Geometría Computacional

Francisco Rivero
Departamento de Matemáticas
Facultad de Ciencias
Universidad de Los Andes.

Diagramación de texto: Carlos Cova y Edgar Iturriaga.

ÍNDICE GENERAL

INTRODUCCIÓN

¿ Qué es la geometría computacional?

La geometría computacional se ocupa del diseño y análisis de algoritmos de computación para resolver problemas de tipo geométrico. En este curso, a menos que se diga lo contrario, los problemas geométricos se refieren al plano de dos dimensiones. La clase de objetos estudiados serán los puntos del plano, definidos mediante un par de coordenadas cartesianas, las rectas, los triángulos, los polígonos y los círculos.

El tema de la geometría computacional es de data reciente. Los orígenes se encuentran en la tesis doctoral de M. I. Shamos en 1975. Desde entonces, el campo se ha expandido considerablemente con una cantidad apreciable de resultados. La investigación en esta área ha encontrado muchas aplicaciones en la vida real: robótica, reconocimiento de voz y de patrones, diseño gráfico, sistemas de información geográfica...etc.

Gracias a la iniciativa del Departamento de Matemáticas de la Facultad de Ciencias de la Universidad de los Andes, se produce la publicación de esta obra dentro de la colección "Acceso Abierto al Conocimiento Matemático", financiado por el CODEPRE.

Deseo expresar mi más profundo agradecimiento al Dr. Hugo Leiva Coordinador de CODEPRE de la U.L.A. por su valiosa colaboración. Así mismo vaya mi agradecimiento a los colegas Edgar Iturriaga y Carlos Cova por su ayuda desinteresada con el Latex, en el lento trabajo de edición del material y por la diagramación final.

ALGORITMOS EN EN EL PLANO

0.1. El área de un triángulo

Un primer problema bastante simple es el siguiente. Considérese una recta l en el plano y un punto p fuera de ella. El problema es responder a la pregunta: ¿ El punto p se encuentra a la derecha o a la izquierda de l ? (ver la figura).

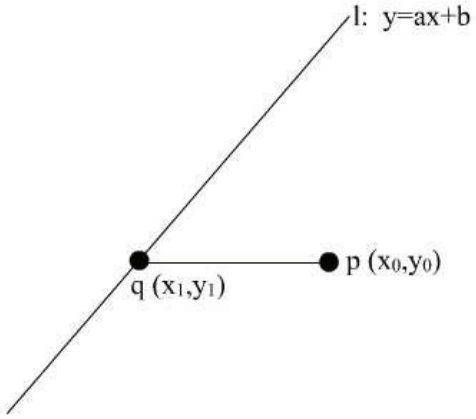

Figura 1: P está a la derecha de L

Supongamos que p tiene coordenadas $P(x_0; y_0)$ y la recta viene dada por la ecuación $y = ax + b$.

Podemos suponer, sin pérdida de generalidad, que la recta no es horizontal ni vertical, esto es, $a \neq 0, \infty$.

Una forma de resolver este problema es la siguiente:

1. Trazamos una recta horizontal pasando por el punto p.
2. Hallamos el punto q de interseccón de l con esta recta horizontal, (Esto se hace sustituyendo $y = y_0$ en la ecuación de la recta y luego se despeja la coordenada x). Sea $q = (x_1, y_1)$ el punto en cuestión. (Nótese que $y_1 = y_0$)
3. Comparamos la coordenada $x = x_0$ del punto p con la coordenada $x = x_1$ del punto q. Entonces:
   - 1) si $x_0 = x_1$ p está en l ,
   - 2) si $x_0 > x_1$ p está a la derecha de l.
   - 3) si $x_0 < x_1$ p está a la izquierda de l.

De esta manera, el problema queda resuelto.

El lector podriía darse por satisfecho con esta manera de trabajar. Sin embargo, desde el punto de vista computacional, este algoritmo presenta algunas debilidades. En primer lugar hay que hacer consideraciones sobre funciones. En segundo lugar podríamos dividir entre números muy pequeños, lo cual es un serio inconveniente a la hora de redondear. Pero alguien un poco más curioso podriá preguntarse: ¿ Existe otra forma de resolver el mismo problema?

La respuesta es positiva. Afortunadamente, existe una fórmula para calcular el área de un triángulo con vértices $a = (a_1, a_2)$, $b = (b_1, b_2)$ y $c = (c_1, c_2)$ dada por:

$$\triangle_{abc} = \frac{1}{2}[a_1(b_2 - c_2) + b_1(c_2 - a_2) + c_1(a_2 - b_2)] \tag{1}$$

Dicha fórmula se obtiene al aplicar el producto vectorial a los vectores $\vec{A} = b - a = (b_1 - a_1, b_2 - a_2)$ y $\vec{B} = (c_1 - a_1, c_2 - a_2)$. Recordemos que el producto vectorial de dos vectores $\vec{A}$ y $\vec{B}$ es otro vector $\vec{A} \times \vec{B}$ cuyo módulo es igual al área del paralelogramo de lados $\vec{A}$ y $\vec{B}$, siendo el área positiva si en ángulo entre ellos está dado en sentido contrario a las agujas del reloj, o negativa si el ángulo está en el sentido de las agujas del reloj.

El área signada es la mitad del determinante

$$\Delta = \begin{vmatrix} a_1 & a_2 & 1 \\ b_1 & b_2 & 1 \\ c_1 & c_2 & 1 \end{vmatrix} \tag{2}$$

Vemos entonces en la figura 2 que si en punto b se encuentra a la derecha de la recta que pasa por a y c, entonces en área es positiva. Si, por el contrario, el punto b está a la izquierda de la recta entonces, el área será negativa.

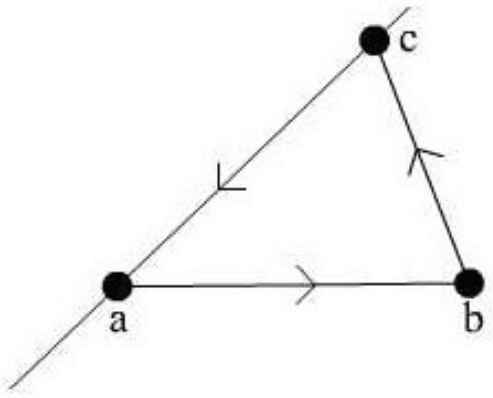

Figura 2: Triángulo formado por tres puntos

Hemos obtenido otro método para resolver el problema: Para determinar si un punto b está a la derecha o a la izquierda de una linea recta, basta con tomar dos puntos de la recta a y c y entonces calculamos el signo del área del triangulo $\triangle abc$ usando la fórmula 1.1.

0.2. La Envolvente Convexa

Veamos ahora un conjunto geométrico de gran importancia dentro de la geometría computacional como la **Envolvente Convexa o Cierre Convexo** de un conjunto de puntos (Convex Hull). Supongamos que tenemos un conjunto S finito, de puntos del plano. Nos interesa hallar el menor conjunto convexo que contiene a S. Dicho conjunto siempre existe y se llama la envolvente convexa de S.

Antes de seguir adelante es menester dar algunas definiciones

Definición 0.2.1. *Un* ***polígono*** *P en el plano es un conjunto de n puntos $\{p_1, .., p_n\}$ llamados* ***vértices*** *, y n segmentos de rectas $\{l_1, .., l_n\}$ llamados* ***lados*** *tales que:*

1. *Los puntos extremos de los lados son vértices del polígono.*

2. *Todo vértice del polígono está en la intersección de exactamente dos lados.*

Definición 0.2.2. *Dos lados que se intersectan en un vértice v se llaman* ***lados consecutivos****.*

Definición 0.2.3. *Un polígono P se llama* ***polígono simple*** *si dos lados no consecutivos no se intersectan.*

Se puede demostrar que todo polígono del plano es una curva cerrada simple. Por lo tanto se puede aplicar el Teorema de Jordan para concluir que el polígono divide al plano en dos regiones disjuntas: el exterior y el interior.

Se acostumbra también definir al polígono P como la región cerrada del plano, dado por P y el interior de P. Cuando se procede de esta manera, entonces se dice que el polígono es la frontera de P y se denota por ∂P.

El siguiente es un resultado de topología general muy conocido.

Lema 0.2.1. *Sea X un conjunto del plano. Entonces se tiene la descomposición disjunta*

$$X = int(X) \cup \partial X \cup int(X^c) \tag{3}$$

En un polígono simple, al recorrer los vertices siguiendo cada uno de los lados llegamos al punto inicial del recorrido. Por lo tanto la trayectoria es cíclica o cerrada. Se acostumbra tomar a v_1 como el vértice inicial y hacer el recorrido en el sentido contrario de las agujas de reloj (ver la figura).

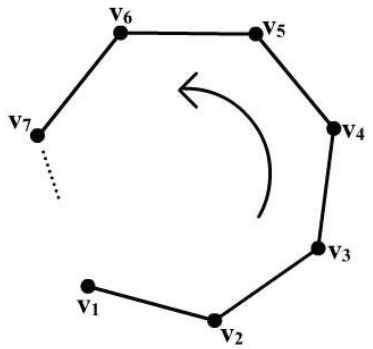

Figura 3: Recorriendo los vértices

Cuando hacemos el recorrido de esta manera siempre tenemos el interior de P a nuestra izquierda.

Definición 0.2.4. *Un conjunto A del plano se llama* ***Conjunto convexo*** *si para todo par de puntos a, b en A el segmento ab está contenido en A*

Definición 0.2.5. *Un polígono P se dice* ***Polígono convexo*** *si la región acotada por P es un conjunto convexo del plano.*

A continuación daremos un par de propiedades importantes de los conjuntos convexos.

Lema 0.2.2. *Si A y B son conjuntos convexos del plano, entonces $A \bigcap B$ es convexo.*

Definición 0.2.6. *Sea S un conjunto finito de puntos del plano. Entonces su* ***Envolvente convexa*** *es el menor conjunto convexo que lo contiene (ver la figura).*

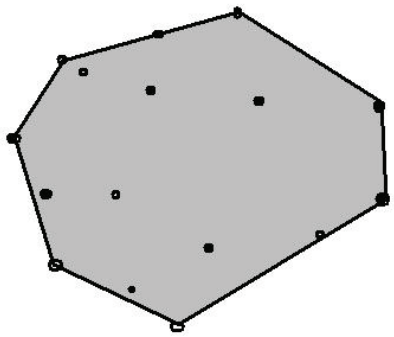

Figura 4: Envolvente convexa

Lema 0.2.3. *S es un conjunto finito de puntos, entonces la frontera de su envolvente convexa es un polígono simple.*

Demostración: Sea P la frontera de la envolvente de S. En primer lugar necesitamos probar que P es convexo.

En efecto, la región acotada por P es la envolvente convexa de S y por lo tanto un conjunto convexo. Luego P es convexo.

En segundo lugar probaremos que P es un polígono cuyos vértices son puntos de S. Si todos los puntos de S son colineales, entonces P es un segmento de recta. Supongamos entonces que no todos los puntos de son colineales.

Como S es finito existe un punto $x_1 \in S$ y una recta L tal que todos los puntos de S que no están sobre L se encuentran de un lado. Entonces x_1 está en la frontera de S y por lo tanto

$x_1 \in S \cap P$. Si sobre dicha recta hay otros puntos de S, entonces elegimos x_1 y x_2 en $L \cap S$ de tal forma que el segmento x_1x_2 tenga longitud máxima. Esto nos da el primer lado de P.

En caso contrario, podemos girar la recta L en sentido antihorario manteniendo el punto x_1 fijo como eje de rotación hasta encontrar el primer punto de S, el cual denotaremos por x_2. Luego el segmento x_1x_2 pertenece a P.

En efecto, existe un punto q en S tal que el triángulo $\triangle(x_1, x_2, q)$ tiene interior no vacío y está dentro de la envolvente de S. Entonces el segmento x_1x_2 es un ado del triángulo. Si p es un punto cualquiera del segmento, entonces cualquier círculo centrado en p contiene puntos tanto del exterior de la envolvente como del interior. Por lo tanto p es un punto frontera y de esta manera queda demostrado que $x_1x_2 \subseteq P$.

Continuando de esta manera usamos ahora el punto x_2 como pivote y podemos hallar otra recta L' y un punto x_3 en S tal que $x_2x_3 \subseteq P$. Como S es finito, después de un número finito k de pasos se completa el polígono frontera P hasta que el último punto x_k sea igua a x_1 y se complete el ciclo.

Finalmente probaremos que P es simple. Si suponemos lo contrario hay dos lados no consecutivos que se cortan. Sean e_i y e_k lados de P tal que:

$$e_i \bigcap e_j \subseteq \{p\}$$

Sea e_i el lado que une los vertices v_i , v_{i+1}, entonces el punto p debe ser igual a alguno de ellos ¿ Por qué?.

Supongamos que $v_i = p$. Entonces en v_i concurren tres lados de P a saber: e_i , e_{i+1} y e_j, supóngase que e_{i+1} está a la derecha de e_i y e_j está a la izquierda. Entonces e_i no es un lado de P.

0.3. Un algoritmo intuitivo

La envolvente convexa es un polígono convexo cuyos vértices son elementos de S. Si, intuitivamente, consideramos los puntos de S como clavos sobre un panel de madera, entonces podemos pensar en la frontera de la envolvente convexa como la forma que toma una liga elástica que encierra a todos los clavos

¿ Cómo calculamos la envolvente convexa?.

Notemos que los puntos del polígono P que forman la frontera de la envolvente convexa tienen la propiedad siguiente:

Propiedad fundamental Dos puntos p y q están sobre el polígono P sí, y sólo si al trazar la linea l que une a p con q, todos los puntos de S que no están sobre l se encuentran a la izquierda o a la derecha de l.

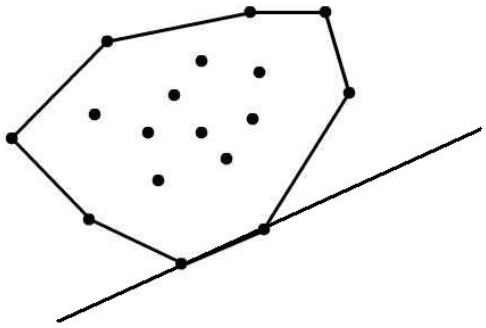

Figura 5: Dos puntos de la envolvente

Esta propiedad fundamental es la base para diseñar un algoritmo que permita calcular la envolvente convexa de S utilizando el algoritmo desarrollado en la sección anterior. Supondremos que S tiene n puntos.

Iniciamos en proceso tomando uno a uno los puntos del conjunto S. Para cada punto p seleccionado se hallan las $(n-1)$ rectas L. Hay que determinar si todos los elementos de S se hallan a la izquierda o a la derecha de L. Si para algún punto $q \epsilon S - \{p\}$ la recta que une p y q divide al plano en dos regiones, una de las cuales contiene todos los elementos de S, entonces p y q son vértices del polígono y el lado $\overline{pq}$ pertenece al polígono. En caso de no obtener este resultado, el punto p se descarta y pasamos a otro punto de S.

Puesto que S es finito, después de aplicar el mismo proceso n veces se obtendrán todos los vértices de la envolvente. A continuación damos el pseudocódigo del algoritmo.

Algoritmo envolvente convexa 1

- `ENTRADA: Un conjunto` S `de` n `puntos del plano.`
- `SALIDA: La envolvente convexa de` S.

BEGIN

1. Para $i = 1, .., n$ DO.
2. Para $j = 1, .., n$, $j \neq i$ DO.
3. Para $s = 1, .., n$, $s \neq i, j$ calcule el área del triángulo $\Delta(p_i, p_j, p_s)$
4. IF área $\Delta(p_i, p_j, p_s) > 0$ o área $\Delta(p_i, p_j, p_s) < 0$.
   Entonces $\{p_i, p_j\} \subseteq$ envolvente de S.
   Else ir a 1.
5. Hacer $p_i = p_j$ Ir a 2

END

0.4. Complejidad

El algoritmo que acabamos de estudiar es bastante simple de entender y puede ser inplementado fácilmente en el computador con cualquier lenguaje de programación. Sin embargo, el número de operaciones básicas necesario para completarlo puede ser muy alto cuando el número de puntos n sea bastante grande. ¿Que tal si n es 1000 o un millón ?. ¿Cuántas operaciones debemos realizar ?

Esto nos conduce al problema de calcular el tiempo de ejecución de un algoritmo o lo que es lo mismo estudiar la **complejidad**.

En computación, cada operación básica viene representada por sumar, restar, multiplicar o dividir números.

Supongamos que para aplicar la fórmula del área del triángulo se requiere de k operaciones básicas donde k es una constante. Entonces el algoritmo para construir la envolvente convexa de un conjunto de n puntos emplea un tiempo total de operaciones básicas denotado por $C(n)$, el cual viene dado por.

$$C(n) = \left[\frac{n(n-1)}{2}\right] \times [(n-2)k] \tag{4}$$

En efecto, hay

$$\frac{n(n-1)}{2}$$

posibles parejas de puntos en S.

Cada pareja origina una recta. Por otro lado, para cada una de estas rectas hay que aplicar el primer algoritmo con cada uno de los $(n-2)$ puntos restantes. Luego el número total de veces que se aplica el primer algoritmo vendrá dado por

$$\frac{n(n-1)}{2} \times (n-2)$$

Expandiendo y reordenando la expresión 1.2 se obtiene:

$$c(n) = c_1 n^3 + c_2 n^2 + c_3 n + c_4$$

donde las c_i, $1 \leq i \leq 4$ son constantes.

Por lo tanto, la complejidad del algoritmo descrito se expresa mediante un polinomio de grado 3. Podemos hallar otro algoritmo distinto y entonces la complejidad nos daría otro polinomio con diferentes constantes. A fin de comparar la complejidad de los distintos algoritmos nos alvidamos de las constantes y sólo consideramos el término principal del polinomio. Este tipo de notación se llama notación de orden "0". Luego el algoritmo tiene una complejidad $0(n^3)$.

Cuando n es grande, entonces n^3 es gigantesco. Más adelante, en el capítulo 3, veremos que se puede construir un algoritmo más eficiente de orden $0(n \ln n)$.

