Ejemplos de Geometría Avanzada

Agustín Santiago Gutiérrez

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Training Camp Argentina 2021

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 ICPC 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.

Discretización de candidatos

- Se pide encontrar algo que tenga cierta propiedad particular
- Por ejemplo, un cierto punto del plano óptimo o especial
- El plano tiene infinitos puntos $\Rightarrow$ no se puede probar todos
- Identificar algunos **candidatos**
- Explorar todos los candidatos y verificar 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.

Ejemplo ($n \leq 1000$)

Torneo Argentino de Programación — ACM–ICPC 2013 1

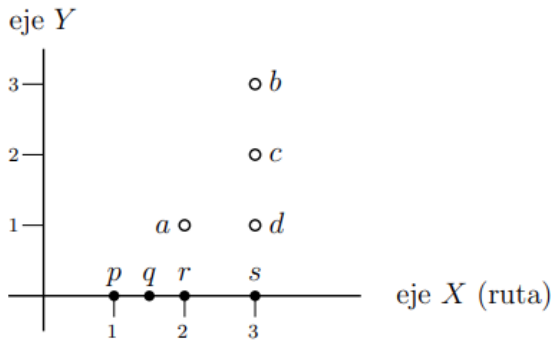
Problema A – Al costado del camino

Al costado del camino, hay palmeras, hay un bar, hay sombra, hay algo más. En este problema, nos interesan particularmente las palmeras.

Ana, Adán, Alan y Amanda organizaron un viaje. Mientras Ana y Adán se encargaban de nimiedades como hacer revisar el auto, preparar el equipaje y conseguir lugar para hospedarse, Alan y Amanda se dedicaban a lo más importante: estudiar los paisajes de palmeras a los que iban a tener acceso a lo largo del camino.

La ruta que ahora recorren es totalmente recta, y para los propósitos de este problema la modelamos como la recta $Y = 0$ del plano XY. Al costado de la ruta con coordenada $Y > 0$ hay palmeras, de modo que modelamos cada una de ellas como un punto diferente del plano XY con coordenada $Y > 0$. Alan y Amanda notaron que desde cada punto de la ruta son visibles determinadas palmeras, y que en general las palmeras visibles varían a lo largo de la ruta. Una palmera es visible desde un punto de la ruta si y sólo si el segmento que une ambos puntos no pasa por ninguna otra palmera.

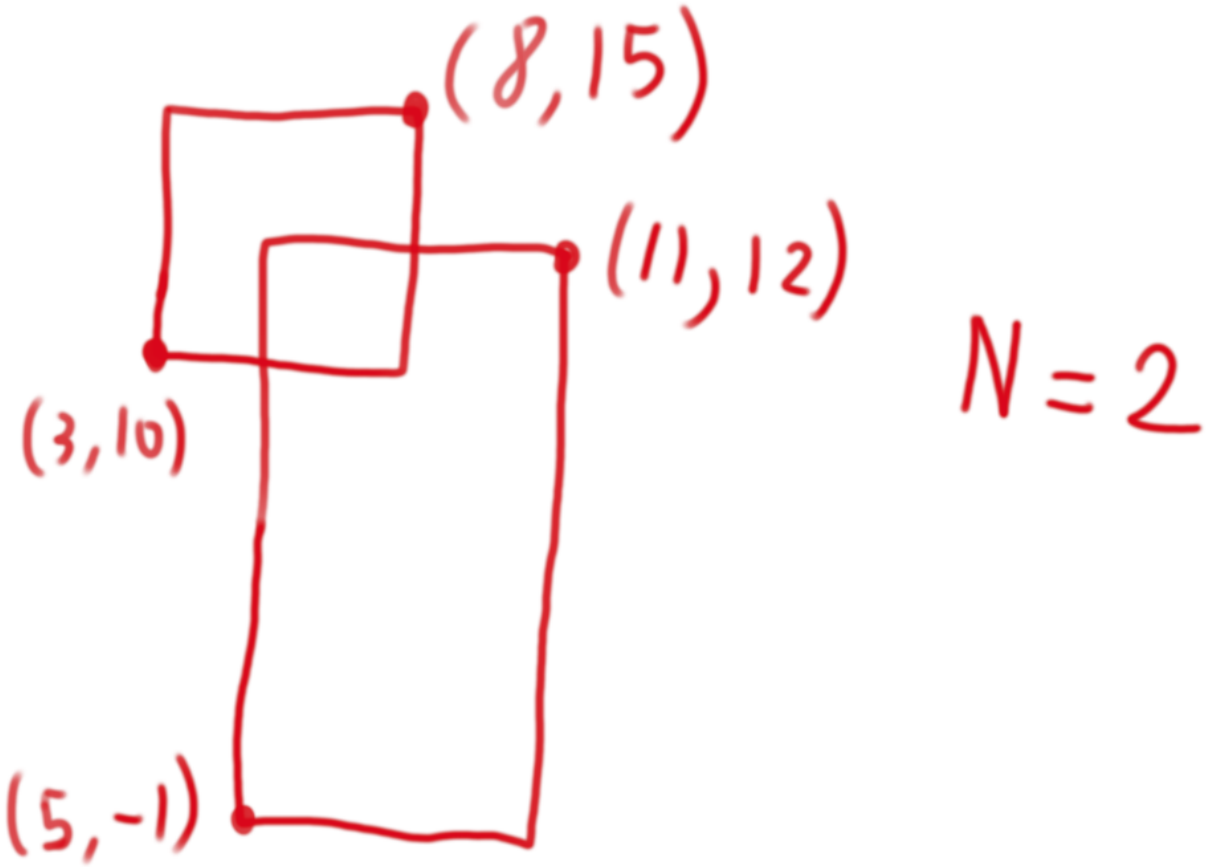
En la siguiente figura los círculos sin relleno representan a las palmeras de la primera entrada de ejemplo, mientras que los círculos con relleno indican puntos posibles de la ruta.



Contenidos

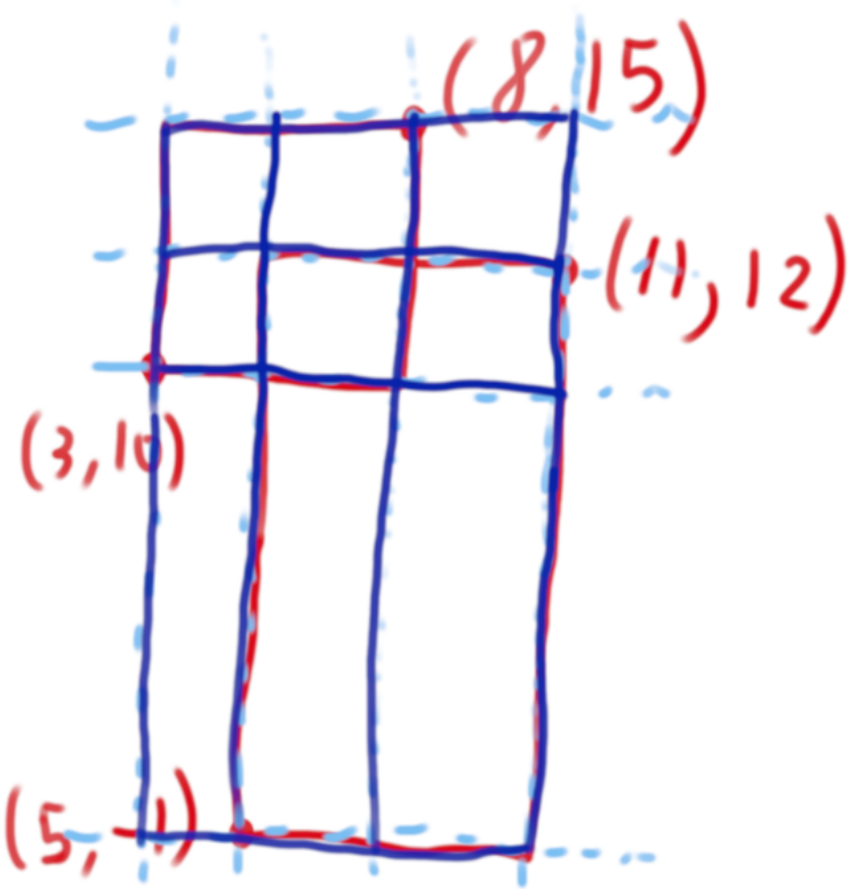
Compresión de coordenadas

Entrada:



Compresión de coordenadas (cont.)

