Trabajo Práctico n.º 1

Teoría de Algoritmos 1 - 1c 2022 Trabajo Práctico 1

Alumno: Pedro Gallino

Padrón: 107587

Fecha de entrega: 05-04-2022

1. Estrategias greedy/heurísticas que no funcionan:

Para resolver el problema en cuestión, lo lógico es calcular el tramo en kilómetros que cubre cada antena de los contratos.

Con los tramos calculados, se podrán buscar las distintas distribuciones posibles para cubrir totalmente la ruta.

-Ordenar los contratos por diámetro:

Puede pensarse en ordenar los contratos por mayor diámetro (suma de los radios hacia ambas direcciones) e ir eligiéndolos hasta cubrir los K kilómetros de la ruta.

Esta heurística no funciona ya que no necesariamente los contratos de mayor diámetro cubren espacios distintos. Pueden superponerse, diferenciándose por unos pocos kilómetros. De esta forma, los elegiríamos en simultaneo sin necesidad alguna.

Contraejemplo 1:

Ruta de 100km con los siguientes contratos:

Antena,posición,radio

1,50,50 diámetro = 100km

2,90,70 diámetro = 140km

3,70,60 diámetro = 120km

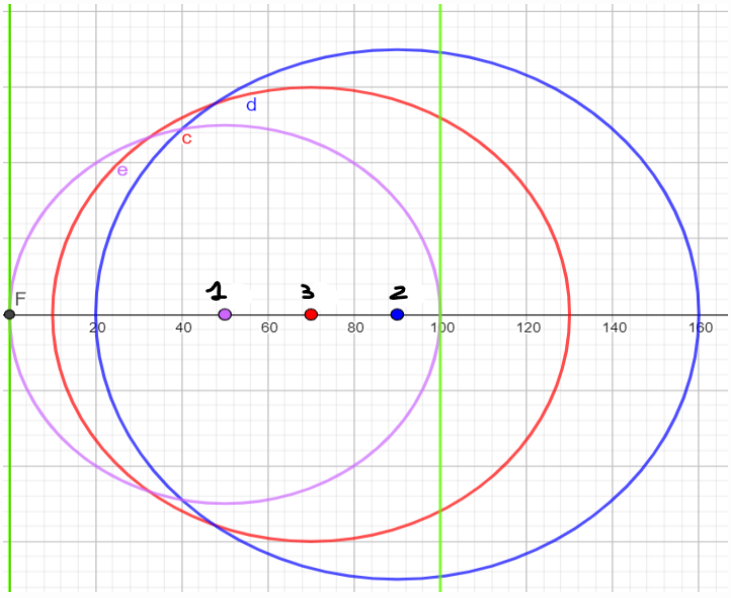

Imagen 1: Contraejemplo 1.

Si los ordenáramos en base a su diámetro, de mayor a menor, obtendríamos [2,3,1].

Colocando la antena 2, quedaría cubierta toda la ruta menos los primeros 20km.

Como aún no cubrió toda la ruta, se coloca una nueva antena, la 3. A pesar de tener un diámetro muy grande, solo cubre 10km de los 20 necesarios. Quedando los primeros 10km de la ruta sin cubrir.

Finalmente, se selecciona la antena 1. Al estar situada en el km 50 y tener un radio de 50, cubre los 10km faltantes.

Con esta heurística, se utilizaron 3 antenas distintas. Se puede ver fácilmente, que, si la ruta tiene 100km, colocando la antena 1 desde el primer momento, la ruta quedaría totalmente cubierta. Por lo tanto, no es óptimo el algoritmo.

Es una estrategia greedy ya que se puede subdividir en problemas resolubles mediante elecciones greedy que en conjunto componen la solución global.

En cada iteración tomamos la antena de mayor diámetro (elección greedy) y verificamos si queda ruta sin cubrir.

-Ordenar los contratos por comienzo de tramo en km y concatenarlos:

También podría proponerse ordenar los tramos que cubre cada contrato por su kilometro de inicio e ir concatenándolos entre sí, de modo que se elija siempre el que finalice más lejos. Esta heurística no es óptima, ya que pueden seleccionarse antenas de más al ir colocándose en orden.

