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

Tenemos que elegir un subconjunto de carreteras y un subconjunto de ciudades, de modo tal que si se elige una carretera, también se elijan todas las ciudades de los que depende este proyecto, y tenemos que maximizar la diferencia entre la suma de los costos de las carreteras elegidas(cantidad de dinero que pasa por Tito) y la suma de los aportes de las ciudades unidas a estas(cantidad de dinero que no pasa por Tito), para así maximizar la cantidad de dinero que pasa por Tito.

Es bastante inconveniente que por los carreteras obtengamos dinero y por las ciudades que están unidas a esta se pierdan ganancias, así que supongamos que obtenemos dinero del país por adelantado por las carreteras y que tenemos que devolver el dinero, si construirla no nos genera ganancias. Estas 2 formulaciones son claramente equivalentes, pero la última es más fácil de modelar.

Una forma sencilla sería crear un grafo con n+m+2 vértices, donde cada vértice representa la fuente(S), el sumidero(T), una carretera o una ciudad. Notaremos a la i-ésima carretera como W_i y la i-ésima ciudad como A_i. Luego, se agregan aristas de peso w_i entre la fuente y la i-ésima carretera, aristas de peso a_i entre la i-ésima ciudad y el receptor, como el dinero que pasa por Tito de las carreteras dependen de las ciudades a las que está unida, para representar esta restricción se agrega una arista de peso ∞ entre la carretera y las ciudades que están unidas a esta. Por último aplicar un corte mínimo sobre este grafo.

Si se corta la arista entre la fuente y la i-ésima carretera, entonces tenemos que devolver el dinero para la i-ésima carretera. Si se corta la arista entre la i-ésima ciudad y el receptor, entonces podemos valorar la construcción de las carreteras unidas a la i-ésima ciudad. Es fácil observar que para todas las dependencias entre una carretera y una ciudad, uniremos las ciudades necesarias o abandonaremos el proyecto ya que sería imposible cortar la arista de peso ∞ entre ellos.

Luego de aplicar el flujo máximo con el corte antes mencionado nos quedaríamos con una red en donde las aristas que están presentes representan las aristas no saturadas, y las que están conectadas al receptor serían las ciudades cuyo aporte es mayor que el costo de la carretera a la que está unida y por tanto la contrucción de esta no le permitiría a Tito lograr su objetivo. Entonces

el conjunto de solución sería el de las carreteras que no estén unidas a estas ciudades.

3. Correctitud

Como problema requiere que se seleccione un conjunto de carreteras y ciudades unidas a estas de tal manera que nos quede el mximo nmero de beneficios. Digamos que inicialmente tenemos todos los ingresos de los proyectos. As que nuestro ingreso total (inicialmente) es igual a $\sum_{i=1} R(w_i)$.

Ahora, como cuando elegimos una carretera, se unen las ciudades correspondientes. Supongamos que las carreteras que no hemos seleccionado estn en un conjunto P y las ciudades que nos hacen perder dinero estn en un conjunto A. Entonces, definitivamente podemos escribir algo como esto:
$\max\{\text{ingresos}\} = \sum_i R(w_i) - \sum_i R(Pi) - \sum_i C(Ai)$.

Podemos ver que la primera de las tres sumas en el lado derecho de la ecuación es constante. Ahora, si podemos minimizar las dos últimas sumas, podemos obtener los ingresos máximos.

Entonces queremos minimizar $\sum_i R(Pi) + \sum_i C(Ai)$.

Así que construimos un grafo donde cada carretera está conectado a la fuente S con capacidad $R(w_i)$ y cada ciudad está conectada al receptor T con capacidad $C(a_i)$. Y como las carreteras dependen de las ciudades que estaán unidas a esta, agregamos aristas desde las carreteras hacia las ciudades de las que depende con capacidad ∞ para garantizar que si se toma una carretera, tambin se toman las ciudades requeridas correspondientes.

Finalmente, calculamos el corte mínimo del grafo construido.

