TP2 - Algo III - Modems

Martín Acuña - Manuel Horn

Mayo 2023

1 Problema

El problema "Modems" consiste en determinar cuál es el mínimo costo necesario para poder proveer de internet a N oficinas mediante W modems ($W < N$), cada una con una posición geográfica representada en un plano cartesiano en $\mathbb{R}^2$. Para esto tenemos los precios de los cables UTP y de fibra óptica, U y V respectivamente ($U \leq V$). También se suma la condición de que sólo podemos usar UTP entre dos oficinas cuando estas están a menos de R centímetros (usando la distancia euclídea).

2 Explicación del algoritmo

El algoritmo para poder resolver en este problema se basa en el algoritmo de Kruskal.

Primero modelamos el problema mediante un grafo pesado completo donde los vértices son las oficinas, con sus posiciones en el plano (posición geográfica), y las aristas son los costos de conectar una oficina 'a' con otra 'b' mediante un cable UTP o fibra óptica.

Para poder computar dicho grafo calculamos los costos con el criterio siguiente: "si dos oficinas están a una distancia menor a R, entonces el costo de conectar ambas va a estar dado por dist(a,b) * U; si la distancia es mayor o igual a R, el costo va a estar dado por dist(a,b) * V"

Así, usamos el algoritmo de Kruskal, terminando en N-W pasos, obteniendo N - (N-W) = W componentes conexas que vendrían a representar "clusters" de oficinas conectadas por uno de los W modems.

A medida de que se corre Kruskal, vamos guardando un costo acumulado para los cables UTP (de costo U). Luego, en cierta iteración i, con $0 \leq i \leq N - W$, el costo de la arista i y todas las de adelante corresponden a usar cables de fibra óptica, por eso ahí empezamos a sumar al costo acumulado para los cables de fibra óptica (de costo V) y nos aseguramos de que el costo acumulado

de UTP será el final.

Así, al terminar Kruskal en N-W pasos, obtenemos los costos de cables de UTP y de fibra óptica que minimizan el costo total de conectar a todas las oficinas.

3 Demostración de correctitud

Para esto nos va a ser útil recordar el invariante de Kruskal:

"Luego de la iteración 'i', tenemos un bosque generador de 'i' aristas que es mínimo entre los bosques de 'i' aristas"

De acuerdo a la inicialización de Kruskal, podemos afirmar que luego de la iteración 'i' tenemos 'N-i' componentes conexas. Así que el algoritmo termina luego de N-W iteraciones, ya que esto garantiza que tengamos W grupos de oficinas conectados por N-W cables.

Por lo tanto, esto es el bosque generador de N-W aristas que es mínimo entre los bosques de N-W aristas.

Tenemos:

$costo_{(N-W)} = costoUTP_{(N-W)} + costoFibraOptica_{(N-W)}$

Donde:

i) $B_{(N-W)}$ es el bosque generador de N-W aristas que es mínimo entre los de N-W aristas

ii) $dist : E \to \mathbb{R}$ la función de distancia (peso) de una arista

iii) $costoUTP_{(N-W)} = U * (\sum_{\{e \in E(B_{(N-W)}) : dist(e) < R\}} dist(e))$, y

$costoFibraOptica_{(N-W)} = V * (\sum_{\{e \in E(B_{(N-W)}) : dist(e) \geq R\}} dist(e))$

$(i) \Rightarrow costo_{(N-W)}$ es mínimo por el invariante de Kruskal (corresponde a $B_{(N-W)}$)

$\Rightarrow (costoUTP_{(N-W)}, costoFibraOptica_{(N-W)})$ es solución al problema.
■

4 Análisis teórico y empírico de la complejidad

En este caso, tenemos un grafo denso (completo), donde la cantidad de aristas 'm' es del orden m = $O(n^2)$, con n la cantidad de vértices.

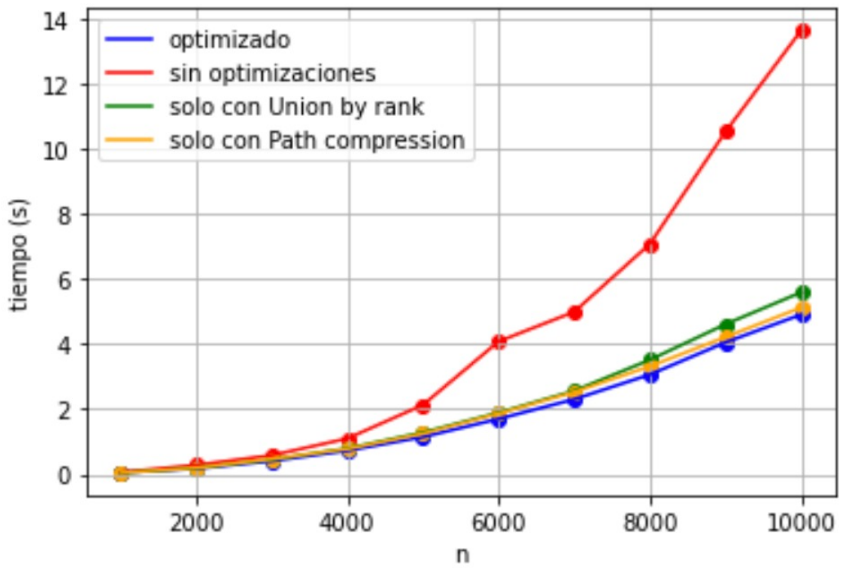
Las implementaciones del algoritmo difieren en la versión de la clase DSU (Disjoint Set Union):

i) Kruskal con DSU Naïve (no optimizado): las operaciones 'Union' y 'Find' son $O(1)$ y $O(n)$ en el peor caso, ya que Union actualiza el padre de sólo la raíz de una de las componentes a unir (esto es reacomodar un puntero), pero esto puede hacer que el bosque que representa dicha unión tenga altura O(n). En consecuencia, Find queda O(n) ya que potencialmente vamos a recorrer n vértices para encontrar la raíz que representa la componente en la que se encuentra el vértice de interés. La complejidad total queda : $O(m * n) = O(n^3)$

ii) Kruskal con DSU optimizado: en esta versión se mejoran las operaciones 'Union' y 'Find' implementando 'Union by Rank' y 'Path Compression'. Dichas implementaciones tienen un costo de O(1) para Union y $O(logn)$ para Find. La complejidad total queda: $O(m * logn) = O(n^2 logn)$

iii) También agregamos dos implementaciones que vienen de alternar el uso de 'Path Compression' y 'Union by Rank'

A continuación hacemos un análisis empírico de los tiempos de ejecución de las implementaciones.



Para hacer el análisis creamos instancias de tamaño (cantidad de oficinas/vértices del grafo) n, con n de 1000 a 10000.

5 Conclusiones

En la práctica, la mejor implementación es la optimizada con 'Path Compression' y 'Union by Rank', pero la complejidad teórica de la versión optimizada contra las que alternan las optimizaciones es igual. En cambio, la implementación sin optimizaciones tiene una complejidad asintótica mayor y en la práctica se refleja, como era de esperarse.