0.5. Algunos problemas interesantes

El problema del robot. La bella robot Robotina desea salir de la casa pero se ha presentado un pequeño problema que debe resolver. La salida se encuentra al final de un pasillo que tiene 60 cm de ancho. Así pues, la robot debe calcular su anchura para evitar una colisión con las paredes. Si la anchura de ella es de 58 cm o menos, entonces podrá pasar a través del pasillo y salir. Si la anchura de robotina es mayor de 58 cm su computadora le dará una orden para no salir.

El robot está compuesto por una serie de partes mecánicas como brazos, antenas, ruedas y otras piezas móviles cuyas posiciones se determinan mediante puntos con coordenadas en el espacio. Podemos representar un robot en el plano como un conjunto finito de puntos S.

¿Cuál es la anchura del robot?

Para dar una definición precisa de anchura de un polígono S, necesitamos una serie de defincciones previas

Definición 0.5.1. *Una **Línea de soporte** de S es una recta que contiene al menos un punto de S y tal que los elementos de S que no se encuentran sobre L, están todos de un mismo lado del plano dividido por L*

Definición 0.5.2. *Sea S un polígono. Un par de líneas de soporte de S, que sean distintas y paralelas, se llaman* ***Paralelas de soporte*** *de S (Ver la figura).*

Definición 0.5.3. *Dos puntos de S situados sobre un par paralelas de soporte se llaman* ***Puntos antipodales****(ver la figura).*

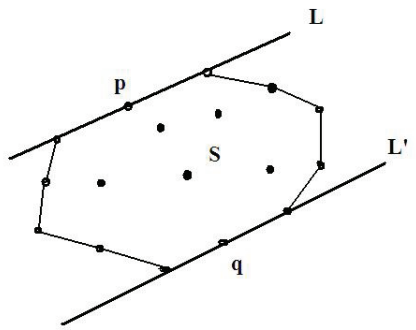

Figura 6: L y L' son líneas de soporte de S

Definición 0.5.4. ***La anchura de un polígono S****, es la menor distancia entre dos rectas paralelas que sean líneas de soporte de S.*

Según estas definiciones tenemos las observaciones siguientes:

1. Para hallar la anchura de S hay que considerar la mínima distancia entre paralelas de soporte.
2. Las paralelas pasan por los puntos antipodales.
3. Los puntos antipodales están sobre la envolvente convexa de S (ver problema 8).

Sabemos que si p es un punto de la envolvente convexa de S, entonces por p debe pasar al menos una línea de soporte L. Si q es un antipodal, entonces por q pasa otra línea de soporte L', paralela a L.

¿Cómo se determinan las ecuaciones de de L y L'?

Si hay otros puntos de la envolvente sobre L, digamos p', entonces L queda determinada como la única recta que pasa por p y p'. Luego L' será la única recta paralela a L y que pasa por q.

Posiblemente, p y q son los únicos puntos de la envolvente sobre las rectas L y L' respectivamente. Entonces en esta situación debemos girar las dos rectas manteniéndolas paralelas y a la misma distancia hasta que alguna de ellas toque dos o más puntos de la envolvente (ver la figura).

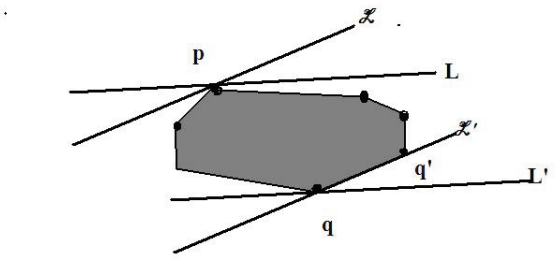

Figura 7: Rotación de las líneas de soporte

En definitiva, para hallar todas las líneas paralelas de soporte de S hay que considerar pares consecutivos de vértices sobre la envolvente y sus respectivos antipodales. Podemos hacer esta búsqueda eficiente con el siguiente algoritmo:

ALGORITMO ANCHURA DE S

- Entrada : Un conjunto S de n puntos
- Salida: La anchura de S.

BEGIN

1. Construya la envolvente de S, CH(S).
2. Ordene los vértices de S en sentido antihorario.
3. Considere el par de vértices consecutivos v_1 y v_2 sobre la envolvente.
4. Buscar un antipodal de v_1, v_2, avanzando en sentido antihorario. Sea v_i el antipodal.
5. Calcule la distancia entre las líneas de soporte L y L' : haga A = d (L, L')

6. Para $i = 1, .., n$ hacer $v_i = v_{i+1}$ y volver a 4.

7. Hacer anchura = min A en 5.

END.

Estudio de la complejidad

1. El paso 1 del algoritmo se puede hacer en tiempo O(nlogn).

2. El paso 4 es de complejidad O(n).

3. El paso 6 es de complejidad O(n), pues a medida que los vértices van avanzando, sus antipodales avanzarán también.

En conclusión , el problema del robot se resuelve en tiempo O(nlogn).

El Problema de la Autopista. La localización de los servicios es parte importante de las ciencias gerenciales y de transporte. Algunos problemas de esta área se pueden resolver usando matemáticas elementales, como en el ejemplo siguiente.

Supongamos que tenemos dos pueblos a y b cercanos a una autopista recta y una compañia de tiendas de ferretería (servicios) quiere instalar una sucursal para atender la demanda de ambos pueblos. Nos preguntamos entonces: ¿En que punto p de la autopista debe instalarse la tienda, de tal manera que las distancias de cada pueblo a la tienda sea mínima?

El problema, desde el punto de vista geométrico, consiste en hallar un punto p sobre la recta l tal que minimice la expresión:

$$d(a, p) + d(b, p)$$

Donde $d(x, y)$ denota la distancia euclideana entre dos puntos.

Este problema era conocido desde hace varios milenios y fue resuelto por el matemático Herón de Alexandría en el año 100 D.C. Herón no hizo mas que emplear las leyes de reflexión de la luz sobre un espejo establecidas por Euclides en su libro *Catoptrica*. Las dos leyes establecen lo siguiente:

1. El ángulo de incidencia de la luz está sobre el mismo plano que el ángulo de refracción.

2. La medida del ángulo de incidencia es igual a la del ángulo de refracción.

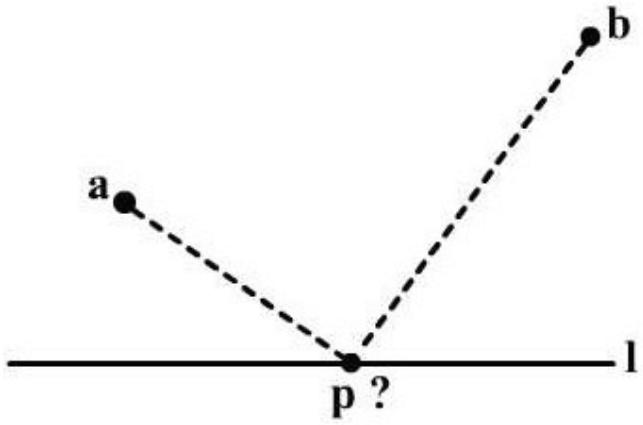

Figura 8: Problema de la Autopista

Inspirados en este principio veremos que la localización del punto p debe ajustarse a estas leyes y de esta manera probaremos también que la luz se propaga siguiendo el camino más corto.

Entonces elegimos en punto p sobre l, tal que $\alpha = \beta$ (ver la figura).

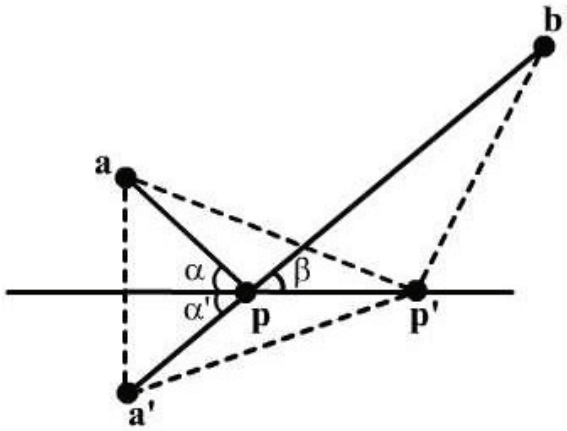

Figura 9: Suma de distancias

Sea p' otra posible localización, con $p' = p$. Probaremos entonces que la suma de distancias $d(a, p') + d(p', b)$ es mayor que $d(a, p) + d(p, b)$.

Sea a' la reflexión de a sobre l. Entonces $\alpha = \alpha'$, y como $\alpha = \beta$ se tiene $\alpha' = \beta$, por lo tanto, los puntos a', p y b estan sobre una misma recta. Luego se tienen las siguientes relaciones enter las distancias

$$\begin{aligned} d(a,p') + d(p',b) &= d(a',p') + d(p',b) \\ &> d(a,b) \\ &= d(a',p) + d(p,b) \\ &= d(a,p) + d(p,b) \end{aligned}$$

Por lo tanto la longitud de la trayectoria $ap'b$ es mayor que apb. luego el punto p es la mejor ubicación de la tienda sobre la autopista.

El problema del círculo mínimo. Este es un problema del área de gerencia, cuya versión clásica se puede plantear de la manera siguiente: Supongamos que tenemos n puntos en el plano representando clientes, plantas de producción para ser abastecidas, escuelas, hospitales, mercados, pueblos o cualquier otro tipo de institución.

El problema consiste en ubicar un punto X en el plano representando un servicio (proveedor, transmisor o despachados) de tal forma que la distancia desde X hasta el punto más alejado sea mínima. Este criterio es de gran utilidad para ubicar hospitales, estaciones de policía, bomberos,..etc, en los cuales es necesario minimizar el peor de los casos en cuanto a tiempo de respuesta.

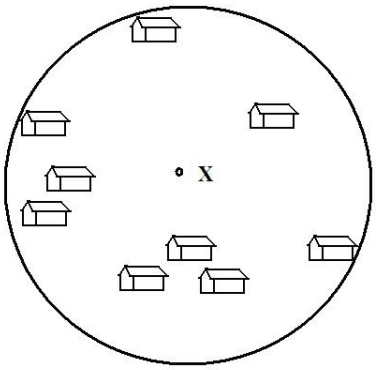

Figura 10: El círculo mínimo

De manera muy breve, clara y elegante se puede enunciar el problema en el área de la geometría.

Hallar el menor círculo que encierra un conjunto de puntos.

Dicho problema se conoce en la literatura especializada como: **Problema del círculo mínimo** (Minimum spanning circle). Apareció por vez primera en 1857 en un trabajo del matemático inglés Sylvester.

Un algoritmo intuitivo como el que veremos a continación, nos da la solución en tiempo $O(n^4)$. En 1972 Elzinga y Hearn hallaron un algoritmo más rápido en tiempo $O(n^2)$. Posteriormente, Shamos desarrolló un algoritmo más rápido aún, en tiempo $O(nlogn)$. Después de una larga búsqueda de algoritmos cada vez más eficientes por parte de los investigadores; finalmente, en 1983 N. Meggido dio un algoritmo de complejidad $0(n)$. Este algoritmo será objeto de estudio en el capítulo 4.

Podemos resolver este problema usando un algoritmo de fuerza bruta inspirados en la siguiente idea.

1. Para todo par de puntos a y b determinamos el **círculo diametral**, es decir, aquel cuyo diámetro es igual a la distancia desde a hasta b, centrado en el punto medio del segmento.

2. Si con esto no cubrimos todos los puntos, para cada tres puntos buscamos el círculo inscrito en el triángulo cuyos vértices son los tres puntos (**círculo triangular**).

3. Revisamos si todos los puntos están dentro del círculo 1) o 2) y nos quedamos con el menor.

Se puede probar fácilmente que este algoritmo tiene complejidad $0(n^4)$, siendo n el número de puntos. A continuación damos el pseudocódigo del algoritmo círculo mínimo.

Algoritmo Círculo Mínimo

- ENTRADA: Un conjunto S de n puntos del plano.
- SALIDA: El círculo mínimo que cubre S.

BEGIN

1. Si S contiene menos de 4 puntos construya el círculo mínimo directamente.
2. Sean p_1 y p_2 dos puntos de S. Sea C el círculo de diámetro p_1p_2y centro en el punto medio de p_1 y p_2.

3. Para $i = 1, .., n$ calcule la distancia $di = d(c, p_i)$
4. Si $di \leq (p_1, p_2)/2$ para todo i, $C =$ círculo mínimo. Else GO TO 2.
5. Sean p_1, p_2 y p_3 tres puntos de S.
   Sea C el círculo determinado por esos tres puntos con centro c y radio r
6. Para $i = 1, .., n$ $di = d(p_i, c)$
7. Si $di \leq r$, $C =$círculo mínimo. Else GO TO 5.
8. Círculo mínimo = menor de los círculos en 7.

END.

Para estudiar la complejidad de este algoritmo, basta considerar los siguiente: Los pasos 2-4 son de orden $0(n^3)$. Los pasos 5-7 son de orden $0(n^4)$. Luego el algoritmo es de orden $0(n^4)$.

El algoritmo de Elzinga y Hearn es el siguiente:

ALGORITMO CÍRCULO MÍNIMO

- ENTRADA: Un conjunto de puntos del plano.
- SALIDA: El círculo mínimo.

BEGIN

1. Elija una pareja de puntos p_i y p_j.
2. Construya el círculo C de diámetro $p_i p_j$.
3. Veriificar si todos los puntos de S están en C.
4. Si 3 es positivo C es el círculo mínimo.
5. Caso contrario, tomar otro punto p_k fuera de C.
6. Si el triángulo $\Delta p_i p_j p_k$ es obtuso o rectángulo renombrar
   p_i, p_j los puntos más alejados y volver a 2.

7. Caso contrario el triángulo $\Delta p_i p_j p_k$ es agudo.
   Construya el círculo C que pase por los tres puntos.

8. Si C contiene todos los puntos de S, entonces C es el círculo mínimo.

9. Caso contrario tomar p_l fura del círculo.

10. Sea Q el punto de $\{p_i p_j p_k\}$ más alejado a a p_l y N el centro del círculo.

11. Sea R el punto de $\{p_i p_j p_k\}$ que está del lado opuesto de p_l con
    relación a la recta que contiene el segmento QN.

12. Con los puntos Q, R, y p_l volver a 6.

END.

EJERCICIOS

1. Demuestre la fórmula (1.1) para calcular el área de un triángulo.

2. Una **Combinación convexa** de puntos x_1,x_2,..,x_k es una suma de la forma $\alpha_1 x_1 + .. + \alpha_k x_k$, con $\alpha_i \geq 0$ para todo i y $\alpha_1 + ... + \alpha_k = 1$. Demuestre que la envolvente convexa de $S = \{x_1, .., x_k\}$ es igual al conjunto de todas las combinaciones convexas de estos k puntos.

3. Un **Semiplano** es el conjunto de puntos del plano que se encuentran a un lado de una línea. Demuestre que la envolvente convexa de S es igual a la intersección de todos los semiplanos que contienen a S.

4. Demuestre que toda circunsferencia queda completamente determinada al conocer tres puntos sobre ella. Halle una fórmula para calcular su centro en función de las coordenadas de los puntos.

5. Sea $S = \{p_1, p_2, .., p_{10}\}$, donde $p_1 = (0, 0)$, $p_2 = (0, 1)$, $p_3 = (1, 1)$, $p_4 = (2, 4)$, $p_5 = (3, 9)$, $p_6 = (4, 16)$, $p_7 = (3, 4)$, $p_8 = (4, 10)$, $p_9 = (5, 12)$, y $p_{10} = (10, 10)$. Entonces hallar la envolvente convexa de este conjunto.

6. Resuelva el problema anterior, pero añadiendo el punto $p_{11} = (10, 20)$.

7. Halle el círculo mínimo que contiene los puntos $p_1 = (0, 0)$, $p_2 = (2, 7)$, $p_3 = (0, 4)$ y $p_5 = (4, 0)$.

8. (Algoritmo de Elzinga y Hearn) Sea C el círculo de diámetro p_ip_j, y sea p_k un punto exterior al círculo, tal que el triángulo $\Delta p_ip_jp_k$ es obtuso. Sean P y Q los puntos más alejados entre $\{p_ip_jp_k\}$. Probar que el círculo C', cuyo diámetro es PQ, contiene a C.

9. Demuestre que el Algoritmo de Elzinga-Hearn termina en algún momento. Notar que en cada paso se incrementa el diámetro de los círculos y se incorpora al menos un nuevo punto a C.

10. Demuestre que los puntos antipodales de un conjunto S están sobre la envolvente convexa.

TRIANGULACIONES

0.6. Problema de la galería de arte

Uno de los aspectos más importantes de la geometría computacional es el de dividir un polígono en triángulos. Una motivación bastante interesante hacia esta teoría es el problema de las galerías de arte, propuesto por Klee en 1976.
La siguiente exposición esta tomada del libro *Computational Geometry in C* de Joseph O'Rourke. Supongamos que tenemos una galería de arte cuya planta tiene la forma de un polígono de n vértices. La pregunta es la siguiente: ¿Cuántos vigilantes son necesarios para proteger la galería?

Cada vigilante se considera como un punto fijo dentro del salón y además puede visualizar todo a su alrededor en un ángulo de 360^o. Los vigilantes no pueden ver a través de las paredes. Se supone que los guardianes no bloquean la visión a nadie.

Antes de atacar el problema, necesitamos precisar un par de términos para obtener una forma rigurosa del planteamiento, desde el punto de vista matemático.

1. (Visibilidad) Diremos que un guardián a puede ver el punto p en el polígono (ó que p es visible desde a), si el segmento ap está en el interior del polígono.

2. (Cobertura) Diremos que un conjunto de guardianes cubre el polígono, si todo punto en el polígono es visible para algún guardián.

Ejemplo 0.6.1. *Si una galería tiene planta rectangular, entonces un solo guardián es suficiente para cuidarla. El guardia puede ser ubicado en cualquier punto del rectángulo (ver la figura).*

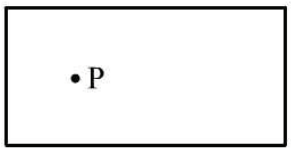

Figura 11: Galería rectangular

Ejemplo 0.6.2. *Si la galería tiene forma de un polígono convexo, entonces un solo guardián es suficiente (ver figura).*

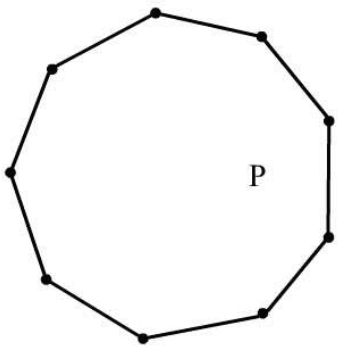

Figura 12: Galería convexa

Vemos entonces que los casos interesantes se presentan para polígonos no convexos.