Contraejemplo:

Ruta de 140km.

Antena,posición,radio

1,44,50

2,80,80

3,100,80

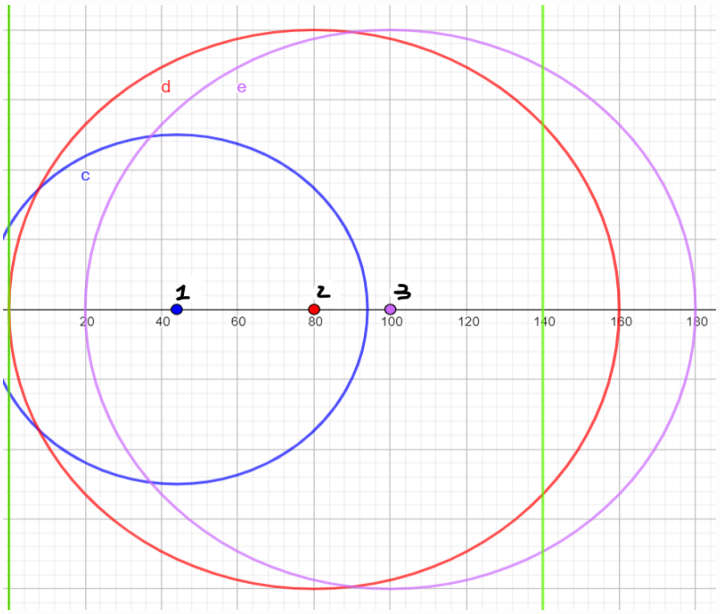

Imagen 2: Contraejemplo 2.

Al ordenarlos por el comienzo del tramo que cubren obtendríamos [1,2,3].

Tramo 1 = [-6;94]km

Tramo 2 = [0;160]km

Tramo 3 = [20,180]km

Según el algoritmo greedy primero seleccionaríamos la antena cuyo comienzo es el “más cercano”. Colocamos la 1, que cubre los primeros 94 km.

En la siguiente iteración debo tomar el próximo tramo que se superponga con el seleccionado anteriormente, que finalice lo más lejos posible. (elección greedy)

Ambos tramos, 2 y 3, se superponen con el tramo 1, pero el tramo 3 finaliza en el kilómetro 180 mientras que el 2 finaliza en el kilómetro 160.

Bajo este razonamiento tomaría el tramo 3, que terminaría de cubrir la ruta de 140km.

Nuevamente, como en el caso 1, se puede ver una solución con menos antenas en uso. Si simplemente hubiese tomado la antena 2, cubriría la ruta utilizando solo una antena, en vez de dos. La heurística no es óptima.

2. Algoritmo greedy óptimo.

A raíz esta última heurística incorrecta, puede proponerse una similar, que con mejoras termina cumpliendo con lo pedido.

Nuevamente se calculan los tramos cubiertos por cada antena y se ordenan de menor a mayor por kilómetro de inicio. Durante todo el proceso, se tendrá una variable “km_actual”, que comenzará en 0. La idea principal es ir seleccionando los tramos de manera inteligente y concatenarlos linealmente.

Esta selección inteligente sería la elección greedy de cada subproblema. El algoritmo itera de menor a mayor tomando los contratos que inicien antes del “km_actual” en adelante. Al comienzo, tomará todos los contratos cuyo tramo inicie antes del km = 0 o en el km = 0. Una vez recorridos todos los tramos posibles, se selecciona al de mayor finalización en km (el que termina lo más alejado del “km_actual”).

Una vez seleccionado el contrato, el “km_actual” se actualiza a la posición de finalización del tramo del contrato seleccionado + un kilómetro (ya que el kilómetro anterior ya estaría cubierto). Luego, continua la iteración sobre los contratos siguientes.

Este proceso se repite hasta que se cubra toda la ruta.

En caso de que se acaben los contratos y no se haya cubierto toda la ruta, quiere decir que no son suficientes. La última antena disponible no tiene el alcance necesario para cubrir el final de la ruta.