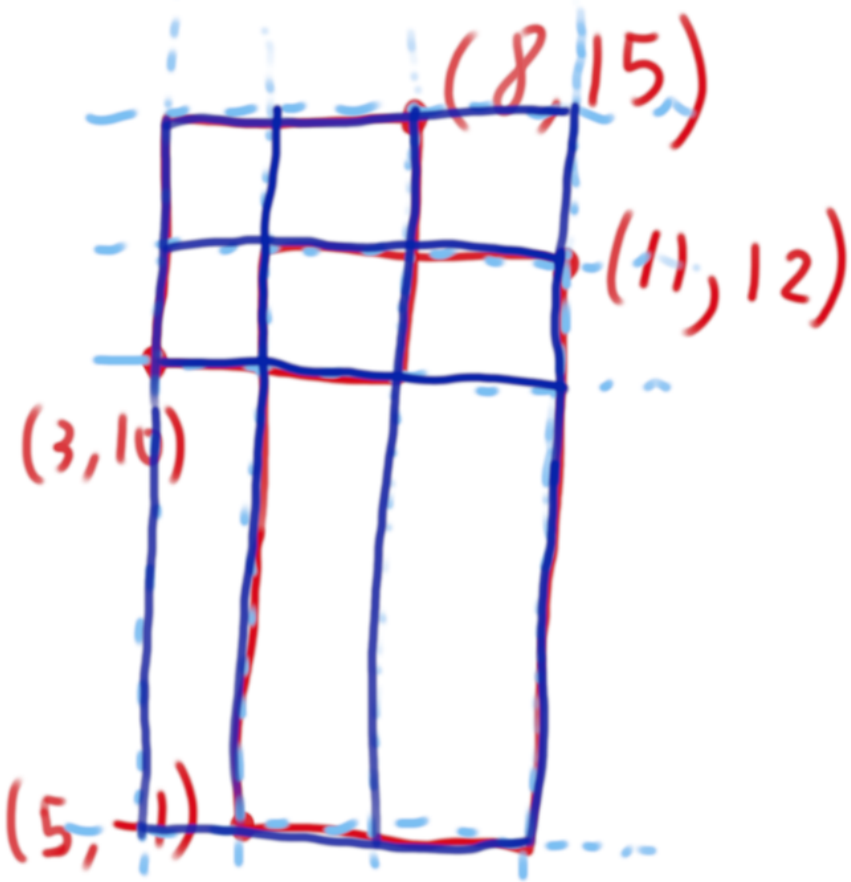
Coordenadas importantes:



$N = 2$

Compresión de coordenadas (cont.)

- Compresión de coordenadas $\rightarrow$ Grilla de hasta $2N \times 2N$ esquinas
- Hasta $2N \times 2N$ esquinas $\rightarrow$ Hasta $(2N-1) \times (2N-1)$ casillas
- Cada casilla tiene un peso: su área original



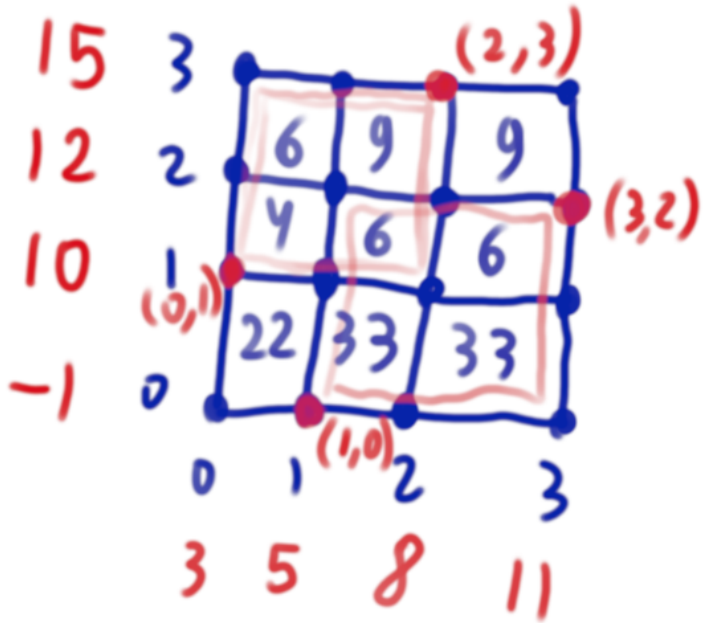


Tabla aditiva

- Grilla de $(n+1) \times (m+1)$ esquinas
- Matriz v de $n \times m$ casillas
- Extendemos a $(n+1) \times (m+1)$ casillas para evitar casos borde

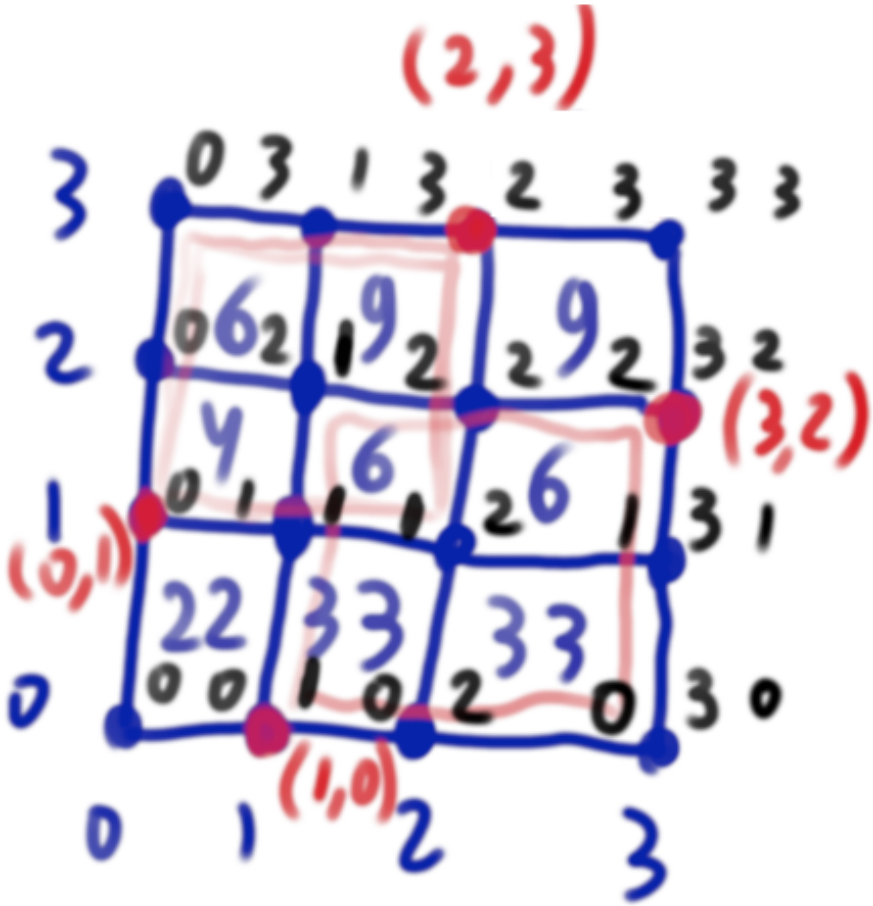


Tabla aditiva (cont.)

- Inicializamos $v_{i,j} = 0 \ \ \forall 0 \leq i \leq n, 0 \leq j \leq m$ (índices de casillas)
- Para llenar $[x_1, x_2] \times [y_1, y_2]$ (índices de esquinas), hacemos:
  - $v_{x_1,y_1} \mathrel{+}= 1 \quad , \quad v_{x_1,y_2} \mathrel{-}= 1 \quad , \quad v_{x_2,y_1} \mathrel{-}= 1 \quad , \quad v_{x_2,y_2} \mathrel{+}= 1$

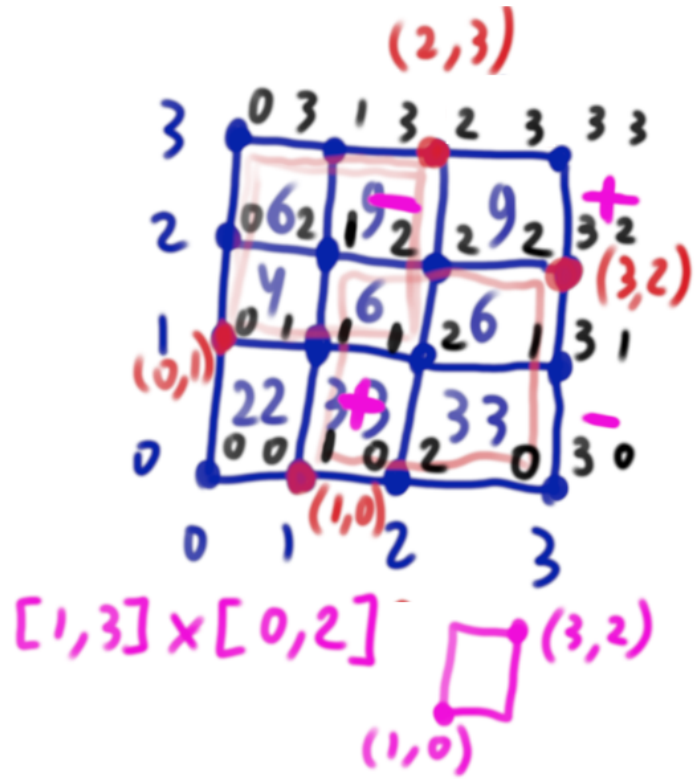


Tabla aditiva (cont.)