Ejemplo 0.6.3. *Consideremos el caso en que la galería esté limitada por un polígono cualquiera de 12 vértices. Entonces hay distintas posibilidades , según la forma de polígono (ver la figura).*

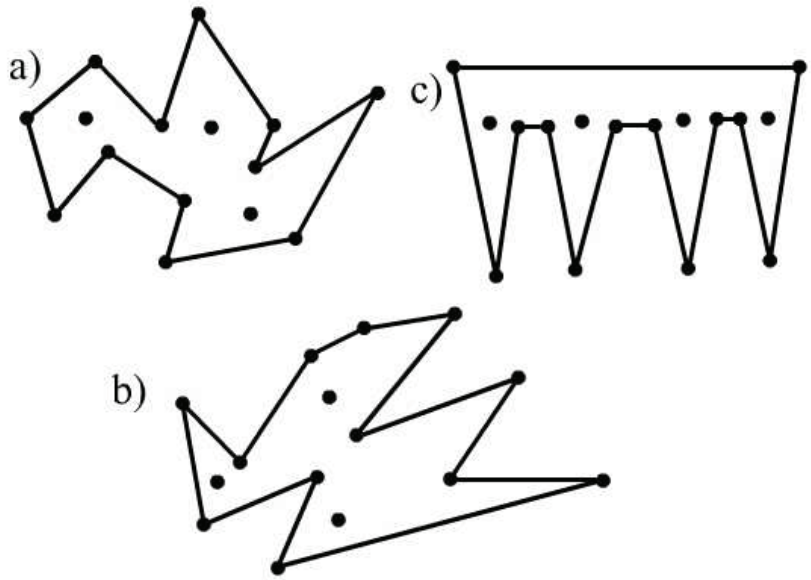

Figura 13: Tres galerías distintas con 12 lados

Antes de continuar daremos unas notaciones apropiadas. Para cada polígono P sea $g(P)$ el mínimo número de guardias necesario para cubrir P. Se puede ver intuitivamente que si P_a, P_b y P_c son los polígonos correspondientes a la figura, entonces.

$$g(P_a) = 3, \quad g(P_b) = 3 \quad y \quad g(P_c) = 4$$

Podríamos entonces preguntarnos: ¿Será posible que 4 guardias sean suficientes para cubrir cualquier polígono de 12 lados?

Si P_n indica un polígono de n-vértices, entonces definimos la función de guardianes.

$$G(n) = min_{P_n}\{g(P_n)\}$$

Es decir, $G(n)$ es el mínimo número de guardianes necesarios para cubrir cualquier polígono de n vertices.

Observación Notemos en primer lugar que $G(n)$ siempre existe para todo n. Ademas, $G(n) \leq n$, pues siempre es posible cubrir todo el polígono de n vértices con n guardianes, ubicando un guardián en cada vértice.

Si $n = 3$, entonces es claro que un guardia es suficiente y por lo tanto $G(3) = 1$.

Si $n = 12$, entonces podemos conjeturar que $G(12) = 4$, con base en nuestra experiencia con el ejemplo anterior.

¿Se ve ahora alguna formula? Sugerimos al lector considerar los casos $n = 4, 5, .., 12$.

Parece ser que

$$G(n) = \left[\frac{n}{3}\right]$$

, donde [ ] denota la parte entera de un número real.

Veremos que si $n = 3k$, entonces $\frac{n}{3} \leq G(n)$. Para $n = 3k$ construiremos un polígono con k dientes, llamado "Peine de Chvátal" (ver la figura).

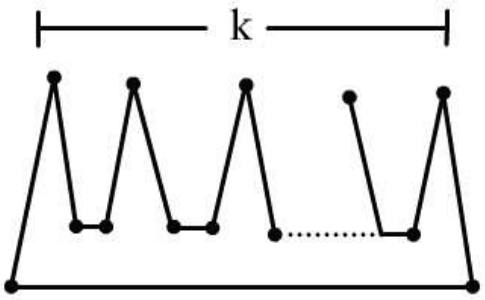

Figura 14: Peine de Chvatal

Veamos que para este polígono se requiere un guardián para cada diente y, por lo tanto, en este ejemplo.

$$G(n) = K = \frac{n}{3}$$

Entonces se tiene la desigualdad $\left[\frac{n}{3}\right] \leq G(n)$. Para probar la igualdad se requiere un poquito mas de trabajo. Eso fué probado por Chvátal en 1975.

Teorema 0.6.1. *Teorema de Chvátal - Para cuidar una galería de n vértices,* $\left[\frac{n}{3}\right]$ *guardianes son suficientes. Esto es*

$$G(n) \leq \left[\frac{n}{3}\right]$$

La prueba original es por inducción sobre el número de vértices. Daremos la prueba de Fisk (1978) basada en la triangulación de un polígono y los grafos coloreados.

0.7. Triangulación de polígonos

El tema de la triangulación de un polígono es uno de los más importantes en Geometría computacional, por la gran cantidad de aplicaciones que se derivan del mismo. En esta sección nos limitamos solamente a dar algunos aspectos teóricos para iniciarnos. Más adelante estudiaremos algunos algoritmos para triangular. También, al final del último capítulo, daremos un algoritmo especial, llamado Triangulación de Delauny, para realizar este proceso de manera bastante eficiente sobre cualquier polígono dado.

Definición 0.7.1. *Una* ***Triangulación*** *de un polígono P es una partición de éste en triángulos de tal forma que los vértices de cada triángulo sean vértices del polígono y no haya ningún corte entre los lados de los triángulos.*

Por ejemplo en la figura 12, vemos dos triangulaciones distintas de un mismo polígono. Los triángulos se han formado añadiendo diagonales.

Definición 0.7.2. *Una* ***Diagonal de un polígono*** *es un segmento de recta que une dos vertices y que está contenida en el interior del poígono. Dos diagonales no pueden intersectarse.*

Probaremos que todo polígono se puede triangular añadiendo diagonales.

Definición 0.7.3. *Un vértice V de un polígono se dice* ***Estrictamente convexo*** *si el ángulo que forman sus lados adyacentes es menor que* π.

Recordemos que en un polígono P los lados están ordenados en el sentido contrario a las agujas del reloj. Una persona que haga una caminata por todo el polígono en este sentido, tendrá siempre el interior del polígono a su izquierda. Al llegar a un vértice estrictamente convexo se produce un giro a la izquierda.

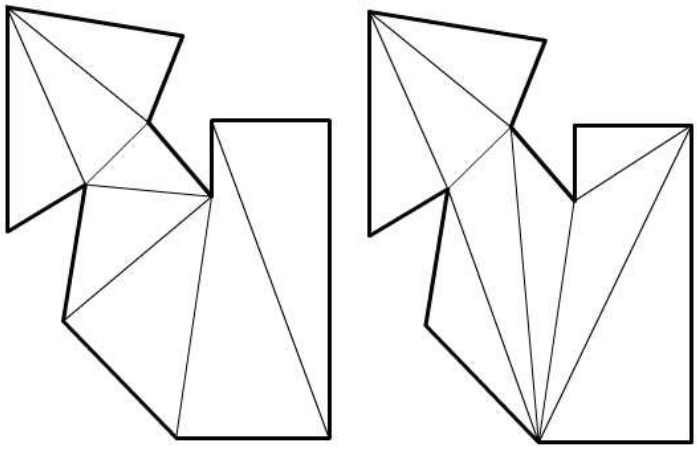

Figura 15: Dos triangulaciones de un mismo polígono

Definición 0.7.4. *Un vértice de un polígono se dice* ***Vértice Reflex*** *si el ángulo entre los lados adyacentes es mayor que* π.

Es claro que todo polígono posee al menos un vértice covexo. Este hecho que nos parece tan trivial e intuitivo, amerita sin embargo, una demostración formal.

Lema 0.7.1. *Todo polígono posee al menos un vértice estrictamente convexo.*

Demostración: Consideremos v_0 como el vértice más bajo del polígono. Es decir, v_0 es el vértice cuya ordenada es mínima entre todos los puntos. Puede haber varios vértices situados a la misma altura que v_0. Si se presenta esta situación, entonces v_0 es aquél situado más hacia la derecha que los otros. Todos estos vértices estarán situados sobre una recta horizontal L(ver la figura).

Entonces v_0 es un vértice estrictamente convexo, pues el lado que une a v_0 con el vértice siguiente está sobre la recta L

Lema 0.7.2 (Meisters)**.** *Todo polígono de n vértices, con* $n \geq 4$*, posee una diagonal.*

Demostración: Sea v un vértice estrictamente convexo. Sean a y b los vértices adyacentes a v. Si ab es una diagonal. Entonces estamos listos. Supongamos que ab no es una diagonal. Como v es convexo, ab no está contenido en el interior de P y por lo tanto hay algunos vértices de P en el triángulo $\triangle avb$ (ver la figura).

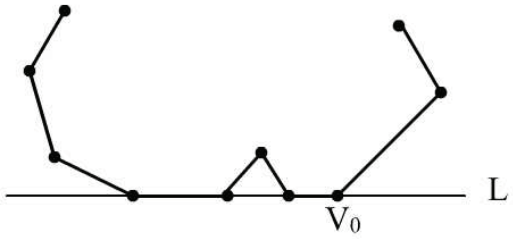

Figura 16: Un vértice estrictamente convexo.

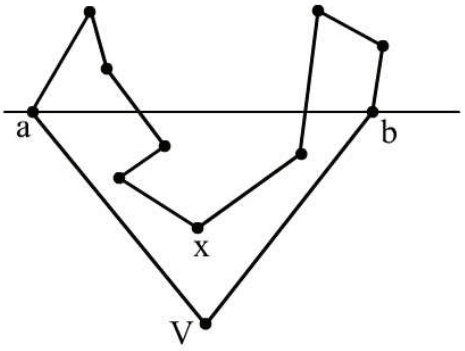

Figura 17: Trazando una diagonal

De todos esos vértices dentro del triángulo $\triangle avb$, sea x el más cercano a V, midiendo las distancias perpendiculares a la linea ab. Afirmamos que $v.x$ es una diagonal.

En efecto el segmento vx está contenido en el interior de P y no intersecta a la frontera salvo en v y en x.

Teorema 0.7.1. ***Triangulación de polígonos*** *Todo polígono de n vértices se puede seccionar en triángulos mediante la adición de diagonales (0 en el caso de $n = 3$)*

Demostración: Usaremos inducción sobre n. Para $n = 3$ no hay que añadir ninguna diagonal y el teorema es cierto.

Supongamos entonces que $n > 3$. Por el lema anterior existen dos vertices de P, a y b , tal que ab es una diagonal de P. Entonces la diagonal divide a P en dos polígonos P_1 y P_2, cada uno de ellos con menos vértices que n y por lo tanto, por hipótesis de inducción, los podemos triangular. Uniendo ambas triangulaciones se obtiene una triangulación de P.

Observación Una pregunta natural que puede surgir en la mente del lector, es la siguiente: ¿Cuántos triángulos aparecen en la triangulacion de un polígono de n lados?

Ejemplo 0.7.1. *Si P es un polígono n lados, entonces la siguiente triangulación en abanico nos proporciona $n-2$ triángulos (ver la figura).*

Teorema 0.7.2. *Todo polígono P de n vértices se puede triangularizar usando $n-3$ diagonales en $n-2$ triángulos.*

Demostración: La demostración es por inducción. Para $n = 3$ es cierto. Sea $n \geq 4$. Trazando una diagonal ab se divide al polígono en dos polígonos menores P_1 y P_2 de n_1 y n_2 vértices respectivamente. Tenemos entonces que $n_1 + n_2 = n + 2$, puesto que ab es un lado común a ambos polígonos y por lo tanto hay una repetición de 2 vértices.

Aplicando la hipotesis de inducción a ambos polígonos se tiene que hay $(n_1-3)+(n_2-3)+1$ diagonales en total para P.

Pero $n_1 + n_2 - 5 = n - 3$. luego hay $n - 3$ diagonales. Por otro lado el número total de triángulos es:

$$(n_1 - 2) + (n_2 - 2) = n_1 + n_2 - 4 = n - 2$$

Con esto se da fin a la demostración.

Corolario 0.7.1. *En todo polígono de n vértices, la suma de los ángulos internos es $(n-2)\pi$*

Demostración. Basta sumar π por cada triángulo.

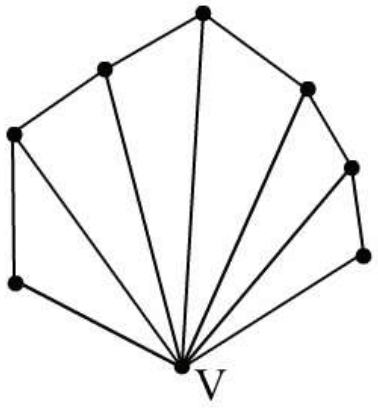

Figura 18: Triangulación en abanico

0.8. Grafos

A continuación daremos una serie de conceptos básicos sobre la Teoría de grafos.

Definición 0.8.1. ***Un grafo*** *G consiste de un conjunto V de puntos llamados* ***vértices*** *o nodos y un multiconjunto E de pares de la forma $[a, b]$, con $a, b \in V$, llamados* ***lados****. Denotamos el grafo G, mediante $G = (V, E)$.*

Definición 0.8.2. *Sea $G = (V, E)$ un grafo. Una* ***cadena*** *C en G es un conjunto de vértices $\{x_1, x_2, \cdots x_k\}$, tales que los lados $\{[x_1, x_2], [x_2, x_3], \cdots, [x_{k-1}, x_k]\}$ están todos en E.*

Definición 0.8.3. *Un* ***ciclo*** *en un grafo $G = (V, E)$ es una cadena $C = \{x_1, x_2, \cdots x_k\}$, tal que el primer vértice x_1 es igual al último x_k.*

Observación Si en el ciclo C todos los vértices son distintos, salvo los extremos, esto es $x_i \neq x_j$ con $1 < i, j < k$, entonces el ciclo se llama un **Ciclo simple**.

Definición 0.8.4. *En un grafo $G = (V, E)$ el* ***orden de un vértice v*** *es igual al número de lados que se conectan con v.*

Observación Si el lado $[x, x]$ está en el conjunto E, entonces hay que contarlo doble, cuando calculamos el orden de x.

Definición 0.8.5. *Diremos que un grafo es* ***conexo*** *si es un solo punto o bien todo par de vértices distintos están conectados por una cadena.*

0.9. Demostración del Teorema de Chvátal

Una triangulación de un polígono puede ser considerada como un grafo. El **Grafo dual** de una triangulación es otro grafo con un nodo dentro de cada triángulo y cada nodo conectado con otro sí, y sólo si los triángulos son vecinos (ver figura).

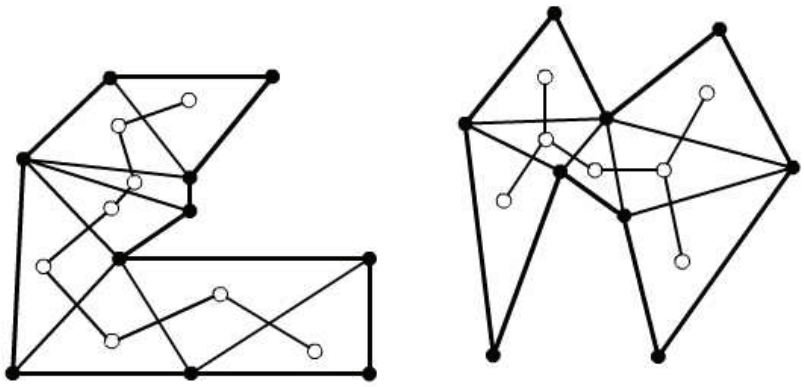

Figura 19: Grafos duales

Lema 0.9.1. *El dual T de una triangulación de un polígono P, es un árbol, siendo cada nodo de grado a lo sumo 3.*

Demostración: Recordemos que un árbol es un grafo conexo que no contiene ciclos. Supongamos que T contenga un ciclo que se puede dibujar en el plano como una trayectoria cerrada π, teniendo por lados segmentos rectilíneos que cortan perpendicularmente las diagonales en su punto medio. Entonces π debe encerrar algunos vértices del polígono P, digamos, un punto extremo de cada diagonal intersectada por π. Entonces π debe intersectar puntos del exterior de P. Esto es una contradicción, pues P es un polígono simple.

Observación: En un árbol, los nodos de grado 1 corresponden a las hojas. Notemos que el triángulo correspondiente en P a un nodo de grado 1, es del tipo $\triangle abc$ con a, b y c vértices consecutivos del polígono y ac una diagonal. Diremos entonces que los tres vértices **Forman una Oreja**(ver la figura).

En la figura, abc es una oreja. También dab es otra oreja que intersecta a la primera.

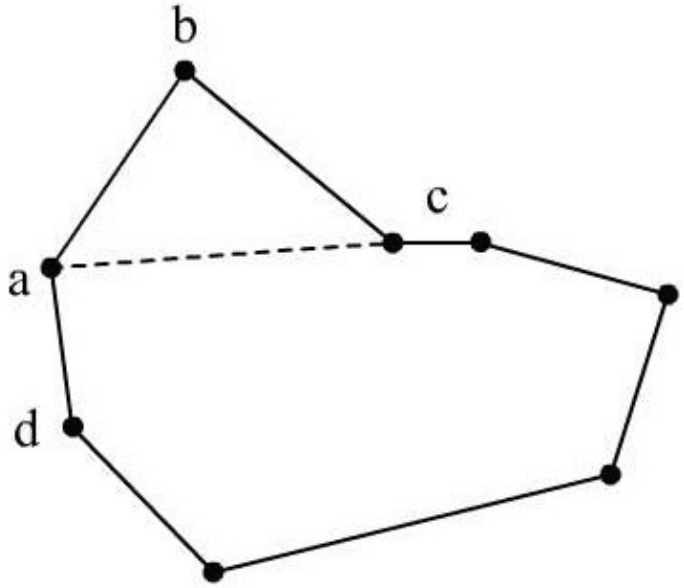

Figura 20: Oreja en un polígono

Teorema 0.9.1 (Meisters de las dos orejas)**.** *Todo polígono de n vértices, con $n \geq 4$ tiene como mínimo dos orejas que no se intersectan.*

Demostración: Sea T una triangulación de polígonos, entonces su dual es un árbol H. Como $n \geq 4$, entonces la triangulación tiene al menos 2 triángulos y por lo tanto H tiene al menos dos nodos. Entonces en H debe haber al menos 2 hojas y cada hoja corresponde a una oreja en el polígono P.

