathcal{CH}_0(P) = \mathcal{CH}(P)$ es su capa convexa más externa, todos los puntos $P \setminus \mathcal{CH}_0(P)$ estarán en el interior de $\mathcal{CH}_0(P)$, de modo que la siguiente capa convexa será $\mathcal{CH}_1(P) = \mathcal{CH}(P \setminus \mathcal{CH}_0(P))$. Si continuamos de esta forma llegará el momento en el que, en el interior de $\mathcal{CH}_n(P)$ no habrá puntos en virtud de que todos han pertenecido a alguna capa convexa anterior. ■

Dado que todos los puntos de un conjunto pertenecen a alguna capa convexa, la comparación entre dos conjuntos se resume a la comparación de las sucesivas capas convexas, es decir, se inicia comparando las capas convexas más externas de ambos conjuntos, después la siguiente capa convexa en ambos conjuntos y así sucesivamente hasta que todas las capas convexas han sido comparadas y, por lo tanto, todos los puntos. Adicional a la comparación individual entre capas convexas del mismo nivel en los dos conjuntos, es necesario comparar la relación que guardan entre sí las capas convexas consecutivas dentro de un mismo conjunto, esto se debe, a que puede ocurrir que las capas convexas individualmente sean semejantes, pero en posiciones diferentes entre sí.

Al tomar de manera individual, una capa convexa de cada conjunto en el mismo nivel, como por ejemplo al inicio la más externa, la comparación entre ellas se logra utilizando la FAC, como se demuestra con el siguiente lema.

Lema 7 *Si las FAC obtenidas de dos capas convexas pertenecientes a dos conjuntos de puntos diferentes son iguales, entonces los conjuntos de puntos formados con los pertenecientes a cada capa convexa, respectivamente, son semejantes.*

Prueba. La FAC proporciona un ordenamiento particular en que son tomados los puntos de la capa convexa de donde se obtienen. Ese ordenamiento define una poligonal convexa donde las medidas de las aristas se encuentran normalizadas por la más pequeña.

De este modo, si las medidas de las aristas son iguales y las medidas de los ángulos son iguales, entonces dichas poligonales son semejantes y, por lo tanto, los conjuntos de puntos de esas capas convexas son semejantes. ■

Como se mencionó anteriormente, no es suficiente con comparar las capas convexas individualmente, es necesario representar analíticamente la posición que guardan entre sí dos capas convexas consecutivas de un mismo conjunto de puntos. Esta representación, se consigue calculando la FRC de la unión de los conjuntos de puntos, pertenecientes a esas capas convexas consecutivas con respecto al centroide de la capa convexa interior, como se demuestra con el siguiente lema:

Lema 8 *La FRC de dos capas convexas consecutivas, pertenecientes a un mismo conjunto de puntos con respecto al centroide de la capa convexa interior, representa la relación de la posición que guardan entre ellas.*

Prueba. Partimos de las siguientes premisas:

- Todos los puntos de la capa convexa interior se encuentran dentro de la capa convexa exterior.
- El centroide de una capa convexa siempre esta dentro de la misma.
- El centroide no depende del cambio de marco de referencia.
- El centroide de la capa convexa interna siempre está en el interior de la capa convexa externa.

Sea un sistema de coordenadas polares con origen en el centroide de la capa convexa interior y el eje polar que pase por el punto con menor distancia al centroide. La posición de todos los puntos queda explícitamente determinada por sus coordenadas, por lo tanto, la secuencia cíclica resultante de ángulos y distancias contiene las posiciones relativas de todos los puntos entre sí. ■

De manera similar a la comparación de capas convexas individuales utilizando la FAC, se demuestra que es posible comparar capas convexas consecutivas de dos conjuntos diferentes, utilizando la FRC.

Lema 9 *Si las FRC obtenidas de capas convexas consecutivas pertenecientes a dos conjuntos de puntos diferentes son iguales, entonces, los conjuntos de puntos formados con los que pertenecen a las capas convexas consecutivas de cada conjunto respectivamente, son semejantes.*

Prueba. La FRC proporciona un ordenamiento particular en que son tomados los puntos de las capas convexas consecutivas de donde se obtienen. Ese ordenamiento define una poligonal estrellada donde las distancias de los vértices al centroide se encuentran normalizadas por la más pequeña.

De este modo, si las distancias son iguales y las medidas de los ángulos son iguales, entonces dichas poligonales son semejantes y, por lo tanto, los conjuntos de puntos de esas capas convexas consecutivas son semejantes. ■

Cuando se comparan todas las capas convexas, primero de forma individual y después por capas consecutivas para dos conjuntos de puntos diferentes, implica la comparación de la totalidad de los puntos en ambos conjuntos. Esto lo formalizamos con el siguiente lema.

Lema 10 *Para dos conjuntos de puntos en posición general en $\mathbb{R}^2$, si las capas convexas entre ambos conjuntos son semejantes y las posiciones, que guardan entre sí capas convexas consecutivas del mismo conjunto de puntos, comparadas con las del otro conjunto de puntos son semejantes, entonces los dos conjuntos de puntos son semejantes.*

Prueba. Partimos de las siguientes premisas:

- Un conjunto de puntos está formado por una secuencia de capas convexas.
- Para dos conjuntos de puntos S y T, es posible comparar todas las capas convexas $\mathcal{CH}_i(S)$ y $\mathcal{CH}_i(T)$, para el mismo nivel i.
- Para dos conjuntos de puntos S y T, si dos capas convexas $\mathcal{CH}_i(S)$ y $\mathcal{CH}_i(T)$ para el mismo nivel i son semejantes, entonces los centroides correspondientes también serán semejantes.
- Para dos conjuntos de puntos S y T, si dos capas convexas consecutivas $\mathcal{CH}_i(S)$ y $\mathcal{CH}_{i-1}(S)$, guardan la misma posición entre sí que sus correspondientes $\mathcal{CH}_i(T)$ y $\mathcal{CH}_{i-1}(T)$ en el otro conjunto de puntos, entonces los centroides correspondientes de las capas convexas internas, son semejantes.

Al comprar todas las capas convexas una a una entre dos conjuntos de puntos, se puede verificar que sean semejantes entre sí. Después, al comparar las capas convexas consecutivas entre ambos conjuntos de puntos, se puede verificar que las posiciones que guardan entre sí las

capas convexas consecutivas para ambos conjuntos de puntos, también sean semejantes entre sí.

Si todos los puntos de ambos conjuntos son semejantes entre sí, entonces los dos conjuntos de puntos son semejantes entre sí. ■

4.2. Identificación del punto diferencia

Cuando estudiamos el comportamiento de los puntos en las capas convexas, observamos lo siguiente de acuerdo a la figura 4-1.

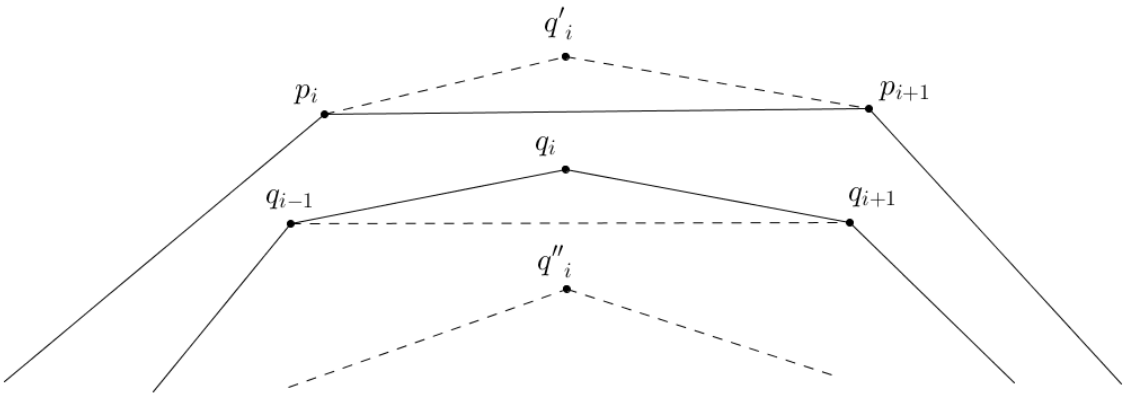


Figura 4-1: Comportamiento de los puntos en las capas convexas.

Si el punto q_i que pertenece a la capa convexa q avanza hacia el segmento $\overline{p_i\ p_{i+1}}$ de la capa convexa p, al rebasarlo, deja de pertenecer a la capa convexa q y se integra a la capa convexa p entre los puntos p_i y p_{i+1}, convirtiéndose en el punto q'_i. Por otro lado, si avanza hacia el segmento $\overline{q_{i-1}\ q_{i+1}}$, al rebasarlo, también deja de pertenecer a la capa convexa q y se integra a la siguiente capa convexa interna, convirtiéndose en el punto q''_i.

Estas observaciones son importantes para determinar el punto diferencia, en el caso de que los conjuntos de puntos tengan diferencia de uno en sus cardinalidades. Dicho análisis procede de la siguiente manera.

Sean S y T dos conjuntos de puntos en posición general en $\mathbb{R}^2$ con $T = S \cup \{x\}$ y $x \notin S$ con sus respectivas capas convexas ya calculadas.