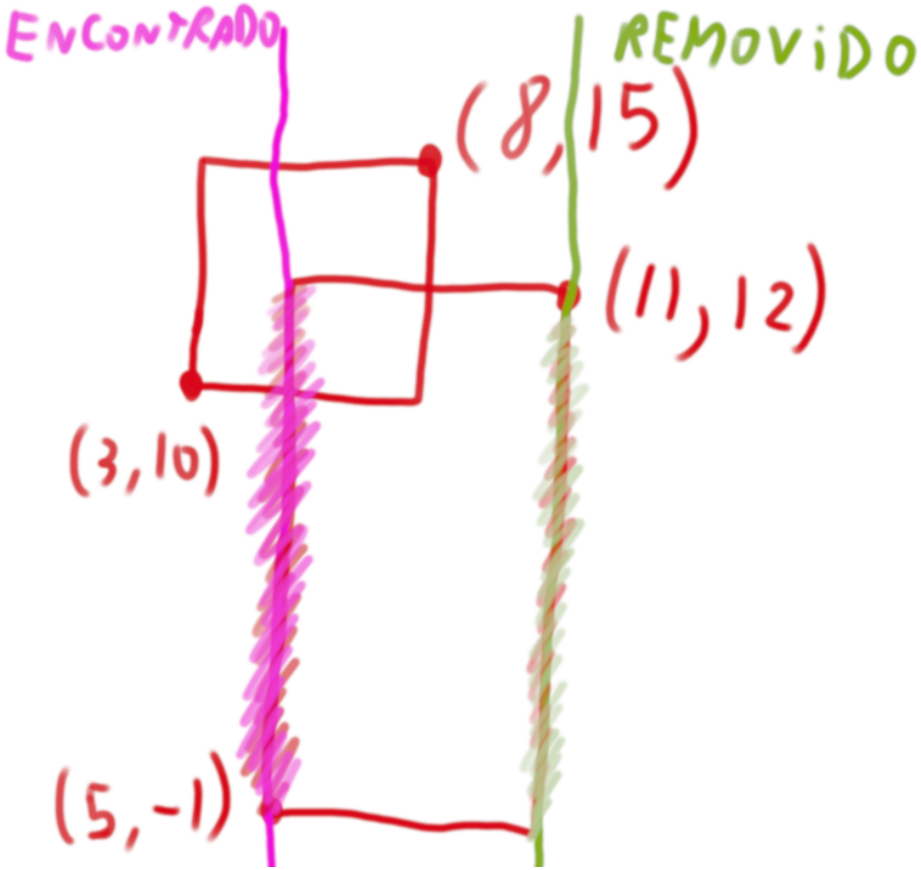
- Computamos el acumulado horizontal y vertical $V_{i,j}$
- $V_{i,j}$ es la cantidad de rectángulos que contienen la casilla (i, j)
- Complejidad: $O(N^2)$

```
forn(step,2)
forsn(i,step==0, n+1)
forsn(j,step==1, m+1)
    v[i][j] += v[i-(step==0)][j-(step==1)];
```

Contenidos

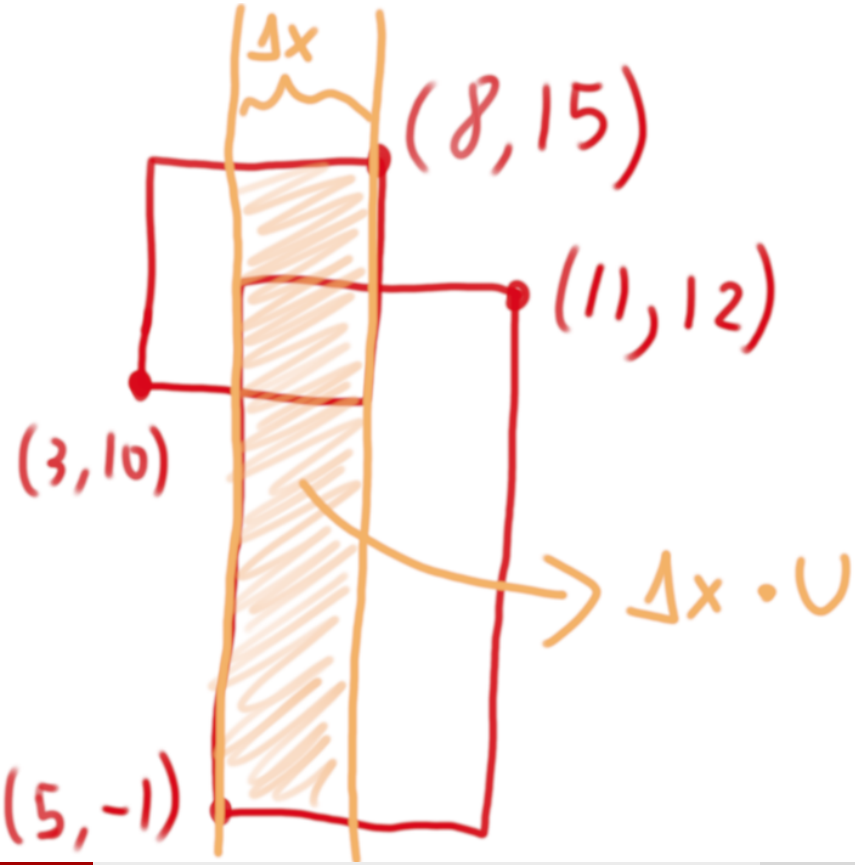
Algoritmo con Sweep Line

- Barrido con una línea vertical, de izquierda a derecha
- Rectángulo $\Rightarrow$ dos eventos:
  - “Es encontrado” con su lado izquierdo
  - “Es removido” con su lado derecho



Algoritmo con Sweep Line (cont.)

- Multiconjunto de intervalos en y:
  - Cuando un lado izquierdo entra, se agrega al multiconjunto
  - Cuando un lado derecho se va, se saca
- Sea U la **longitud total de la unión** de todos los intervalos en y
- Cada avance de la línea barre un área $\Delta x \cdot U$



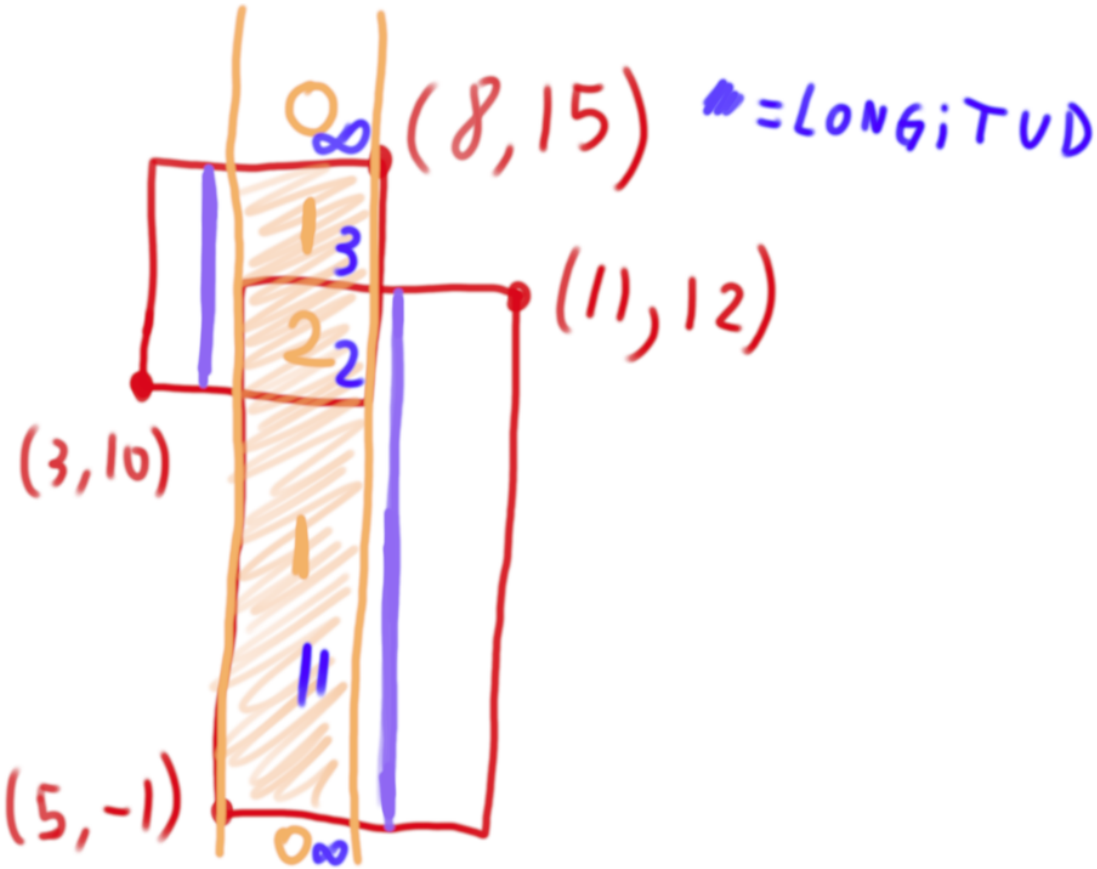
Algoritmo con Sweep Line (cont.)

