Índice general

Proyecto de
Diseño y Análisis de Algoritmos

Belsai Arango Hernández
Daniela Rodríguez Cepero

Universidad de La Habana,
San Lázaro y L. Plaza de la Revolución, La Habana, Cuba
https://github.com/Daroce1012/DAA-problems/tree/Tito_el_corrupto
http://www.uh.cu

1. Problema

Tito el corrupto

Tito se dió cuenta de que la carrera de computación estaba acabando con él y un día decidió darle un cambio radical a su vida. Comenzó a estudiar Ingeniería Industrial. Luego de unos años de fiesta, logró finalmente conseguir su título de ingeniero. Luego de otros tantos años ejerciendo sus estudios (?), consiguió ponerse a la cabeza de un gran proyecto de construcción de carreteras.

La zona en la que debe trabajar tiene n ciudades con m posibles carreteras a construir entre ellas. Cada ciudad que sea incluida en el proyecto aportará a_i dólares al proyecto, mientras que cada carretera tiene un costo de w_i dólares. Si una carretera se incluye en el proyecto, las ciudades unidas por esta también deben incluirse.

El problema estaría en que Tito quiere utilizar una de las habilidades que aprendió en sus años de estudio, la de la malversación de fondos. Todo el dinero necesario para el proyecto que no sea un aporte de alguna ciudad, lo proveería el país y pasaría por manos de Tito. El dinero aportado por las ciudades no pasaría por sus manos. Tito quiere maximizar la cantidad de dinero que pasa por él, para poder hacer su magia. Ayude a Tito a seleccionar el conjunto de carreteras a incluir en el proyecto para lograr su objetivo

Resumen Se tienen n ciudades y m posibles carreteras a construir, cada ciudad que sea incluida en el proyecto aportará a_i dólares y la construcción de la carretera cuesta w_i dólares. Todo el dinero necesario para el proyecto que no sea un aporte de alguna ciudad, lo proveería el país y entonces pasaría a manos de Tito. Se quiere seleccionar un conjunto de carreteras a incluir en el proyecto que maximice la cantidad de dinero que pasa por Tito.

2. Solución

Se desea conocer la longitud del menor recorrido para visitar las $n+1$ ciudades iniciando por la ciudad k. Se tienen n ciudades con coordenadas $(x, 0)$ que se ordenan de forma creciente. La ciudad p tiene coordenadas (x, y) con $y \neq 0$.

Si el recorrido comienza en la ciudad p el menor de ellos tendrá una longitud igual a la distancia entre las dos ciudades más alejadas que se encuentran sobre el eje x, más la menor distancia de estas con la ciudad p.

Cuando comienza en alguna de las ciudades con coordenadas $(x, 0)$ se determina, primero, por cada ciudad i comenzando en $i = k$, el camino de menor longitud entre:

- A partir de k visitar todas las ciudades a la izquierda, para luego regresar hasta la ciudad i y visitar la ciudad p.
- A partir de k visitar todas las ciudades hasta i, para luego regresar a la ciudad más a la izquierda y visitar la ciudad p.

Las $n - i$ ciudades restantes se resuelven como el caso inicial en que se comenzaba por p terminando en n.

Luego del primer análisis la menor longitud obtenida se compara con el resultado de verificar el proceso inverso. Equivalente a revertir el orden en que se encuentran las ciudades y aplicar el mismo algoritmo.

Por cada ciudad i desde la primera hasta la $k-$ésima se escoge el camino que menor longitud genera entre:

- A partir de k visitar todas las ciudades de la izquierda hasta i, para luego regresar hasta la del extremo derecho y visitar la ciudad p; resolviendo las $i - 1$ primeras como el caso inicial en que se comienza en p y termina en la primera o en la $i-$ésima - 1 ciudad.

- A partir de k visitar todas las ciudades hasta llegar a la del extremo derecho, para luego regresar hasta la $i-$ésima y visitar la ciudad p; resolviendo las $i - 1$ primeras como el caso inicial en que se comienza en p y termina en la primera o en la $i-$ésima - 1.

El resultado final es la menor longitud para recorrer todas las ciudades si se comienza en k.

3. Correctitud

Se comienza demostrando que para un recorrido que inicia en p la menor longitud está dada por la distancia entre las dos ciudades más alejadas sobre el eje x más la menor longitud de ellas con p.

Demostración:
Supongamos que existe un recorrido que comenzando en p y visitando como primera ciudad una que no está en los extremos, tiene una longitud menor o igual que un recorrido que comienza en p y su primera visita es su extremo más cercano. Sean a, b, c, d las ciudades sobre el eje x. Si p visita a b entonces de visitar luego las ciudades a y d necesariamente se tiene que pasar por la ciudad b ó p nuevamente. De ser por b la arista ab ó bd se repite y por desigualdad triangular $pb + ba > pa$ y $pb + bd > pd$ por tanto de no tomar la arista pa ó pd que toma el segundo camino, consume dos aristas adicionales que su longitud total la hace mayor. Ahora, de ser por p entonces existen pb, bj y pj para visitar las ciudades de b a j, que el otro camino resuelve solo con bj, por tanto la longitud es mayor. Luego no existe un camino óptimo para recorrer las $n + 1$ ciudades empezando por k, que no sea visitando primero al extremo más cercano.