El corte sobre la red de flujo G con flujo máximo f^* es de capacidad mínima.
Demostración:

Sea el corte (S,T) formado por $S = \{s, v_1, v_2, ..., v_k\}$ y $T = \{v_{k+1}, v_{k+2}, ..., v_{n-2}, t\}$ donde todo vértice $v_i \in S$ es alcanzable por s a través de un camino en la red residual G_f y todo vértice en T serían los no alcanzables por s bajo este criterio. Ambos conjuntos son no vacíos, pues a S al menos pertenece el vértice s y a T pertenece el vértice t, pues si no perteneciera significa que existe un camino de s hacia t en G_f lo que implica que sería un camino aumentativo y sería posible incrementar el flujo utilizando dicho camino lo cual entra en contradicción con el hecho de que f^* es un flujo máximo en G. Luego, toda arista $< u, v >$ que cruza el corte (S,T) de S hacia T, donde $u \in S$ y $v \in T$ cumple que $f(< u, v >) = c(< u, v >)$, pues si existiese $< u, v >$ que cruza el corte tal que $f(< u, v >) < c(< u, v >)$ entonces la arista $< u, v >$ no estaría saturada y existiera en G_f por lo que el camino de s hacia u de aristas no saturadas, adicionando la arista $< u, v >$ forman un camino s hacia v en G_f lo cual es una contradicción pues $v \in T$.

Además toda arista $< v, u >$ que cruza el corte desde T hacia S donde $u \in S$ y $v \in T$ cumple que $f(< u, v >) = 0$, pues si existiese $< v, u >$ que cruza el corte tal que $f(< v, u >) > 0$ entonces dicha arista da origen a una arista inversa $< u, v >$ en G_f, como $u \in S$, entonces existe un camino desde s hacia u en G_f que al unirse con la arista $< u, v >$ daría lugar a un camino de s hacia v en G_f lo cual es una contradicción con el hecho de que $v \in T$. Por lo tanto todas las aristas que salen de S están saturadas y todas las que entran tienen flujo 0. Demostremos entonces que la capacidad del corte (S, T) es igual al valor del flujo máximo f^* en G. Sabemos que $|f^*| = f(S, T)$ por el *Lema* 26.4 visto en conferencia. Luego:
$|f^*| = f(S,T) = \sum_{u \in S} \sum_{v \in T} f(u,v) - \sum_{u \in S} \sum_{v \in T} f(v,u)$
Pero como demostramos anteriormente que $f(u, v) = c(u, v)$ y $f(v, u) = 0$ donde $u \in S$ y $v \in T$ entonces:
$|f^*| = f(S,T) = \sum_{u \in S} \sum_{v \in T} c(u,v) - \sum_{u \in S} \sum_{v \in T} 0 = \sum_{u \in S} \sum_{v \in T} c(u,v) = C(S,T)$
Luego la capacidad del corte (S, T) es igual al valor del flujo máximo f^* en G por lo que su capacidad es mínima ya que $|f^*|$ es una cota mínima de la capacidad de todo corte por *Corolario* 26.5 visto en conferencias en EDA.

Para hayar el flujo máximo con corte mínimo se utiliza el algoritmo Edmonds-Karp que genera una red residual en donde las aristas que están presentes representan las aristas no saturadas, y las que están conectadas al receptor serían las ciudades cuyo aporte es mayor que el costo de la carretera a la que está unida y por tanto la contrucción de esta no le permitiría a Tito lograr su objetivo. Entonces el conjunto de solución sería el de las carreteras que no estén unidas a estas ciudades.

4. Complejidad Temporal

La solución del problema utiliza el algoritmo Edmonds-Karp para obtener la red de flujo máximo con corte mínimo.

El algoritmo Edmonds-Karp se ejecuta $O(|V| * |E|)$ veces (ver demostración en I. to A. 3ra. Edicin Cap. 26) Pero sabemos además, que cada iteración toma tiempo $O(|E|)$ ya que el camino aumentativo se halla utilizando un BFS (BFS es $O(|V| + |E|)$ pero en la red de flujo, $|V|$ es $O(|E|)$). El resto de las operaciones, o sea actualizar la Red Residual y el flujo que se va calculando, se pueden hacer también en ese mismo orden de tiempo Por tanto, el tiempo de ejecución del algoritmo es $O(|V| * |E|^2)$.