- Comprimimos coordenadas **en y**
- Las coordenadas de los intervalos estarán entre 0 y N
- Si podemos hacer en $O(\lg N)$:
  - Agregar intervalo
  - Sacar intervalo
  - Consultar longitud de la unión
- La complejidad final será $O(N \lg N)$
- **Problema de estructuras de datos en una sola dimensión**

Contenidos

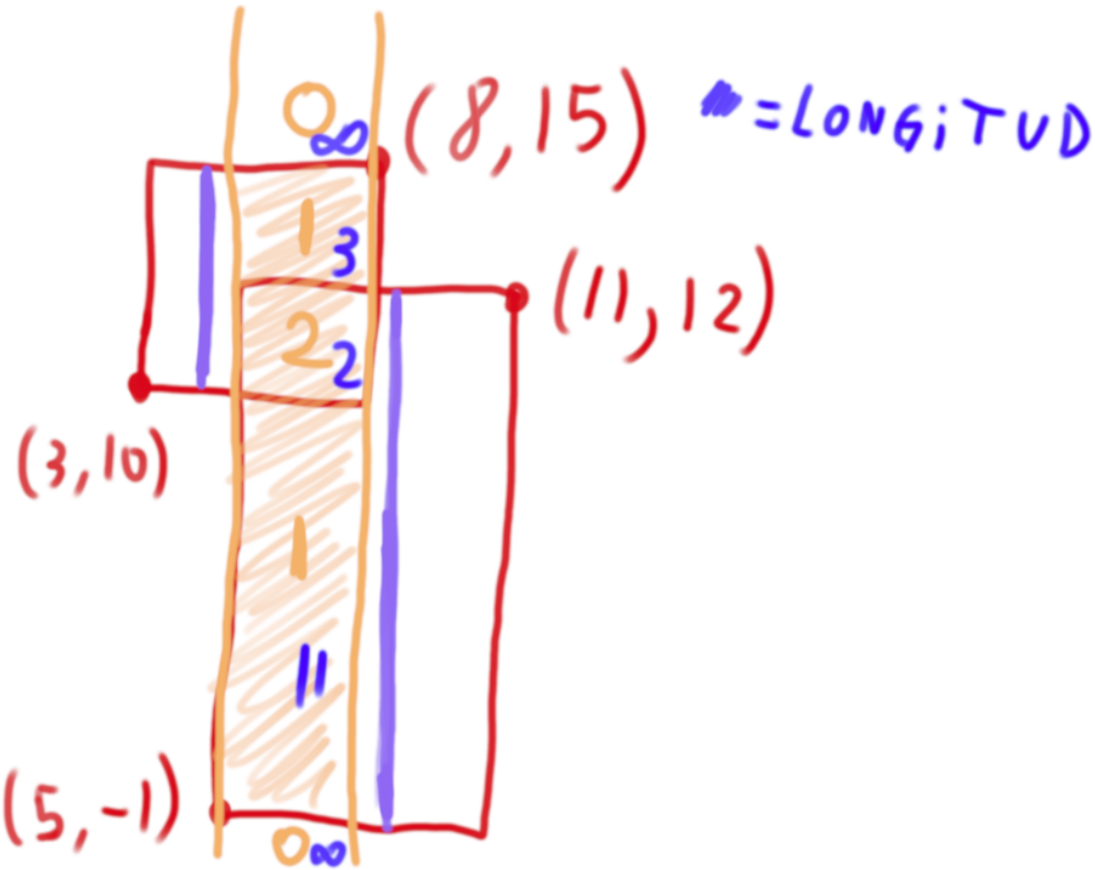
Implementación eficiente 1: Lazy Propagation

- Segment Tree con Lazy Propagation
- Celda del arreglo $=$ **cantidad de intervalos** en ese tramo
- Hacemos $+1$ o -1 a un **rango** (cuando un intervalo se va o viene)
- Cada celda tiene **longitud distinta** (compresión de coordenadas)



Implementación eficiente 1: Lazy Propagation (cont.)

- Necesitamos la **longitud total de celdas no nulas**
- Guardamos en los nodos del Segment Tree dos valores:
  - El **mínimo valor** en el rango
  - La **longitud total** correspondiente a **ese mínimo valor**



Implementación eficiente 1: Lazy Propagation (cont.)

- Estos pares se combinan de forma asociativa:

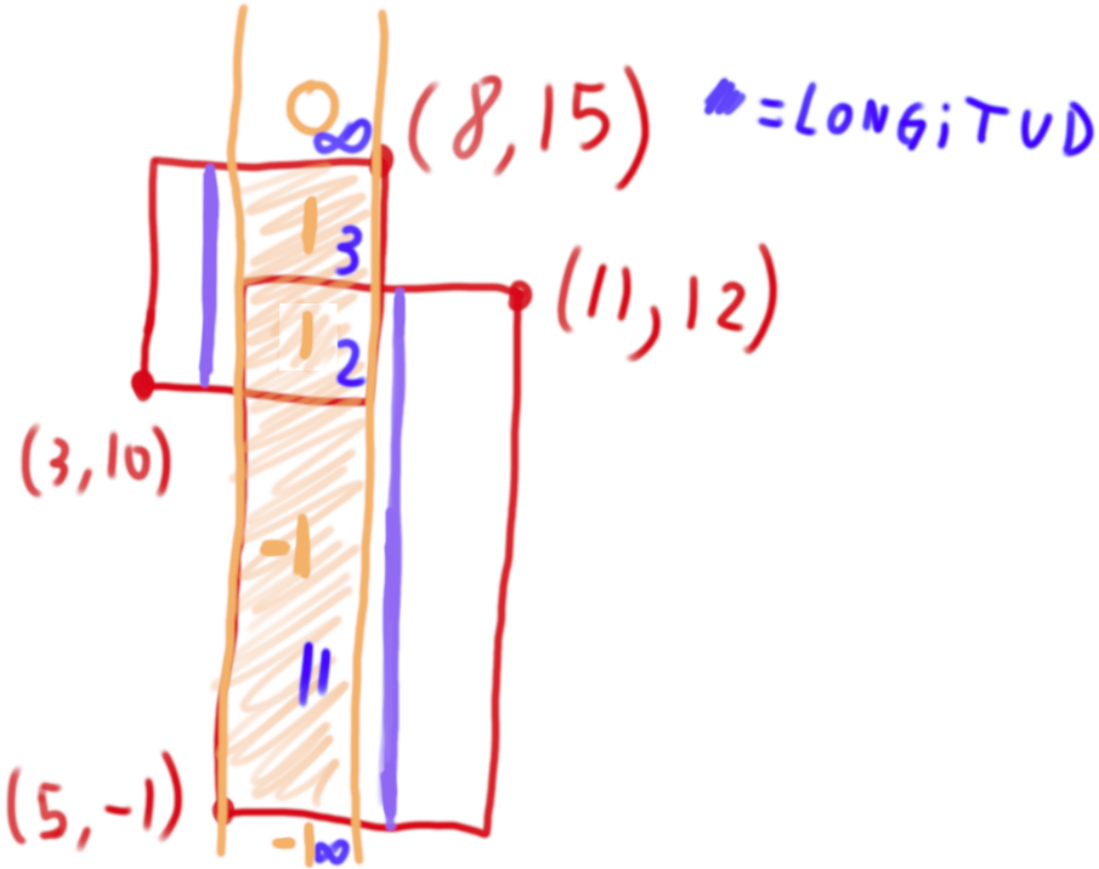
$$(min_1, l_1) \triangle (min_2, l_2) = \begin{cases} (min_1, l_1) & min_1 < min_2 \\ (min_2, l_2) & min_1 > min_2 \\ (min_1, l_1 + l_2) & min_1 = min_2 \end{cases}$$

- Son fáciles de actualizar ante un update de rango:
  - Si se suma k a todo el rango, se pasa de (m, l) a $(m + k, l)$
- Para saber la longitud total de no nulos en un rango $[i, j)$:
  - Se hace la consulta al Segment Tree y se obtiene (m, l)
  - Si $m > 0$:
    - No hay ninguna celda nula
    - La longitud total no nula es $T(i, j) = y_j - y_i$
    - Es decir, la longitud total del rango (i, j)
  - Si $m = 0$:
    - Hay celdas nulas
    - La longitud total no nula es $T(i, j) - l$

Contenidos

Implementación eficiente 2: Sin Lazy

- Segment Tree común y corriente
- Celda del arreglo = **diferencias consecutivas** del caso anterior
- Si a esto le hiciéramos sumas parciales, quedaría lo anterior
- Ahora hacemos $+1$ y -1 a los **extremos del rango**



Implementación eficiente 2: Sin Lazy (cont.)

