Algoritmos y Estructuras de Datos III

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Informe sobre el problema *"Modems"*

Integrante	LU	Correo electrónico
Fabian Gómez Laguna	522/20	fabiangl2011@gmail.com
Nicolás Sergio Aquino	290/20	aquino_nicolas@hotmail.com

Reservado para la cátedra

Instancia	Docente	Nota
Primera entrega		
Segunda entrega		

1. Presentación y descripción del problema

El problema sobre el que se basará el siguiente informe (Ejercicio 3 del 2° Trabajo Práctico de la materia: *problema de modems*) tiene como objetivo ubicar $\mathcal{W}$ cantidad de Modems en *'el subsuelo del pabellón de 0-inf'* de tal forma que se gaste la menor cantidad de dinero en cables UTP y de fibra óptica para conectar las N oficinas del subsuelo. Para ello se tiene en cuenta las siguientes condiciones:

§ $\mathcal{W} < N$

Es decir, se deberán comprar cables obligatoriamente para conectar algunas oficinas entre sí.

§ Pueden comprarse cables de Fibra óptica/UTP

Donde el centímetro de UTP tiene costo de $\mathcal{U}$ y el de fibra óptica, $\mathcal{V}$.

Además, tener en cuenta que los cables UTP pueden usarse solo si dos oficinas están a menos de o igual a $\mathcal{R}$ centímetros.

Si el costo por centímetro de ambos cables es el mismo, entonces se prefiere usar UTP por sobre fibra óptica (siempre y cuando la restricción de longitud R lo permita).

Siempre se cumple que $1 \leq U \leq V \leq 10$

En particular se pide saber cuál es la mínima cantidad de dinero que debe pagarse en cables para conectar todas las oficinas teniendo W modems. Para la resolución se hizo la implementación de un algoritmo eficiente (cuya explicación está en la siguiente sección) basado en el algoritmo de Kruskal visto en clase.

2. Explicación del algoritmo

Sean U, V, W, R números naturales, $d : E \to \mathbb{R}$ la distancia entre las oficinas conectadas por la arista y $G = (O, E)$ el grafo donde las oficinas son los vértices y las aristas indican las distancias entre oficinas. En particular, observemos que G es un grafo completo ya que todas las oficinas están conectadas entre sí indicando sus distancias en las aristas. Buscamos obtener los valores UTP y FIBRA tal que sus precios sean lo más pequeño posible.

En primer lugar se copian las aristas de G a *cand* para luego ordenarlas y así poder utilizar el algoritmo de Kruskal. Tanto las variables UTP como FIBRA empiezan en 0 y E_T se inicializa como vacío, donde E_T son los cables que forman parte de la solución. Posteriormente se entra a un bucle *while* hasta que no haya más componentes conexas que W, es decir hasta que sólo haya W componentes conexas, ya que en ese caso podemos colocar un módem en cada componente conexa (en cualquier oficina dentro de estas) y así conseguir abastecer a todas las oficinas.

Ya dentro del *while* primero se le asigna la arista más chica de *cand* a (u, w) para luego eliminarla de *cand* y en caso de que las oficinas u y w pertenezcan a distintas componentes conexas en $T = (O, E_T)$ (es decir que no haya un camino de cables que las conecten), entonces se agrega (u, w) a E_T (se agrega un cable entre las oficinas u y w) y se suma su costo de UTP (si su distancia es $\leq R$) o de Fibra Óptica a las variables, que seguirán siendo los mínimos ya que $d((u, w))$ era la distancia mínima entre dos oficinas de componentes conexas distintas.

Finalmente, se retorna una tupla con ambos valores. De esta forma, usando esta implementación basada en Kruskal nos aseguramos que siempre el mínimo costo sea sumado tanto a UTP como a FIBRA y al retornarlos está garantizado que sean los mínimos.

