TP 2: Recorridos y árbol generador mínimo

28 de Mayo, 2023 Algoritmos y Estructuras de Datos III

Integrante	LU	Correo electrónico
Zaid, Pablo	869/21	pablozaid2002@gmail.com
Arienti, Federico	316/21	fa.arienti@gmail.com

Facultad de Ciencias Exactas y Naturales

Universidad de Buenos Aires

Ciudad Universitaria - (Pabellón I/Planta Baja)

Intendente Güiraldes 2610 - C1428EGA

Ciudad Autónoma de Buenos Aires - Rep. Argentina

Tel/Fax: (++54 +11) 4576-3300

http://www.exactas.uba.ar

RESUMEN

En la teoría de grafos, el problema del *árbol generador mínimo*[1] —o *minimum spanning tree*, en inglés— se refiere al problema de encontrar, para un grafo conexo $G = (V,\ E)$ con función de peso $w : E \to \mathbb{R}$ asociada, un subgrafo conexo y acíclico de G —es decir, un árbol generador— que minimice la suma total del peso de sus aristas.

Existen diversos métodos para la resolución de este problema. Entre ellos, los algoritmos *golosos* de *Prim* y de *Kruskal*, que se basan en la selección de aristas de peso mínimo *seguras*[2] para la construcción de una solución.

El siguiente informe evalúa el problema de los *módems*, explicado en el próximo apartado, y lo reformula como un problema de *árbol generador mínimo* que aprovecha el invariante del algoritmo de *Kruskal*. Además, evalúa la eficiencia de la solución propuesta de manera empírica en función de la aplicación de diferentes heurísticas. En particular, *union by rank* y *path compression*[3].

Palabras clave: *árbol generador mínimo, algoritmo de Kruskal.*

Índice

[1]Ver Thomas H. Cormen; Charles E. Leiserson; Ronald L. Rivest y Clifford Stein. Introduction to algorithms. 2009. Sección 23: *Minimum Spanning Trees.*

[2]Una arista es *segura* si y sólo si, al agregarla a un subgrafo de un árbol generador, el resultado sigue siendo un subgrafo de algún árbol generador.

[3]Ver nota al pie (1). Sección 21: *Data Structures for Disjoint Sets.*

1. El problema de los módems

El problema de los *módems* que consideraremos tiene la siguiente premisa. Dado un conjunto de N oficinas

$$O := \{o_1 \dots o_N\}$$

donde, para cada $1 \leq i \leq N$, la oficina o_i se representa por su posición $(x_i,\ y_i)$ en el plano cartesiano, queremos encontrar el costo mínimo que deberemos pagar en cables UTP y de fibra óptica para conectar todas las oficinas a internet[4].

Para ello, vamos a contar con $W < N$ módems a distribuir entre las oficinas e incurriremos en un costo de U o V pesos, $0 \leq U \leq V$, respectivamente, por cada centímetro de cable UTP o de fibra óptica utilizado. Dado que los cables UTP tienen ciertas restricciones, estos se podrán utilizar si y sólo si la distancia entre dos oficinas es menor o igual a R centímetros.

1.1. Modelado como un problema de árbol generador mínimo. Dada la descripción anterior, vamos a mostrar que el problema se puede modelar como una variante del problema del árbol generador mínimo. Lo detallamos a continuación.

Sea G_{M} un grafo pesado completo cuyos vértices son el conjunto de oficinas O. Por cada par de vértices $o_i,\ o_j \in O$, definimos el costo de la arista $(o_i,\ o_j)$ como

$$w(o_i,\ o_j) = \begin{cases} d_{ij} \cdot U & \text{si } d_{ij} \leq R \\ d_{ij} \cdot V & \text{si no} \end{cases}$$

donde d_{ij} es la distancia euclideana entre ambas oficinas.

Luego, de haber al menos una oficina con un módem, G_{M} modela una solución no mínima en la que todas las oficinas están conectadas entre si con la opción más barata disponible entre un cable UTP y uno de fibra óptica.

Si $W = 1$, sigue que, para mantener a todas las oficinas conectadas y minimizar el costo empleado en los cables, basta encontrar un árbol generador mínimo de G_{M}.

Sin embargo, si $W > 1$, podemos reducir aún más el costo si, en vez de encontrar un árbol generador mínimo, encontramos un bosque generador mínimo de W componentes de G_{M}. Es decir, un bosque de W árboles que incluya todos los vértices de G_{M} y tenga peso mínimo.

Vamos a demostrar que este bosque es óptimo y que basta modificar el algoritmo de *Kruskal* para que termine en la iteración $N - W$ para encontrarlo.

1.2. Demostración de optimalidad. Supongamos, por absurdo, que un bosque generador mínimo de W componentes de G_{M} no es una solución óptima al problema de los módems. Luego, debe existir otro subgrafo $B \subseteq G_{\mathrm{M}}$ cuyo peso es el mínimo posible y que, una vez dispuestos los W módems, provee de internet a todas las oficinas.

