Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencias de la Computación
ICC2133 – Estructuras de Datos y Algoritmos – Sección 2
Profesores Yadrán Eterovic y Cristobal Gazali
Ayudante Jefe Trinidad Vargas
Primer Semestre 2021

Alumno: José Antonio Castro

Informe Tarea 3:
Problema del vendedor Dewey

1 Problema de arboles extendibles

Un árbol de expansión (minimum spanning tree) de un gráfico es solo un subgráfico que contiene todos los vértices y es un árbol. Un gráfico puede tener muchos árboles de expansión. Ahora supongamos que los aristas del gráfico tienen pesos o longitudes. El peso de un árbol es solo la suma de los pesos de sus aristas. Obviamente, los diferentes árboles tienen diferentes longitudes. El problema básicamente es ¿cómo encontrar el árbol de expansión de longitud mínima?

La mejor idea es encontrar alguna propiedad clave del MST que nos permita estar seguros de que algún arista es parte de él, y usar esta propiedad para construir el MST un arista a la vez. Para simplificar, asumimos que hay un árbol de expansión mínimo único. Puede hacer que ideas como esta funcionen sin esta suposición, pero se vuelve más difícil establecer sus teoremas o escribir sus algoritmos con precisión.

1.1 Algoritmo Kruscal (con las operaciones de conjuntos disjuntos)

¿Cuántas aristas tiene un árbol de expansión mínimo?
Un árbol de expansión mínimo tiene (V - 1) aristas donde V es el número de vértices en el gráfico dado.

Pasos a seguir

1) Ordenar todos los bordes en orden no decreciente de su peso.
2) Eligir el borde más pequeño. Comprobando si forma un ciclo con el árbol de expansión formado hasta ahora. Si el ciclo no está formado, incluir este borde. De lo contrario, deséchalo. Acá se utiliza el algoritmo $Union - Find$ para detectar ciclos. Por eso recomendamos leer la siguiente publicación como requisito previo. El algoritmo es un algoritmo codicioso. La elección codiciosa es elegir el borde de peso más pequeño que no cause un ciclo en el MST construido hasta ahora.
3) Repetir el paso 2 hasta que haya bordes (V-1) en el árbol de expansión.

```
kruskal(G(V, E)):
    ordenar E por costo, de menor a mayor
    foreach v ∈ V : make set(v)
    T <- ∅
    foreach (u, v) ∈ E, en el orden obtenido arriba :
        if find (u) != find (v):
            T <- T ∪ { (u, v) }
            union(u, v)
    return T
```

(pseudocódigo rescatado de clase 18 diapositiva 29)

Tengamos en cuenta que, cada vez que agrega una arista (u, v), siempre es el más pequeño que conecta la parte de T accesible desde u con el resto de G, por lo que según el lema debe ser parte del MST.

Complejidad
V es número de vértices y E es número de aristas.

(a) Ordenar las aristas toma $\mathcal{O}(E \times Log(E))$

(b) Luego iteramos por todas las aristas y aplicamos el algoritmo $Union-Find$ (esto puede tomar como maximo $\mathcal{O}(E \times Log(E) + E \times Log(V))$).

(c) Ahora bien el máximo valor de E puede ser $\mathcal{O}(V^2)$. Ergo $\mathcal{O}(Log(E))$ llega a ser lo mismo que $\mathcal{O}(Log(V))$

La compejidad es de a los más $\mathcal{O}(Log(V))$.

1.2 Algoritmo Prim

Al igual que el algoritmo de Kruskal, el algoritmo de Prim también es un algoritmo codicioso. Comienza con un árbol de expansión vacío. La idea es mantener dos conjuntos de vértices. El primer conjunto contiene los vértices ya incluidos en el MST, el otro conjunto contiene los vértices aún no incluidos. En cada paso, considera todos los aristas que conectan los dos conjuntos y selecciona el arista de peso mínimo de estos aristas. Después de elegir el arista, mueve el otro punto final del arista al conjunto que contiene MST.

Un grupo de aristas que conecta dos conjuntos de vértices en un gráfico se llama corte en teoría de grafos. Entonces, en cada paso del algoritmo de Prim, encontramos un corte (de dos conjuntos, uno contiene los vértices ya incluidos en MST y el otro contiene el resto de los vértices), escogemos el arista de peso mínimo del corte e incluye este vértice en MST Set (el conjunto que contiene vértices ya incluidos).

La idea detrás del algoritmo de Prim es simple, un árbol de expansión significa que todos los vértices deben estar conectados. Entonces, los dos subconjuntos disjuntos (discutidos anteriormente) de vértices deben estar conectados para hacer un árbol de expansión. Y deben estar conectados con el arista de peso mínimo para convertirlo en un árbol de expansión mínimo.

Pasos a seguir

1) Creamos un conjunto $mstSet$ que realiza un seguimiento de los vértices ya incluidos en MST.

2) Asignar un valor clave a todos los vértices en el gráfico de entrada. Inicializamos todos los valores clave como infinito. Asignamos un valor de clave como 0 para el primer vértice para que se elija primero.

3) Si bien mstSet no incluye todos los vértices.

   A) Elegir un vértice u que no esté en $mstSet$ y tenga un valor de clave mínimo.

   B) Incluor u en $mstSet$.

   C) Actualizar el valor clave de todos los vértices adyacentes de u. Para actualizar los valores clave, recorra en iteración todos los vértices adyacentes. Para cada vértice adyacente v, si el peso del arista $u - v$ es menor que el valor clave anterior de v, actualice el valor clave como el peso de $u - v$.

   La idea de utilizar valores clave es elegir el arista de peso mínimo del corte. Los valores clave se utilizan solo para vértices que aún no están incluidos en MST, el valor clave para estos vértices indica los aristas de peso mínimo que los conectan al conjunto de vértices incluidos en MST.

Complejidad
V es número de vértices y E es número de aristas. Su complejidad es de $\mathcal{O}(V \times log(V) + E \times log(V)) = O(E \times log(V))$, haciendolo en el peor caso igual al agortimo Kruscal en el pero caso. Sin embargo según los papers que busqué, Prim se puede hacer más eficiente usando los Heaps de Fibonacci llegando a $\mathcal{O}(E + log(V))$.

2 Bibliografía

1) GeeksforGeeks. (2021, 19 abril). Kruskal's Minimum Spanning Tree Algorithm — Greedy Algo-2. `https://www.geeksforgeeks.org/kruskals-minimum-spanning-tree-algorithm-greedy-algo-2/`

2) GeeksforGeeks. (2020, 3 noviembre). Prim's Minimum Spanning Tree (MST) — Greedy Algo-5. `https://www.geeksforgeeks.org/prims-minimum-spanning-tree-mst-greedy-algo-5/`

3) Minimum spanning trees. (1996, 6 febrero). ics.uci.edu. `https://www.ics.uci.edu/%7Eppstein/161/960206.html`

4) Data Structures and Algorithms: Graph Algorithms. (1998). Data Structures and Algorithms: Graph Algorithms. `https://www.cs.auckland.ac.nz/software/AlgAnim/prim.html`