A continuación mostramos el pseudocódigo del algoritmo:

entrada: $G = (O, E)$ de n oficinas, $d : E \to \mathbb{R}$, U, V, W, R
salida: (UTP, FIBRA)

UTP, FIBRA $\leftarrow 0$
$E_T \leftarrow \emptyset$
$cand \leftarrow E$
ordenar($cand$)
while haya más de W componentes conexas **do**
  $(u, w) \leftarrow$ mín($cand$)
  $cand \leftarrow cand \setminus \{(u, w)\}$
  if u y w no pertenecen a la misma componente conexa de $T = (O, E_T)$ **then**
    $E_T \leftarrow E_T \cup \{(u, w)\}$
    if $\mathrm{d}((u, w)) \leq R$ **then**
      UTP $\leftarrow$ UTP + $\mathrm{d}((u, w)) \cdot U$
    else
      FIBRA $\leftarrow$ FIBRA + $\mathrm{d}((u, w)) \cdot V$
    end if
  end if
end while
return (UTP, FIBRA)

3. Justificación de correctitud del algoritmo

Recordemos el invariante de Kruskal: *Dado $G = (O, E)$ un grafo conexo. $T_k = (O, E_{T_k})$, es un bosque de exactamente $n - k$ componentes conexas (con $0 \leq k < n$) y es subgrafo de un árbol generador mínimo de G.*

El algoritmo utilizado es esencialmente el mismo algoritmo de Kruskal, pero en lugar de terminar al armar el árbol generador mínimo (al agregar $n - 1$ aristas a E_T), termina cuando se forma un bosque de exactamente W componentes conexas. Lo único adicional es que cada vez que se agrega una arista, se actualizan los valores de UTP o de FIBRA, manteniendo el siguiente invariante: *Para T_k, UTP y FIBRA son el gasto mínimo de cables que se necesita para conectar todas las oficinas, teniendo $n - k$ módems.*

Antes de entrar al ciclo, cada oficina es una componente conexa distinta (aún no hay ningún cable) y UTP = FIBRA = 0 son los gastos mínimos de cables con n módems. En cada iteración, si se agregó una k-ésima arista, el algoritmo de Kruskal eligió la arista (u, w) con menor peso (distancia) que unía dos componentes conexas distintas y la agregó a E_T, obteniendo el bosque $T_k = T_{k-1} \cup \{(u, w)\}$. Para la arista elegida, se incrementó el valor de UTP si $d((u, w)) \leq R$ o de FIBRA si no, manteniendo el invariante de que UTP y FIBRA indican el gasto mínimo en cables utilizando $n - k$ modems. El ciclo termina cuando se agregan $n - W$ aristas, es decir cuando T_k es un bosque de W componentes conexas y es subgrafo de un AGM de G. Al terminar el ciclo, UTP y FIBRA contienen el mínimo gasto necesario para conectar las oficinas con W modems.

4. Experimentación

4.1. ¿Cuál es la mejor implementación teoricamente de Kruskal?

La versión que vimos en la materia de la implementación de Kruskal es la que utiliza DSU (Disjoint Set Union), que es eficiente cuando tenemos grafos con aristas esparsas ($m \ll n^2$). Esta implementación tiene una complejidad de $\mathcal{O}(m \log m)$ si hay que ordenar las aristas y una complejidad de $\mathcal{O}(m\ \alpha(m))$ si ya están ordenadas (donde $\alpha()$ es una función que crece muy lentamente).

Sin embargo, existe otra implementación de Kruskal que está pensada para grafos densos, es decir cuando $m = \Theta(n^2)$. Esta implementación utiliza una matriz de adyacencia para buscar las aristas candidatas de cada nodo (que sean de costo minimo y unan dos componentes conexas distintas) en tiempo lineal en los vertices, con lo cual al necesitar $n - 1$ iteraciones para armar el árbol, el algoritmo queda con complejidad $\Theta(n^2) = \Theta(m)$, es decir es líneal en la cantidad de aristas.

Como el grafo que armamos en nuestro problema es denso (ya que cada nodo tiene una arista a los demás nodos indicando su distancia), la mejor implementación teórica es la que está pensada para grafos densos, ya que tiene complejidad $\Theta(m)$.

