

Juan Diego Balanta

Parcial 2

42 + 15

Punto 1:

Tener en cuenta que si D es menor ag el radar no podra alcanzar la isla incluso si esta en su coordenada x , lo que daria como respuesta que no se pueden colocar radares que alcancen a TODAS las islas

Entrada: $P[0..N][0..2]$, $N \geq 0$, con coordenadas enteras de las islas y $D > 0$

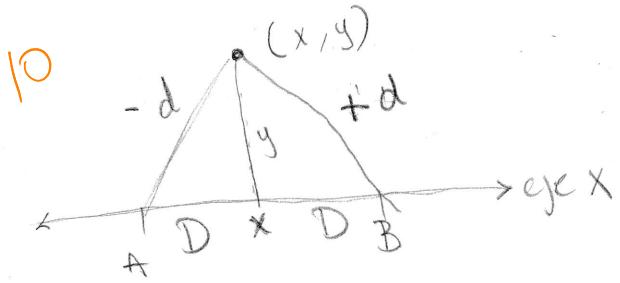
Salida: Minima cantidad de radares con radio de cubrimiento D que se deben instalar sobre el borde costero para que se cubran todas las islas en $P[0..N][0..2]$

punto a): Transformar la entrada para un algoritmo de mínimo cubrimiento de intervalos.

Respuesta:

Debemos transformar el problema a una sola dimensión la cual me va a dictar los intervalos necesarios para el minimo cubrimiento de intervalos.

Así lo transformo con respecto a x ya que es el eje en donde los radares son colocados. entonces de debe proyectar por medio de una transformación de Pitágoras esta distancia de cubrimiento. Donde para cada isla tengo:

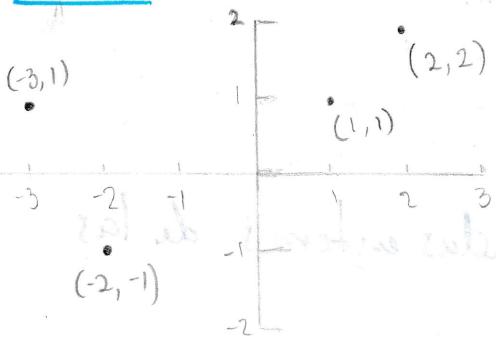


La transformación de (A, B) son los intervalos a la izquierda y derecha de x donde se encuentra la isla, utilizando la parte positiva de la operación $d = \sqrt{D^2 - y^2}$ donde D es el $D > 0$ de la entrada y "y" es el valor o coordenada y de la isla.

Entonces para lograr A y B se debe restar y sumar respectivamente:
 $D^2 - y^2$ debe ser positivo. $\begin{cases} A = x - \sqrt{D^2 - y^2} \\ B = x + \sqrt{D^2 - y^2} \end{cases}$ → intervalo (A, B) en x . donde esta en forma de minimo cubrimiento de intervalo.