Se observa que los conjuntos son iguales salvo por el punto x, el cual se encuentra en T pero

no en S, sin pérdida de generalidad digamos que la secuencia

$$..., \ p_{i-1}, \ p_i, \ p_{i+1}, \ p_{i+2}, \ ...$$

pertenece al cierre convexo p de S, y que la secuencia

$$..., \ p_{i-1}, \ p_i, \ x, \ p_{i+1}, \ p_{i+2}, \ ...$$

pertenece al cierre convexo p de T, como se muestra en la figura 4-2.

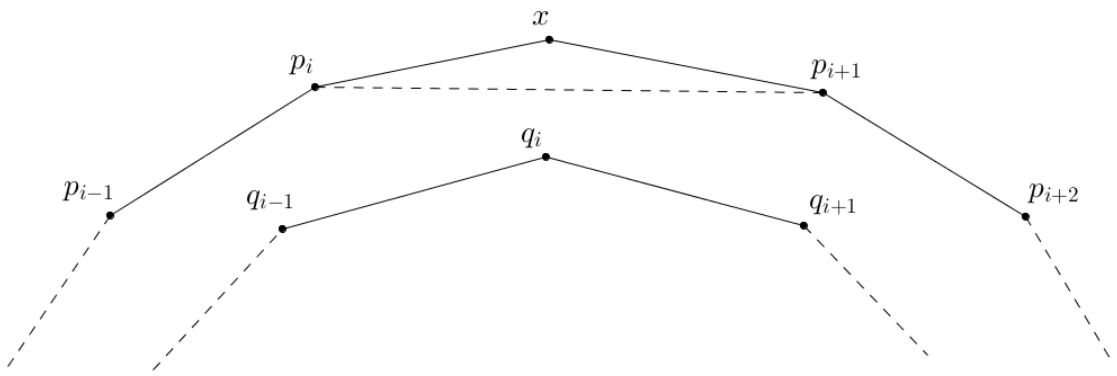


Figura 4-2: Capas convexas para conjuntos con un punto de diferencia.

Dado que el único punto de diferencia entre S y T es x, los puntos de la siguiente capa convexa q son iguales en ambos conjuntos, ello significa que la eliminación de x de T provocaría que las capas convexas p en ambos conjuntos, también sean iguales. Sin embargo, la posición de los puntos de q provocarían que la capa convexa p de S cambiara y ese cambio no se refleja en la capa convexa p de T, ya que todos los puntos de q están en el interior de p.

Dependiendo de la posición de los puntos de q respecto al triángulo $\Delta p_i x p_{i+1}$, se deben analizar los siguientes casos considerando que, la capa convexa p de T siempre incluye la secuencia

$$..., \ p_{i-1}, \ p_i, \ x, \ p_{i+1}, \ p_{i+2}, \ ...$$

4.2.1. Caso 1. $\Delta p_i x p_{i+1}$ está vacío

Es el caso mostrado en la figura 4-2. Al estar todos los puntos de q en el interior de $p_i\ p_{i+1}$, la capa convexa p de S tendrá la secuencia

$$..., \ p_{i-1}, \ p_i, \ p_{i+1}, \ p_{i+2}, \ ...$$

y las capas convexas q de S y T serán iguales.

4.2.2. Caso 2. $\Delta p_i x p_{i+1}$ contiene un punto

En este caso el punto y se encuentra en el interior del triángulo $\Delta p_i x p_{i+1}$, como se observa en la figura 4-3.

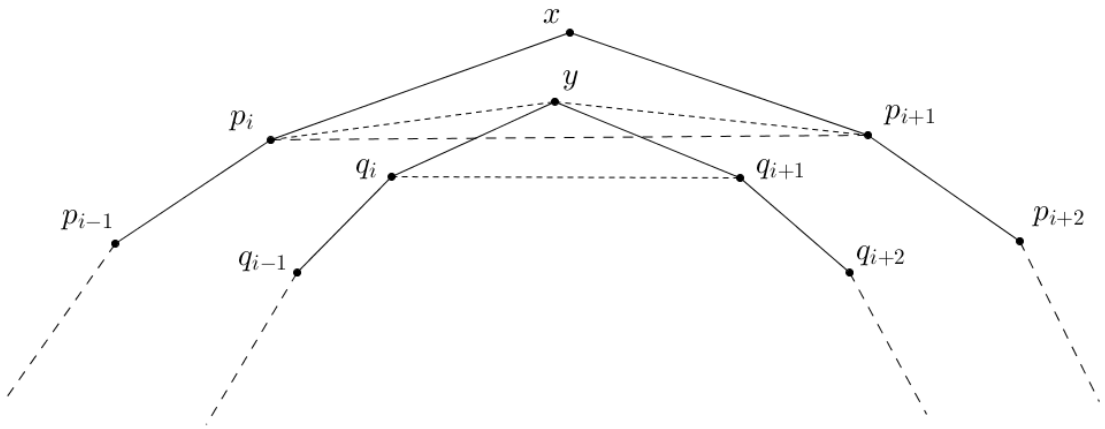


Figura 4-3: Capas convexas diferentes con igual cantidad de puntos.

Aquí la capa convexa p de S tendrá la secuencia

$$..., \ p_{i-1}, \ p_i, \ y, \ p_{i+1}, \ p_{i+2}, \ ...$$

y las capas convexas q serán, para S

$$..., \ q_{i-1}, \ q_i, \ q_{i+1}, \ q_{i+2}, \ ...$$

y para T

$$..., \ q_{i-1}, \ q_i, \ y, \ q_{i+1}, \ q_{i+2}, \ ...$$

4.2.3. Caso 3. $\Delta p_i x p_{i+1}$ contiene más de un punto

En este caso el triángulo $\Delta p_i x p_{i+1}$ contiene los puntos $y_1, \ y_2, \ ... \ , \ y_{m-1}, \ y_m$, como se muestra en la figura 4-4.

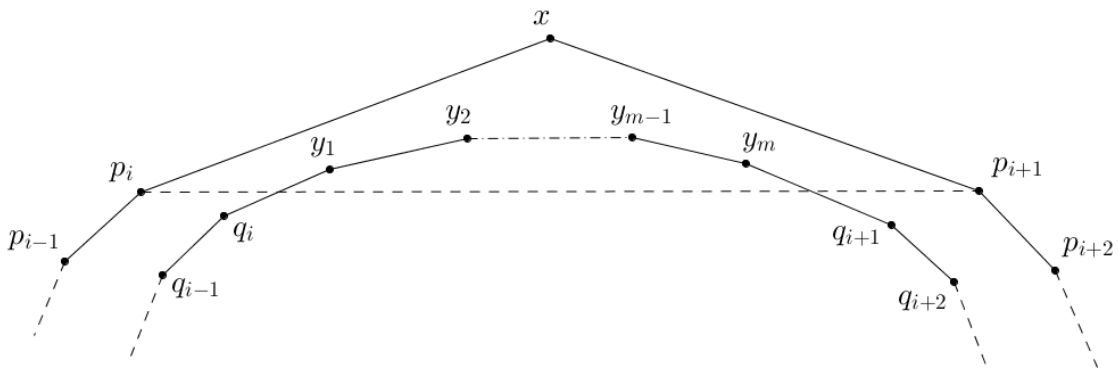


Figura 4-4: Capas convexas diferentes con diferente cantidad de puntos.

Aquí la capa convexa p de S tendrá la secuencia

$$..., \ p_{i-1}, \ p_i, \ y_i, \ y_{i+1}, \ ... \ , \ y_{j-1}, \ y_j, \ p_{i+1}, \ p_{i+2}, \ ...$$

y las capas convexas q serán, para S

$$..., \ q_{i-1}, \ q_i, \ y_1, \ y_2, \ ... \ , \ y_{i-1}, \ y_{j+1}, \ ... \ , y_m, \ q_{i+1}, \ q_{i+2}, \ ...$$

y para T

$$..., \ q_{i-1}, \ q_i, \ y_1, \ y_2, \ ... \ , y_m, \ q_{i+1}, \ q_{i+2}, \ ...$$

donde los puntos y_i y y_j son los puntos de corte a la capa convexa p.

En el interior del triángulo $\Delta p_i x p_{i+1}$ puede haber más puntos de los ya mencionados, pero se consideran sólo los que pertenecen a la capa convexa interior siguiente a p, ya que los demás se encuentran en el interior de dicha capa convexa.

4.2.4. Análisis

Todos los puntos con excepción de $x \in T$ son iguales en ambos conjuntos, por lo que se busca identificar a x.

La capa convexa p de T, siempre incluye la secuencia

$$..., \ p_{i-1}, \ p_i, \ x, \ p_{i+1}, \ p_{i+2}, \ ...$$

pero la capa convexa p de S puede ser

$$..., \ p_{i-1}, \ p_i, \ p_{i+1}, \ p_{i+2}, \ ... \tag{4-1}$$

$$..., \ p_{i-1}, \ p_i, \ y, \ p_{i+1}, \ p_{i+2}, \ ... \tag{4-2}$$

$$..., \ p_{i-1}, \ p_i, \ y_i, \ y_{i+1}, \ ... \ , \ y_{j-1}, \ y_j, \ p_{i+1}, \ p_{i+2}, \ ... \tag{4-3}$$

siendo el resto de la secuencias idéntica para ambos conjuntos.