Definición 0.9.1. *Un grafo G admite una* ***3-Coloración*** *o puede ser coloreado con tres colores si cada vértice se puede colorear de tal forma que los vértices adyacentes tengan colores distintos.*

Sea P ahora un polígono cualquiera de n lados. Tenemos entonces que P se puede triangular añadiendo diagonales. Sea T el grafo de dicha triangulación. Entonces podemos enpezar a colorear un vértice con un color (rojo) los otros dos vértices del triángulo los coloreamos con colores distintos (verde y amarillo). Como cada vértice de la triangulación es de orden 3 a lo sumo, es posible continuar coloreando el grafo hasta el final. La demostración de este hecho viene dada en el siguiente teorema.

Teorema 0.9.2 (Tricolor)**.** *El grafo de la triangulación de un polígono P admite una 3-coloración.*

Demostración: Inducción sobre el número de vértices n. Si $n = 3$, entonces un triángulo puede colorearse con tres colores

Supongamos $n \geq 4$. Por el teorema de Meisters, el polígono P tiene una oreja $\triangle abc$ tal que ac es una diagonal. Cortamos esta oreja y tenemos un grafo P' con un vértice menos. Por hipótesis de inducción P' se puede colorear. Coloreamos entonces el vértice b con un color distinto al de a y al de c. Esto nos proporciona una 3-coloración de P.

Tenemos ahora todos los elementos necesarios a la mano para dar la demostración del Teorema de Chvátal sobre galerías de arte.

Supongamos que tenemos una galería de arte cuya planta tiene la forma de un polígono P de n-vértices.

Daremos la demostración dada por Fisk, la cual parte de triangular el polígono P. Sea T el grafo asociado a la triangulación de P.

Hemos demostrado que P se puede colorear con 3 colores. Por ejemplo rojo, verde y amarillo. El siguiente paso es elegir un color cualquiera, por ejemplo, el rojo. Esto garantiza que se puede cubrir todo el polígono.

En efecto, como el interior de P queda dividido en triángulos y en cada triángulo debe haber un vértice rojo, entonces cada guardían tiene completa visibilidad sobre su triángulo y al considerar todos los guardianes, éstos cubren todo el polígono.

El número de guardianes no se puede calcular. Puede haber más vértices de un color que otro. Pero, usando el principio de los casilleros de Hilbert, hay un color que colorea $\left[\frac{n}{3}\right]$ vértices a lo sumo. Luego, tomamos este color y colocamos a los guardias en los vértices de este color.

0.10. Algoritmos de triangulación

En esta sección daremos algunos algoritmos para triangular un polígono cuando se conocen sus vértices. La búsqueda de diagonales cumple un papel central en todos estos métodos.

La triangulación de un polígono es un tipo de grafo que contiene una gran cantidad de datos. Se requiere entonces manejar estos datos correctamente para poder diseñar algoritmos.

Entre las operaciones más usadas se encuentran

1. Visitar todos los lados o vértices de un grafo usando un camino trazado de antemano.
2. Recorrer cada una de las caras en ambos sentidos.
3. Eliminar lados o vértices.
4. Incluir lados o vértices.

Una representación que soporta todas estas operaciones es la **Lista doblemente ligada** (DCEL, por sus iniciales en inglés). Ésta es una lista de los lados en donde a cada elemento se le asigna un registro con la siguiente información.

1. Vértice inicial y vértice final del lado.
2. Caras o regiones adyacentes al lado.
3. Un par de apuntadores: Uno hacia el lado siguiente y otro hacia el lado anterior.

Ejemplo 0.10.1. *Para el grafo G de vértices $\{a, b, c, d\}$ y lados $\{e_1, e_2, e_3, e_4\}$. se ha establecido el siguiente camino para recorrer los vértices:*

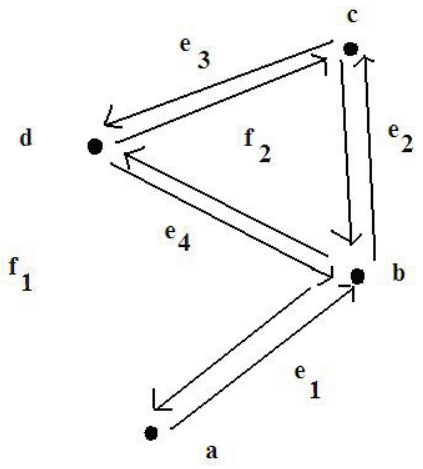

Figura 21: Un grafo con un camino de recorrido

La lista de los lados junto con el recorrido es la siguiente:

Lado	V_1	V_2	F_1	F_2	P_1	P_2
e_1	v_1	v_2	f_1	f_1	e_2	e_4
e_2	v_2	v_3	f_1	f_2	e_3	e_1
e_3	v_3	v_4	f_1	f_2	e_2	e_4
e_4	v_4	v_2	f_1	f_2	e_1	e_2

Sea P un polígono de n lados y a y b un par de vértices. Veamos bajo qué condiciones el segmento ab forma una diagonal del polígono. Recordemos que una condición necesaria y suficiente viene dada por:

- ab no corta ningún lado del polígono.
- ab no corta ninguna diagonal.
- ab está contenido dentro de P.

¿Qué condiciones deben cumplirse para que el segmento ab corte al segmento cd ?

Podemos dar un criterio sencillo basado en las posiciones relativas de los puntos extremos(ver la figura).

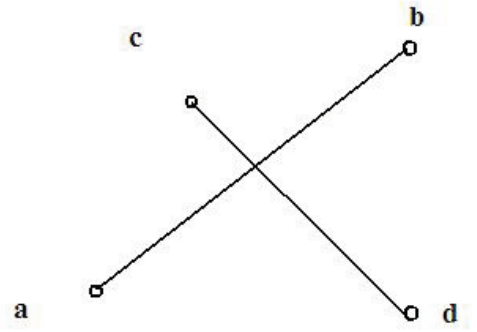

Figura 22: Intersección de segmentos

La condición se expresa de la siguiente manera:

- Los puntos a y b están en lados distintos del segmento cd.
- Los puntos c y d están en lados distintos del segmento ab.

Para hacer esta prueba hay que calcular las áreas signadas de los cuatro triángulos: $\Delta(a,b,c)$, $\Delta(a,b,d)$, $\Delta(c,d,a)$ y $\Delta(c,d,b)$. Luego, se deben estudiar los signos correspondientes para determinar las posiciones relativas de los puntos.

¿Cómo se determina si la diagonal ab cae dentro del polígono?

Esto requiere de un procedimiento especial, el cual depende del tipo de vértice en a. Si a es un vértice convexo y c, d son los vértices adyacentes, entonces el ángulo ϕ entre los lados ac y ad es menor que π, medido en el sentido contrario a las agujas del reloj. Este arco proyecta un cono sobre el polígono. La diagonal ab será interior al polígono si se halla dentro de este cono.

Para chequear esta condición basta con determinar si los puntos c y d están en lados distintos de la recta que une los puntos a y b (ver la figura).

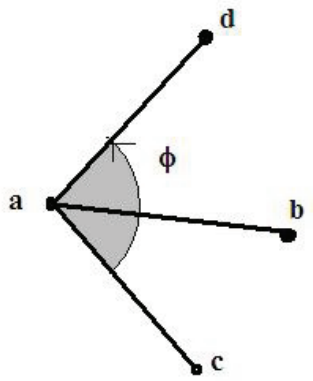

Figura 23: vértice convexo

En caso de ser un vértice reflex, el ángulo α entre los lados adyacentes es mayor que π. Vemos entonces que en este caso el criterio anterior puede fallar. Sin embargo, el complemento del cono proyectado por α es un cono convexo y entonces podemos decir que ab está dentro del polígono si no se encuentra en este cono (ver la figura).

Según estos criterios podemos diseñar un par de algoritmos de triagulación de un polígono. Nuestro primer algoritmo es de fuerza bruta, pues busca trazar todas las posibles diagonales en el polígono. Para cada par de vértices v_i, v_j se determina si el segmento $v_i v_j$ es una diagonal. En caso de ser positivo el test, se traza la diagonal. Esto tiene un costo de $O(n^2)$.

Como hay $\dfrac{n(n-1)}{2}$ posibles parejas, entonces este algoritmo es de una complejidad de

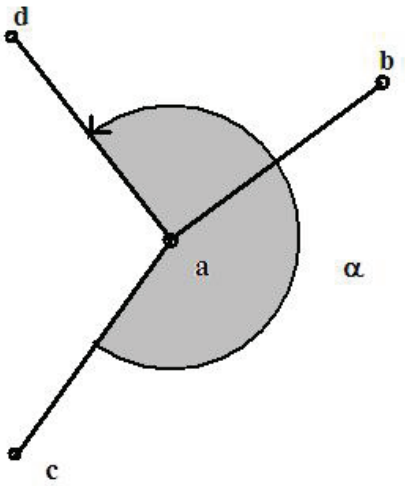

Figura 24: vértice reflex

orden $O(n^4)$.

Podemos bajar la complejidad a $O(n^2)$ diseñando un algoritmo más eficiente, que en cada paso corta una oreja del polígono. En primer lugar, para cada uno de los vértices v_i chequeamos si es o no una punta de oreja, o lo que es lo mismo, determinar si el segmento $v_{i-1}v_{i+1}$ es una diagonal. Esto tiene un costo de $O(n)$, pues hay que chequear intersección con los n lados del polígono. Una vez hecho esto, si v_i es punta de oreja trazamos la diagonal $v_{i-1}v_{i+1}$. Por supuesto, al trazar una diagonal estamos modificando el polígono y la condición de punta de oreja de los puntos restantes. Pero esto sólo afecta a los dos vértices vecinos de v_i, los cuáles son v_{i-1} y v_{i+1}. Así pues, en cada paso se debe revisar la condición de punta de oreja sólo para un par de puntos maás.

Como hay un total de n vértices en el polígono, el algoritmo es de complejidad $O(n^2)$. Más adelante veremos otros algoritmos de triangulación más eficientes.

Ejercicios

1. Diseñe una galería de n lados y una colocación de guardianes de tal forma que sean capaces de visualizar todas las paredes de la galería, pero no algunos puntos del interior.

2. Escriba un código para un algoritmo intuitivo de triangulación que trace diagonales entre los vértices

3. Escriba un código para un algortimo de triangulación basado en el corte de orejas.

4. **El problema de Euler** ¿Cuántas triangulaciones distintas posee un polígono regular de n lados?

5. **Centro de masa** Supongamos que tenemos un triángulo plano hecho de un material uniforme en cuanto a su densidad. Demuestre que el centro de masa o centro de gravedad del mismo está ubicado en su centroide, cuyas coordenadas son iguales a las coordenadas promedios de los vértices. Si S es un conjunto cualquiera el centro de gravedad es un vector que será denotado por $\gamma(S)$. Si S es la unión disjunta de dos conjuntos A y B, entonces el centro de gravedad de S es el promedio de los pesos de cada conjunto. Sea $\omega(S) = \omega(A) + \omega(B)$ el peso de S, entonces

$$\gamma(S) = \frac{\omega(A)\gamma(A) + \omega(B)\gamma(B)}{\omega(S)}. \tag{5}$$

   Siendo el peso de cada conjunto igual a su área, si asumimos que la densidad es uniforme.

6. Usando el problema anterior, escriba un algoritmo para hallar el centro de masa de un polígono.

LA ENVOLVENTE CONVEXA

En este capítulo estudiaremos uno de los conjuntos fundamentales de la geometría computacional, la envolvente convexa. Si S es un conjunto finito de puntos en el plano, la envolvente es el mayor convexo que los contiene. El borde o frontera del mismo es un polígono cerrado convexo. Entre las aplicaciones prácticas de la envolvente mencionaremos las siguientes.

1) **Robótica** . El conocimiento de la envolvente convexa de un robot evita las colisiones del mismo con los obstaculos en su trayectoria de desplazamiento. De esta manera el conocimiento de la envolvente ayuda a buscar las posibles trayectorias del desplazaientodel robot.

2) **Diámetro de un conjunto de puntos**. Definimos el diámetro de un conjunto de puntos como la máxima distancia que puede existir entre sus pares. Se puede probar que este diaḿetro se obtiene para un par de puntos situados sobre la envolvente convexa.

3) **La caja mas pequeña.** Un problema interesante es hallar el rectángulo de área mínima que encierre a un polígono. Este rectangulo tendrá siempre un lado que contiene a uno de los lados de la envolvente convexa.

4) **Análisis de formas.** Algunas estructuras necesitan ser reconstruidas y para esto necesitan ser reconocidas, primero, mediante su envolvente convexa.

En estas notas presentamos cuatro algoritmos distintos para calcular la envolvente convexa.

0.11. Algoritmo de envolvimiento de regalo

Hemos visto en el capítulo 1 un algoritmo bastante intuituvo para calcular la envolvente un conjunto finito S, de complejidad $O(n^3)$. Se puede modificar un poco este algoritmo de una manera inteligente para hacer menos cálculos y así poder bajar la complejidad.

Sea $s = \{x_1, .., x_n\}$ un conjunto del plano. Vamos a construir la envolvente convexa usando un método muy intuitivo conocido como envolvente de regalo ó marcha de Jarvis, descubierto por Chand y Kapur [10] en 1970, y Jarvis [8]. La idea ya fue esbozada en la demostración del lema 2.1

Supongamos que conocemos un lado e de la envolvente, el cual yace en posición horizontal en la parte de abajo y cuyo extremo derecho es igual al punto x. Sabemos que x debe estar conectado con el siguiente punto de la envolvente cuando este polígono se recorre en sentido contrario a las agujas del reloj. Llamaremos a este punto hipotético y. La pregunta es: ¿Cómo hacemos para calcular y?

La idea consiste en trazar una línea L en el plano que tiene un extremo pivote en x, y la cual hacemos girar en sentido antihorario observando el ángulo que forma con el lado e. Entonces el primer punto de S que toca esta línea es y. Este paso se lleva a cabo considerando todas las rectas del tipo L_{xs} con $s \in S$. Podemos entonces comparar las pendientes de todas ellas y tomamos el punto y en S tal que la pendiente de L_{xy} sea mínima (ver la figura).

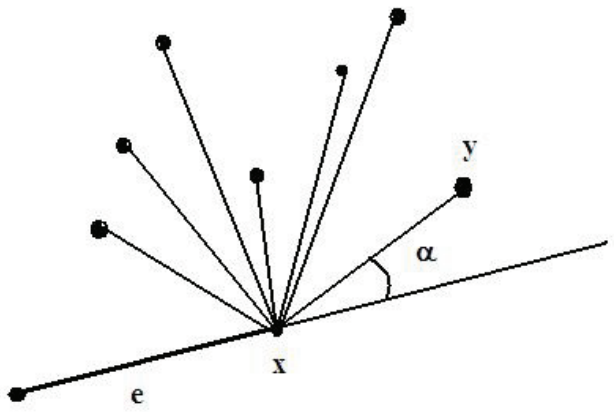

Figura 25: El menor ángulo α con respecto al lado e.

En realidad, en la etapa de implementación del algoritmo no es necesario calcular en ningún momento las pendientes de estas rectas. Calcular ángulos requiere usar aritmética de números reales, funciones trigonométricas y, por lo tanto, caer en errores de redondeo y otro tipo de problemas que debilitan los algoritmos. Una forma bastante delicada de hacer esto, es comparar las pendientes usando nuestra fórmula de área signada para un triángulo.

Es fácil ver entonces que la recta xa tiene menor pendiente que la recta xb sí y sólo si el área del triángulo $\Delta(xba)$ es positiva.

A medida que vamos avanzando caminamos sobre todos los puntos de la envolvente. Por eso se llama también a este algoritmo Marcha de Jarvis. Al llegar a la parte más alta del recorrido, los ángulos empiezan a ser negativos y los medimos de arriba hacia abajo. Después que la línea de barrido ha hecho un giro completo de 360^0 volvemos a la posición inicial y se ha completado el algoritmo.

Finalmente, para hallar el primer punto x en este proceso ordenamos todos los puntos de S según su coordenada X y tomamos el mínimo. Esto tiene un costo de $O(nlogn)$

Este método de trabajo es una técnica muy común en geometría computacional conocido por el nombre de Barrido Geométrico (Geometric Sweeping).

Podemos entonces dar el código del algoritmo.

ALGORITMO ENVOLVIMIENTO DE REGALO

- ENTRADA: Un conjunto S de n puntos.
- SALIDA: La envolvente convexa S.

BEGIN

- Sea $p(1)$ el punto de S con menor coordenada x.
- Sea $p(2)$ el punto de S tal que la pendiente de la recta $p(1)p(2)$ tenga pendiente mínima.
- Plot $(p(1), p(2))$
- $i := 2$;
- Wrile $p(i) \neq p(1)$ DO
- Sea $p(i+1)$ el punto en S tal que el ángulo $< p(i-1)p(i)p(i+1)$ sea mínimo.
- $i := i + 1$
- Plot $(p(i), p(i+1))$

END

Veamos cuál es la complejidad de la Marcha de Jarvis.

Si tenemos K puntos en la envolvente convexa y hay n puntos S, entonces el algoritmo tiene un tiempo de ejecución dado por $O(K.n)$.

En efecto, una vez hallado el primer punto $p(1)$, para hallar el segundo hay que calcular la pendiente de la recta $p(1)X_i$ para los n puntos de S. Lo mismo sucede para cada uno de los puntos de la envolvente $p(i)$ que van apareciendo. Luego, hay que calcular $K.n$ pendientes.