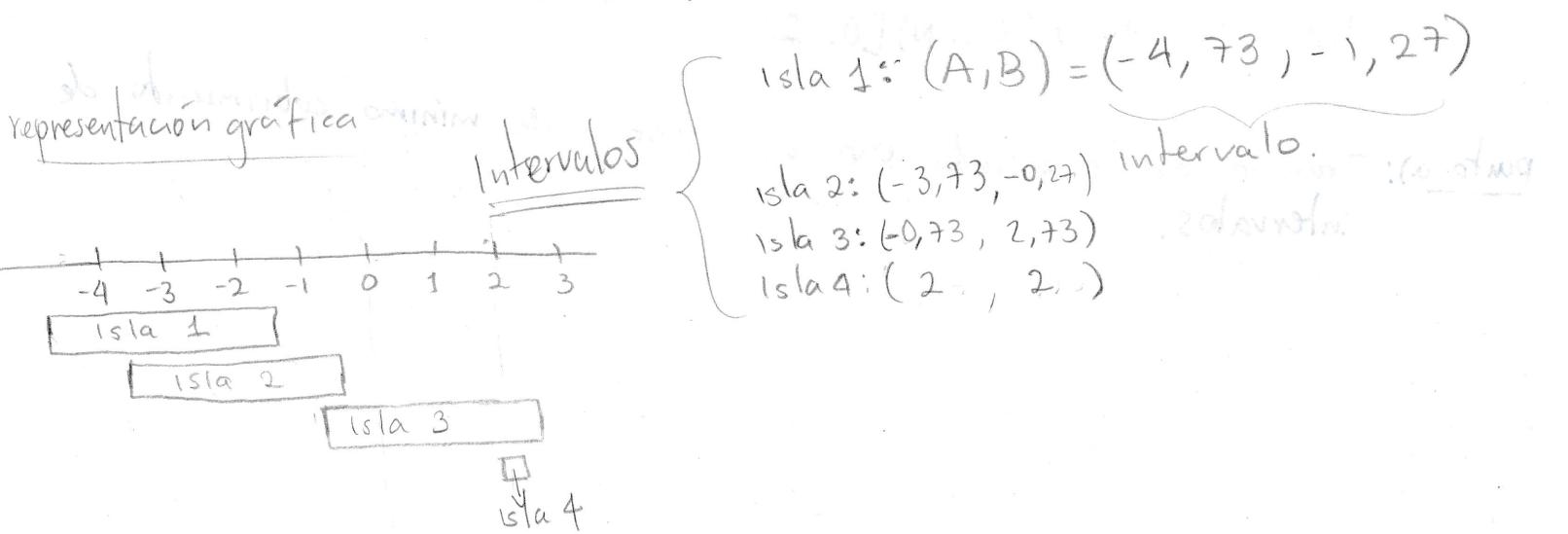
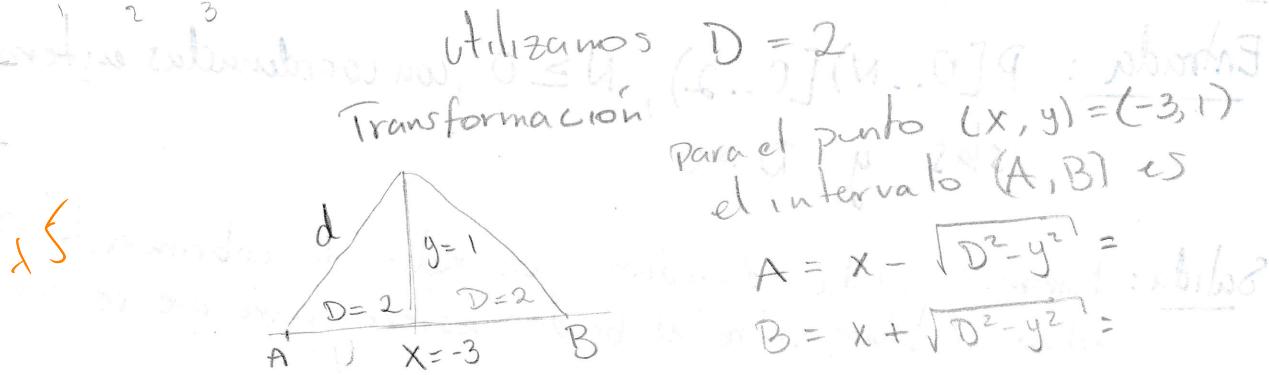
Nota: Se debe usar solo si $D^2 - y^2$ es positivo, puesto que al ser negativo no se podrían cubrir todas las islas independiente de cuantos radares se usen, ya que la isla está muy lejos.

Punto b):



isla 1: (-3, 1)
 isla 2: (-2, -1)
 isla 3: (1, 1)
 isla 4: (2, 2)

coordenadas originales



Punto c):

Donde la entrada es el arreglo P y el valor D.

Transformación $(P[0..N][0..2], D)$

for i en rango $(0..N)$

$$d \leftarrow D * D - y * y$$

if $d > 0$: # Debemos utilizar las distancias
Solo si d es positivo

$$d \leftarrow \text{sqrt}(d)$$

se procede a transformar x y en A y B

$$A \leftarrow P[i][0] - d$$

$$B \leftarrow P[i][0] + d$$

+ transformado.append((A, B))

else:

retornar -1 # esto para terminar el algoritmo o representar de que va a haber una isla a la que no se puede llegar con un radar de distancia D. Entonces la salida sería -1 o -oo representando que no tiene solución

Transformado: tiene valores numéricos y dimensiones iguales a $P[0..N][0..2]$

minimo cubrimiento de intervalos (Transformado [0..N][0..2])

1. $\text{N} \leftarrow \text{len}(\text{Transformado})$
2. Transformado se organiza de menor a mayor según el final del intervalo.
3. Transformado.sort(key=lambda x: x[1]) # siguiendo la estrategia de minimizar busco tener la mayor cantidad de islas dentro del rango del mismo radar.
4. en un ciclo donde ($n! = N$) # antes de llegar al final del arreglo.
5. best = n; n = n + 1
6. en un ciclo donde se cumpla ($n! = N \wedge A[n][0] \leq A[\text{best}][1]$)
7. $n \leftarrow n + 1$ # ir revisando hasta que se rompa el ciclo.
esto me genera todos las islas que serían cubiertas por 1 radar, se incumple cuando deje de cubrir una isla previamente cubierta por el radar.
8. En caso de que ya no se cumpla entonces aumento la cantidad de radares para las otras islas. e iterar.
 $\text{ans} \leftarrow \text{ans} + 1$
9. Al llegar al final debo de retornar la cantidad de radares utilizados
retornar ans.