También puede darse que entre dos contratos quede ruta sin cubrir. Mediante el algoritmo, esta situación puede identificarse y terminar el proceso. Dentro de estos casos se contempla que la primera antena no llegue a cubrir el comienzo de la ruta.

Ejemplo a modo de visualizar el algoritmo: Ruta 500km.

```
1,44,50

2,402,150

3,219,35

4,100,80

5,80,80

6,300,160

7,150,30
```

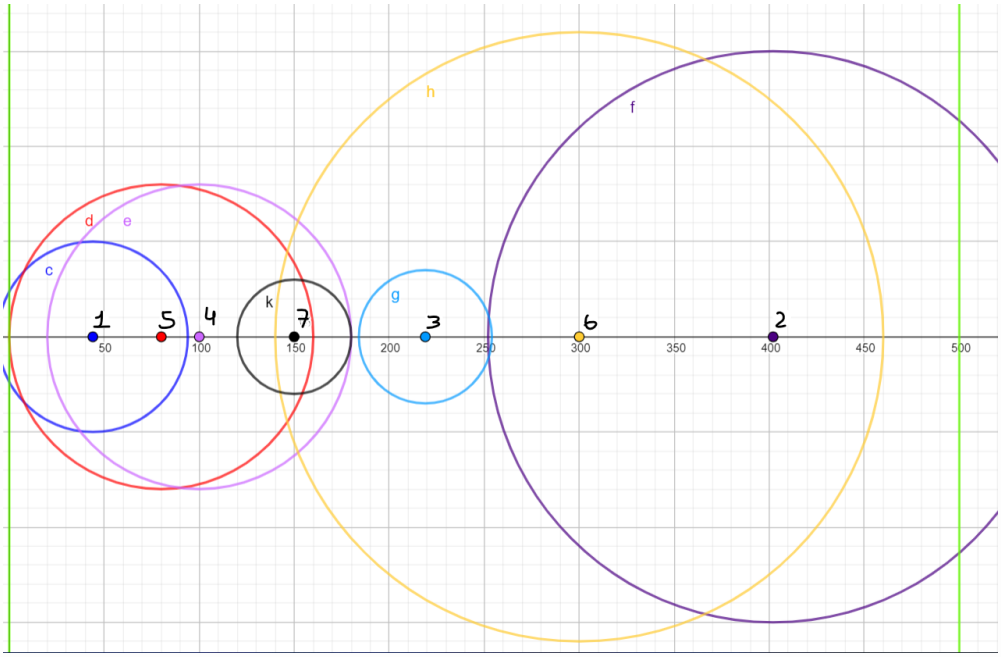

Imagen 3: Ejemplo óptimo.

Paso 1: Calcular los tramos de cada contrato y ordenarlos por inicio de menor a mayor.

Ordenados = [1,5,4,7,6,3,2]

Paso 2: Tomar los menores que contengan al “km actual” = 0km.

Contrato 1 = [-6;94]km y contrato 5 = [0;160]km.

Paso 3: Seleccionar el tramo de finalización “mayor” y actualizar “km actual”.

Coloca el contrato 5. Y actualiza “km actual” a 161km.

Paso 4: Si aún no cubrió la ruta en su totalidad, continúa iterando. (Similar a volver al paso 2)

Contrato 4 = [20;180]km, contrato 7 = [120;180]km y contrato 6 = [140;460]km

Selecciona el contrato 6 y actualiza “km actual” a 461km.

Aún faltan cubrir 39km por lo que continúa iterando desde el último de los contratos evaluados. (Que inicien antes que el 461km).

Contrato 3 = [184; 254]km y contrato 2 = [252;552]km.

Selecciona el contrato 2 y como cubre los kilómetros faltantes, termina el algoritmo.

El resultado final es [5,6,2].

Es un algoritmo greedy porque dividide el problema en subproblemas pequeños, que en conjunto nos llevan a la solución global y los resuelve con una estrategia greedy. Estos subproblemas constan de elegir el mejor contrato entre los posibles dependiendo del km_actual. Elegir el mejor contrato, es justamente, la elección greedy. Toma el contrato de finalización más lejana, sin dejar tramos de ruta descubiertos.