Lo primero que se observa es que:

- En 4-1, la secuencia de S tiene un elemento menos que la de T y los restantes son iguales.
- En 4-2, la secuencia de S y la de T tienen la misma cantidad de elementos, pero uno de ellos es diferente.
- En 4-3, la secuencia de S tiene más de un elementos que la de T.

De modo que los casos son completamente excluyentes, ya que las FAC o son del mismo tamaño o tienen un elemento de diferencia, o tienen más de un elemento de diferencia.

Los casos 4-1 y 4-3 quedan diferenciados por el tamaño de las FAC. En el caso 4-2 el ángulo comprendido por $\hat{y}$ siempre será mayor al ángulo comprendido por $\hat{x}$, al estar el triángulo $\Delta p_i y p_{i+1}$ contenido en el triángulo $\Delta p_i x p_{i+1}$.

El algoritmo KMP que se describe en el apéndice B debe ser modificado para usarlo en la comparación de firmas cíclicas, dicha modificación consiste en:

- Comparación de los elementos individuales de las firmas, en lugar de caracteres.

- Elección de un punto de inicio, al ser la firmas sucesiones cíclicas.
- Definición de un sentido de avance.
- Contabilizar los elementos individuales como criterio de finalización, al no existir un final de la cadena.

Para la identificación del punto diferencia se busca la subcadena común más larga, esto se logra iniciando la comparación en un sentido hasta que las secuencias sean diferentes, a partir de ahí se repite la comparación pero en sentido inverso. Al ser sucesiones cíclicas, la primera comparación alineará los círculos a un posible inicio común, de modo que en la segunda comparación se alcanzará el elemento común más lejano.

Es necesario considerar que como los ángulos tomados para la firma cíclica se calculan a partir de puntos adyacentes, la sucesión de la firma cíclica tendrá tres elementos consecutivos diferentes para un punto diferencia.

4.3. Complejidad

Con lo analizado anteriormente podemos formular el siguiente teorema.

Teorema 11 *El algoritmo 1 compara dos conjuntos de puntos en posición general en* $\mathbb{R}^2$ *en tiempo* $\mathcal{O}(n \log n)$.

Prueba. La verificación de la identidad del posible punto diferencia (algoritmo 2) se efectúa de manera secuencial y posterior al proceso de comparación. El algoritmo de Bernard Chazelle (apéndice A) se ejecuta en tiempo $\mathcal{O}(n \log n)$. Cada capa convexa se procesa sólo una vez. La validación de la posición entre capas convexas consecutivas se efectúa también sólo una vez. El cálculo de las firmas cíclicas y su comparación con el algoritmo KMP se ejecuta en tiempo lineal sobre el tamaño de la firma, de modo que para todas las capas convexas se efectúa en tiempo total $\mathcal{O}(n)$.

La complejidad total del algoritmo es $\mathcal{O}(n \log n) + \mathcal{O}(n)$, quedando al final $\mathcal{O}(n \log n)$. ■

Capítulo 5

Ampliación a más de un punto diferencia

5.1. Diferencias en un sólo conjunto

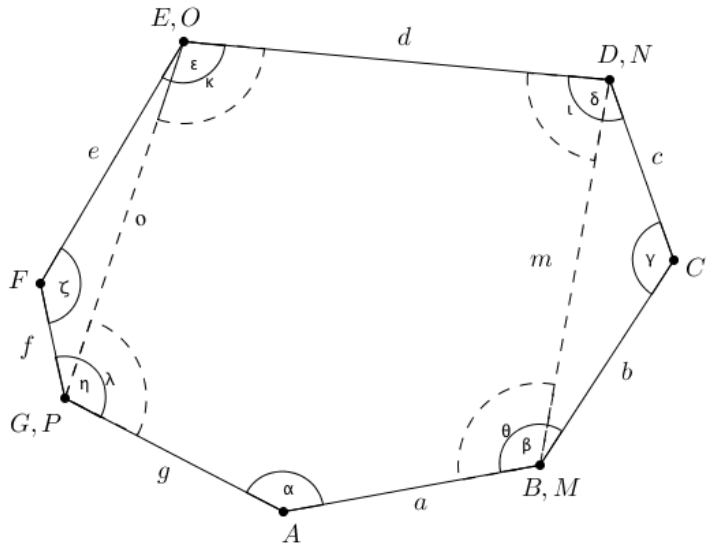
Este caso se describe de la siguiente manera, sean S y T dos conjuntos de puntos en posición general en $\mathbb{R}^2$ y sea U otro conjunto de puntos tal que:

$$\forall u \in U,\ u \in T\ \wedge\ u \notin S\ \wedge\ (T \setminus U) = S$$

La identificación de los elementos de U inicia desde que la cardinalidad de T es mayor que la de S, es decir se sabe que $S \subset T$.

La aplicación del algoritmo descrito, se basa en la generalización del análisis para el punto diferencia, como se mostró en la figura 4-1. Pero, la comparación de firmas requiere de la existencia de al menos tres puntos comunes consecutivos, debido a la variación de los ángulos como se muestra en la figura 5-1. Aquí podemos observar que los puntos B, D, E, G del conjunto de puntos T, son los mismos que los puntos M, N, O, P del conjunto de puntos S respectivamente, pero las FAC para ambos conjuntos sólo tienen en común al punto A.

La identificación de los puntos diferencia se efectúa a partir de las subcadenas de tamaño múltiplo de tres, siendo el central en cada terna el candidato a punto diferencia. Ya lograda esta identificación, es posible marcar dichos puntos como se indica en el algoritmo, a fin de



$$T$$
$$A = (a, \alpha)$$
$$B = (b, \beta)$$
$$C = (c, \gamma)$$
$$D = (d, \delta)$$
$$E = (e, \varepsilon)$$
$$F = (f, \zeta)$$
$$G = (g, \eta)$$
$$FAC(T) = (A, B, C, D, E, F, G)$$

$$S$$
$$A = (a, \alpha)$$
$$M = (m, \theta)$$
$$N = (d, \iota)$$
$$O = (o, \kappa)$$
$$P = (g, \lambda)$$
$$FAC(S) = (A, M, N, O, P)$$

Figura 5-1: Conjuntos con dos puntos diferencia.

eliminarlos del cálculo de las capas convexas y verificando de este modo que los conjuntos resultantes sean iguales.

Por ejemplo, en el caso de los conjuntos mostrados en la figura 5-1, la subcadena diferente de T comparada con S es de tamaño 6 y es $BCDEFG$, siendo los elementos centrales de las dos ternas C y F respectivamente, los que corresponden a los puntos diferencia entre ambos conjuntos.

Estas observaciones permiten establecer restricciones a la capacidad del algoritmo para identificar más de un punto:

1. Los puntos diferencia sólo se pueden encontrar en uno de los dos conjuntos.

2. Deben existir al menos tres puntos iguales en la misma capa convexa, para poder alinear las subcadenas diferentes.

3. Las subcadenas diferentes para el conjunto más grande, deben tener tamaño múltiplo de tres.

4. No pueden existir dos puntos diferencia consecutivos en la misma capa convexa.

Las restricciones 1 y 2 requieren la identificación de las subcadenas comunes entre las dos firmas, para lo cual se puede utilizar un árbol generalizado de sufijos que se construye en tiempo lineal con el algoritmo desarrollado por Esko Ukkonen. Dicho algoritmo se debe modificar para almacenar en cada nodo las posiciones de inicio y fin de la subcadena, en las cadenas originales [Ukkonen, 1995]. En tiempo lineal es posible recorrer el árbol y marcar en las cadenas originales, los lugares donde se encuentran las subcadenas comunes.

Las restricciones 3 y 4 van de la mano en términos de que, dos puntos diferencia consecutivos generarían una subcadena de tamaño no múltiplo de tres. Sin embargo, no es posible generalizar, ya que en caso de existir en una cantidad múltiplo de tres, la subcadena diferente también tendría tamaño múltiplo de tres.

5.2. Diferencias en ambos conjuntos

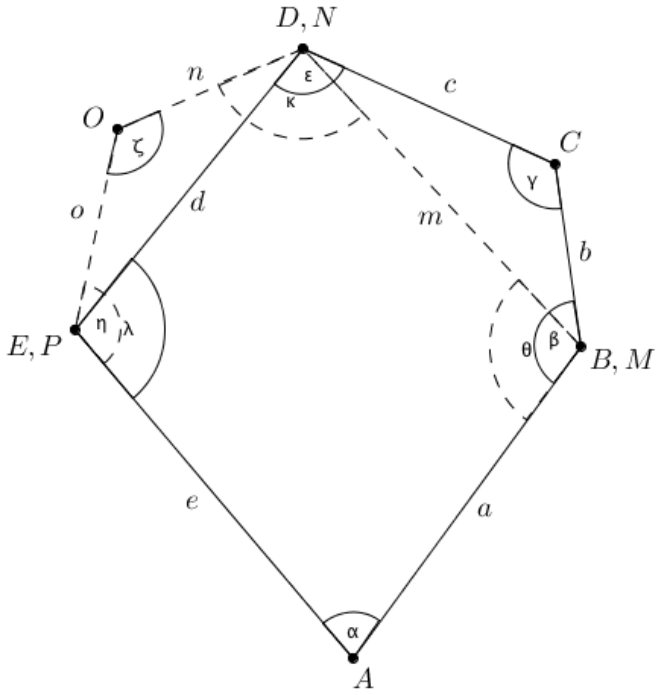
Si los puntos diferencia existen en ambos conjuntos, ya no es posible utilizar de manera directa el algoritmo, debido a que no es posible establecer una correspondencia entre las firmas.