- Antes teníamos mínimo valor y su longitud: (m, l)
- Ahora corresponde a **mínima suma de prefijos**
- Guardaremos también la **suma total** del rango
- ¡La recursión es casi idéntica!

$$(m_1, s_1, l_1) \triangle (m_2, s_2, l_2) = \begin{cases} (m_1, s_1 + s_2, l_1) & m_1 < s_1 + m_2 \\ (s_1 + m_2, s_1 + s_2, l_2) & m_1 > s_1 + m_2 \\ (m_1, s_1 + s_2, l_1 + l_2) & m_1 = s_1 + m_2 \end{cases}$$

Contenidos

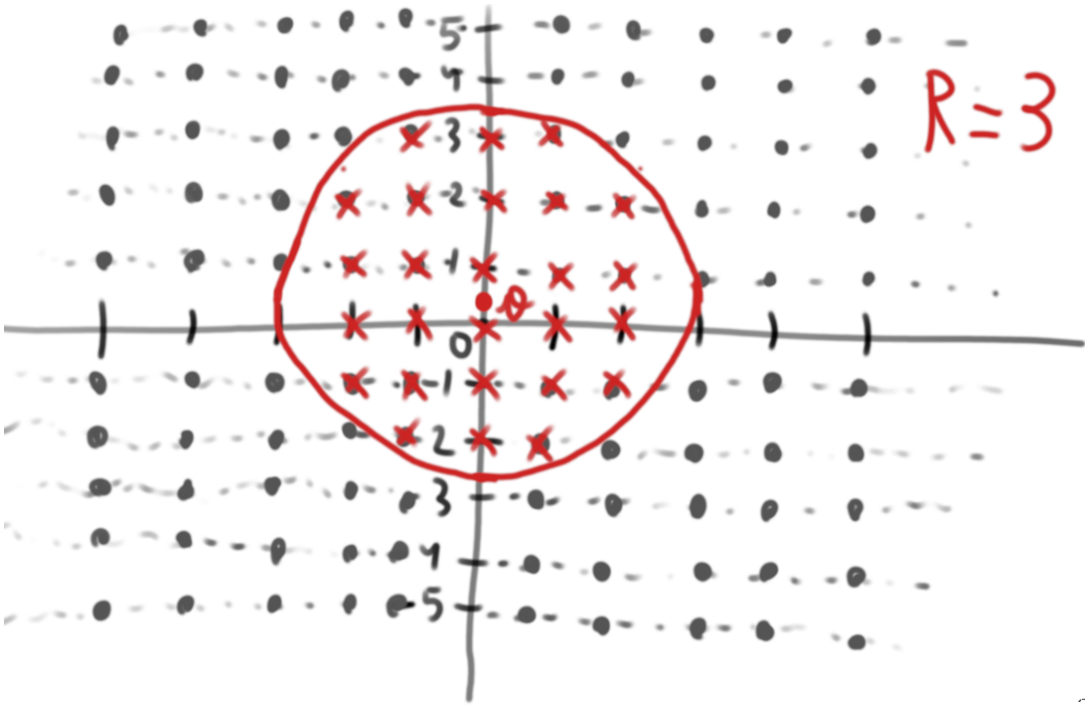
¿Qué es sweep circle?

- Misma idea que sweep line, pero con una circunferencia
- La circunferencia puede moverse:
  - En linea recta (traslación)
  - Alrededor de un centro fijo (rotación)
  - En cualquier otra trayectoria bien definida
- Un círculo es una **figura acotada** (como un intervalo en 1D)
- El choque con los puntos interesantes produce eventos
  - De *entrada* al círculo
  - De *salida* del círculo

Ejemplo: Ubicación ideal de un círculo en el eje Y

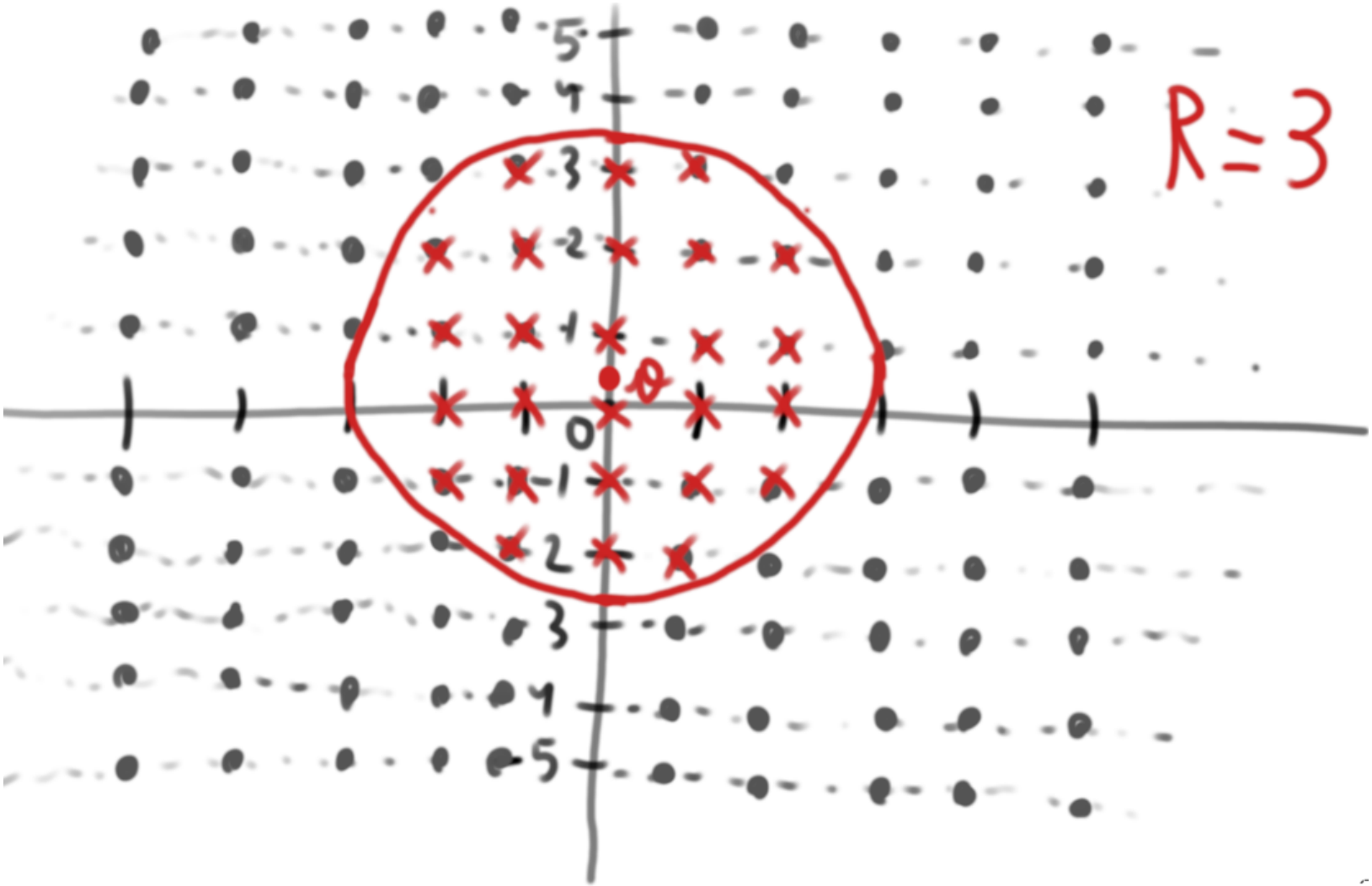
Problema ("Inscripción", OIA Nacional Nivel 3, año 2002)