Para obtener la solución luego de aplicar el algoritmo Edmonds-Karp se realiza un ciclo por la cantidad de vértices adyacentes a la fuente en la red de flujo, que en el grafo principal representa la cantidad de carreteras(aristas) por tanto du tiempo de ejecución es $O(|E|)$

Luego por cada vértice adyacente al receptor que representa las ciudades en la red de flujo se buscan sus vértices adyacentes que son las carreteras para marcarlos como carreteras no válidas a construir, es decir no forma parte de la solución. Por tanto el tiempo de ejecución es la cantidad de ciudades por la cantidad de carreteras $O(|V| * |E|)$

El último ciclo es un ciclo en la cantidad de carreteras por tanto es $O(|E|)$

También se utilizaron métodos auxiliares en la construcción del grafo y la matriz adjunta como *adylist* cuyo costo es $O(|V| * |E|)$, $converttoflow$ con costo de $O(|E|)$.

Luego por regla de la suma el costo de la solución del problema es $O(|V| * |E|^2)$.

5. Tester

```
def tester_solution(grafo, edges_to_build, cities_to_build, pos):

1     if pos == len(grafo.edges):
2         presp = Obtener_presupuesto(grafo, edges_to_build, cities_to_build)
3         return presp

4     max_pres = 0

5     for i in range(pos, len(grafo.edges)):

6         visitado1= False
7         visitado2 = False


8         pres_1 = tester_solution(grafo, edges_to_build, cities_to_build, i+1

9         edges_to_build[i] = 1

10         vert1 = grafo.edges[i].node1
11         vert2 = grafo.edges[i].node2

12         if not cities_to_build[vert1.name] == 1:
13             visitado1 = True
14             cities_to_build[vert1.name] = 1
15         if not cities_to_build[vert2.name] == 1:
16             visitado2 = True
17             cities_to_build[vert2.name] = 1

18         pres_2 = tester_solution(grafo, edges_to_build, cities_to_build, i+

19         edges_to_build[i] = 0

20         if visitado1:
21             cities_to_build[vert1.name] = 0
22         if visitado2:
23             cities_to_build[vert2.name] = 0

24         if max(pres_1, pres_2) > max_pres:
25             max_pres = max(pres_1, pres_2)

26     return max_pres
```

El método tiene como parámetro de entrada el grafo, una lista donde se encuentran las carreteras a construir que se representan a través de las aristas, una lista

con los vértices del grafo a construir que representan las ciudades y un entero que representa la posición actual.
Lineas 1-2: es la condición de parada del tester_solution, es decir si la posición actual llego al final de la lista de las aristas del grafo significa que tenemos una solución. Se calcula el presupuesto teniendo en cuenta las ciudades y las carreteras que se van a construir que están en las listas cities_to_build y edges_to_build respectivamente y devuelve ese valor.
Linea 4: se incializa la variable max_pres que representa el presupuesto máximo que se tiene hasta el momento.
Lineas 5-25: la idea es por cada arista del grafo se quiere hallar el máximo presupuesto entre el presupuesto que se obtiene al construir una arista(carretera) y el presupuesto que se obtiene al no construirla, teniendo en cuenta que construir una arista lleva implícito la construcción de los vértices(ciudades) que están conectados a ella.
Linea 8: se realiza la llamada al método para construir una solución en la que la arista actual(carretera) no forma parte de la listas de carreteras a construir, el valor que se obtiene que representa el presupuesto se guarda en la variable pres_1.
Linea 9: se marca la arista actual en la lista de carreteras a construir.
Lineas 10-11: Se guardan los vértices que están conectados a la arista actual para marcarlos como vértices(ciudades) a construir.
Linea 18: se hace el llamado para hallar el presupuesto teniendo en cuenta que la arista actual forma parte de las aristas a construir y se guarda en la varible pres_2.
Lineas 24: Se compara el máximo presupuesto entre el pres_1 y el pres_2 y el presupuesto máximo obtenido hasta el momento.
Linea 26: se retorna el presupuesto máximo.