Algoritmos y Estructuras de Datos III

Trabajo Práctico 2

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

Modems

Informe

Integrante	LU	Correo electrónico
Braginski Maguitman, Leonel Alan	385/21	leobraginski@gmail.com
Deganis, Gaston Lucas	295/20	gastondeganis@gmail.com

Parte I

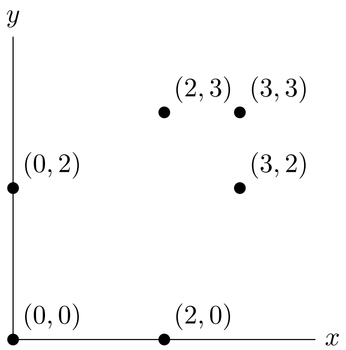
Problemática

1. Problema

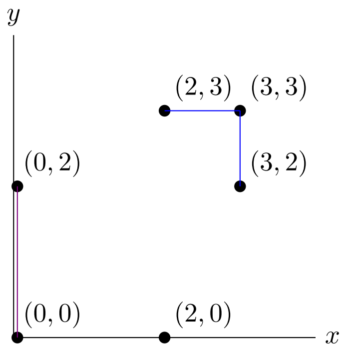
La administración del pabellón $0-\infty$ decidió contratar a estudiantes del DC para hacer la instalación de internet en el subsuelo del edificio. Sabemos que hay N oficinas y sus respectivas posiciones en el plano cartesiano (cm). Contamos con W módems, $W < N$. Las oficinas que no puedan tener un módem deben estar conectadas a otras oficinas con módems mediante cable UTP o de fibra óptica. Si usamos UTP estaremos gastando U por cada cm de cable, si usamos fibra óptica estaremos gastando V por cada cm de cable. Solo podemos usar cable UTP si la conexión entre oficinas es de menos de R cm. Nuestro objetivo es determinar cuanto es el menor costo posible para tener a todas las oficinas con conexión a internet

2. Ejemplo

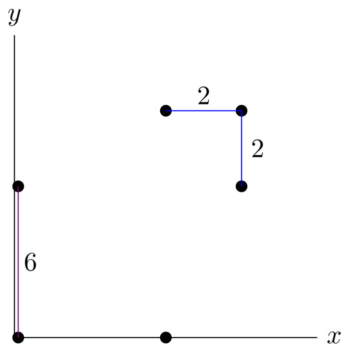
Supongamos el siguiente diagrama de oficinas:



Si contamos con $W = 3$ módems, $R = 1$ como distancia máxima para conexiones UTP y $U = 2$, $V = 3$ para los precios de los cables. Entonces nos combiene hacer las siguientes conexiones



Conexiones



Precios

La linea violeta representa un cable de fibra óptica mientras que las lineas azules son cables UTP. Como contamos con 3 módems la mejor que podemos hacer es poner un módem en (2,0), uno en (0,0) y el ultimo en (3,3). De esta forma como la distancia de las oficinas vecinas de (3,3) es de 1 podemos hacer conexiones UTP. Solo quedaría la oficina (0,2) sin módem la cual recibe su conexión por fibra óptica de (0,0)

Parte II

Algoritmo y demostración

3. Algoritmo

Podemos resolver este problema utilizando un grafo pesado. Representamos cada oficina como un vértice, todas las oficinas tienen $N-1$ aristas conectando con las demás del plano. En total contamos con N^2 aristas, vamos a ponerle de peso a cada arista el precio de instalar un cable en esa distancia.
Sabemos que este grafo pesado es conexo y denso, nuestro objetivo es quedarnos con las conexiones más baratas posibles. Teniendo esto en cuenta usamos el algoritmo de `Kruskal` para armar un árbol generador mínimo que determine estas conexiones buscadas. Luego sabiendo que contamos con W módems cortamos `Kruskal` al llegar a W componentes conexas, es decir $N - W - 1$ aristas del `AGM`.

Complejidad:
Armado de grafo de oficinas $\rightarrow O(N^2)$; Kruskal (Grafos densos)$\rightarrow O(N^2)$
Total $\rightarrow O(N^2)$

3.1. Kruskal

El algoritmo modifcado de `Kruskal` se usa para conseguir bosques de `AGM` en cualquier grafo pesado. Esto lo logra representando el grafo como un conjunto de aristas, las aristas se ordenan ascendentemente por pesos. En ese orden se crea el nuevo grafo, agregando cada arista de la lista si esta no genera un ciclo con las aristas agregadas previamente.

4. Correctitud

Recordemos la definición de AGM: Un grafo T es AGM si es un árbol generado de un grafo pesado G, de costo mínimo.
T es un árbol $\iff$ T es acíclico y tiene $N-1$ aristas.
T es de costo mínimo $\iff$ Sea S otro árbol generador de G, el costo total de sus aristas es mayor o igual al costo total de T.
Llamamos bosque generador a un grafo no conexo donde todas las componentes son árboles generadores de algún subgrafo de G

Para demostrar la correctitud de este problema queremos comprobar que los precios de las conexiones elegidas son los mínimos, es decir que en el grafo G nos quedamos con las aristas más livianas. Y que contamos con W componentes conexas.

