Ejemplos de Geometría Avanzada

Agustín Santiago Gutiérrez

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Training Camp Argentina 2017

Contenidos

“He who despises Euclidean Geometry is like a man who, returning from foreign parts, disparages his home.”

H. G. Forder.

“Que nadie entre aquí si no sabe Geometría.”

Platón

“Las ecuaciones son sólo la parte aburrida de la matemática. Trato de ver las cosas en términos de geometría”

Stephen Hawking

Generalmente, los problemas de geometría de ACM requieren calcular alguna cantidad (por ej., una distancia, un área, etc) relacionada con elementos geométricos como puntos, líneas, círculos, etc. Para resolverlos, debemos poder ser capaces de:

1. Representar los objetos geométricos involucrados en el problema, para poder operar con ellos.
2. Desarrollar un algoritmo para buscar la respuesta deseada.

En esta charla nos concentraremos totalmente en la segunda parte, y suponemos que ya somos capaces de llevar a cabo la primera.

Como idea general, la técnica más básica para problemas geométricos es la de **discretización de candidatos**:

- Muchas veces se pide encontrar, por ejemplo, un punto del plano que cumpla alguna característica particular.
- Pero como el plano contiene infinitos puntos, no podemos probarlos todos.
- Casi siempre podemos identificar un subconjunto de **candidatos**, de manera que la respuesta buscada sí o sí sea alguno de ellos.
- En este caso, una solución posible consiste en explorar todos los candidatos y verificar lo que corresponda para cada uno.

Es fácil pasar por alto estas ideas, y tratar de buscar una solución más complicada “que encuentre la solución directamente” si uno no está atento.

Contenidos

Compresión de coordenadas + Tabla aditiva

- Luego de comprimir coordenadas, “el mundo” es una matriz de $2N \times 2N$
- Cada casilla tiene un peso, correspondiente a su área original
- Inicializamos $v_{i,j} = 0 \ \ \forall i, j$
- Para llenar por completo un rectángulo $[x_1, x_2] \times [y_1, y_2]$, hacemos:
  - $v_{x_1, y_1} \mathrel{+}= 1$
  - $v_{x_1, y_2} \mathrel{-}= 1$
  - $v_{x_2, y_1} \mathrel{-}= 1$
  - $v_{x_2, y_2} \mathrel{+}= 1$
- Al final, computando el acumulado $V_{i,j}$ (ver clase de estructuras, tabla aditiva), $V_{i,j}$ resulta indicar la cantidad de rectángulos que contienen esa casilla.
- Complejidad: $O(N^2)$

Contenidos

Algoritmo con Sweep Line

- Realizamos un barrido con una línea vertical, de izquierda a derecha.
- Cada rectángulo aporta dos eventos: “es encontrado” con su lado izquierdo, y “es removido” con su lado derecho.
- Mantendremos un multiconjunto de intervalos en y: cuando un lado izquierdo entra, se agrega al multiconjunto, y cuando un lado derecho se va, se saca.
- Cada vez que la línea avanza, barre un área total igual a $\Delta x \cdot U$, siendo U la longitud total de la unión de todos los intervalos en y.
- Será necesario comprimir coordenadas **en y**: Todas las coordenadas de los intervalos estarán entre 0 y N.
- Si podemos agregar intervalo, sacar intervalo, y consultar longitud de la unión en $O(\lg N)$, la complejidad final es $O(N \lg N)$
- Todo queda reducido a un problema de estructuras de datos en una sola dimensión

Contenidos

Implementación eficiente de la estructura de datos auxiliar

- La estructura de datos puede mantenerse en un Segment Tree con Lazy Propagation.
- La idea es que cada celda del arreglo almacena la **cantidad de intervalos** que lo contienen.
- Queremos hacer $+1$ o -1 a un rango (cuando un intervalo se va o viene)
- Cada celda tiene una longitud distinta, por la compresión de coordenadas
- Queremos saber la longitud total de celdas no nulas en un rango
- Guardaremos en los nodos del Segment Tree pares con dos valores:
  - El mínimo valor en el rango
  - La longitud total correspondiente a ese mínimo valor

Implementación eficiente de la estructura de datos auxiliar (cont)

- Observemos que estos pares se combinan de forma asociativa:

$$(min_1, l_1) \triangle (min_2, l_2) = \begin{cases} (min_1, l_1) & min_1 < min_2 \\ (min_2, l_2) & min_1 > min_2 \\ (min_1, l_1 + l_2) & min_1 = min_2 \end{cases}$$

- A su vez, son fáciles de actualizar ante un update de rango: Si se suma k a todo el rango, se pasa de (m, l) a $(m + k, l)$
- Luego, se pueden mantener en un Segment Tree con Lazy Propagation.
- Para saber la longitud total de no nulos en un rango $[i, j)$, se hace la consulta y al obtener (m, l):
  - Si $m > 0$, no hay ninguna celda nula, y la longitud total no nula es T, la longitud total del rango (i, j)
  - Si $m = 0$, hay celdas nulas, y la longitud total no nula es $T - l$

Contenidos

¿Qué es sweep circle?