Como se mostró en la sección anterior, los puntos coincidentes permiten alinear las secciones no coincidentes de las capas convexas, sin embargo, la existencia de diferencias en ambos conjuntos, provoca que éstas se puedan compensar unas a otras imposibilitando la alineación de secciones.

En la figura 5-2 se muestran dos conjuntos de puntos que tienen un punto diferencia cada uno, sin embargo, tienen la misma cantidad de elementos por lo que no es posible la identificación de diferencias. El punto C que pertenece al conjunto T y el punto O que pertenece al conjunto S, provocan que ambos conjuntos de puntos sean del mismo tamaño y como consecuencia, las subcadenas diferentes no son de longitud múltiplo de tres. Si se aplicara el algoritmo en este caso, se obtendría para el conjunto T la subcadena diferente $BCDE$ y para el conjunto S, la subcadena diferente sería $MNOP$, de las cuales no es posible identificar a los puntos C y O.

En general para más de un punto diferencia, si no se cumplen las restricciones antes mencionadas, el problema aumenta de complejidad considerablemente y puede ser visto con diferentes enfoques, por ejemplo, particionamiento de conjuntos, subsucesiones comunes, etc.

En varios de estos casos, el efectuar comparaciones entre ellos no se puede hacer con com-



$$T$$
$$A = (a, \alpha)$$
$$B = (b, \beta)$$
$$C = (c, \gamma)$$
$$D = (d, \varepsilon)$$
$$E = (e, \lambda)$$
$$FAC(T) = (A, B, C, D, E)$$

$$S$$
$$A = (a, \alpha)$$
$$M = (m, \theta)$$
$$N = (n, \kappa)$$
$$O = (o, \zeta)$$
$$P = (e, \eta)$$
$$FAC(S) = (A, M, N, O, P)$$

Figura 5-2: Conjuntos con puntos de diferencia en ambos.

plejidad inferior a $\mathcal{O}(n^2)$, sin embargo, como se menciona en las conclusiones, el contar con un algoritmo subcuadrático para comparaciones de conjuntos de puntos, es el inicio para otros proyectos de investigación.

Capítulo 6

Conclusiones

6.1. Siguientes pasos

El algoritmo presentado es en sí mismo un avance para el desarrollo de algoritmos determinísticos, que sean útiles en tareas de reconocimiento. A partir de aquí es posible continuar por varios caminos.

Por un lado, la creación del concepto de firmas cíclicas y su implementación con el algoritmo KMP, es la clave para la comparación de conjuntos bien definidos de puntos. Sin embargo, como se mostró anteriormente no resultó útil para identificar combinaciones de diferencias en ambos conjuntos. Un camino para siguientes investigaciones es encontrar nuevos mecanismos para efectuar dichas comparaciones, con algoritmos que no superen la cota de $\mathcal{O}(n)$, aunque sea de forma amortizada.

Otra línea de investigación es sobre el problema original, que implica espacios de solución que crecen de forma exponencial, aquí el algoritmo presentado puede ser la base de un proceso heurístico de búsqueda, de subconjuntos de puntos en uno mayor, con la finalidad de reconocer patrones dentro de universos de puntos más grandes.

Kreher y Stinson [1999] muestran diversas técnicas de búsqueda heurística, en donde de manera general se define una métrica de optimización para una solución factible dada. Se desarrollan más soluciones factibles a partir de la anterior y, se evalúan sus respectivas métricas para seleccionar la siguiente mejor solución, continuando de esa forma hasta encontrar la solución deseada o agotar el número de intentos.

La métrica de optimización y la técnica para elegir la siguiente solución factible, son factores arbitrarios, por eso son búsquedas heurísticas. Para las técnicas de elección, los autores muestran algunas de las más populares como la escalada de la colina, el recocido simulado, la búsqueda tabú y los algoritmos genéticos. En todos los casos, se requiere de la evaluación de la solución factible, que para nuestro caso, es la comparación de un subconjunto seleccionado del universo contra el patrón de prueba y, la métrica de optimización que determine, que tan buena o mala es la solución evaluada.

La eficiencia lograda en el algoritmo, permite poder utilizarlo como función de evaluación, de la solución factible en el proceso de la búsqueda heurística, ya que al tener que ser ejecutado iterativamente, no sería útil en conjuntos grandes de puntos si tuviera una complejidad menos eficiente.

Para la métrica de optimización se pueden considerar varias alternativas, en donde de manera enunciativa y no limitativa, listamos las siguientes:

- El tamaño de la subcadena común más larga obtenida antes de la primera diferencia.
- El valor absoluto de la diferencia entre cardinalidades de las capas convexas más externas.
- El valor absoluto de la diferencia entre perímetros de las capas convexas más externas.
- La profundidad de la primera diferencia dentro de la secuencia de capas convexas.
- Combinaciones de las anteriores.

En todos los casos, es necesario modificar el algoritmo para que calcule la métrica durante el proceso de comparación de soluciones. La construcción del vecindario de soluciones factibles, también puede presentar mejoras para la búsqueda, en esta tarea de construir dicho vecindario es necesario sustituir uno o más puntos de la solución factible evaluada con puntos del universo, así, la elección de esos nuevos puntos, se puede mejorar con la implementación de una lista tabú, combinando más de una heurística.

Quizá la linea de investigación de las que aquí se proponen, que puede resultar más interesante, es el cambio de $\mathbb{R}^2$ a $\mathbb{R}^3$ en donde ya no es posible utilizar el algoritmo de Chazelle, lo que compromete la eficiencia de la complejidad lograda. Preparata y Shamos [1985], muestran un algoritmo para encontrar el cierre convexo en tres dimensiones de una nube de puntos en

el espacio, que puede servir de partida en futuras investigaciones sobre conjuntos de puntos en $\mathbb{R}^3$.

6.2. Aplicaciones

En el área de procesamiento digital de imágenes, la comparación de conjuntos de puntos en el plano, es una tarea que se presenta frecuentemente para la detección de rasgos y texturas, sobre todo en el procesamiento de imágenes médicas, topográficas y fotografía satelital [Gonzalez y Woods, 2008].

El problema de ajustar patrones de puntos a clases de patrones, se ha resuelto de diversas maneras en la actualidad; dos de los métodos usados más frecuentemente, están basados en el diseño de redes neuronales o en técnicas de cuantización vectorial. Dichos métodos han probado su efectividad para conjuntos relativamente pequeños de puntos, como los mostrados en la figura 6-1, sin embargo, cuando la clase de patrones se compone de cientos o miles de puntos, dichas técnicas presentan problemas de desempeño, en términos de que no producen respuestas confiables en tiempos cortos, o bien los tiempos son inaceptables para obtener respuestas confiables [Hagan, Demuth, y Beale, 1996]. El algoritmo presentado se puede adaptar para que compare los conjuntos muestra con las clases de patrones y obtenga resultados confiables de manera rápida, en conjuntos de miles de puntos.

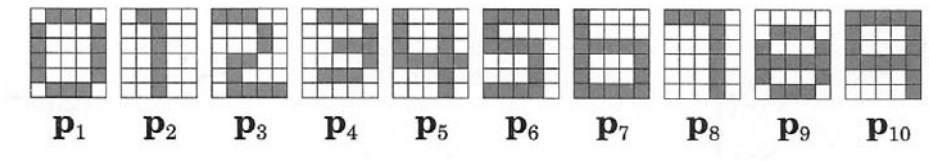


Figura 6-1: Reconocimiento de caracteres mediante cuantización vectorial.

La reconstrucción de señales, se efectúa a partir de los descriptores de Fourier que son series de números complejos, a mayor cantidad de descriptores la reconstrucción de la señal será de mejor calidad, como observamos en la figura 6-2 [Gonzalez y Woods, 2008] [Pajares Martinsanz y de la Cruz García, 2008].