Correctitud

Sea Transformado un conjunto finito de intervalos que cubren $[l..h]$. Sea Transformado un conjunto finito de intervalos que cubren $[l..h]$ donde $l \leq h$ o. sea $A = (s, e)$ en Transformado donde podemos tener que $s \leq l \leq e$ y A tiene el máximo e que pertenece al grupo o conjunto de Transformado. Esto hace que A sea parte de un cubrimiento mínimo de intervalos para $[l..h]$.

Demostación

Teniendo un mínimos \mathbf{C} Transformado un cubrimiento de intervalos mínimos para $[l..h]$ tenemos que encontrar que mínimos $\neq \emptyset$ gracias a que $l \leq h$.

Primer caso: Si $A \in \mathbf{mínimos}$, entonces confirmamos que A es parte de un cubrimiento de intervalos mínimo y mínimos no es vacío.

Segundo: Si $A \notin \mathbf{mínimos}$, debe haber un intervalo del **radar** = (s_r, e_r) que cumpla $s_r \leq l \leq e_r$.

Como $A = (s, e)$ y no pertenece, entonces A tiene que ser menor que radar entonces radar cumpliría la condición del teorema donde radar (s_r, e_r) , e_r es un máximo de Transformados, pero según el teorema entonces decimos que e_r es el mayor de todos. Entonces e_r tiene que ser menor o igual al mismo Transformados, entonces e_r tiene que ser menor o igual al mismo e , es decir $e_r \leq e$. Al ser e el máximo de todos lo hace parte de un cubrimiento de intervalos mínimos. Entonces mínimos debe incluir a A .

Como A hace parte del cubrimiento mínimo de intervalos para $[l, h]$

Para este problema se debe sortear de menor a mayor pero el algoritmo lo come de manera lineal, lo que haría el tiempo $O(N \log N)$ y espacio N .

IS

radar

Cubrimiento A

*

Cubrimiento A
x2

Punto 2

17

Entrada: un arreglo de enteros $A[0..n]$ y un $k \geq 0$

Salida: Existe una LIS de tamaño k ? $N \geq 0$

Punto a): para ser exactos: ¿Existe una subsecuencia creciente (no descendente) de tamaño k ?

Tomando el material de clase y la propuesta de erickson se procede a resolver el problema de la siguiente manera:

Instancias:

$$A = [1, 2, 3, 4] \quad k = 3$$

$$A = [1, 8, 3, 5] \quad k = 12$$

Estrategia de solución:

Transformamos el problema recursivo para utilizar por medio de 2 indices (i, j) donde $0 \leq i < j \leq N$, donde vamos a encontrar la cadena de subsecuencia más larga (LIS) donde la subsecuencia $[j..n]$ en que cada elemento es mayor a $A[i]$. y A es global

Para $0 \leq i < j \leq n$ "quién es una función recurrente en la que retorna una LIS." $\phi(i, j)$ $\uparrow i, j$??

Objetivo: $(0, 1)$ ¿Cuál es el apoyo?

Definición recurrente
Para $0 \leq n \leq N$ y $0 \leq i \leq j \leq n$

$$\phi(i, j) = \begin{cases} \phi(i, j+1) & A[i] > A[j] \\ \max(\phi(i, j+1), 1 + \phi(j, j+1)) & \text{para cualquier otro caso.} \end{cases}$$

Esta estrategia al revisar todas las soluciones tiene un $O(n)$ y n espacio.

Para saber si esto funciona, entonces tenemos que una secuencia de enteros vacía es la subsecuencia para la entrada $A = []$. También podemos ver que una subsecuencia de $A[0..N]$ se puede interpretar como una subsecuencia de $A[1..N]$ o una subsecuencia formada por el elemento $A[0] + \text{una subsecuencia del arreglo } A[1..N]$ en donde cada elemento es mayor a $A[0]$.

Ahora para definir que A' es una LIS decimos que $A[x]$ es un elemento del Arreglo, si $A[0] < A[x]$, entonces los elementos de A' que es una LIS serán mayores que $A[x]$ y $A[0]$, por ende una subsecuencia de $A[0..N]$ va a ser igual a la de $A[t..N]$ siendo esta subsecuencia A' que es LIS. De no ser el caso entonces es $A[0] + A'$ del arreglo $A[1..N]$ con elementos mayores a $A[0]$.