- La idea de sweep circle es exactamente la misma que la de sweep line, pero en lugar de mover una recta imaginaria, movemos una circunferencia.
- La circunferencia puede moverse en linea recta (traslación) o alrededor de un centro fijo (rotación).
- Como un círculo es una figura acotada, el choque de la circunferencia con los puntos interesantes produce eventos de *entrada* y de *salida* en el círculo.

Ejemplo: Ubicación ideal de un círculo en el eje Y

Problema

Dado un radio $R > 0$ entero, se debe indicar cuál es la máxima cantidad de puntos de la grilla de coordenadas enteras que es posible encerrar con un círculo de radio R, cuyo centro se encuentre posicionado sobre la recta $x = 0$ (el eje y).

Ejemplo: Ubicación ideal de un círculo en el eje Y

Problema

Dado un radio $R > 0$ entero, se debe indicar cuál es la máxima cantidad de puntos de la grilla de coordenadas enteras que es posible encerrar con un círculo de radio R, cuyo centro se encuentre posicionado sobre la recta $x = 0$ (el eje y).

Observación: Alcanza con considerar las posiciones $0 \leq y \leq 1$

Planteo con sweep circle

- Comenzamos con el círculo ubicado en $(0, 0)$, y todos los correspondientes puntos de la grilla adentro.
- “Movemos” el círculo en vertical, hasta llegar a $(0, 1)$, procesando los eventos de entrada y salida de puntos.
- La máxima cantidad de puntos que tengamos dentro del círculo en cualquier momento, es el resultado.

Planteo con sweep circle

- Comenzamos con el círculo ubicado en $(0, 0)$, y todos los correspondientes puntos de la grilla adentro.
- "Movemos" el círculo en vertical, hasta llegar a $(0, 1)$, procesando los eventos de entrada y salida de puntos.
- La máxima cantidad de puntos que tengamos dentro del círculo en cualquier momento, es el resultado.

- Notar que hay solamente $O(R)$ eventos de entrada / salida, y además la cantidad de puntos totales dentro del círculo inicial puede computarse en $O(R)$.
- Con todo esto y la técnica de barrido, el problema se resuelve en $O(R \lg R)$

Contenidos

Ejemplo: Radio óptimo para cubrir K puntos

Problema

Dado un conjunto de N puntos en el plano, encontrar el mínimo radio posible de un círculo que cubra al menos K de ellos.

Ejemplo: Radio óptimo para cubrir K puntos

Problema

Dado un conjunto de N puntos en el plano, encontrar el mínimo radio posible de un círculo que cubra al menos K de ellos.

- Observación 1: Podemos asumir que el círculo toca alguno de los puntos.

Ejemplo: Radio óptimo para cubrir K puntos

Problema

Dado un conjunto de N puntos en el plano, encontrar el mínimo radio posible de un círculo que cubra al menos K de ellos.

- Observación 1: Podemos asumir que el círculo toca alguno de los puntos.
- Observación 2: Podemos hacer búsqueda binaria en la respuesta.

Ejemplo: Radio óptimo para cubrir K puntos

Problema

Dado un conjunto de N puntos en el plano, encontrar el mínimo radio posible de un círculo que cubra al menos K de ellos.

- Observación 1: Podemos asumir que el círculo toca alguno de los puntos.
- Observación 2: Podemos hacer búsqueda binaria en la respuesta.
- Idea: Para verificar si un cierto radio R puede cubrir K puntos, probamos todos los valores posibles de un cierto punto que el círculo toca (por Obs 1).
- Más idea: Para cada punto, **giramos** el círculo a su alrededor.

Contenidos

Ancho de polígono

- Observación: El ancho se produce en dirección normal a algún lado.
- Esto da un algoritmo $O(N^2)$ (para cada lado, buscar el punto más lejano). ¿Podemos mejorar más?

Ancho de polígono

- Observación: El ancho se produce en dirección normal a algún lado.
- Esto da un algoritmo $O(N^2)$ (para cada lado, buscar el punto más lejano). ¿Podemos mejorar más?
- ¡Sí! Fijado un lado l del polígono, la distancia $f(x)$ de un vértice x a l es unimodal en x. $O(N \lg N)$

Ancho de polígono

- Observación: El ancho se produce en dirección normal a algún lado.
- Esto da un algoritmo $O(N^2)$ (para cada lado, buscar el punto más lejano). ¿Podemos mejorar más?
- ¡Sí! Fijado un lado l del polígono, la distancia $f(x)$ de un vértice x a l es unimodal en x. $O(N \lg N)$
- Incluso se puede mejorar a $O(N)$, notando que el valor óptimo de x siempre avanza al avanzar el lado l (two-pointers).