Si el número de puntos de la envolvente convexa es grande (muy cerca a n), entonces el algoritmo es muy lento. En el peor de los casos, cuando $K = n$, entonces el tiempo de ejecución es de orden $O(n^2)$. Este tipo de algoritmo cuya velocidad depende del resultado buscado, se llama de **Algoritmo de salida sensitiva.**

0.12. Algoritmo de Graham

Veamos ahora otro algoritmo bastante eficiente para calcular la envolvente convexa, el cual tiene una complejidad de orden $O(n.\lg n)$. Fue desarrollado por Graham a comienzos de los 70 [9], cuando trabajaba en los laboratorios Bell, en una aplicación que requería calcular la envolvente convexa para un conjunto de aproximadamente 10.000 puntos.

La idea del algoritmo es bastante simple. Tomamos un punto sobre la envolvente y ordenamos en forma radial todos los puntos de S. Al hacer el recorrido en este orden vamos conectando con lados los puntos adyacentes y de esta manera vamos construyendo un polígono. Cuando un punto sea del tipo reflex, entonces lo eliminamos.

Para iniciar el algoritmo elegimos el primer punto de S, denotado por p_0, como el punto de más baja altura. En caso de empate se toma el del extremo derecho. Luego ordenamos todos los puntos de S en el sentido contrario a las agujas del reloj según el angulo que forma el rayo que parte de p_0 y termina en el punto (ver la figura).

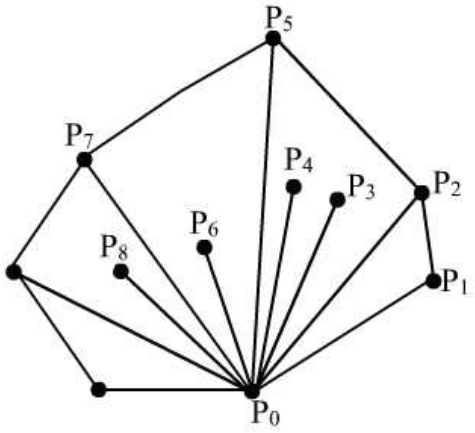

Figura 26: Ordenando los puntos de acuerdo a la pendiente

Entonces p_1 es un punto de la envolvente, al igual que p_2, pues $p_0p_1p_2$ es un giro a la izquierda.

Continuando de esta manera podemos decir que p_3 es también de la envolvente, pues $p_1p_2p_3$ es un giro a la izquierda. Sin embargo, en el siguiente paso vemos que $p_2p_3p_4$ es

un giro a la derecha. Por lo tanto eliminamos el punto p_3 y nos quedamos con los puntos p_0, p_1, p_2, p_4. Para el paso siguiente vemos que $p_2p_4p_5$ es un giro a la derecha y por lo tanto eliminamos al punto p_4 y nos quedamos con p_5. Continuando de esta manera, despues de dar un recorrido completo antihorario, llegamos al punto de inicio p_o.

A fin de poder implementar este algoritmo introducimos una nueva estructura de datos. **La pila** (stack) es una estructura de datos muy usada que permite guardar objetos como números, puntos, figuras, etc.

Comenzamos a almacenar los objetos en el fondo (Bottom) de la pila poniéndolos unos sobre otros. El último objeto almacenado se encuentra en el tope (Top) (ver la figura).

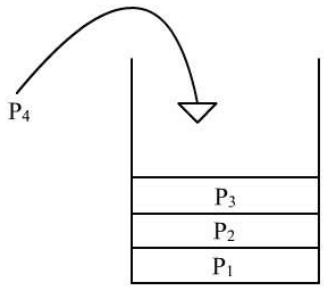

Figura 27: Una pila

Con una pila S se manejan dos operaciones básicas

1. PUSH (p, S), la cual inserta un objeto p en el tope de la pila
2. POP (S), mediante la cual se elimina el objeto que se encuentra en el tope de la pila.

A continuación damos el código para el algoritmo.

Algoritmo de GRAHAM

- `ENTRADA: Un conjunto` S `de` n `puntos del plano.`
- `SALIDA: La envolvente convexa` S`.`

```
BEGIN
```

1. `Sea` p_0 `el punto más bajo de` S `y más a la derecha.`

2. Ordene los otros puntos según el ángulo polar con p_0 el orígen. En caso de empate, elimine los puntos más cercanos al orígen

3. Stack $S = (p_1, p_0) = p_t, p_{t-1}$ con t en el tope
   $i = 2$

4. WHILE $i < n$ do

   - Si $p_{t-1}p_tp_i$ es un giro a la izquierda
     entonces PUSH (p_i, S) y haga $i = i + 1$

   - Else Pop(S)

END

Veamos cuál es la complejidad del algoritmo de Graham.

El paso 1 se ejecuta comparando todas las coordenadas y de los puntos y por lo tanto es de complejidad $O(n)$.

Para el paso 2 se puede usar un algoritmo eficiente de ordenamiento como el Quick Sort, el cual es de complejidad $O(n \lg N)$.

El paso 3 consiste en construir la pila e insertar los 2 primeros puntos. Esto se hace en tiempo constante.

El paso 4 requiere n camparaciones (giro a la izquierda o a la derecha) y como máximo n entradas (Push) en la pila y $n - 3$ salidas (Pop). Todo esto es de una complejidad $O(n)$.

0.13. Algoritmo Quick-Hull

Una idea muy efectiva para acelerar el proceso de cálculo de la envolvente convexa consiste en desechar la mayor cantidad de puntos interiores, pues éstos no califican para la envolvente. Dentro de esta corriente se inscribe el presente algoritmo. Fué desarrollada por distintos investigadores a finales de la deécada de 1970, Eddy(1977)[5] ; Bykat (1978)[6]; Green y Silverman (1979), Floyd (1976). Su nombre, Quick-Hull, se debe a su semejanza con el algoritmo de ordenamiento Quick-Sort.

Iniciamos el proceso dividiendo al conjunto al conjunto S en dos partes casi iguales mediante una recta horizontal L. Denotamos las partes por S_1 y S_2. Tanto S_1 como S_2) contienen un gran triángulo cuyos vértices forman parte de la envolvente convexa de S_1 (resp. de S_2). Entonces este triángulo será una primera aproximación de la envolvente convexa de S_1 (resp. de S_2)(ver la figura).

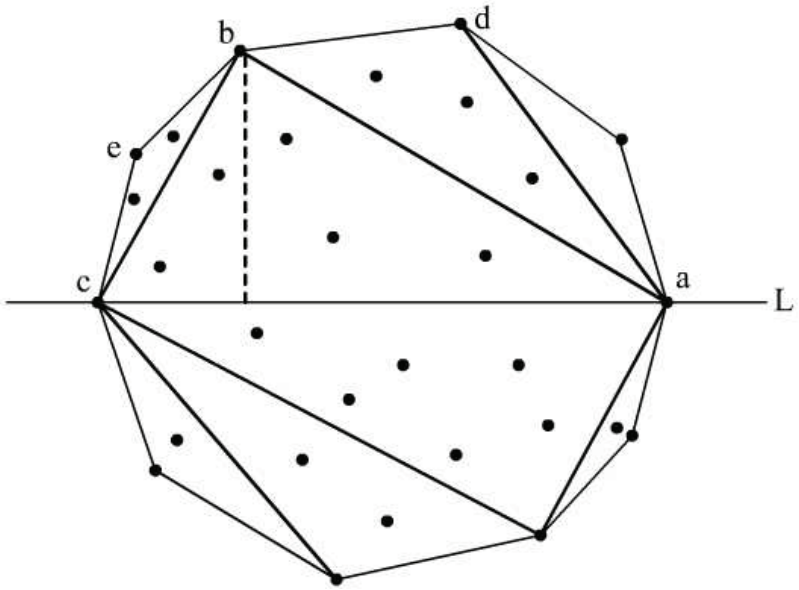

Figura 28: Algoritmo Quick Hull

El punto c (respec. a) es aquél cuya coordenada X es mínima (respec. máxima). En la figura,la recta L divide a S, en dos partes, una arriba y la otra hacia abajo; S_1 y S_2. El punto b es el punto de S_1 mas alejado de la recta L en la dirección perpendicular a L. El triángulo $\triangle abc$ es una primera aproximación de la envolvente convexa de S_1. Los puntos interiores los desechamos.

El punto d es el más alejado de la recta ba dentro de S_1.
El punto e es el m ás alejado de la recta bc dentro de S_1.

Luego la poligonal $adbec$ es otra aproximación a la envolvente convexa de S_1. Como se ve, después de aplicar recursivamnete este algoritmo un número finito de pasos llegamos a la envolvente convexa de S_1 y S_2. El paso final consiste en concatenar o unir las dos envolventes para obtener la envolvente de S. Podemos dar un pseudocódigo a este algoritmo.

Algoritmo: QUICK-HULL

- Dado: Un conjunto S de puntos del plano.
- Salida: La envolvente convexa de S.

Begin

1. Halle los puntos a y b, tal que la coordenada X de A es mínima entre los puntos de S y la coordenada X de b es maxima.
2. Sea S_1 el conjunto de puntos de S por encima de la recta L quer une a y b.
   Sea S_2 el conjunto de puntos de S situado por debajo de L.
3. Call Upper Hull (S_1, a, b)
   Call Lower Hull(S_2, a, b)
4. Retun
   (a)+Upper Hull Call Lower Hull(S_1, a, b)+(b)
   +Lower Hull Call Lower Hull(S_1, a, b)

END.

La subrutina Upper Hull (S_1, a, b) calcula la envolvente convexa de los puntos de S_1, por el método recursivo ya expuesto. Similarmente la subrutina Lower Hull calcula la envolvente convexa de los puntos de S_2.

Seguidamente damos el pseudocódigo para Upper Hull(S_1, a, b), el de Lower Hull(S_1, a, b) es similar.

Algoritmo Upper Hull (S, a, b)

- Entrada: un conjunto de puntos del plano S por encima de una línea L que une los puntos a y b
- Salida: La envolvente convexa de $S + \{a, b\}$

Begin

1. Halle el punto p más alejado de la línea que pasa por a y b
2. Sea S_1 el conjunto de puntos encima de la línea que pasa por a y p. Sea S_2 el conjunto de puntos de S por encima de la línea que pasa por p y b.
3. Aplicar en forma recursiva
   Upper Hull (S_1, a, p) y Upper Hull (S_2, p, b)
4. Concatenar los dos resultados obtenidos en la parte 3

END.

Podemos analizar la complejidad del algoritmo Quick Hull.
Los pasos 1 y 2 se pueden efectuar en $O(n)$ operaciones. Sin embargo, la subrutina Upper Hull puede tener una complejidad $T(n) = O(n \lg n)$ en el mejor de los casos y $T(n) = O(n^2)$ en el peor.

No obstante, este algoritmo es bastante rápido en la mayoría de los casos.

0.14. El método Divide y Vencerás

Divide y Vencerás es uno de los grandes paradigmas en la ciencia de la computación. Es una técnica de resolución de problemas que tiene diversas aplicaciones en el campo de la Geometría Computacional. El método consiste en partir un problema grande en problemas pequeños, resolverlos recursivamente y luego, la solución general se obtiene al empalmar las soluciones de los subproblemas.

Mediante este método se desarrollan algoritmos que son bastantes prácticos, intuitivos y eficientes. Supongamos que tenemos un problema de tamaño n (la medida n puede ser el número de puntos de un conjunto y el problema podría ser hallar la envolvente convexa de un conjunto de n puntos). Entonces aplicamos los pasos siguientes:

1. Dividimos el problema en K subproblemas del tamaño n/k.
2. Resolvemos cada uno de K subproblemas para obtener la solución general.
3. Combinamos las soluciones de los K subproblemas para obtener la solución general.

Si el tiempo de máquina para resolver el problema de tamaño S es $T(S)$, entonces tendremos en cuenta las siguientes observaciones para calcular la complejidad.

El tiempo de máquina para resolver el problema de tamaño 1 es constante: $T(1) = b$.
Los pasos 1 y 2 se pueden ejecutar en tiempo $C.n$ donde C es una constante.
Cada subproblema de tamaño n/K se resuelve en tiempo $T(n/K)$.
Entonces tenemos la siguiente relación de recurrencia.

$$\begin{aligned} T(1) &= b \\ T(n) &= KT(n/K) + C.n \end{aligned}$$

Supondremos ahora que $k = 2$ y que el entero n es una potencia de 2, esto es, $n = 2^s$, donde $s = log\ n$. Podemos entonces hallar un estimado del tiempo total en forma explícita. En el peor de los casos, dividimos por la mitad cada problema. Luego se tienen las ecuaciones.

$$\begin{aligned} T(n) &= 2T(n/2) + C.n \\ T(n/2) &= 2T(n/4) + C.(n/2) \\ &\vdots \\ T(n/2^S) &= 2T(n/2^{S-1}) + C.(n/2^S) \\ T(1) &= b \end{aligned}$$

Agrupando se obtiene $T(n) = 2^S b + C.n.S = n.b + C.n.\log n$.

Por lo tanto $T(n) = O(n.\log n)$.

0.15. Algoritmo Divide y Vencerás

De todos los algoritmos vistos hasta el momento, ninguno admite una generalización para calcular la envolvente convexa en tres dimensiones. El presente algoritmo fue diseñado con este fin por Preparata y Hong [7](1977). También ha sido llamado Kirkpatrik - Seidel.

La idea consiste en dividir el conjunto S de n puntos en dos subconjuntos: S_1 y S_2 cada uno de tamaño $[\frac{n}{2}]$, mediante una línea vertical, para garantizar que ambos sean disjuntos. Luego, se calcula la envolvente convexa de cada uno de ellos y finalmente, uniendo las envolventes mediante un proceso de de **concatenación**, se puede obtener la envolvente de S.

Si esto se hace en forma recursiva y en un tiempo lineal, entonces el paradigma Divide y Vencerás nos dice que el tiempo total del algoritmo será de O(nlogn).

Podemos hacer un esquema de este algoritmo:

ALGORITMO de Kirkpatrik-Seidel

- `ENTRADA: Un conjunto de S puntos en el plano`
- `SALIDA: La envolvente convexa de S.`

`BEGIN`

1. `Halle la mediana de las cordenadas x de los puntos de S.`
2. `Divida` S `en dos subconjuntos disjuntos` S_1 `y` S_2`.`
3. `Hallar recursivamente` $CH(S_1)$ `y` $CH(S_2)$`.`
4. `Concatenar` $CH(S_1)$ `y` $CH(S_2)$`.`

`END.`

El paso más importante es el 4. Daremos un algoritmo de concatenación para dos envolventes en forma eficiente con una complejidad de O(n). La clave en este proceso será construir una línea tangente a ambos conjuntos, por la parte de arriba y otra por debajo, que nos sirvan

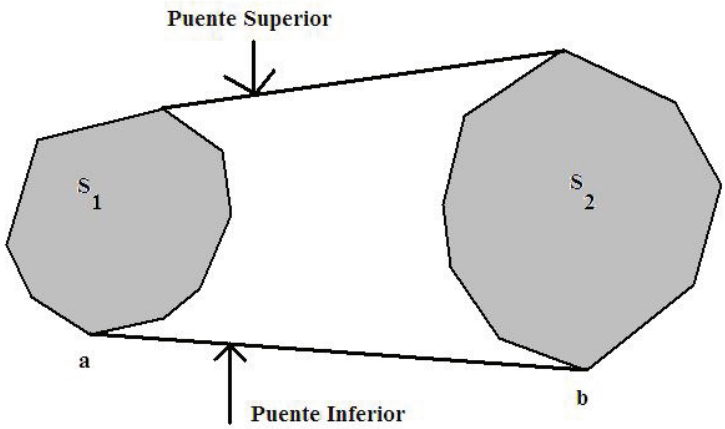

Figura 29: Puentes de empalme

de puente para empalmar. Estas líneas serán llamadas **puente superior** y **puente inferior** respectivamente. Daremos en detalle la construcción del puente inferior. El puente superior se construye en forma parecida.

Es claro que al determinar las líneas de puente podemos empalmar ambos conjuntos realizando el siguiente procedimiento:

1. Eliminando aquellos lados de S_1 que se encuentran entre los vértices de contacto (parte derecha).

2. Eliminando aquellos lados de S_2 que se encuentran entre los vértices de contacto (parte izquierda).

3. Incluyendo los puentes.

¿ Cómo se calculan los vértices extremos a y b?

Comenzamos con una primera aproximación del puente considerando la línea que une a x (el punto más a la derecha de S_1) con y (el punto más a la izquierda de S_2). La idea es ir bajando hasta llegar a la línea tangente a ambos polígonos.

Recordemos que los vértices de cada uno de los polígonos están ordenados por subíndices en sentido contrario a las agujas del reloj. A medida que vamos bajando, recorriendo los vértices del lado derecho de S_1, los índices disminuyen. Para los vértices del lado izquierdo de S_2, los índices van aumentando.

Antes que nada debemos establecer algunas condiciones necesarias y suficientes para que una línea L que una un par de vértices de contacto x_i y y_j nos de un puente inferior. Estas condiciones simultáneas son:

1. L es tangente en S_1, esto es ,x_{i+1} y x_{i-1} están por encima de L
2. L es tangente en S_2, esto es , y_{i+1} e y_{i-1} están por encima de L

Podemos dar ahora un método para bajar la línea L. Si L no es una tangente en S_2, entonces ir bajando el punto y hasta conseguir una tangente. Si x no es tangente en S_1, entonces bajar x hasta conseguir una tangente. Continuando de esta manera, al final se llega al resultado deseado(ver la figura).

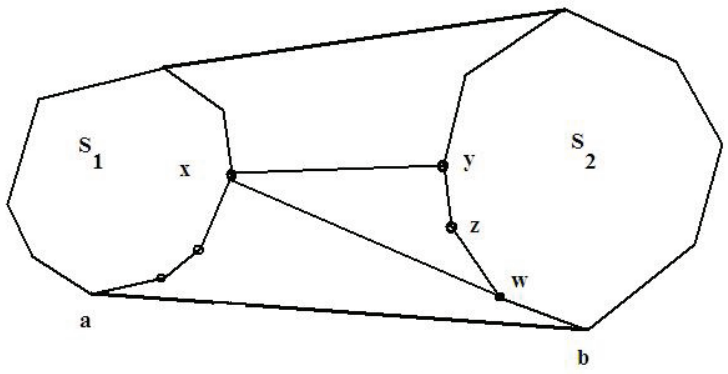

Figura 30: Algoritmo de línea puente inferior

Podemos dar ahora un pseudocódigo para este algoritmo.