4.2. Experimentación entre las implementaciones

Podemos ver en la figura 1 que efectivamente, para este problema, la implementación de Kruskal hecha para grafos densos funciona mucho mejor que la implementación con DSU:

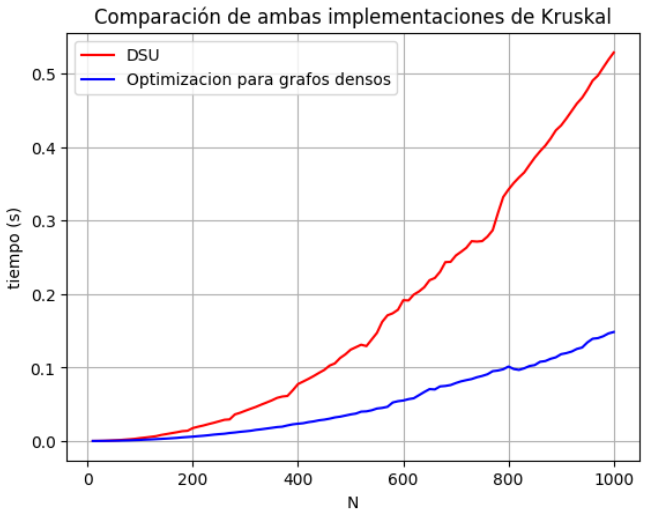

Figura 1: Experimentación de las implementaciones de Kruskal con entradas aleatorias

4.3. Experimentación con distintas DSU

Ahora surge la duda de si cambiando de alguna forma las implementaciones de DSU se puede lograr alguna mejoría (o si empeora) para ver que tanto influye la implementación de la estructura de datos.

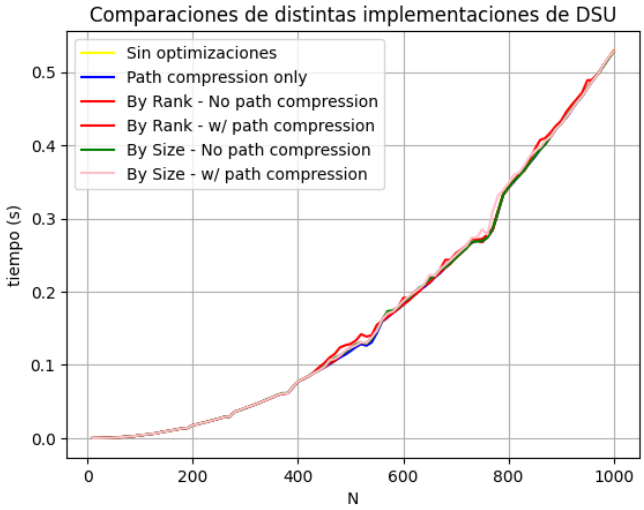

Figura 2: Experimentación con distintas DSU con entradas aleatorias

Podemos observar en la figura 2 que para una entrada aleatoria cualquiera no hay diferencias significantes en rendimiento para las distintas implementaciones de DSU. Por lo tanto, para este tipo de problemas, la forma en que está implementada DSU no impacta significativamente en el desempeño del algoritmo de Kruskal.

5. Conclusiones

Para finalizar, podemos concluir que en este caso, es decir grafo denso donde todas las distancias entre todos los vértices fueron tomados en cuenta, son más eficientes los algoritmos basados en Kruskal orientados a resolver problemas con grafos densos. Además luego de la experimentación podemos se puede observar que la implementación de la estructura de datos de DSU no generan diferencias significantes en rendimiento en entradas aleatorias, esto hace llegar a la conclusión de que la forma de implementar DSU no implica ningún cambio significativo en la eficiencia del algoritmo de Kruskal, para este tipo de problemas.

Referencias

A dense version of Kruskal's algorithm. URL: https://fedelebron.com/a-dense-version-of-kruskals-algorithm.