Contenidos

Triángulo máximo

- Solución fuerza bruta: $O(N^3)$
- ¿Podemos mejorar?

Triángulo máximo

- Solución fuerza bruta: $O(N^3)$
- ¿Podemos mejorar?
- ¡Sí! Fijados A, B, el área $f(A, B, C)$ es unimodal en C. $O(N^2 \lg N)$

Triángulo máximo

- Solución fuerza bruta: $O(N^3)$
- ¿Podemos mejorar?
- ¡Sí! Fijados A, B, el área $f(A, B, C)$ es unimodal en C. $O(N^2 \lg N)$
- Incluso se puede mejorar a $O(N^2)$, notando que el valor óptimo de C siempre avanza al avanzar B.

Referencia

Solución $O(N)$:
(consiste en una especie de búsqueda local, rotando los 3 puntos)

`https://stackoverflow.com/questions/1621364/how-to-find-largest-triangle-in-convex-hull-aside-from-brute-force-search`

Contenidos

Par de puntos más lejano

- Solución fuerza bruta: $O(N^2)$.
- ¿Se podrá mejorar?

Par de puntos más lejano

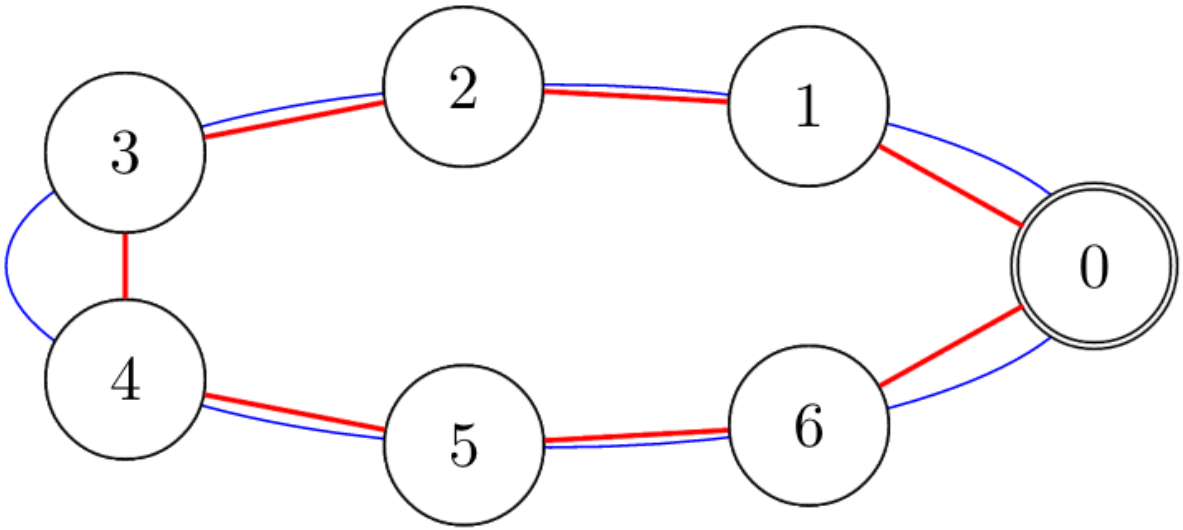
- Solución fuerza bruta: $O(N^2)$.
- ¿Se podrá mejorar?
- Uno puede pensar en un algoritmo $O(N \lg N)$ como venimos haciendo, usando la convexidad.
- Para cada vértice v fijo, buscamos con búsqueda binaria o ternaria el vértice w a máxima distancia de v.
- ¿Es $d(v, w)$ unimodal en w, para un v fijo?

Par de puntos más lejano

- Solución fuerza bruta: $O(N^2)$.
- ¿Se podrá mejorar?
- Uno puede pensar en un algoritmo $O(N \lg N)$ como venimos haciendo, usando la convexidad.
- Para cada vértice v fijo, buscamos con búsqueda binaria o ternaria el vértice w a máxima distancia de v.
- ¿Es $d(v, w)$ unimodal en w, para un v fijo?
- **NO.**