ALGORITMO: LÍNEA PUENTE INFERIOR

- ENTRADA: Las envolventes convexas de S_1 y S_2
- SALIDA: La envolvente convexa de $S = S_1 \cup S_2$

BEGIN

1. Calcule $x_i = x$ el punto más a la derecha de S_1
2. Calcule $y_j = y$, el punto más a la izquierda de S_2.
3. While L = xy no sea una tangente doble para S_1 S_2 DO
4. While L no sea una tangente para S_1 DO
   $a \longleftarrow x_{i-1}$.
5. While L no sea una tangente para S_2 DO
   $b \longleftarrow y_{j+1}$

END

Ejercicios

1. Demuestre que un polígono que contenga un vértice reflex no es convexo.
2. En el algoritmo de envolvimiento de regalo aquí presentado se pueden presentar problemas cuando tres puntos son colineales. Haga las modificaciones correspondientes en el código, para solventar esta situación.
3. Un **Punto Extremo** de un conjunto de puntos S es un vértice de la envolvente convexa cuyo ángulo interno es menor que π. Demuestre que un punto de S no es extremo sí y sólo si se encuentra dentro de algún triángulo cuyos vértices son puntos de S y no es un vértice del triángulo.
4. Usando lo anterior, diseñe un algoritmo para hallar los puntos extremos de S.
5. Demuestre que el algoritmo Línea Puente inferior tiene una complejidad de O(n).
6. Demuestre que el diámetro de S es igual a la máxima distancia entre puntos antipodales del polígono P, frontera de la envolvente.

DIAGRAMA DE VORONOI

0.16. Problemas de aproximación

Un operador de una torre de control, o controlador aéreo, tiene varios puntos en la pantalla de la computadora representando cada uno de ellos a un avión en movimiento. En todo momento el operador debe saber cuáles son los dos aviones más cercanos entre sí para enviarles un mensaje de radio y así evitar una colisión en el aire. ¿Cuáles son los dos puntos más cercanos en la pantalla de un conjunto de n puntos?

Este problema se resuelve usando los métodos de la geometría computacional y pertenece a una clase de los llamados problemas de aproximación. Éstos se presentan cuando se requiere calcular la distancia euclideana entre puntos, líneas, polígonos y círculos. A continuación enunciamos algunos de ellos:

1) **Pares más cercanos.**Hallar los dos pares de puntos más cercanos en un conjunto S de n puntos.

2) **El vecino más cercano.** Para todo punto P en un conjunto S, hallar el punto S mas cercano.

3) **Árboles de longitud minimal** (Euclidean-Minimum-Spanning-Tree) conectar todos los puntos de un conjunto S, mediante un grafo de árbol de longitud mínima.

4) **Máximo círculo vacío.** Hallar el mayor círculo que no contenga puntos de S y cuyo centro sea interior a la envolvente convexa de S.

0.17. El Diagrama de Voronoi

Todos los problemas de proximidad enunciados en la sección anterior pueden ser resueltos usando una herramienta que contiene toda la información sobre la proximidad de los puntos. El Diagrama de Voronoi es uno de los conjuntos más importantes en geometría computacional.

Supongamos que tenemos un conjunto S formado por n puntos del plano. Para cada pareja de puntos del plano más cercano a p_i que a p_j en un semiplano determinado por el bisectriz del segmento $\overline{p_i p_j}$ y se denota por $H(p_i, p_j)$ (ver la figura).

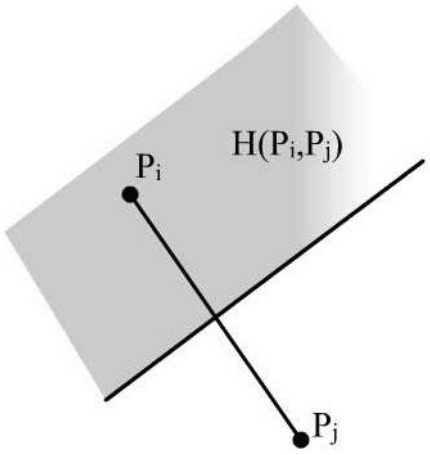

Figura 31: Hiperplano que contiene a p_i

Si para cada punto p_i en S denotamos por V_i el conjunto de puntos del plano mas cercano a p_i que a los restantes puntos de $S - \{p_i\}$, entonces se puede probar que:

$$V_i = \bigcap_{j \neq i} H(p_i, p_j)$$

Cada conjunto $H(p_i, p_j)$ es convexo. Entonces V_i es un conjunto convexo. Tambien se puede probar además que V_i es una reguión pogonal convexa y que la unión de todos ellos es el plano, esto es:

$$\Re^2 = \bigcup_{j=1}^{n} V_i$$

Esto motiva la siguiente:

Definición 0.17.1. *Un **Diagrama de Voronoi** para un conjunto $S = \{p_1, .., p_n\}$ de puntos del plano es una partición del plano en n regiones poligonales convexas $V_1, .., V_n$ tal que para cada i todos los puntos de la reguión V_i están mas cercanos a p_i que a cualquier otro punto de $S - \{p_i\}$*

Ejemplo 0.17.1. *En la figura siguiente mostramos un Diagrama de Voronoi para un conjunto S de 8 puntos.*

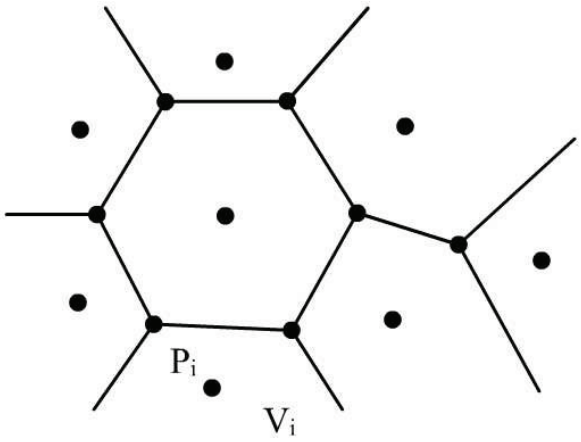

Figura 32: Diagrama de Voronoi

El polígono convexo V_i que contiene al punto p_i se llama **Polígono de Voronoi** del punto p_i. Los vértices del diagrama se llaman **Vértices de Voronoi** y los segmentos de recta del diagrama se llaman los **Lados de Voronoi**.

Una primera observación al diagrama de voronoi nos permite llegar a las siguientes conclusiones.

1. Si tenemos un punto p_i en S, entonces su vecino más próximo se halla en alguno de los polígonos de Voronoi adyacentes a V_i.

2. Si ordenamos en una lista cada punto P_i con su vecino más cercano, entonces podemos buscar en dicha lista el par de elementos de S mas cercanos.

Esto nos permite construir algoritmos para resolver los dos problemas de aproximación: El vecino más cercano y el par más cercano.

Más adelante estudiaremos algoritmos para construir el diagrama de Voronoi en tiempo $0(n\ log\ n)$. Entonces, los dos primeros problemas de aproximación se pueden resolver en tiempo $0(\log n)$.

0.18. Propiedades del diagrama de Voronoi

En esta sección damos una serie de propiedades sobre los vértices, caras y lados de los diagramas de Voronoi.

Lema 0.18.1. *Si los cuatro puntos de un conjunto S estan sobre un círculo C, entonces el centro del círculo es un vértice de Voronoi de grado 4.*

Demostración: En efecto sean p_1, p_2, p_3 yp_4 los puntos de S circulares. Entonces cada segmento de recta bisectora de $\overline{p_i p_j}$ pasa por el centro del círculo C. Luego el punto c es un vértice de Voronoi, pues allí se produce una intersección de dos segmentos bisectores (ver la figura).

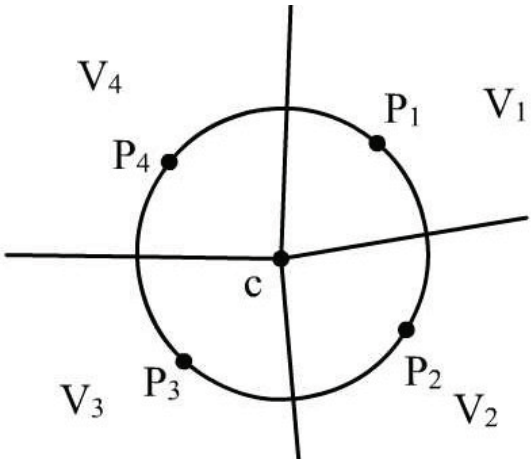

Figura 33: cuatro puntos cocirculares

Nótese que c es el único nodo en $V_{or}(S)$ y es de orden 4, pues allí concurren las 4 rectas bisectoras.

De ahora en adelante supondremos que en S no hay 4 puntos cocirculares.

Lema 0.18.2. *Todo vértice de Voronoi de S tiene grado exactamente tres.*

Demostracíon: Sea $S = \{p_1, p_2, ..., p_n\}$ y v un vértice de Voronoi. Cada vértice de $Vor(S)$ se obtiene como una intersecci ón de los lados de los polígonos de Voronoi.

Supongamos que los lados de Voronoi $\{l_1, ..., l_k\}$ convergen en v, y además l_i divide los polígonos de Voronoi V_i y V_{i+1}, Y l_i divide a V_1 y V_k (ver la figura).

Entonces v es equidistante de los puntos p_1 y p_2, puesto que v está sobre l_2, la línea que divide las regiones V_1 y V_2. De igual manera, v es equidistante de $p_3, .., p_k$. Luego los puntos $p_1, .., p_k$ están sobre un círculo y por lo tanto $K \leq 3$, pues no hay 4 puntos cocirculares en S.

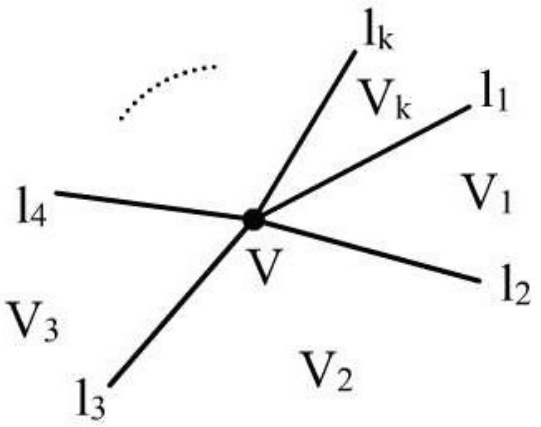

Figura 34: Nodo de grado tres

Si $k = 2$ entonces l_1 y l_2 dividen a los polígonos V_1 y V_2 y por lo tanto ambos lados pertenecen al segmento bisector de $\overline{p_1p_2}$. Luego v no es vértice de Voronoi.

Si $k = 1$, entonces hay una sola semirrecta que divide la región V_1 y por la tanto la región V_1 no es convexa, lo cual es una contradicción.

Luego $k = 3$ y todo nodo de $Vor(S)$ tiene grado exactamente tres.

Teorema 0.18.1. ***Fórmula de Euler para Grafos*** *Sea G un grafo planar. Entonces se cumple la relación.*

$$v - e + f = 2 \tag{6}$$

donde v es el número de vértices, e es el número de lados y f es número de caras.

Para una demostración de este resultado, ver [12] p. 564.

Lema 0.18.3. *El diagrama de Voronoi satisface las relaciones:*

1. $v \leq \frac{2}{3}e$
2. $e \leq 3f - 6$
3. $f \leq \frac{2}{3}e$
4. $v \leq 2f - 4$.

Demostración Podemos suponer que todos los lados infinitos del diagrama de Voronoi se conectan con un punto al infinito. Luego la fórmula de Euler vale bajo esta consideración. Sabemos que cada vértice del grafo es de orden 3. Para el vértice al infinito el grado es mayor o igual a 3. Luego el número de lados satisface 1.

Una prueba bastante intuitiva de este hecho es la siguiente: Tome una pareja cualquiera de vértices. Desconéctelos de los vértices ajenos y rehaga las conexiones de tal forma que se conecten entre ellos madiante tres o más lados y queden aislados del resto del grafo. Por cada dos vértices vemos que hay tres o más lados. Luego 1 es cierto.

Los restantes 2-4 salen de combinar la primera desigualdad con la fórmula de Euler.

Sea v un vértice de Voronoi. Entonces, por el lema anterior hay tres polígonos de Voronoi que concurren en v. Sean p_1, p_2 y p_3 los puntos de S que definen estos tres polígonos. Entonces el punto v equidista de estos tres puntos.

Teorema 0.18.2. *El **Círculo de Voronoi tangente en v**, denotado por C(v), es el círculo con centro en v y tangente a cada uno de los puntos p_1, p_2 y p_3 .*

Dicho círculo será de mucha importancia en el desarrollo de esta teoría.

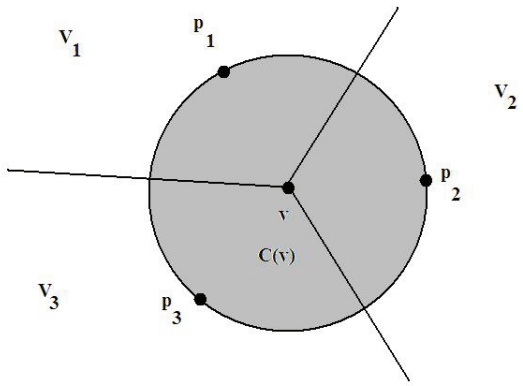

Figura 35: Círculo de Voronoi

Lema 0.18.4. *Sea v un vértice de Voronoi. Entonces el círculo C(v) no contiene puntos de S en su interior.*

Demostración Sean V_1, V_2 y V_3 los polígonos que coinciden en v. Éstos vienen determinados por los puntos de S, p_1, p_2 y p_3. Si q es otro punto de S dentro de C(v), entonces v estaría

más cerca de q que de p_1, y por lo tanto, el polígono de Voronoi $V(q)$ debe incidir en v, lo cual es imposible. Luego no hay puntos de S dentro de C(v).

Lema 0.18.5. *Sea p_i un punto en S, y p_j el punto en S más cercano. Entonces las correspondientes regiones de voronoi V_i y V_j comparten un lado en común.*

Demostración: Sea v el punto medio del segmento $\overline{p_i p_j}$. Sea C el círculo de radio $r = \frac{p_i p_j}{2}$ y con centro en p_i.
Entonces este círculo C debe estar contenido en el polígono de Voronoi V_i, que contiene a p_i. En efecto, si el círculo se sale de V_i, es cortado por algún lado de Voronoi e (ver la figura).

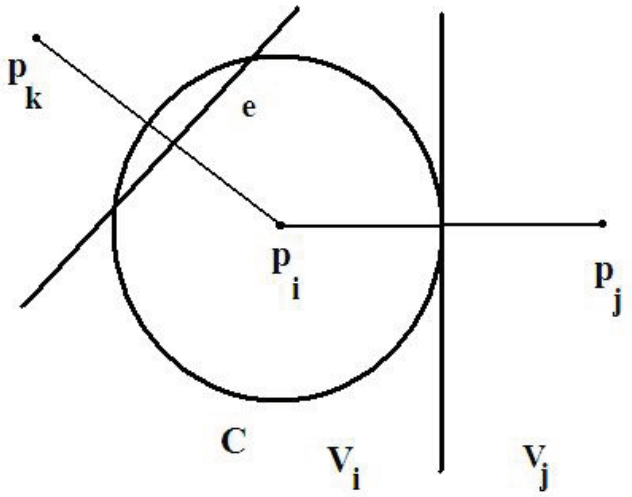

Figura 36: El vecino más cercano

Luego existe un punto u de e, contenido en el interior de C. Por otro lado e es parte de la recta bisectora del segmento $\overline{p_k p_j}$, donde p_k es un punto de S, y $p_k \neq p_j$.

Entonces p_k está mas cercano a p_i que p_j, pues:

$$\begin{aligned} d(p_i, p_k) &\leq d(p_i, u) + d(u, p_k) \\ &= 2d(p_i, u) < 2d(p_i, v) = d(p_i, p_j) \end{aligned}$$

Esto es una contradicción y, por lo tanto, $C \subseteq V_i$.

Como el punto v está a igual distancia de p_i y p_j, v pertenece a la frontera de V_i. Es decir, v pertenece a un lado de Voronoi.

Probaremos que v es punto interior de un lado. Si suponemos que v está en la frontera de un lado, entonces es un vértice de Voronoi. Sean e_1 y e_2 dos lados en la frontera de V_i, adyacentes a v. Como V_i es convexo, el ángulo entre e_1 y e_2 debe ser menor de 180^o y, por lo tanto, alguno de los dos lados intersecta a C. Esto es imposible por lo visto anteriormente. Luego v está en el interior de algún lado de Voronoi que divide los polígonos V_i y V_j. Con esto termina la demostración.

0.19. Diagrama de Voronoi y la Envolvente Convexa

Veamos ahora la relación que existe entre el diagrama de Voronoi y la envolvente convexa de un conjunto S finito.

Sean p_1 y p_2 dos puntos consecutivos de la envolvente convexa de S. Entonces, si L es una recta que contiene al segmento p_1p_2, todos los puntos de S estarán a un lado de S. Sea L' la línea perpendicular a L y que pasa por el punto medio entre p_1 y p_2. Es claro que L' es la mediatriz del segmento p_1p_2. Luego L' contiene parte de un lado de Voronoi e de $V(S)$, que separa los polígonos V_1 y V_2 (ver la figura).