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 $0 \leq y \leq 1$

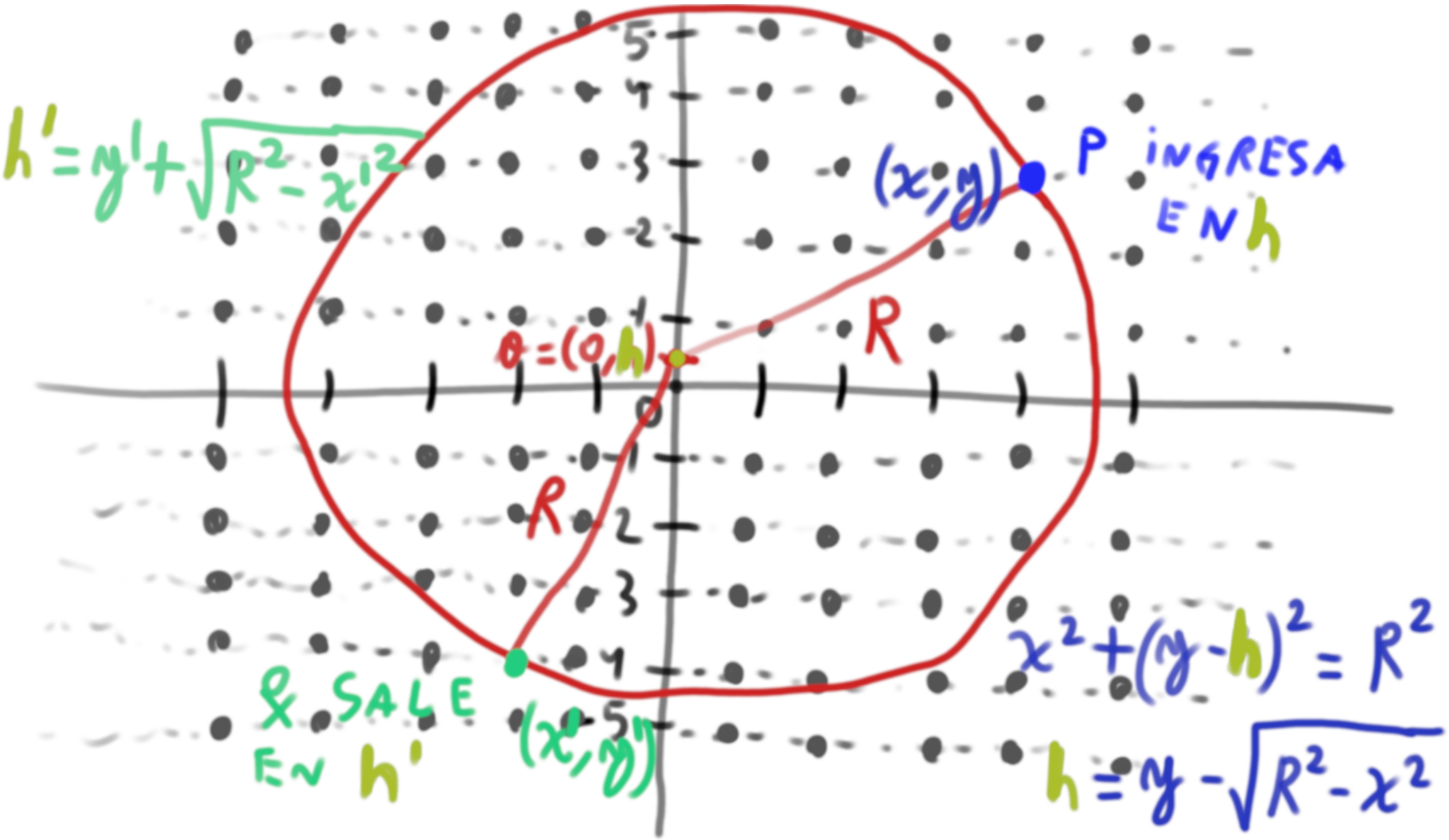


Planteo con sweep circle

- Comenzamos con el círculo ubicado en $(0, 0)$
- Los correspondientes puntos de la grilla inician adentro
- “Movemos” el círculo en vertical, desde $(0, 0)$ hasta llegar a $(0, 1)$
- Procesamos **en orden** los **eventos** (entradas y salidas de puntos)
- Respuesta: Máxima cantidad de puntos dentro en algún momento

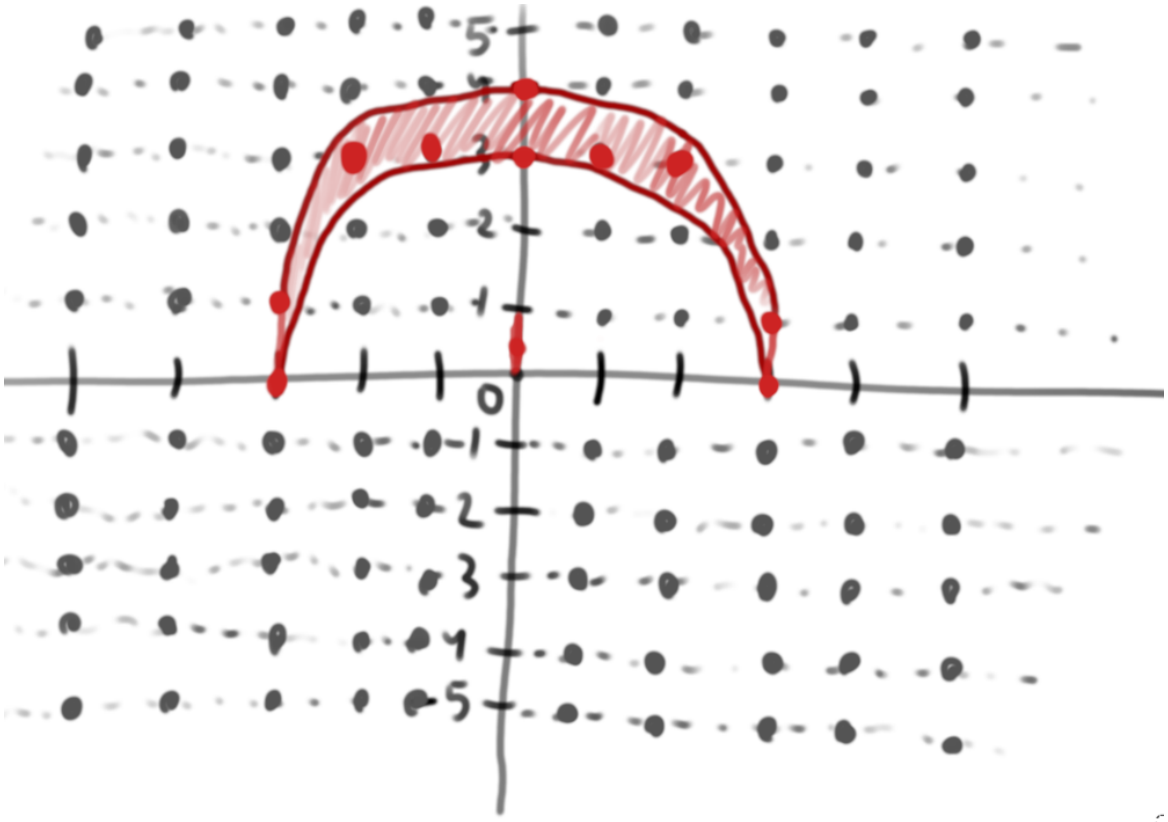
Planteo con sweep circle (cont.)

- El cálculo del *h* "tiempo" de los eventos es geométrico
- Los ϵ "sumando o restando" $\Rightarrow$ bordes "inclusive o exclusive"



Planteo con sweep circle (cont.)