Contraejemplo

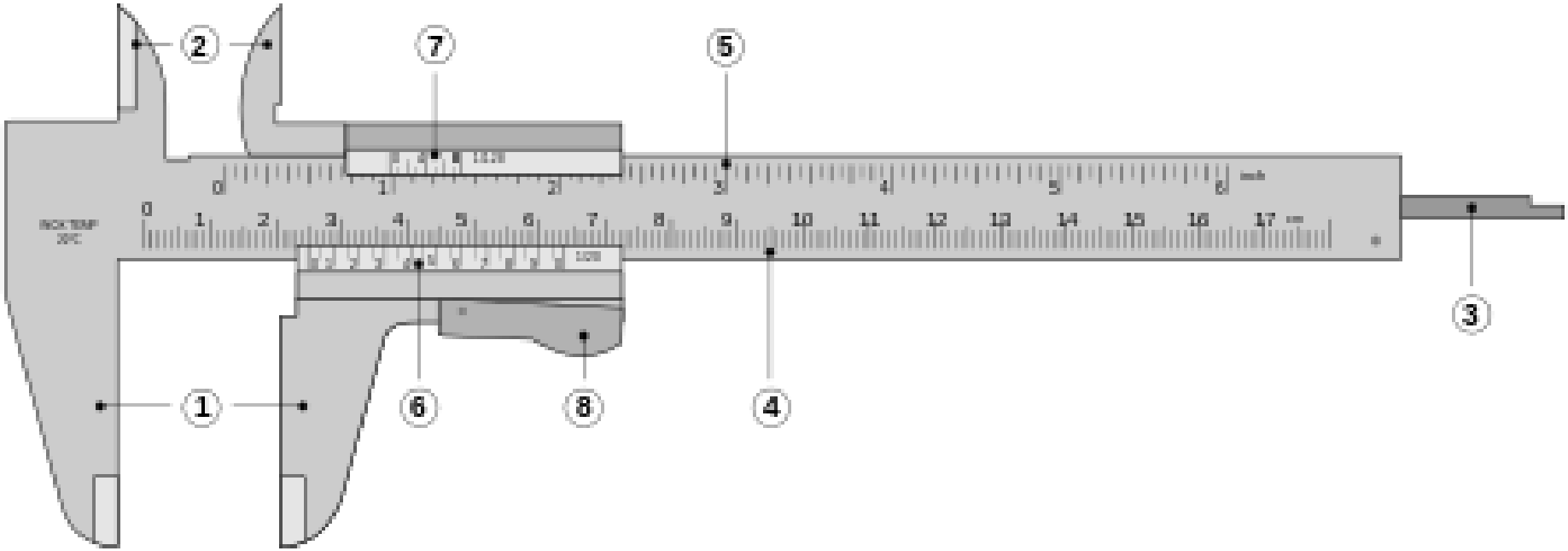
- Si fijamos $v = 2$, al mover w en sentido horario, $f(v, w)$ sube hasta el nodo 0, luego baja hasta el nodo 5, luego vuelve a subir hasta el 4 y finalmente baja por última vez.
- Resumiendo:
  - En un polígono convexo, las distancias de los vértices a un **lado(recta)** fijo **forman** una función unimodal
  - En un polígono convexo, las distancias de los vértices a un **vértice** fijo **NO forman** una función unimodal



Contenidos

Rotating calipers

- Caliper en inglés es calibre.
- Esto es un calibre:



Rotating calipers (cont)

- La analogía surge de imaginar que vamos girando un calibre ajustado al polígono todo el tiempo.
- Nuestro calibre siempre toca **dos vértices** del polígono.
- Un par de vértices que son tocados al mismo tiempo para alguna rotación del calibre, se llama un par **antipodal**
- El método de rotating calipers consiste en simular eficientemente en $O(N)$ este proceso de rotación, enumerando todos los pares antipodales
- Caso borde del recorrido: Si el polígono tiene dos lados paralelos, cuando el calibre se ajusta a ellos hay 4 pares de vértices antipodales, correspondientes a esos dos lados.

Rotating calipers: Algoritmo

- En nuestro caso, giraremos en sentido **horario**
- De un lado, comenzamos con un vértice p de mínimo x, y en caso de empate el de mínimo y
- Del otro lado comenzamos con un vértice q de máximo x, y en caso de empate el de máximo y
- (p, q) es nuestro primer par antipodal.
- En cada paso consideramos los vértices siguientes a p y q en sentido horario, que son p_n y q_n
- De esos dos, elegimos avanzar por "el que primero es tocado por los calibres", cambiando así a:
  - (p_n, q) si $(p_n - p) \times (q_n - q) > 0$
  - (p, q_n) si $(p_n - p) \times (q_n - q) < 0$
  - Si $(p_n - p) \times (q_n - q) = 0$, (p, p_n) y (q, q_n) son lados paralelos. Los procesamos y avanzamos a (p_n, q_n)

Aplicación al problema original

- Se puede observar que la distancia máxima entre vértices se alcanza necesariamente en un par antipodal.
- Con el método de rotating calipers, los enumeramos en $O(N)$ y tomamos el par más lejano encontrado.