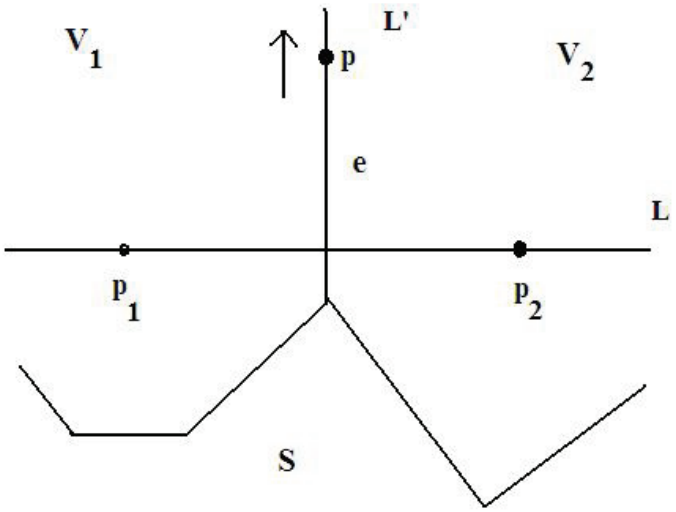

Figura 37: El lado e es infinito

Si p es un punto que se mueve arbitrariamente sobre L', se tendrá siempre $d(p_1, p) = d(p_2, p)$, no importando qué tan lejos se separe el punto$p$ del conjunto S. Es más, podemos escojer p_0 suficientemente alto, de tal manera que el círculo de radio $r = d(p_0, p_1)$ no contenga puntos de S en su interior. Entonces el rayo infinito que corre desde p_0 hacia arriba, dentro de la recta L', estará contenida dentro del lado de Voronoi que divide los polígonos V_1 y V_2.

Reciprocamente, sean p_1 y p_2 dos puntos de S tales que el lado de Voronoi e, que divide los polígonos V_1 y V_2 es infinito. Entonces, estos polígonos no son acotados.

En particular, como V_1 no es acotado, para todo $r > 0$, existe un punto p, dentro de V_1, tal que $d(p, p_1) > r$ y p está más cerca de p_1 que cualquier otro punto de S (Ver la figura).

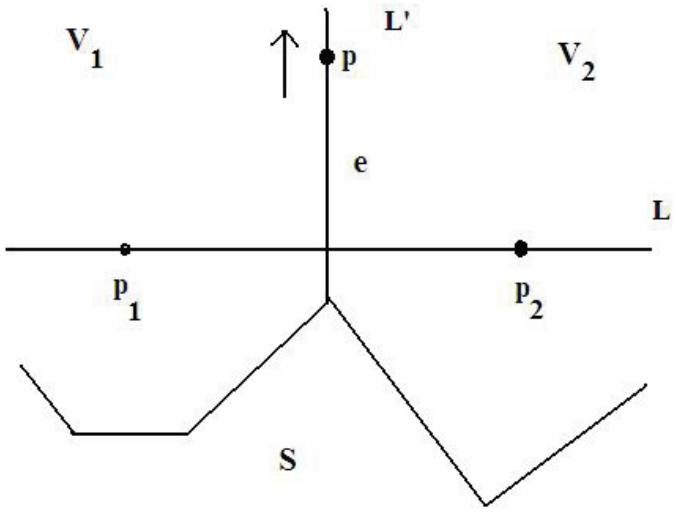

Figura 38: Punto de S más cerca de p

Entonces, para r muy grande el arco del círculo C es casi una recta. Se tendrá pues que ningún punto de S está del lado de L donde se encuentra p, lo cual implica que p es un punto de la envolvente convexa.

Hemos demostrado entonces el siguiente resultado

Lema 0.19.1. *Dos puntos p_1 y p_2 son vértices consecutivos de la envolvente de S sí, y sólo si, sus correspondientes polígonos de Voronoi V_1 y V_2 están separados por un lado infinito.*

0.20. Construcción del Diagrama de Voronoi

En esta sección presentamos un algoritmo bastante eficiente para construir el diagrama de Voronoi de un conjunto S del plano con n puntos. Usaremos el método de Divide y Vencerás.

El método aquí empleado nos da un algoritmo con una complejidad de O(nlogn) y fue expuesto por primera vez en 1975 por Shamos y Hoey [4]. La idea del algoritmo consiste en dividir al conjunto S en dos partes S_1 y S_2 mediante una línea vertical L de tal forma que ambos conjuntos tengan casi el mismo número de puntos (si n es par se tiene exactamente la mitad). Supondremos en toda la exposición que S_1 está a la izquierda de L y S_2 está a la derecha de L.

Supongamos que hemos construido recursivamente los respectivos diagramas de Voronoi tanto para S_1 como para S_2, los cuales serán denotados por $V(S_1)$ y $V(S_2)$. La gran pregunta es: ¿Cómo hacemos para concatenarlos de alguna manera para obtener el diagrama de Voronoi del conjunto S?

0.21. Un proceso de concatenación

El proceso de concatenación o fusión es algo complicado y ocupará la mayor parte de esta exposición. Cuando se unen los dos diagramas hay que eliminar algunos segmentos y también se deben reponer partes. De allí pues surgen las dificultades. Sin embargo, daremos unos procedimientos claros sobre como solventarlas, basados en las propiedades geométricas de los grafos.

Veamos el proceso de concatenación en sentido inverso. Es decir, partimos del diagrama de Voronoi de S y luego a S lo desmembramos en dos mitades $S1$ y S_2. ¿Qué partes de $V(S)$ desaparecen?

Sea e un lado del diagrama de Voronoi de S definido por los puntos p_i y p_j. Esto es, e se encuentra sobre la recta mediatriz determinada por los dos puntos y además divide las regiones de Voronoi V_i y V_j (ver el dibujo).

Nos interesa saber qué puede ocurrir con este lado al desmembrar los dos diagramas.

La respuesta depende de dónde se encuentren los puntos al momento de dividir a S. Entonces tenemos

1. Si p_1 y p_2 están en S_1, entonces $e \in V_1$ y no desaparece.

2. Si p_1 y p_2 están en S_2, entonces $e \in V_2$ y no desaparece.

3. Si p_1 está en un conjunto y p_2 está en el otro entonces e desaparece.

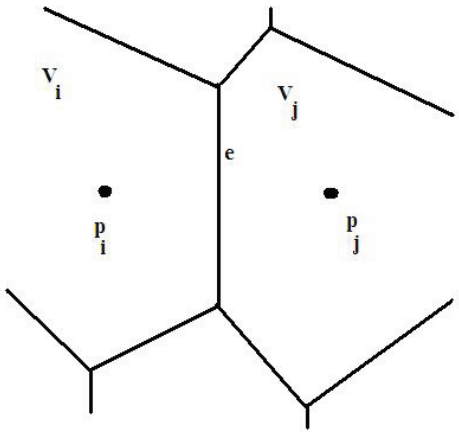

Figura 39: El lado e divide dos polígonos de Voronoi

El lado e desaparece, cuando los puntos p_1 y p_2 están en lados distintos de la línea divisoria L, pues al separar los conjuntos S_1 y S_2, entonces los puntos dejan de ser vecinos. Entonces, al reunir los dos diagramas nuevamente habrá que incluir a e como parte de una nueva frontera.

Veremos ahora como se construye **El grafo frontera.**

Denotaremos por σ el subgrafo de Voronoi de S tal que los lados de σ están determinados por parejas de puntos p_i y p_j tales que $p_i \in S_1$ y $p_j \in S_2$. Este σ será llamado el **Grafo frontera**. Seguidamente daremos una serie de resultados teóricos sobre este grafo.

Lema 0.21.1. *Cada vértice de σ tiene grado exactamente igual a dos.*

Demostración Sea v un vértice de σ. Como cada σ es un subgrafo de $V(S)$, entonces v es también un vértice de Voronoi y por lo tanto tiene grado menor o igual a 3.

Si grado de v = 3, entonces existen lados e_1, e_2 y e_3 de σ incidentes en v. Dichos lados delimitan polígonos V_1, V_2 y V_3. Podemos suponer que e_1 está en la frontera entre V_1 y V_2, e_2 está en la frontera entre V_3 y V_2 y e_3 está en la frontera entre V_1 y V_3 (ver la figura).

Supóngase que $p_1 \in S_1$ y $p_2 \in S_2$. Entonces como $e_1 \in \sigma$ se tiene que $p_3 \in S_2$. Y también como $e_2 \in \sigma$ se tiene que $p_3 \in S_1$. Esto es una contradicción.

Si el vértice tiene grado 1, entonces e_1 es el único lado de Voronoi incidente con v. Pero e_1 es un lado que separa dos regiones de Voronoi V_1 y V_2, definidas por los puntos p_1 y p_2.

Supongamos que $p_1 \in S_1$ y $p_2 \in S_2$. Entonces el punto p_3 en el dibujo no pertenece a ninguno de los conjuntos S_1 o S_2. En efecto, si asumimos que $p_3 \in S_2$, entonces $e_3 \in \sigma$ y e_3 es adyacente a v, lo cual es imposible. Si, por otra parte, suponemos que $p_3 \in S_1$, llegamos a la misma contradicción. Luego grado (v) = 2.

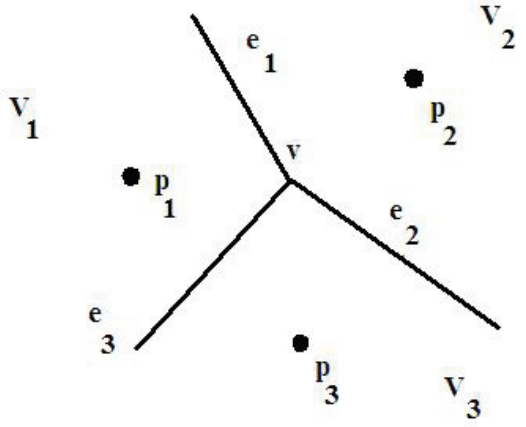

Figura 40: Un vértice de grado 3

Hasta el momento conocemos muy poco acerca del grafo frontera σ. Este puede tener varias componentes conexas. Algunas podrán ser ciclos y otras serán cadenas. Sin embargo, si σ tiene alguna componente cíclica, digamos σ_1, entonces σ_1 encierra algún punto p de S y esto es imposible. Luego no puede haber ciclos entre las componentes de σ.
Recordemos que una cadena es un tipo de grafo G donde todos sus nodos son de grado dos y G, no es cíclico.

Teorema 0.21.1. *Una grafo G se llama* ***Cadena monótona*** *si G es una cadena, y la intersección de G con cualquier recta horizontal contiene exactamente un punto.*

De acuerdo a la definición, toda cadena monótona con un número finito de vértices posee dos lados que son semirectas infinitas, uno en la parte de arriba y otro abajo.

Lema 0.21.2. *Toda componente convexa de σ es una cadena monótona*

Demostración En primer lugar notemos que en σ no hay lados horizontales. Si e es un lado horizontalde σ, entonces existen puntos $p_1 \in S_1$ y $p_2 \in S_2$ tales que la mediatriz del segmento p_1p_2 contiene a e. Luego p_1 y p_2 están sobre una misma recta vertical. Esto es imposible, pues los conjuntos S_1 y S_2 están separados por una recta vertical.

Supongamos que tenemos una cadena $C \in \sigma$, la cual no es monótona. Luego existe una recta horizontal L y tres vértices v_1, v_2 y v_3 de σ tal que v_1 y v_3 están por encima de L y v_2 está por debajo (ver la figura).

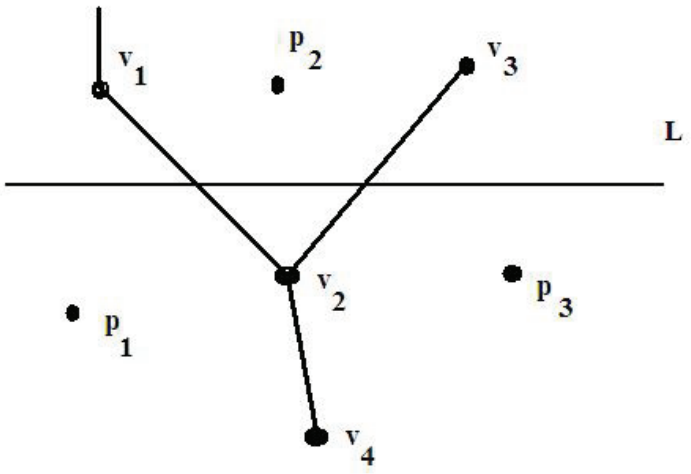

Figura 41: Una cadena de σ

Como v_2 es un vértice de Voronoi, su grado es tres y por lo tanto existe un cuarto vértice v_4 adyacente a v_2. Claramente, v_4 está por debajo de L, pues los polígonos de Voronoi son convexos.

Tenemos entonces tres lados de Voronoi v_1v_2, v_2v_4 y v_2v_3. Asociados a estos tres lados hay tres puntos en S, p_1, p_2 y p_3. Supóngase que $p_2 \in S_2$. Entonces s debe tener que p_1, $p_3 \in S_1$. Pero como S_1 y S_2 están divididos por una línea vertical L', ésta debe pasar entre p_1 y p_2. Luego p_3 está la derecha de L' y por lo tanto $p_3 \in S_2$. Esto es una contradicción. Luego la cadena C debe ser monótona .

Lema 0.21.3. *El grafo frontera σ tiene una sola componente conexa.*

Demostración En primer lugar, σ debe tener al menos una componente conexa, pues el diagrama de Voronoi es conexo y al menos existe un lado e que divide dos regiones de Voronoi; una en S_1 y la otra en S_2 y por lo tanto $e \in \sigma$.

Supongamos que σ tenga más de una componente conexa. Por los dos lemas enteriores ellas son cadenas monótonas, de longitud ifinita, que no se intersectan. Cada cadena divide al plano en dos regiones disjuntas: una a la derecha y otra a la izquierda. Supóngase que C_1 y C_2 son dos cadenas de σ que están una al lado de la otra, es decir, no hay otra cadena entre ellas. Sin pérdida de generalidad podemos asumir que C_1 está a la izquierda de C_2 (ver la figura).

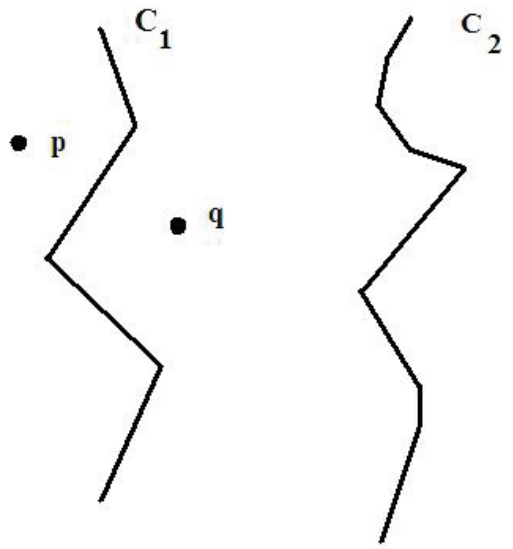

Figura 42: Dos cadenas de σ consecutivas

Entonces todos los polígonos de Voronoi que están a la izquierda de C_1 pertenecen a V_1 y los polígonos que están a la derecha pertenecen a V_2. Sea p un punto intermedio entre ambas cadenas. Entonces, por estar a la derecha de C_1 se tiene que $p \in S_2$. Por otro lado, el punto p se halla a la izquierda de la cadena C_2 y por lo tanto $p \in S_1$. Esto es una contradicción. Por lo tanto no hay puntos intermedios entre ambas cadenas. Esto implica que el diagrama de Voronoi de S no es conexo, lo cual es falso. Por lo tanto no hay más de una cadena en σ.

0.22. Algoritmo Divide y Vencerás

Estamos ahora en condiciones de dar un algoritmo para construir la cadena de frontera σ que concatenará los dos diagramas de Voronoi V_1 y V_2. Sabemos que σ posee dos lados que son semirrectas infinitas: Uno en la parte alta, denotado por L_u, y otro en la parte baja, denotado por L_d (ver la figura).

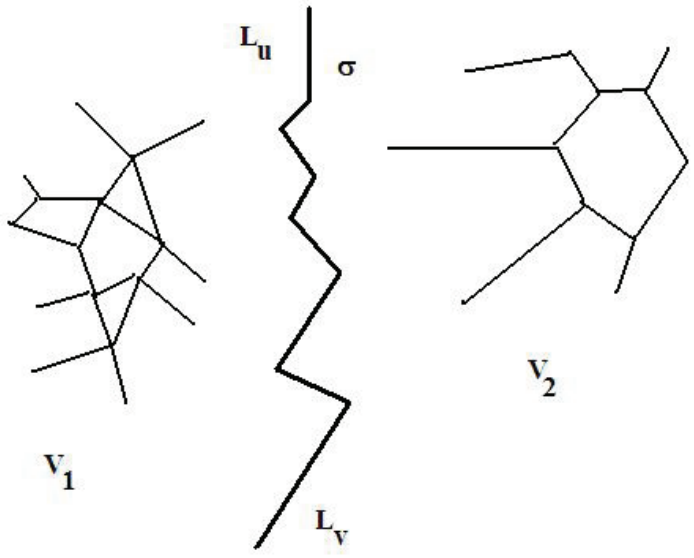

Figura 43: Construyendo la frontera σ

En primer lugar calculamos las envolventes convexas de S_1 y S_2. Luego hallamos las dos líneas de puentes para empalmar ambas envolvente (véase el algoritmo de concatenación para construir la envolvente en el capítulo 3). La línea del puente superior contiene un punto p en S_1 y un punto q en S_2 que son los puntos de contacto. La semirrecta L_u está contenida en la mediatriz que une los puntos p y q.

Comenzamos a recorrer la cadena σ desde la parte de arriba bajando por la semirrecta L_u hasta que se consiga con el primer lado de Voronoi de alguno de los dos diagramas. Supongamos que esta línea se corte con un lado e perteneciente a V_2. Entonces este lado divide dos regiones de Voronoi V_2 y V_3, determinadas por los puntos q y q'.

Entonces debemos cambiar de dirección en el punto de corte. La nueva dirección viene dada por la mediatriz que une el segmento pq'. Tenemos entonces que hacer un giro a la derecha (ver la figura). Continuamos bajando por esta mediatriz hasta que cortemos otro lado de Voronoi de alguno de los diagramas.

Si la semirrecta L_u hubiese cortado un lado de Voronoi de S_1 desde el inicio del recorrido entonces al entrar en un nuevo polígono de Voronoi de S_1, determinado por un punto p', cambiamos de dirección siguiendo la mediatriz del segmento $p'q$. En este caso damos un giro hacia la izquierda.