Cuando se busca empatar la señal con alguna clase de patrón, es necesario efectuar la comparación de dichos números, sin embargo, la señal puede estar amplificada o retrasada en

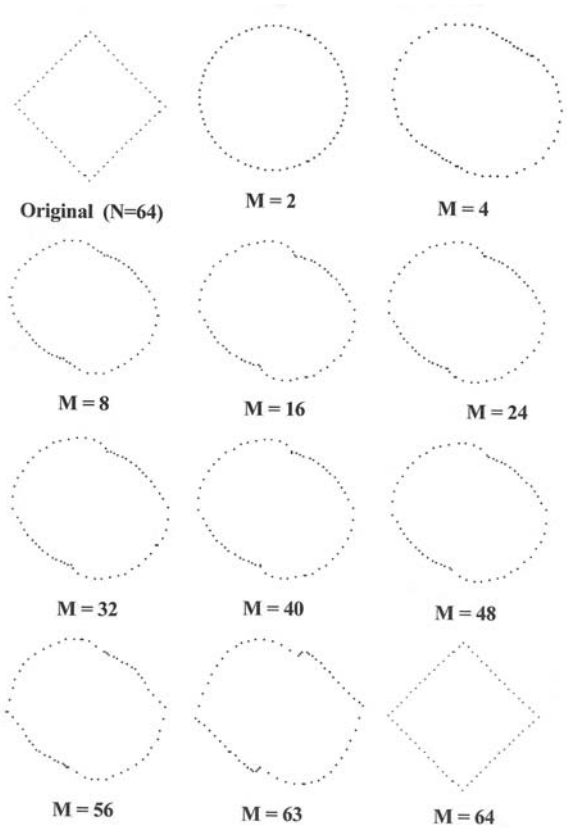


Figura 6-2: Reconstrucción de señales mediante descriptores de Fourier.

tiempo, lo que genera secuencias de números diferentes pero, que conservan en conjunto el mismo patrón a reconocer. Si los descriptores que son números complejos, se llevan al plano complejo y se hace un mapeo con $\mathbb{R}^2$, como podemos ver en la figura 6-3, podría utilizarse el algoritmo para efectuar la comparación de una gran cantidad de descriptores.

Aunado a lo anterior, el reconocimiento de patrones para localización mediante marcas de referencia, requiere la comparación en ocasiones, de conjuntos de cientos de puntos, sobre todo en lo referente a visión computacional en sistemas de navegación de vehículos no tripulados [Pajares Martinsanz y de la Cruz García, 2008]. Los descriptores obtenidos a partir de las imágenes captadas por las cámaras, deben ser comparadas con múltiples patrones, a fin de identificar objetos y lugares, todo esto en tiempo real, por lo que es necesario un algoritmo que pueda efectuar dichas comparaciones en tiempo subcuadrático.

En el desarrollo de la minería de datos, donde se buscan nuevas formas de interpretar

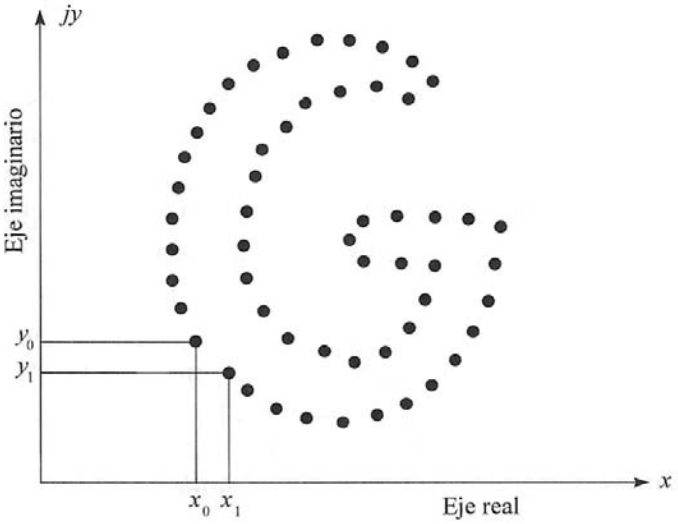


Figura 6-3: Mapeo de puntos del plano complejo al plano real.

volúmenes enormes de información, es posible identificar patrones de comportamiento, pudiendo describir tendencias, preferencia, conductas, etc; a partir de modelos precargados que sirvan de comparación para los generados dinámicamente. Estos modelos, están compuestos comúnmente por cientos o miles de puntos, que representan una medida contra un marco de referencia, de modo que pueden ser mapeados al plano $\mathbb{R}^2$ y ser comparados utilizando el algoritmo [Maimon y Rokach, 2010].

Apéndice A

Algoritmo de Bernard Chazelle para la obtención de capas convexas

Este algoritmo encuentra todas las capas convexas, de una nube de puntos en posición general en tiempo $\mathcal{O}(n \log n)$, como se observa en la figura A-1, fue desarrollado por Bernard Chazelle en 1985, mientras trabajaba en el Departamento de Ciencias de la Computación, de Brown University en Rhode Island, Estados Unidos de América [Chazelle, 1985].

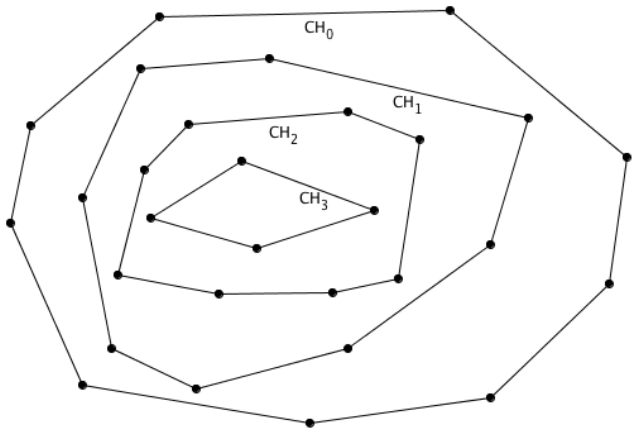


Figura A-1: Nube de puntos y sus capas convexas.

Se basa en el cálculo de la capa convexa más externa de la nube de puntos, la eliminación de los puntos de esa capa convexa del conjunto original y así obtener la siguiente capa y, la

repetición de dicho proceso hasta agotar los puntos de la nube.

El cálculo de la capa convexa de un conjunto de puntos es un problema estudiado ampliamente, de modo que existen diversas soluciones para el mismo, sin embargo, Chazelle se basó en los trabajos de Preparata y Hong [1977] y Overmars y van Leeuwen [1981] para el desarrollo de su algoritmo, el cual se explica a continuación.

Una capa convexa está formada por dos cadenas, una que cubre a todos los puntos de la nube por arriba y otra que los cubre por abajo, como se muestra en la figura A-2, donde la superior está representada con lineas continuas y la inferior con líneas punteadas.

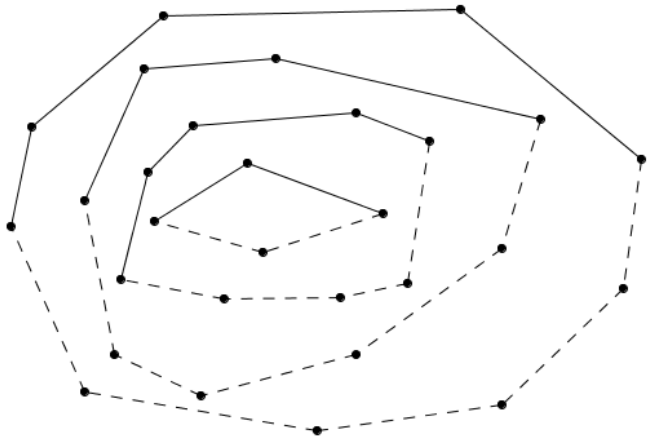


Figura A-2: Cadenas superiores e inferiores de las capas convexas.

Se construyen dos estructuras de datos, una para las cadenas superiores y otra para las inferiores en tiempo $\mathcal{O}(n \log n)$, dichas estructuras proporcionan la información para recuperar la cadena más externa en ese momento, posteriormente las estructuras se actualizan eliminando los puntos pertenecientes a ambas cadenas, de modo que al finalizar dicha actualización, proporcionarán la información para recuperar la siguiente. Esta actualización se debe efectuar en tiempo $\mathcal{O}(\log n)$ ya que es ejecutada por cada punto y así, el tiempo total de ejecución se mantenga $\mathcal{O}(n \log n)$. Al finalizar la eliminación de puntos de las estructuras, se habrán calculado la totalidad de capas convexas.

La estructura mencionada es un árbol, que contiene las cadenas superiores o inferiores según el caso y que organiza la jerarquía de dichas cadenas en la cobertura de los puntos. Consideremos un árbol binario completo y denotémoslo por T, en donde las hojas representarán los puntos

de la nube ordenados de izquierda a derecha. Sea $S(v)$ el conjunto de puntos en las hojas del subárbol de T que tiene raíz en v y sea $U(v)$ la cadena superior del cierre convexo de $S(v)$. La unión de todas las aristas en $U(v)$ para todos los nodos v de T, crean una gráfica plana G que es acíclica y forma un árbol libre. Cada arista de G es una tangente a dos cadenas superiores y existe una correspondencia uno a uno, entre las aristas de G y los nodos de T, como se muestra en la figura A-3.