- Hay $O(R)$ eventos de entrada / salida.
- La cantidad de puntos totales dentro del círculo inicial puede computarse en $O(R)$
- Con 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 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 toca alguno de los puntos
- Observación 2: Podemos hacer búsqueda binaria en la respuesta

Ejemplo: Radio óptimo para cubrir K puntos

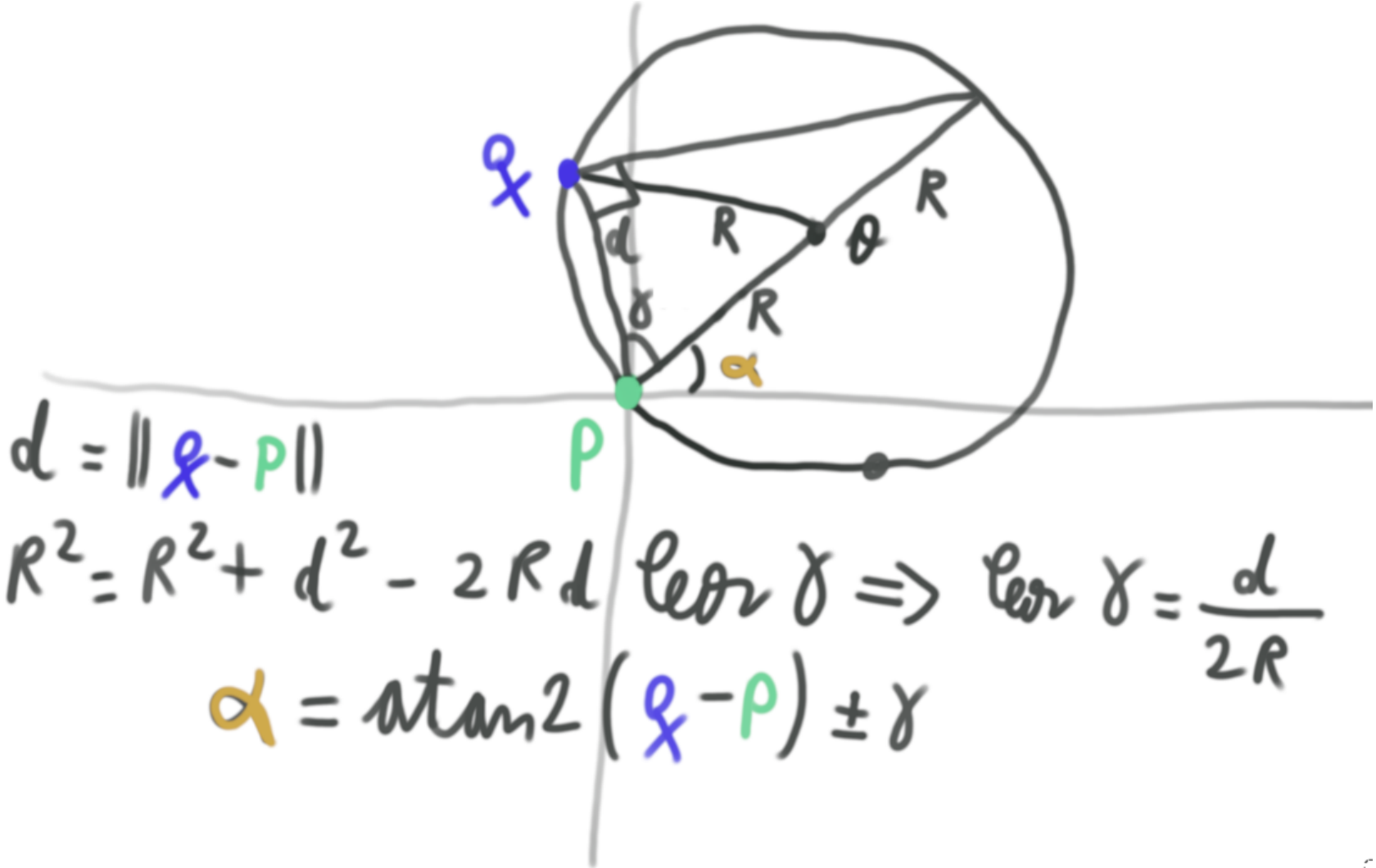
Problema

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 toca alguno de los puntos
- Observación 2: Podemos hacer búsqueda binaria en la respuesta
- Idea: Para verificar si con radio R se puede cubrir K puntos, probamos cada punto a ver si es el de observación 1
- Más idea: Para cada punto, **giramos** el círculo a su alrededor

Ejemplo: Radio óptimo para cubrir K puntos (cont.)

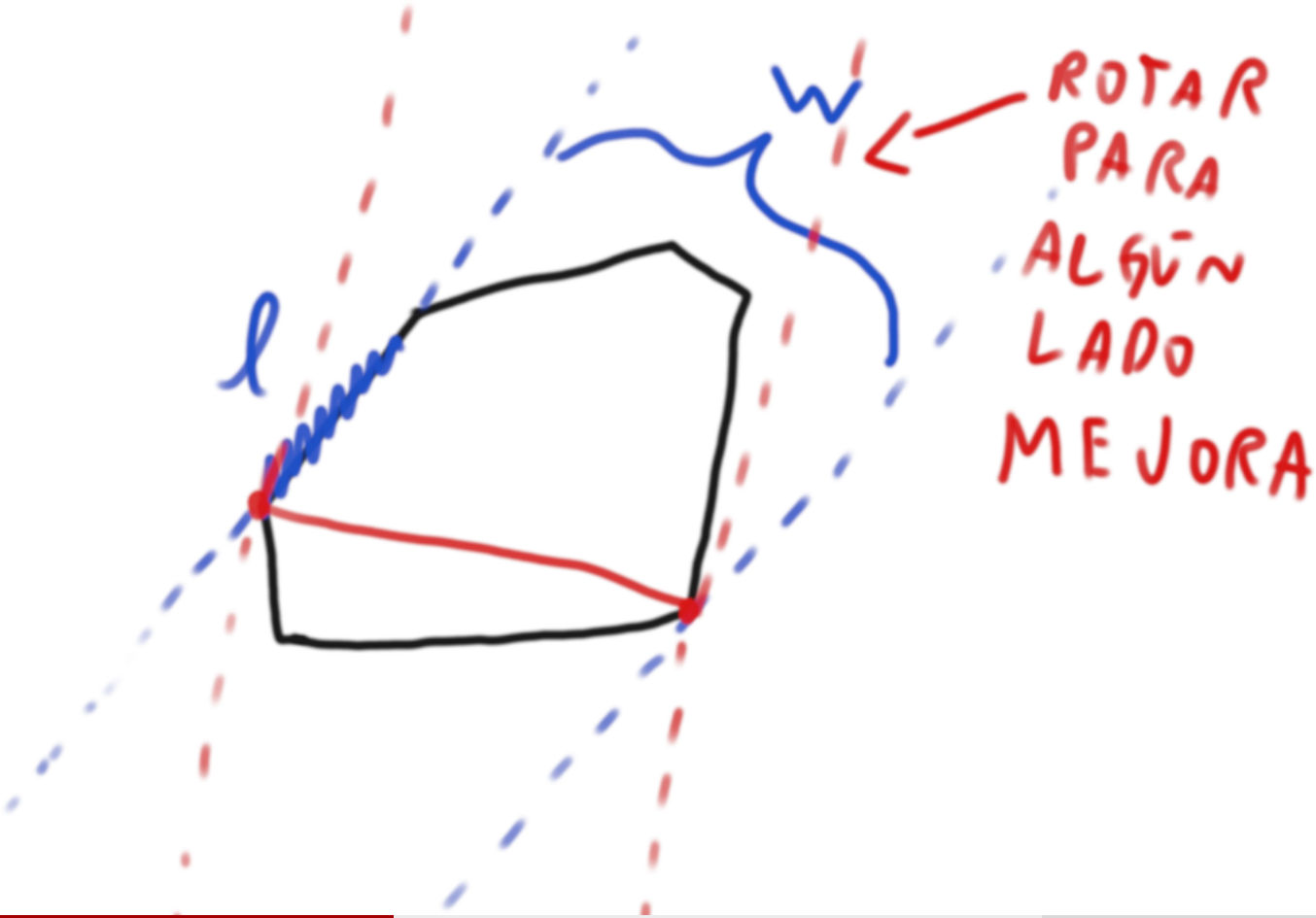
Cálculo de los tiempos de los eventos:



Contenidos

Ancho de polígono

- Observación: Ancho en dirección normal a algún lado
- Algoritmo $O(N^2)$ (para cada lado, buscar el punto más lejano)
- ¿Podemos mejorar más?