De esta manera continuamos bajando, siguiendo la trayectoria de las mediatrices de los puntos de S_1 y S_2, cuyos polígonos de Voronoi son finitos. Esta regla se enuncia así:

REGLA DE BAJADA. "Si Ud. viene bajando por una mediatriz cualquiera que une los puntos a y b , con $a \in S_1$ y $b \in S_2$ y se consigue con un lado de Voronoi y entra a una nueva región determinada por el punto c, entonces haga un cambio de dirección siguiendo la mediatriz del segmento ac si $c \in S_2$ o cb, si $c \in S_1$".

Al final nos conseguiremos con la línea infinita L_d y aquí termina el proceso.

Algoritmo concatenamiento

- ENTRADA: Conjuntos de Voronoi S_1 y S_2 de tamaño $\frac{n}{2}$ cada uno.
- SALIDA: La línea de frontera que los divide.

BEGIN

1. Construya la envolvente convexa de S_1 y S_2
2. Construya los puentes de unión entre ambas envolventes, usando los algoritmos PUENTE SUPERIOR y PUENTE INFERIOR
3. Construya σ siguiendo las reglas establecidas*. Hallar las mediatrices L_u del puente superior y L_d del puente inferior.
4. Comience en un punto suficientemente alto de L_u y comience a bajar disminuyendo la coordenada y de los puntos.
5. Continúe bajando, siguiendo la trayectoria de las mediatrices determinadas por los puntos a la derecha de la envolvente deS_1 y los puntos a la izquierda de la envolvente de S_2
6. Detenerse cuando se intersecte la semirrecta L_d.
7. Elimine aquellas partes de los vértices de Voronoi que intersectan a σ.
8. Hacer $Vor(S) = Vor(S_1) \cup Vor(S_2) \cup \sigma$.

END.

Finalmente enunciamos el algoritmo de Vornoi.

Observación. Se puede demostrar que la complejidad de este algoritmo es O(nlogn).

Algoritmo Voronoi

- ENTRADA: Un conjunto S de n puntos del plano.
- SALIDA: El diagrama de Voronoi de S.

BEGIN

1. Usando el algoritmo de las medianas, divida el conjunto S en dos partes iguales S_1 y S_2 mediante una línea vertical
2. Recursivamente aplique el algoritmo de concatenamiento a S_1 y S_2.

END.

0.23. Aplicaciones

Veamos ahora algunas de las aplicaciones del diagrama de Voronoi en la resolución de problemas de proximidad mencionados al comienzo de este capítulo.
1) **El vecino más cercano**. Supongamos que alguien está en una ciudad y quiere dirigirse hacia un centro de comunicaciones. La persona desea saber entonces cuál es el más cercano a su posición. Este problema se conoce en la literatura como el del vecino más cercano y se puede plantear matemáticamente de la siguiente forma: Supongamos que tenemos un conjunto S de n puntos del plano $\{x_1, x_2, ...x_n\}$. Sea p un punto cualquiera.

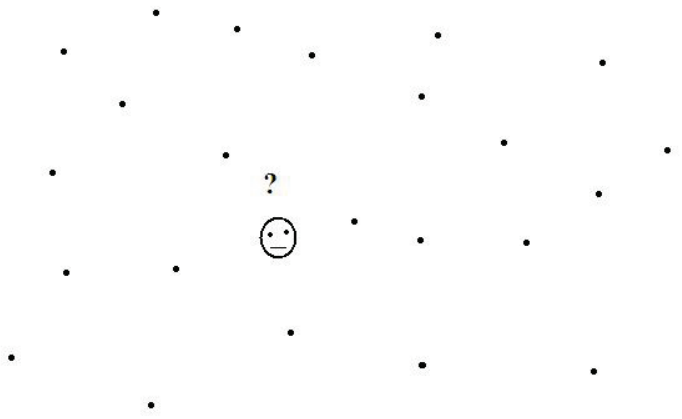

Figura 44: El vecino más cercano

Entonces podemos preguntarnos: ¿Cuál es el punto de S más cercano a p?

Se puede construir un algoritmo de fuerza bruta que nos de la solución en un tiempo O(n), calculando las n distancas a cada uno de los puntos y luego tomando la distancia mínima. Si se conoce el diagrama de Voronoi de los n puntos, entonces se puede optimizar la solución mediante un proceso de búsqueda en un tiempo considerablemente más bajo de O(log). Sabemos que el diagrama de Voronoi divide al plano en polígonos. Entonces, para determinar el punto de S más cercano a p basta con determinar en que polígono de Voronoi se encuentra. Esto se lleva a cabo en un tiempo de O(logn) usando un árbol de búsqueda binario.

2) **Los pares más cercanos.** Supongamos que un controlador aéreo tiene en la pantalla de su computador las posiciones de todos los aviones cercanos a su torre de control. Para hacer su trabajo debe saber en todo momento cuáles son los dos aviones más cercanos para evitar una posible colisión entre los mismos.

Como siempre, sea S el conjunto de los puntos del plano que representan las posiciones de los aviones. Se desea buscar un algoritmo eficiente y rápido que nos de los pares más cercanos en S, a medida que S va cambiando en el tiempo. Un algoritmo de fuerza bruta puede resolver este problema en tiempo O(n^2). Sin embargo se puede mejorar el tiempo de cálculo mediante un algoritmo que use el diagrama de Voronoi.

Sabemos que los pares más cercanos están separados por un lado de Voronoi. Si p y q es el par más cercano, ellos están en regiones de Voronoi vecinas y la mediatriz del segmento que une a ambos contiene un lado de Voronoi. Luego el problema se reduce a buscar, entre todos los lados de Voronoi, aquel que separe los puntos más cercanos. Esto tiene una complejidad de O(nlogn).

3) **Problema del Máximo Círculo Vacío.** Este es un problema de localización de servicios. Supóngase que hay n servicios comerciales en una ciudad, de una cierta clase, por ejemplo, farmacias, Panaderías, abastos,...etc, y queremos hallar dentro del polígono de los límites de la ciudad el lugar más alejado a los servicios ya existentes. Se quiere ubicar una nueva tienda en el lugar más alejado de los posibles competidores.

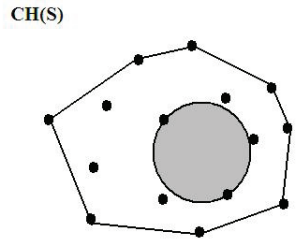

Figura 45: Máximo Círculo Vacío

Podemos hacer un modelo matemático de este problema considerando a cada servicio como un punto en el plano. Sea $S = \{x_1, x_2, ...x_n\}$ el conjunto de puntos en cuestión. Se quiere entonces hallar un punto x sobre la envolvente convexa CH(S) tal que maximice la función :

$$f(x) = min\ d(x, x_i) \tag{7}$$

Es decir, hallar el centro del máximo círculo vacío que se pueda formar dentro de CH(S)(ver la figura).

Es fácil ver que el máximo círculo vacío es del tipo C(v), siendo v un vértce de Voronoi o bien un círculo centrado en la intersección de uno de los ejes de Voronoi con la envolvente convexa. Luego este problema se puede resolver en tiempo O(nlogn). Los detalles se pueden ver en [3], pág. 256.

0.24. Triangulación de Delauny

Sea S un conjunto de puntos del plano. Una triangulación de S es una región dividida en triángulos, tal que los vértices de cada uno de los triángulos son puntos de S y la frontera de la región es la envolvente convexa de S.

En el capítulo 2 vimos cómo la triangulación de un polígono nos permitió resolver el problema de la vigilancia en las galerías de arte. Las triangulaciones también son importantes en el diseño de carrocerías de autos.

En Ingeniería, las triangulaciones se usan para analizar las formas de objetos complejos y estudiar mejor su estructura bajo una técnica conocida como *Análisis de Elementos Finitos*. El objeto grande, sometido a una fuerza, se divide mediante una malla o partición en pequeños elementos, que pueden ser tetraedros, cubos,...etc. Sobre estos elementos se plantean más facilmente las ecuaciones diferenciales, con lo cual se puede conocer mejor la dinámica de todo el objeto.

Una triangulación se puede representar como un grafo, cuyos nodos son los puntos de S. Asociado a S también tenemos otro grafo: el diagrama de Voronoi. Podemos entonces preguntar: ¿Qué relación existe entre ambos grafos?

El resultado siguiente fue obtenido por Delauny en 1934.

Teorema 0.24.1. *Sea S un conjunto finito de puntos del plano y Vor(S) su diagrama de Voronoi. Entonces el grafo dual de Vor(S), el cual se obtiene tomando los lados como segmentos rectos que conectan los vértices de S, es una triangulación de S.*

Demostración La triangulación así obtenida se denomina *Triangulación de Delauny* (ver la figura).

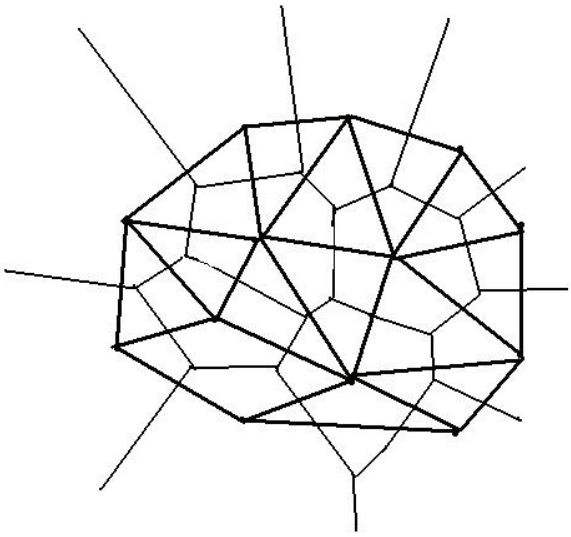

Figura 46: Triangulación de Delauny y diagrama de Voronoi

Tenemos antes que nada las siguientes observaciones

1. Cada vértice x de la triangulación de Delauny, denotada de ahora en adelante por D, se corresponde con la región de Voronoi V_x.

2. Cada lado de D es un segmento recto con extremos p_1 y p_2, que corta perpendicularmente al lado de Voronoi que conecta las regiones donde se hallan los puntos dados.

3. Cada cara de D es un triángulo que contiene un vértice de Voronoi.

El proceso de triangulación ocurre de la siguiente manera. Si e_1, e_2 y e_3 son tres lados de Voronoi que convergen en un vértice v, entonces hay tres regiones de Voronoi V_1, V_2 y V_3 tales que: e_1 separa a V_1 de V_2, e_2 separa a V_2 deV_3 y e_3 separa a V_3 de V_1.
Cada región V_i contiene un único punto de S, denotado por p_i. Luego los duales de los lados

e_1, e_2 y e_3 son repectivamente: p_1p_2, p_2p_3 y p_3p_1.

El dual del vértice v es el triángulo $d(v) = \Delta(p_1, p_2, p_3)$. Para demostrar que D es realmente una triangulación, necesitamos un par de resultados.

Lema 0.24.1. *Si v_1 y v_2 son dos vértices de Voronoi, entonces los triángulos $d(v_1)$ y $d(v_2)$ no se solapan en su interior. Esto es*

$$int(d(v_1) \cap d(v_2)) = \phi \tag{8}$$

Demostración Consideremos los círculos $C(v_1)$ y $C(v_2)$, como en la definición 11 . Si $C(v_1) \cap C(v_2) = \phi$, entonces $d(v_1) \cap d(v_2) = \phi$.

Supongamos que hay puntos comunes en el interior de$C(v_1)$ y $C(v_2)$. Es claro que ninguno de los círculos puede contener al otro, pues dentro de cada círculo no hay puntos de S.

Luego los círculos se cortan en un par de puntos q_1 y q_2 (ver la figura).

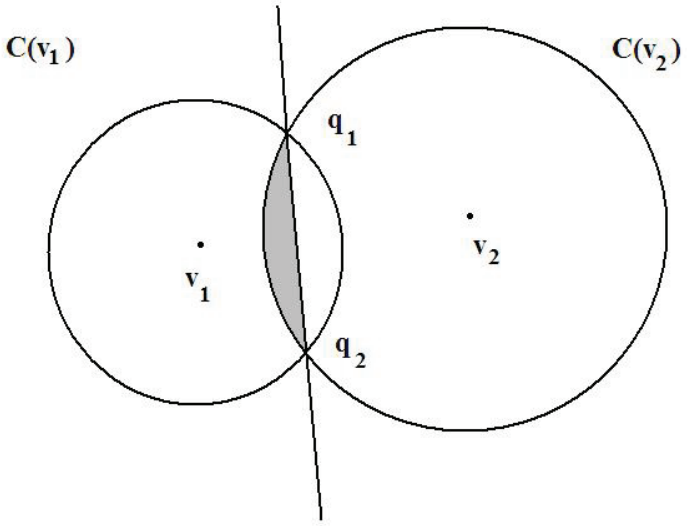

Figura 47: Intersección de círculos

El segmento L separa los puntos v_1 y v_2. Afirmamos que L también separa los triángulos $d(v_1)$ y $d(v_2)$. En efecto, si hay un punto de $d(v_2)$ en el área sombreada, este debe ser un vértice del mismo. Esto contradice el lema 8, pues no puede haber puntos de S dentro de los círculos, a excepción del centro. Con esto termina la demostración.

Lema 0.24.2. *Los triángulos de Delauny llenan todo el interior de CH(S).*

Demostración. Sea x un punto de la envolvente convexa de S y supóngase que x no esté contenido en ningún triángulo de D. Entonces existe un círculo de centro en x y radio δ, contenido en el complemento de D. Denotmos este círculo por C. En el lenguaje de la topología del plano esto es equivalente a afirmar que D^c es abierto, lo cual es cierto pues el conjunto D es un cerrado.

Sea v el vértice de Voronoi más cercano a C (ver la figura.)

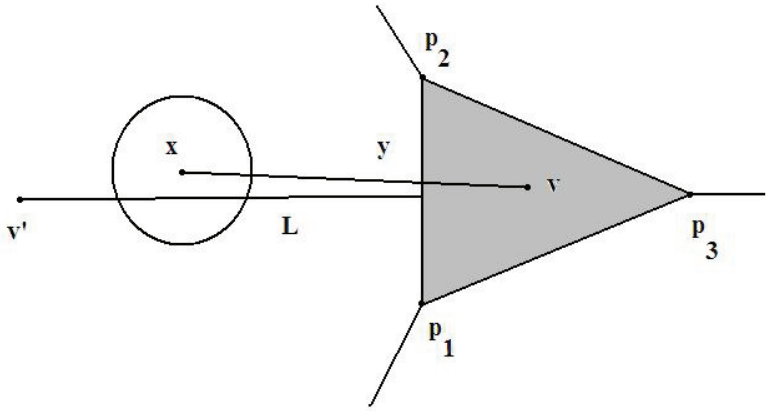

Figura 48: Un punto x en el complemento de D

Sea y el punto del segmento xv que se encuentra en D, más cercano a x. Entonces y está en la frontera de D y por lo tanto sobre un lado del triángulo $d(v) = \Delta(p_1, p_2, p_3)$. Si asumimos que x está sobre el lado p_1p_2, entonces la mediatriz L contiene un lado de Voronoi, el cual no es infinito, pues x está en el lado izquierdo de p_1p_2 y dentro de la envolvente convexa. Luego L contiene un lado finito de Voronoi y por lo tanto hay un vértice de Voronoi v' del lado opuesto a p_3.

Como $d(v) \bigcap d(v') = \phi$, por el lema anterior se tiene que hay puntos en $d(v')$ más próximos a x que y. Con esto termina la demostración.

Corolario 0.24.1. *Sea e el número de lados y v el número de vértices del diagrama de Voronoi de un conjunto de n puntos. Entonces se tiene*

$$e \leq 3n - 6$$

$$v \leq 2n - 4$$

Análisis de la complejidad La triangulación de Delauny de un conjunto S de n puntos se puede hacer en tiempo O(nlogn). En efecto, basta construir el diagrama de Voronoi de S y observar que por cada lado del diagrama se genera un lado de un triángulo. Por el corolario anterior, hallar los lados de estos triángulos es de una complejidad O(n). Por otro lado, el diagrama de Voronoi se construye en tiempo O(nlogn). Luego el proceso total se lleva cabo en un tiempo O(nlogn).

Ejercicios

1. Describa con detalle el algoritmo que calcula σ
2. Calcule la complejidad del algoritmo anterior.
3. Calcule la complejidad del algoritmo Voronoi.
4. Diseñe un algoritmo para hallar el máximo círculo vacío de un conjunto de n puntos.
5. Enuncie y demuestre un algoritmo para hallar la mediana de un conjunto de números $A = \{a_1.a_2, ..., a_n\}$

BIBLIOGRAFÍA

[1] de Berg, Mark; van Kreveld, Marc et al. (2000) *Computational Geometry: Algorithms and Applications*. Springer Verlag.

[2] O'Rourke, Joseph . (1998) *Computational Geometry in C.*Cambridge University Press.

[3] Preparata, Franco P.; Shamos, Michael Ian . (1985) *Computational Geometry: An Introduction* Springer Verlag.

[4] Shamos, M.; Hoey D. (1975) *Closest Point problems* Proc. 16th. Annu. IEEE Sympos. Found. Comp. Science, pp 151-62.

[5] Eddy, W. (1977) *A new convex hull algorithm for planar sets*ACM Trans. Math.Software3(3), 398-403.

[6] Bykat, A. (1978)*Convex hull of a finite set of points in two dimensions.*Info. Proc. Lett. 9, 223-228.

[7] Preparata, Franco; S. J. Hong (1977) *Convex hull of finite sets of points in two and three dimensions*Commun, ACM 20, 87-93.

[8] Jarvis, R.A. (1973) *On the identification of the convex hull of a finite set of points in the plane*Info.Proc.Lett. 2, 18-21.

[9] Graham, R. L. (1972) *An efficient algorithm for determining the convex hull of a finite set planar set* .Info.Proc. Lett. 1, 132-133.

[10] Chand, D. R.; Kapur, S.S. (1970) *An algorithm for convex polytopes.*JACM 17(1), 78-86 .

[11] Elzinga, J.; Hearn, D.W. (1972) *Geometrical solutions to some minimax location problems.*Trasp. Sci., 6: 379-394.

[12] Grimaldi, Ralph P. (1997) *Matemáticas discretas y Combinatoria.*Addison-Wesley Iberoamericana. Delaware U.S.A.

[13] Diestel, Reinhard. (2005) *Graph Theory*Electronic Edition Springer-Verlag New York.