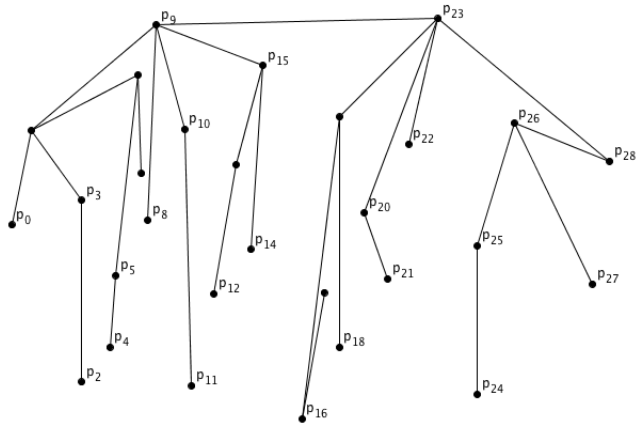


Figura A-3: Gráfica de cadenas superiores.

El árbol T no existe realmente, es sólo conceptual para la construcción de la gráfica de cadenas superiores G, en el sentido que indica los puntos que forman la cadena superior en turno, por ejemplo, en la figura A-3 se tienen 29 puntos, etiquetados de izquierda a derecha como $p_0, p_1, ..., p_{28}$.

En la primera iteración se forman las cadenas superiores del primer nivel del árbol T, es decir, el punto p_0 con el punto p_1, el punto p_2 con el punto p_3, etc. Posteriormente la cadena formada por p_0p_1, se une con la cadena p_2p_3 con una tangente a ellas, en este caso p_1p_3, en el caso de las cadenas $p_{20}p_{21}$ y $p_{22}p_{23}$ la tangente es $p_{20}p_{23}$.

En el siguiente nivel tenemos por ejemplo, la cadena $p_8p_9p_{10}p_{11}$ y la cadena $p_{12}p_{13}p_{15}$, se unen con la tangente p_9p_{15} formando la cadena $p_8p_9p_{15}$. Nótese que esta cadena superior cubre los puntos $p_{10}, p_{11}, p_{12}, p_{13}, p_{14}$. En el último nivel, las cadenas $p_0p_1p_9p_{15}$ y $p_{16}p_{19}p_{23}p_{28}$ se unen con la tangente p_9p_{23} para formar la cadena $p_0p_1p_9p_{23}p_{28}$, que corresponde a la cadena superior de la capa convexa más externa.

De manera análoga se construye la gráfica de cadenas inferiores. La unión de cadenas me-

diante tangentes se efectúa en tiempo lineal, de acuerdo al algoritmo de Preparata y Hong [1977]. Ya con las dos cadenas, superior e inferior, se construye la capa convexa más externa de la nube de puntos en tiempo $\mathcal{O}(n \log n)$.

El siguiente paso es eliminar de ambas gráficas los puntos pertenecientes a esta capa convexa calculada, pero manteniendo la estructura de cadenas, de modo que al finalizar la actualización se tenga calculada la siguiente capa.

Para poder efectuar la actualización en tiempo $\mathcal{O}(\log n)$, la gráfica se almacena con una lista de adyacencias que tendrá las siguientes modificaciones.

Para cada vértice se almacenan dos listas ligadas, una contendrá los vértices conectados del lado izquierdo y la otra los vértices del lado derecho.

En el momento de la generación, cada vértice que se conecte se agregará al inicio de la lista y como las aristas corresponden con las tangentes de las cadenas, la existencia de una nueva arista implica que está por encima de las existentes para ese vértice en el caso de la gráfica de cadenas superiores, o por debajo, en el caso de la gráfica de cadenas inferiores.

Al final de la generación, las listas tendrán los vértices con que se conectan, ordenados por la pendiente con respecto al punto en cuestión.

Por ejemplo, para la gráfica de la figura A-3 tendríamos la estructura de datos de lista de adyacencias mostrada en la tabla A-1. Aquí se puede observar que el punto 9 es adyacente por la izquierda con los puntos 1 y 8, y por la derecha, con los puntos 23, 15 y 10, en ambos casos, se encuentran ordenados de manera descendente por la pendiente.

La secuencia de conexiones que tuvo el punto 9 para llegar a este resultado fue el siguiente: en la primera iteración, el punto 9 se conecto por la izquierda con el punto 8 el cual se agrega al inicio de la lista; en la segunda iteración, se conectó por la derecha con el punto 10 el cual se agrega al inicio de la lista; en la tercera iteración, se conectó por la derecha con el punto 15 el cual se agrega al inicio de la lista desplazando al 10 a la siguiente posición; en la cuarta iteración, se conecta por la izquierda con el punto 1 el cual se agrega al inicio de la lista desplazando al 8 a la siguiente posición. Finalmente, en la quinta iteración, se conecta por la derecha con el punto 23 el cual se agrega al inicio de la lista desplazando al 15 y al 10 que ya se encontraban ahí.

La recuperación de la cadena superior a partir de la estructura de datos, se puede hacer

Vértice	Izquierda	Derecha
0		1
1	0	9,6,3
2		3
3	1,2	
4		5
5	4	6
6	1,5	7
7	6	
8		9
9	1,8	23,15,10
10	9	11
11	10	
12		13
13	12	15
14		15
15	9,13,14	
16		19,17
17	16	
18		19
19	16,18	23
20		23,21
21	20	
22		23
23	9,19,20,22	28
24		25
25	24	26
26	25	28,27
27	26	
28	23,26	

Tabla A-1: Lista de adyacencia de la gráfica de cadenas superiores.

recorriendo los inicios de la lista, empezando por el primero o por el último punto. En el ejemplo, si empezamos con el punto 0 el primero a la derecha es el 1, del 1 el primero a la derecha es el 9, del 9 el primero a la derecha es el 23, del 23 el primero a la derecha es el 28, el cual es ya el último punto, por lo que la cadena es 0,1,9,23,28.

Para la eliminación de los puntos de la estructura de datos, no importa el orden en que se tomen, ya que esa eliminación deja a la estructura siempre actualizada, con las cadenas superiores correctas.

El proceso de eliminación se puede visualizar de la siguiente manera, se hace la suposición que las aristas de la gráfica son de un material elástico, el vértice a eliminar se empieza a mover hacia abajo para el caso de las cadenas superiores o hacia arriba en el caso de las cadenas inferiores, entonces las aristas empezarán a estirarse.

A medida que el vértice se aleja, las aristas empezarán a unirse a los vértices que tienen cerca, en un comportamiento de envoltura alrededor de ellos, cuando el vértice a eliminar llega a ∞ o $-\infty$, las aristas que él mismo tenía desaparecen, pero las que unieron puntos alrededor se conservan.

El retirado de las aristas de un punto a eliminar, se tiene que efectuar en el orden inverso al que fueron agregadas, con el fin de mantener la propiedad de gráfica de cadenas superiores, ya que cuando fue creada, las aristas que se agregaban iban cubriendo a las aristas existentes.

La estructura de datos tiene dos listas para almacenar las adyacencias de los vértices en el orden correcto de inserción, sin embargo, están separados los vértices de la izquierda y de la derecha, así que es necesario mezclarlas durante el proceso de retirado.

Para seleccionar de que lista tomar el siguiente vértice usaremos la siguiente observación: los puntos fueron numerados de izquierda a derecha y corresponden con las hojas del árbol binario completo, ello implica que si vemos las etiquetas de los puntos como números binarios, los prefijos que tienen entre sí corresponden al subárbol al cual pertenecen; para dos puntos diferentes, la longitud del prefijo común indica que tan lejos están uno del otro, mientras más cerca se encuentren, la longitud del prefijo será mayor y ello implica, que fueron unidos en las primeras iteraciones de la construcción de la gráfica.

Esta comparación de prefijos binarios puede efectuarse con la operación lógica NXOR *(o exclusivo negado)*, entre la etiqueta del vértice a eliminar y los vértices finales de ambas listas, el que resulte mayor es la lista a considerar; por ejemplo, en la tabla A-1 tenemos el punto 9 que pertenece a la capa convexa y por lo tanto deberá ser eliminado, su lista izquierda tiene los puntos 1,8 y su lista derecha tiene los puntos 23,15,10, para seleccionar de que lista se debe tomar el siguiente vértice de la arista a retirar, hacemos las operaciones mostradas en las tabla A-2, en donde observamos que se elige el vértice 8 para retirar su arista, ya que en su representación binaria tiene un prefijo común más largo con el 9 que el 10, eso significa que el 8 se agregó antes que el 10 en la estructura de datos.

$(9)_2 = 01001$	**NXOR**	$(8)_2 = 01000$	=	$(30)_2 = 11110$
$(9)_2 = 01001$	**NXOR**	$(10)_2 = 01010$	=	$(28)_2 = 11100$

Tabla A-2: Comparación de prefijos binarios.

La operación de retirado de aristas de la gráfica de cadenas, implica la unión de dos cadenas superiores. Debido a que la arista en sí es una tangente, la primera arista que se retira conecta siempre un punto único, ya que se encuentra por debajo de todas las demás. Ese punto único es la primer cadena a unir de forma ascendente. A medida que se van retirando las aristas, tenemos por un lado la cadena superior obtenida al retirar la arista anterior y por el otro, la cadena que esta conectada con la arista a retirar.

Lo que se necesita es crear una nueva tangente entre estas dos cadenas, dicha tarea se muestra en la figura A-4, donde sin pérdida de generalidad, digamos que el punto a eliminar es p y la arista anterior retirada fue $\overline{pa}$ o $\overline{pb}$ y la arista por retirar es $\overline{pc}$.