Ancho de polígono

- ¡Sí! Fijamos 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

- ¡Sí! Fijamos un lado l del polígono.
- La distancia $f(x)$ de un vértice x a l es unimodal en x
- $O(N \lg N)$
- ¡Se puede mejorar más!
- El valor óptimo de x siempre avanza al avanzar el lado l
- Técnica de dos punteros: $O(N)$

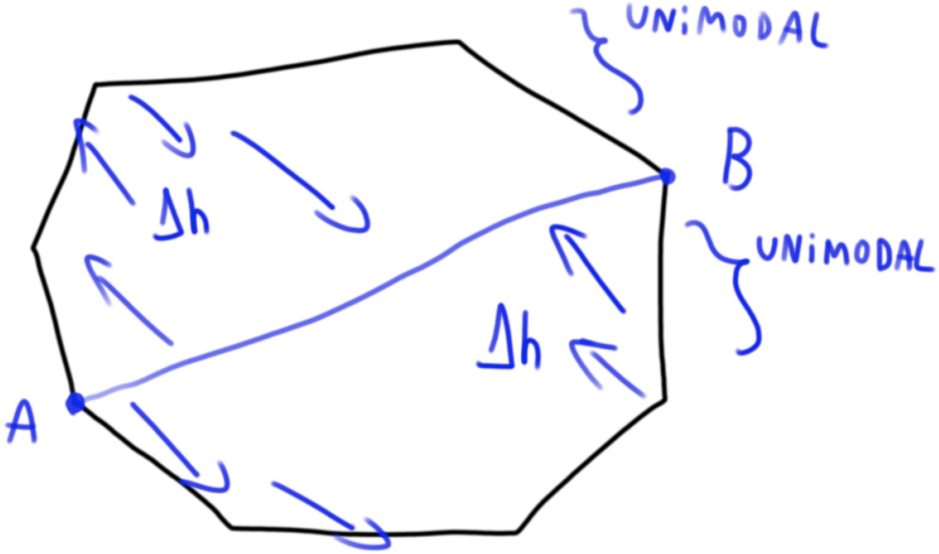
Contenidos

Triángulo máximo

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

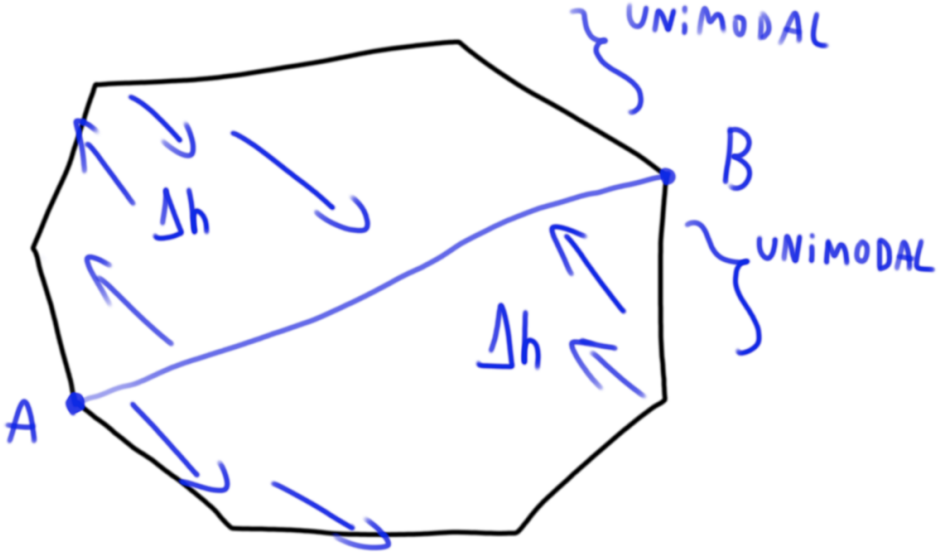
Triángulo máximo

- 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 $O(N \lg N)$:

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

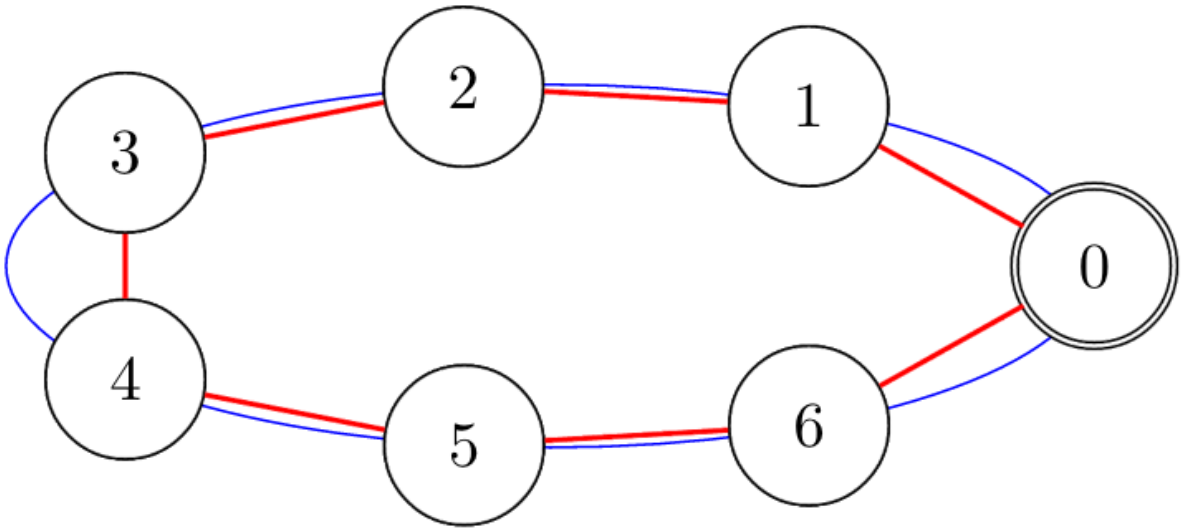
- Solución fuerza bruta: $O(N^2)$.
- ¿Se podrá mejorar?
- Uno puede pensar en un algoritmo $O(N \lg N)$ como venimos haciendo, usando la convexidad.
- Para cada vértice v fijo, buscamos con búsqueda binaria o ternaria el vértice w a máxima distancia de v.
- ¿Es $d(v, w)$ unimodal en w, para un v fijo?

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

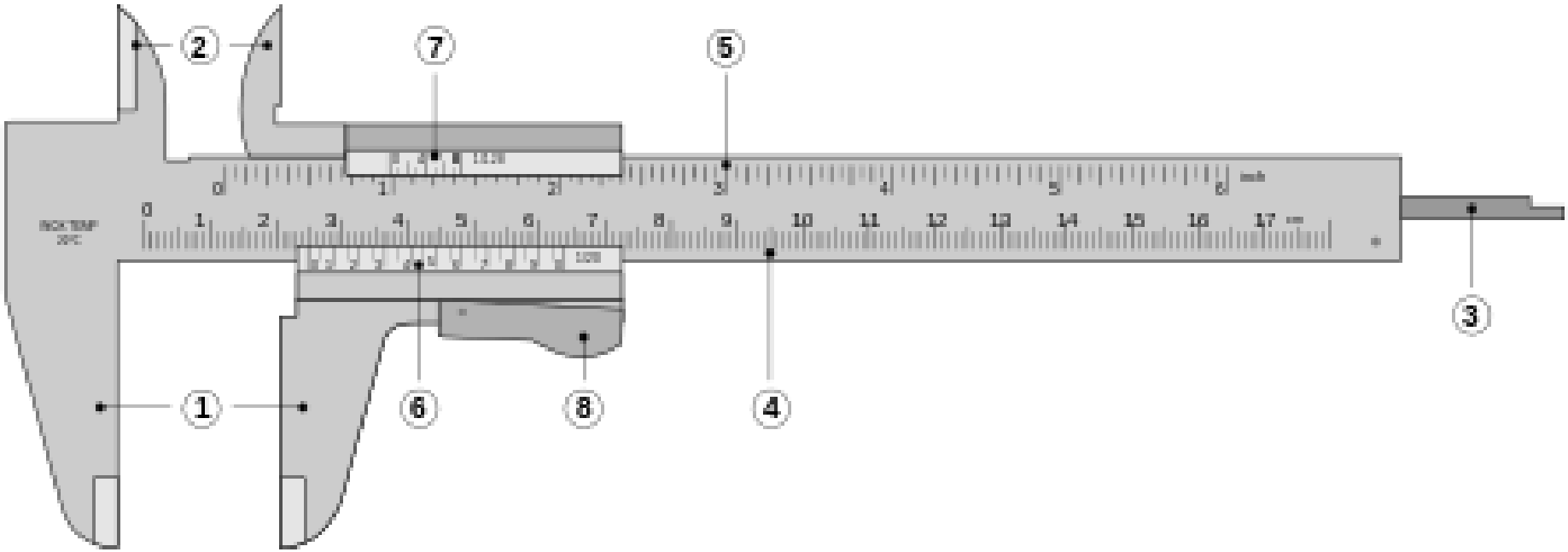
- Si fijamos $v = 2$, al mover w en sentido horario, $f(v, w)$ sube hasta el nodo 0, luego baja hasta el nodo 5, luego vuelve a subir hasta el 4 y finalmente baja por última vez.
- Resumiendo:
  - En un polígono convexo, las distancias de los vértices a un **lado(recta)** fijo **forman** una función 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.