Finalmente como ya tenemos una LIS, podemos reconectar la solución para responder a la pregunta de si existe una subsecuencia de tamaño K . Resulta que si tenemos un sentinela provisional para la primera iteración y obtenemos el valor de LIS podemos demostrar que la cadena de tamaño K existe, ya que si $K \geq$ al tamaño de LIS entonces eso significa que el tamaño de LIS, llamado t es: $K \geq t$. Sabemos que se tuvo que haber formado una cadena de tamaño K para llegar al tamaño de t .

Función que soluciona el problema, con la entrada especificada:

LIS_K(A[0..N], K):

A.insert(0, -∞) \leftarrow sentinel artifical para la primera iteración

$i=0$.

$t \leftarrow \phi(0, 1)$

if $t \geq k$:

return True

else:

return False.

!opt.

Si sumamos los $O(n)$ de $\phi(i, j) + LIS_K$ tenemos entonces que esta solución tiene un $O(2^n)$

Para $LIS = k$ tenemos entonces implementando una operación o una comparación donde si el k del problema es mayor a t entonces la respuesta es falso, no existe ya que no van a haber cadenas o subsecuencias crecientes no descendientes que lleguen a k por lo tanto esa cadena no existe.

Si $k \leq t$ tenemos entonces que si existe ya que tenemos la igualdad $0 \leq k \leq t$ donde t al ser mayor que k significa que tuvo que armar al menos una subcadena de tamaño k , luego $k+1$ hasta t . Por lo tanto existe esa cadena creciente de tamaño k .

Punto b):

La idea de correctitud entonces se muestra como una función recurrente que calcula LIS y que puede ser extendida al problema original de una cadena tamaño K .

Para $0 \leq n \leq N$ y $\phi(i, j)$ no anterior

Teorema: Si $0 \leq n \leq N$ y $\phi(i, j)$ obtiene un valor entero entonces

$\phi(n, n+1)$

Calcula el LIS de $A[i..n]$ que extiende j .

Corolario: $\phi(0, -\infty)$ Calcula el LIS de $A[0..N]$

Como mencionamos antes el caso base es utilizar $A[]$

Caso base:

$i=0$ y $j > n$ no estaría revisando ningún elemento, la respuesta es 0. Es en definición una LIS

Caso inductivo

$\phi(i, j+1)$, si $n = i$ entonces $j+1$ es $n+1$, lo que haría el caso base de lo contrario si no lo es pasará a ser una propuesta donde se revise si $A[i]$ es mayor a $A[j]$, recursir hasta un caso base que retorna una LIS. De encontrar que $A[i]$ sea menor a $A[j]$ entonces se seguirá iterando hasta llegar al final del arreglo por ende $A[i..j]$ es una LIS, de lo contrario recursir.

Punto 3 Bonus

+10

En este caso debemos asumir que para L y R debe haber al menos un punto en P que se puede expresar como p_i ya que L[i] y R[i] son los respectivos bordes izquierdos y derechos del intervalo. De no encontrar dicho punto p_i se debe agregar para obtener al menos 1 p_i "which stabs" x en cada intervalo.

El p_i debe estar posicionado en un punto específico siendo este en un intervalo que se este sobrelapando, pero debería ser en una parte del intervalo en la que tome la mayor cantidad de intervalos sobrelapados y así tener la menor cantidad de puntos P, ya que se busca "minimizar" o optimizar el posicionamiento de estos puntos.

Una estrategia de greedy es organizar los intervalos en orden de "end points", es decir organizar L y R de menor a mayor correspondiente a R, o decir que intervalos "terminan de primero". Es decir que el orden de L depende de R para seguir con sus valores respectivos.

Luego de esto debo encontrar que intervalos me generan la mayor cantidad de conflictos para así encontrar un punto p_i que organizarlo por "end points" o cual acaba primero. entonces encuentre el primer sobrelap. Si hay un punto, seguir buscando, de lo contrario se debe colocar un punto para "apuñalar" varios intervalos sobrelapados. Se debe recurrir hasta llegar al ultimo elemento o el ultimo intervalo que no esté "apuñalado" por ningún punto p_i .

Demostración:

Si existe una solución óptima de puntos P donde no se haya agregado un punto p_i en el primer sobrelap, entonces significa que p_0 se encuentra antes o después de ese primer sobrelapamiento. Si es después entonces la solución no es valida porque hay elementos o intervalos que no están "apuñalados" si la solución tiene antes del primer solapamiento, entonces la solución no es óptima ya que estaría tomando puntos con pocos intervalos, o uno solo entonces habría p_i+1 puntos y la solución no sería óptima por ende

la solución debe tomar el primer sobrelapamiento para la solución óptima ya que se busca minimizar la cantidad de puntos.

Por eso buscar por sobrelapamientos y agregar los puntos p_i para formar la mayor cantidad de intervalos usando la menor cantidad de puntos es una solución valida para el problema.