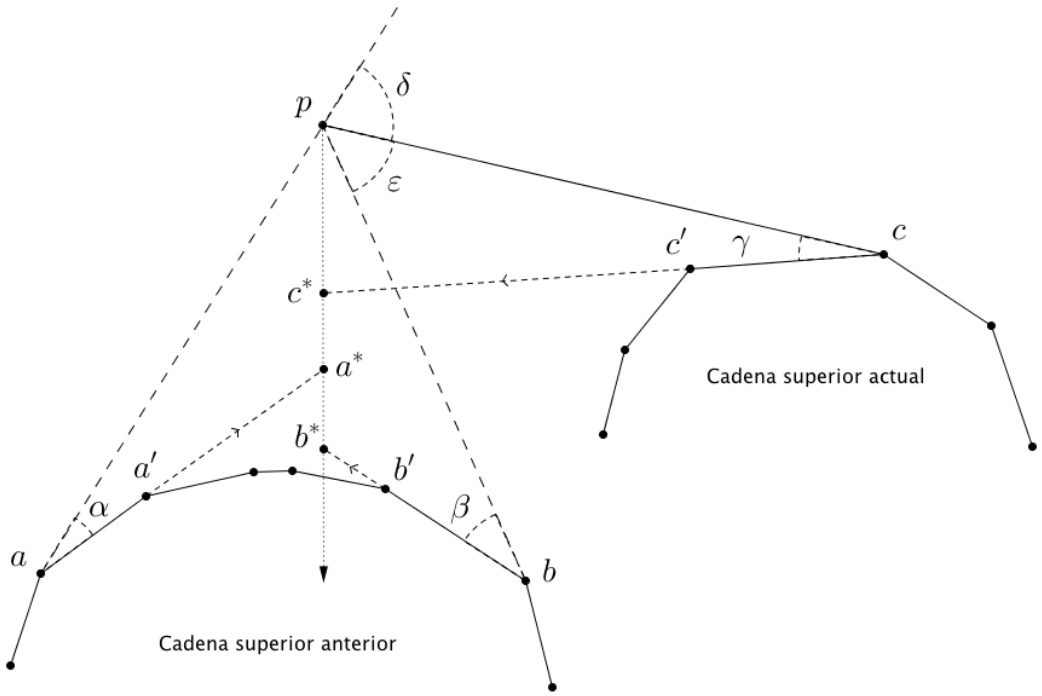


Figura A-4: Unión de cadenas superiores.

Dicha tangente cubrirá los puntos que se encuentren entre a o b y c y dado que todos ellos pertenecen o pertenecían a una cadena superior, se encontrarán al inicio de sus respectivas listas

de adyacencias en la estructura, así que pueden ser accedidas en tiempo constante.

A medida que p va descendiendo, pa o pb y pc irán envolviendo los puntos al interior y puede ser necesario desplazar a hacia a', b hacia b' o c hacia c', eso depende de que los ángulos α, β o γ se hagan nulos. Al mismo tiempo los ángulos δ y ε irán variando, en el momento que a o b, p y c sean colineales, δ o ε serán nulos y tendremos la nueva tangente a las dos cadenas formada por $\overline{ac}$ o $\overline{bc}$.

La evaluación de la nueva posición de p, se debe efectuar sin calcular el valor de los ángulos propiamente, ya que afectaría el desempeño del algoritmo, por otro lado, el punto de evaluación debe ser tal que alguno de los cinco ángulos mencionados sea nulo. Esta condición se obtiene fácilmente si observamos la intersección de los segmentos involucrados con la vertical a partir de p como sigue:

- $\alpha = 0$, segmento $\overline{aa'}$ (punto a^*).
- $\beta = 0$, segmento $\overline{bb'}$ (punto b^*).
- $\gamma = 0$, segmento $\overline{cc'}$ (punto c^*).
- $\delta = 0$, segmento $\overline{ac}$.
- $\varepsilon = 0$, segmento $\overline{bc}$.

y se toma como nueva posición la que resulte con la mayor coordenada y, para el ejemplo mostrado en la figura A-4 sería el punto c^*. Hay que recordar que se debe actualizar a, b o c al siguiente punto en la cadena a', b' o c', en caso de que el ángulo nulo haya sido α, β o γ, o bien terminar el proceso si el ángulo nulo fue δ o ε. Por otro lado, no siempre existen los puntos a y b simultáneamente, pero ello no afecta el cálculo, la evaluación se debe hacer sobre los segmentos existentes, así sea uno solo, en cuyo caso sería el seleccionado.

La finalización del algoritmo está determinada cuando al eliminar los puntos de la última capa, las estructuras de datos queden vacías.

Apéndice B

Algoritmo de Knuth-Morris-Pratt para comparación de cadenas

La comparación de cadenas de caracteres es un problema que surgió desde los orígenes mismos de la computación, de manera simple la comparación de dos cadenas implica la revisión de una dentro de la otra para todos los caracteres, sin embargo, el costo en tiempo de ejecución de dicho proceso es cuadrático. Por el contrario, el algoritmo propuesto por Knuth, Morris y Pratt, encuentra todas las ocurrencias de una cadena en otra, en tiempo proporcional a la suma de los tamaños de ambas cadenas y se explica a continuación [Knuth *et al.*, 1977].

Supóngase que se tienen las cadenas T y P y que el proceso de comparación se encuentra en el desplazamiento s, como se muestra en la figura B-1.

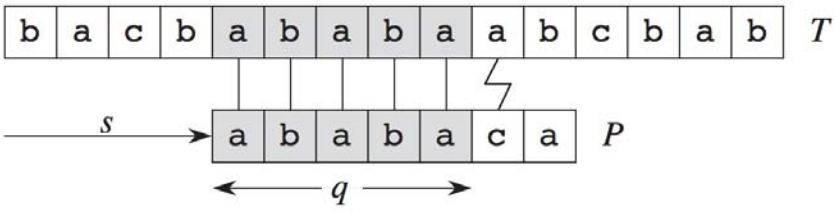


Figura B-1: Comparación de cadenas.

Se tienen $q = 5$ caracteres que han coincidido y el sexto ha fallado en la comparación. El conocimiento de que q caracteres han coincidido es útil, para saber de inmediato que ciertos desplazamientos serán inútiles de probar, en este caso se observa que el desplazamiento $s + 1$

es inútil, ya que el primer carácter de P en este caso a, no va a coincidir con el primer carácter evaluado en el texto, pero si con el segundo. Por otro lado, el desplazamiento $s' = s + 2$ que se muestra en la figura B-2 hace coincidir tres caracteres.

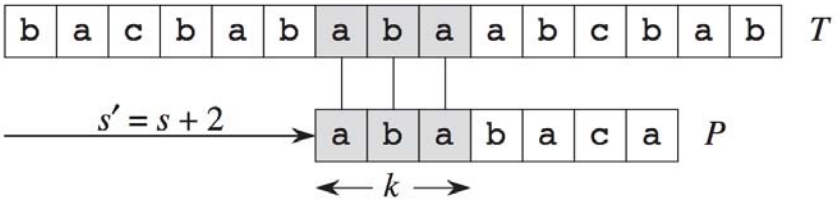
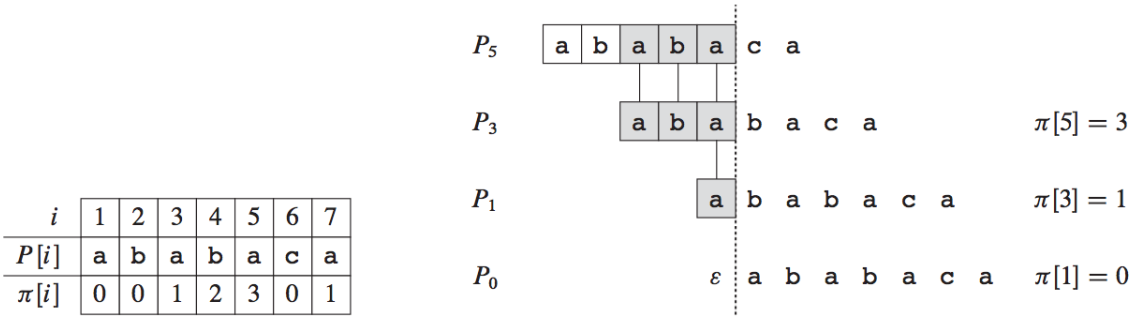


Figura B-2: Comparación de cadenas con un desplazamiento útil.

En general, si estamos comparando las cadenas de caracteres T y P y para un desplazamiento s coincidieron q caracteres, significa que en T desde el carácter $s + 1$ hasta el $s + q$ son iguales al prefijo de tamaño q de P, es decir, $P[1..q]$ es sufijo de $T[1..s + q]$; si conocemos el prefijo más grande de $P[1..q]$ que es sufijo de $T[1..s + q]$ esos caracteres ya no será necesario compararlos.