Optimalidad:

El algoritmo propuesto es óptimo ya que para toda instancia nos provee la solución óptima. La razón principal de esto es la elección greedy de tomar el tramo que inicie antes que el “km_actual” con el final de tramo más distante. De esta forma, se asegura de cubrir desde el inicio, cada kilómetro de ruta.

Por la naturaleza del algoritmo es imposible utilizar antenas de más, ya que se asegura de tomar únicamente la que finaliza más lejos, cubriendo así, la mayor superficie de ruta posible.

Al tomar las antenas que cubren el mayor tramo, yendo linealmente desde el km 0, inevitablemente se está colocando la menor cantidad de antenas posibles.

En caso de que no ser posible la cobertura de la ruta en su totalidad, el algoritmo lo detectaría fácilmente. Basta con verificar que el tramo cubierto por la antena más próxima, no está en contacto con el “km_actual”. Por lo que los kilómetros que los separan serían imposibles de cubrir. De esta forma, el algoritmo siempre informaría la insuficiencia de antenas, para la ruta en cuestión.

3. Pseudocódigo y estructuras de datos.

```
Sea C conjunto de contratos con sus tramos (inicio y final)
Ordenar C por inicio de tramo
Km_actual = 0
Seleccionados = []
Por cada Contrato:
    Si km inicio <= Km_actual:
            Si es el que finaliza más lejos:
                    Posible contrato a colocar = contrato.
    Si con el posible contrato a colocar alcanza para cubrir lo que quede de ruta:
            Retorno los contratos seleccionados.
    Si es el último contrato o el siguiente genera un tramo sin cubrir:
            Printear “no es posible”
            Retornar None
```

```
    Si el siguiente contrato comienza luego de km_actual:
        Agrego a seleccionados el posible contrato a colocar.
        Km_actual = fin_contrato_a_colocar + 1
Retornar seleccionados
```

Para el mismo solo es necesario utilizar arreglos y variables para guardar información provisoria.

4. Complejidad temporal y espacial

El ordenamiento de los contratos (de menor a mayor) según el inicio de sus tramos es implementado con MergeSort. Este algoritmo de ordenamiento tiene una complejidad temporal de O(nlog(n)) siendo n los elementos del arreglo a ordenar. En este caso serían los contratos.

Luego, en el peor de los casos, se iteran los n contratos, siendo la complejidad temporal O(n).

Por lo tanto, la complejidad temporal total es O(nlog(n)).

La complejidad espacial es O(n). En el peor de los casos, los seleccionados serían los n contratos, por lo que necesitaría almacenarlos todos en un arreglo para retornarlos.

5. Programe su algoritmo:

Se adjunta código fuente con el informe.

6. Analice el resultado que obtiene mediante su algoritmo:

Observando el algoritmo, puede notarse que en el caso de que dos antenas candidatas a ser colocadas, cuyo fin de cobertura sea sobre el mismo kilómetro, siempre va a quedar colocada la que tenga un inicio de tramo en un kilómetro más chico. Esto puede revertirse, al poner “>=” en vez de “>” en la comparación de antenas, beneficiando a las antenas con inicio de tramo más grande.

También es posible que no sea necesario elegir siempre el contrato con la mayor distancia al “km_actual”, puede haber más de una opción de contratos. Si tuviese una ruta de 10km y tres contratos cuyos tramos de cobertura son: Tramo 1 = [0;10], Tramo 2 = [-5;15], Tramo 3 = [-3;17]. Puede verse que cualquiera de los tres tramos cumple lo pedido, pero el algoritmo elegirá el tramo 2, por ser el primero en la iteración.

7. ¿Su programa mantiene la complejidad espacial y temporal de su algoritmo?

Sí, en mi programa aplico MergeSort para ordenar los tramos de ruta, el cual es O(nlog(n)) siendo n la cantidad de contratos. Luego itero un arreglo lo cual es lineal, O(n). Además todas las operaciones durante la iteración son O(1). (comparaciones, sumas, condicionales, etc)

Programándolo en Python no tuve inconvenientes para pasar del algoritmo al código.