Falta demostrar que luego de visitar el extremo más cercano, la menor longitud se obtiene visitando en esa misma dirección hasta el otro extremo. Si existe un camino que no emplea una arista ij que va desde una ciudad i a una j sobre el eje x entonces tienen que estar las aristas ip y pj, pero por desigualdad triangular $ip + pj > ij$, por tanto tiene una longitud mayor.
Queda demostrado que un recorrido que inicia en p si primero visita a su extremo más cercano y termina el recorrido por el eje x en el otro extremo es óptimo.

Si el recorrido comienza en una ciudad con coordenadas $(x, 0)$ el algoritmo determina para cada ciudad i, la menor longitud de llegar a p visitando las i-primeras ciudades, las restantes las visita de forma óptima a partir de p.

Para encontrar la menor longitud de llegar a p visitando las i-primeras ciudades se verifica la menor longitud entre:

- A partir de k visitar todas las ciudades a la izquierda, para luego regresar hasta la ciudad i y visitar la ciudad p.
- A partir de k visitar todas las ciudades hasta i, para luego regresar a la ciudad más a la izquierda y visitar la ciudad p.

Si se tienen i ciudades conectadas y tres de ellas están conectadas dos a dos entonces existe un camino de longitud menor que las conecta.
Demostración:
Sea a, b, c, d ciudades, donde a, c, d están sobre el eje X. Existen las aristas ab, ac, bc y cd. Por desigualdad triangular $bc + cd > bd$, luego si se deshechan las aristas bc y cd y se añade bd se tiene un camino de menor longitud.

El algoritmo deshecha el caso en que a partir de k se visita a la ciudad p no habiendo visitado antes las k-primeras ciudades y a la derecha de k existiendo ciudades también sin visitar.
Si en un recorrido se decide visitar primero a p y dejar alguna de las k-primeras ciudades sin visitar entonces existen dos posibilidades: visitarlas a partir de k o visitar a la derecha y luego volver para visitarlas. Para el primer caso, se llega a que se conectan tres ciudades dos a dos (kp, pi y ik), no es óptimo. En el segundo caso, si d es la ciudad de la derecha visitada por k entonces existen las aristas kp, pd y kd, luego existen tres ciudades conectadas dos a dos, por tanto no es óptimo. Se demuestra entonces que la solución propuesta de buscar la menor longitud para a partir de la ciudad k terminar visitando a la ciudad p habiendo visitado antes las k-primeras ciudades es factible. Solo falta verificar que es óptima.

Sea una solución óptima y la solución que ofrece el algoritmo greedy, ambas desde k hasta p. Tanto el greedy como el óptimo tienen al menos las aristas de las ciudades a la izquierda de k comunes. Sea pj la arista que termina en p desde j que pertenece al camino de k a p en el algoritmo óptimo y pd la arista en el caso del algoritmo greedy. Sea ij la arista que conecta a la primera ciudad con la siguiente, en el algoritmo óptimo. Si j es igual a p entonces el algoritmo greedy la toma, $d = i$, y los caminos de k a p ambos son iguales. En el caso de ser diferentes, j es igual a la ciudad anterior, lo mismo ocurre con el greedy. Luego el óptimo visita mínimo hasta la $k-$ésima ciudad porque por desigualdad triangular la longitud desde la primera ciudad hasta la última que retrocedió más la de la arista con que llega a p es mayor que la arista que sale desde la primera ciudad hasta p, sin visitar ninguna ciudad nueva. Por tanto, mínimo visita hasta k luego visita tantas ciudades a la derecha como menor sea su distancia total a p. El camino óptimo se parece más al greedy hasta p. El greedy en p decide recorrer las restantes ciudades recurriendo como el caso base donde la ciudad de partida es p, lo cual se demostró que es óptimo. Por tanto la longitud del camino greedy es menor o igual a la del camino óptimo, entonces también es una solución óptima.

También se deben analizar los caminos con los cuales se decide tomar primero las ciudades más a la derecha. La solución inmediata es revertir las posiciones de referencia de las ciudades y analizar de la misma forma anterior. El algoritmo chequea dichos caminos sin revertir, comenzando desde la derecha a buscar la menor forma de llegar a p.

4. Complejidad Temporal

El algoritmo lee la entrada del problema e itera hasta n (cantidad de ciudades ubicadas en $X0$) para colocar las coordenadas en una arreglo a. El orden de realizar dicha operación es $O(n)$. Luego ordena las coordenadas utilizando el algoritmo de ordenación de python que tiene una complejidad temporal de $O(n \log n)$. Para colocar en la primera posición de la lista un elemento se empleó dos veces el reverse() de las listas de python lo cual es $O(n)$. Para ofrecer la solución se realizan dos recorridos, el primero desde k hasta n, y el segundo desde 1 hasta k, en cada uno se realiza una cantidad constante de operaciones matemáticas elementales $O(1)$, como $0 \leq k \leq n$, la complejidad temporal será en el orden de $O(n)$. Por regla de la suma la complejidad temporal del algoritmo es de $O(n \log n)$.