La información anterior es posible calcularla antes de iniciar la comparación, en virtud de que $T[s + 1..s + q]$ es igual a $P[1..q]$, en otras palabras, la información para el desplazamiento adicional se puede precalcular obteniendo el prefijo más largo que sea sufijo de todos los prefijos de P, a esta información se le conoce como *función de prefijos* y se muestra en la figura B-3 para nuestro ejemplo.



i	1	2	3	4	5	6	7
$P[i]$	a	b	a	b	a	c	a
$\pi[i]$	0	0	1	2	3	0	1

Figura B-3: Función de prefijos.

Aquí puede verse que para el prefijo $P[1..5]$, el prefijo más largo que es sufijo de él mismo es $P[1..3]$, $P[1..1]$ también es sufijo de $P[1..5]$ sin embargo no es el más largo.

El algoritmo 7 se utiliza para la comparación de las cadenas de caracteres.

Algoritmo 7 Algoritmo Knuth-Morris-Pratt

Entrada: Cadenas $\boldsymbol{T}$ y $\boldsymbol{P}$

Salida: Despliega las ocurrencias de P en T

1: $\boldsymbol{n} = longitud(\boldsymbol{T})$
2: $\boldsymbol{m} = longitud(\boldsymbol{P})$
3: Usar el algoritmo 8 con $\boldsymbol{P}$ y dejar la salida en π
4: $\boldsymbol{q} = 0$
5: **para** $\boldsymbol{i} = 1$ a $\boldsymbol{n}$ **hacer**
6: **mientras** $\boldsymbol{q} > 0$ y $\boldsymbol{P}[\boldsymbol{q}+\boldsymbol{1}] \neq \boldsymbol{T}[\boldsymbol{i}]$ **hacer**
7: $\boldsymbol{q} = \pi[\boldsymbol{q}]$
8: **fin mientras**
9: **si** $\boldsymbol{P}[\boldsymbol{q}+\boldsymbol{1}] = \boldsymbol{T}[\boldsymbol{i}]$ **entonces**
10: $\boldsymbol{q} = \boldsymbol{q} + 1$
11: **fin si**
12: **si** $\boldsymbol{q} = \boldsymbol{m}$ **entonces**
13: Despliega *Cadena encontrada en la posición* $\boldsymbol{i} - \boldsymbol{m}$
14: $\boldsymbol{q} = \pi[\boldsymbol{q}]$
15: **fin si**
16: **fin para**

En T la comparación siempre se hace sobre el carácter i, el cual no retrocede, siempre avanza. En q se lleva la cuenta de cuantos caracteres de P han coincidido, es decir, la comparación en P se hace sobre el carácter $q + 1$.

En caso de que la comparación sea exitosa, se incrementa el contador de caracteres coincidentes, es decir q. Cuando la comparación falla, se verifica si ya hubo caracteres coincidentes, en cuyo caso se retoma la comparación con P a partir de la posición indicada por la función de prefijos, si vuelve a fallar se toma el siguiente prefijo de menor tamaño hasta que no queden caracteres coincidentes.

Cuando la comparación falla y no hay caracteres coincidentes, el contador q se mantiene en 0, indicando que la comparación contra P es en el inicio de la cadena. Si el número de caracteres coincidentes es igual a la longitud de P, se ha encontrado una ocurrencia.

El algoritmo 8 se utiliza para el cálculo de la función de prefijos.

Algoritmo 8 Algoritmo para el cálculo de la función de prefijos

Entrada: Cadena $\boldsymbol{P}$

Salida: Arreglo π

1: $\boldsymbol{m} = longitud(\boldsymbol{P})$
2: $\pi[1..\boldsymbol{m}] = 0$
3: $\boldsymbol{k} = 0$
4: **para** $\boldsymbol{q} = 2$ a $\boldsymbol{m}$ **hacer**
5: **mientras** $\boldsymbol{k} > 0$ y $\boldsymbol{P}[\boldsymbol{k}+\mathbf{1}] \neq \boldsymbol{P}[\boldsymbol{q}]$ **hacer**
6: $\boldsymbol{k} = \pi[\boldsymbol{k}]$
7: **fin mientras**
8: **si** $\boldsymbol{P}[\boldsymbol{k}+\mathbf{1}] = \boldsymbol{P}[\boldsymbol{q}]$ **entonces**
9: $\boldsymbol{k} = \boldsymbol{k} + 1$
10: **fin si**
11: $\pi[\boldsymbol{q}] = \boldsymbol{k}$
12: **fin para**
13: **devolver** π

La lógica de los algoritmos 7 y 8 es extremadamente similar, en términos que la función de prefijos lo que hace es buscar las ocurrencias de P consigo misma.

Bibliografía

Aho, A.V., Hopcroft, J.E., y Ullman, J.D. *Estructuras de Datos y Algoritmos*. Addison Wesley Iberoamericana, S.A., USA (1988)

Aichholzer, O., Aurenhammer, F., y Krasser, H. Enumerating Order Types for Small Point Sets with Applications. En *Proc.* 17^{th} *Ann. ACM Symp. Computational Geometry*, págs. 11–18. Medford, Massachusetts, USA (2001)

Algoritmo. *Enciclopedia Hispánica*, *Macropedia*, tomo 1. 2ª edición. Editorial Barsa Planeta, Inc., USA (2001)

Aloupis, G., Iacono, J., Langerman, S., y Ozkan, O. *The Complexity of Order Type Isomorphism*. Cornell University Computing Research Repository **abs/1311.0928** (2013)

Baldor, A. *Geometría Plana y del Espacio*. Publicaciones Cultural, S.A., México (1984)

Chazelle, B. *On the Convex Layers on a Planar Set*. IEEE Transactions on Information Theory **IT-31**(4):509–517 (1985)

Colley, S.J. *Cálculo Vectorial*. 4ª edición. Pearson Education, Inc., México (2013)

Cormen, T.H., Leiserson, C.E., Rivest, R.L., y Stein, C. *Introduction to Algorithms*. 3ª edición. The MIT Press, USA (2009)

Equivalence, R. *Encyclopedic Dictionary of Mathematics*, tomo 2. 2ª edición. The MIT Press, USA (1977)

Foley, J.D., Hughes, J.F., Van Dam, A., McGuire, M., Sklar, D.F., Feiner, S.K.,

y Akeley, K. *Computer Graphics: Principles and Practice.* 3ª edición. Addison Wesley Publishing Company, Inc., USA (2014)

Folkman, J. y Lawrence, J. *Oriented matroids.* Journal of combinatorial theory **B**(25):199–236 (1978)

Garey, M.R. y Johnson, D.S. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Bell Telephone Laboratories, Inc., USA (1979)

Gonzalez, R. y Woods, R. *Digital Image Procesing.* 3ª edición. Pearson Education, Inc., USA (2008)

Goodman, J.E. y Pollack, R. *Multidimensional sorting.* SIAM J. Computing **12**(3):484–507 (1983)

Hagan, M.T., Demuth, H.B., y Beale, M. *Neural Network Design.* PWS Publishing Company, USA (1996)

Khwârizmî, A. *The New Encyclopædia Britannica*, *Micropædia*, tomo 6. 15ª edición. Encyclopædia Britannica, Inc., USA (2007)

Kleene, S.C. *Introduction to Metamathematics.* D. Van Nostrand Company, Inc., Netherlands (1950)

Knuth, D.E., Morris, J.H., y Pratt, V.R. *Fast Pattern Matching in Strings.* SIAM J. Computing **6**(2):323–350 (1977)

Knuth, D.E. *The Art of Computer Programming*, *Computer Science and Information Processing*, tomo 1. 3ª edición. Addison Wesley Publishing Company, Inc., USA (1997)

Kolman, B. *Álgebra Lineal.* Fondo Educativo Interamericano, S.A., México (1986)

Krasser, H. *Order Types of Points Sets in the Plane.* Tesis Doctoral, Graz University of Technology, Graz, Austria (2003)

Kreher, D.L. y Stinson, D.R. *Combinatorial Algorithms Generation, Enumeration and Search.* CRC Press LLC, USA (1999)

Maimon, O. y Rokach, L. *Data Mining and Knowledge Discovery Handbook*. 2ª edición. Springer, USA (2010)

Overmars, M.H. y van Leeuwen, J. *Maintenance of Configurations in the Plane*. Journal of Computer and System Sciences **23**:166–204 (1981)

Pajares Martinsanz, G. y de la Cruz García, J. *Visión por Computador Imágenes Digitales y Aplicaciones*. 2ª edición. Alfaomega Ra-Ma, México (2008)

Percepción. *Enciclopedia Hispánica*, *Macropedia*, tomo 11. 2ª edición. Editorial Barsa Planeta, Inc., USA (2001)

Preparata, F.P. y Hong, S.J. *Convex Hulls of Finite Sets of Points in Two and Three Dimensions*. Communications of the ACM **20**(2):87–93 (1977)

Preparata, F.P. y Shamos, M.I. *Computational Geometry An Introduction*. Texts and Monographs in Computer Science. Springer, USA (1985)

Relation. *Encyclopedic Dictionary of Mathematics*, tomo 2. 2ª edición. The MIT Press, USA (1977)

Ukkonen, E. *On-line construction of suffix trees*. Algorithmica **14**(3):249–260 (1995)

Union, I.A. The Constellations. `http://www.iau.org/public/themes/constellations/` (2015-06-11)