Sin embargo, B también debe ser un bosque generador de W componentes. Esto se debe a que: si no fuera generador, entonces no estaríamos considerando todas las oficinas en nuestro modelo; y, si no fuera un bosque de W componentes, entonces podríamos reducir su peso si eliminamos suficientes aristas —el peso de toda arista es positivo en G_{M}— hasta formar uno. Notar que tampoco puede tener más componentes, ya que no habría suficientes módems para proveer de internet a cada grupo de oficinas.

En consecuencia, B es un bosque generador de W componentes conexas de G_{M} que tiene un peso menor que un bosque generador mínimo de W componentes de G_{M}. $\rightarrow\leftarrow$ $\square$

[4]Notar que una oficina tendrá acceso a internet si y sólo si tiene un módem o está conectada a una oficina con acceso a internet.

1.3. Demostración de correctitud. Vamos a demostrar primero la siguiente proposición. Dado un grafo conexo G, un subgrafo B de un árbol generador mínimo $T \subseteq G$ es un bosque generador mínimo de k componentes de G si se compone de las $n-k$ aristas de peso mínimo de T.

Demostración. Por propiedad de árboles, está claro que si B tiene $n-k$ aristas, entonces B es un bosque generador de k componentes conexas. Vamos a demostrar entonces, por el absurdo, que es mínimo.

Supongamos que existe un bosque generador B' de k componentes de G que pesa menos que B. Luego, podemos construir un árbol generador de G tomando un conjunto E de $k-1$ aristas de T que unan a las k componentes conexas diferentes en B'. Esto lo podemos hacer ya que, si tal conjunto no existierá, entonces habría, al menos, un par de vértices, ubicados en dos componentes conexas diferentes de B', para los cuales no existe un camino que los una en T. Lo que es absurdo, dado que T es un árbol generador.

Dicho esto, como estas aristas tienen, a lo sumo, el peso de las $k-1$ aristas máximas en T, sigue que $B' + E$ es un árbol generador de G con peso menor que el árbol generador mínimo T. $\rightarrow\leftarrow$ $\square$

En consecuencia, dado que el invariante del algoritmo de *Kruskal* afirma que, para la k-ésima iteración, el grafo B_k construido por el algoritmo es un subgrafo de un árbol generador mínimo de G y, en cada iteración, el algoritmo agrega una arista segura de peso mínimo a B_k, para todo $1 \leq k \leq n-1$, entonces, B_{n-k} es un subgrafo de un árbol generador mínimo de T compuesto por las $n-k$ aristas de peso mínimo de T. Luego, por su cantidad de aristas, B_{n-k} es un bosque generador mínimo de k componentes. $\square$

1.4. El algoritmo. Dicho todo esto, el siguiente pseudo-algoritmo presenta una solución al problema de los *módems*.

```
proc modems(O: sec<ℤ×ℤ>, N, W, R, U, V: ℕ₀) → ℝ×ℝ:
    a, b ← 0, 0
    E ← {(w(O[i], O[j]), O[i], O[j]) | i, j: 1 .. N}
    X ← ∅
    para i en 1 .. N - W:
        u, v ← minima arista segura en E para B = (O, X)
        X ← X ∪ {(u, v)}
        si (u, v) es de tipo UDP:
            a ← a + w(u, v)
        si no:
            b ← b + w(u, v)
    retornar a, b
```

Algoritmo 1. Pseudocódigo para el problema de los *módems*.

El mismo construye el conjunto de aristas pesadas E de G_{M} a partir del conjunto de entrada O y la función w definida en el apartado (1.1). A su vez, emplea una versión modificada del algoritmo de *Kruskal* que, además de terminar antes, acumula el costo incurrido en cada tipo de cable, en las variables a y b, a medida que agregamos aristas al bosque B. Notar que, para el algoritmo de *Kruskal*, un arista es segura si, además de mantener su invariante, conecta a dos componentes conexas diferentes del subgrafo construido hasta esa iteración.

1.5. Complejidad temporal y espacial. El algoritmo presentado en la sección anterior tiene un costo fijo asociado a la construcción del conjunto E: dado que el grafo G_{M} es completo, la complejidad de construir E, tanto temporal como espacial, será $O(m) = O(n^2)$, donde m y n refieren a la cantidad de aristas y vértices, respectivamente, del grafo.

Sin embargo, el costo asociado al algoritmo de *Kruskal* puede variar acorde a su implementación. En particular, el costo del algoritmo está atado al ordenamiento[5], por peso, del conjunto E. Dado que no tenemos garantías respecto al peso de las aristas, ordenarlas tendrá un costo en $O(m \log m) = O(m \log n)$. Sigue que, si utilizamos una estructura de conjuntos disjuntos con las heurísticas de *union by size* y *path compression*, podemos mantener la complejidad temporal de *Kruskal* en $O(m \log n)$, por un costo espacial en $O(n)$[6]. Teóricamente, esta es la mejor implementación de *Kruskal* para este caso de uso.

2. Evaluación empírica

[5]El mismo está implícito en la línea 6 de nuestro pseudocódigo.

[6]Ver nota al pie (3) y la sección 23.2 del mismo libro.