Empecemos probando que el grafo generado T es mínimo, para esto vamos a demostrar la correctitud de `Kruskal`. Esto lo haremos haciendo inducción en el invariante de Kruskal.

$$P(i) = \text{"Tenemos un bosque generador mínimo de } i \text{ aristas que es subgrafo de algún AGM de G"}$$

Caso base: $P(0)$
Tenemos 0 aristas y N componentes conexas de 1 vértice. $\Rightarrow$ T es subgrafo de cualquier AGM de G.

Paso inductivo: Tomando como `Hipotesis Inductiva` la propiedad $P(i)$. Sea T_i el grafo en $P(i)$ vamos a probar que vale $P(i+1)$, es decir T_{i+1} cumple el invariante.
Por Kruskal sabemos que T_{i+1} toma una arista segura de costo mínimo de G, esto significa que la arista agregada en T_{i+1} no esta en T_i ni forma ciclos con las aristas de T_i. Seguimos teniendo entonces un bosque generador de $i+1$ aristas. ✓
Sea e la arista seleccionada por `Kruskal` tal que $T_i = T_{i+1} - e$ Supongamos que existe otra arista e' tal que $T_{i+1} - e + e'$ sigue siendo bosque generador de G. Sabemos que e es mínima por lo que $c(e) \leq c(e')$. Sea T'_{i+1} el bosque generador de agregar e' en vez de e. Entonces T_{i+1} tiene costo menor o igual que T'_{i+1}, con esto garantizamos que agregando e T_{i+1} es bosque generador `mínimo`. ✓

$\therefore$ El invariante es válido y el grafo T generado con `Kruskal` para G es `AGM`

Solo nos queda probar que hay W componentes conexas: Vamos a llamar T_c al `AGM` generado de `Kruskal` con N iteraciones. T_c es árbol por lo que es conexo y todas sus aristas son puentes, esto significa que $T_c - \{e_1, e_2, ..., e_k\}$ es un grafo disconexo de $k+1$ componentes. Sea T el `AGM` de `Kruskal` cortando $W-1$ iteraciones antes que T_c, este grafo es equivalente a $T_c - \{e_1, ..., e_{w-1}\}$ y por lo tanto T tiene $W - 1 + 1 = W$ componentes conexas. Aparte como estas aristas que no estamos agregando son las de costo mayor T sigue siendo mínimo.

Parte III

Demostración empírica

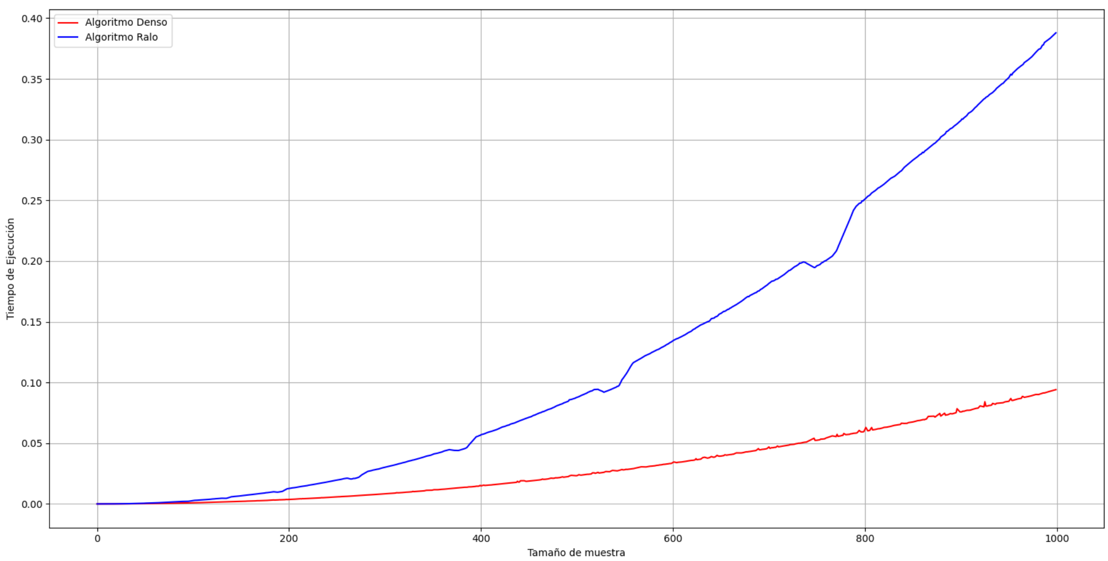
5. Implementaciones de Kruskal

El algoritmo de `Kruskal` se puede implementar en $O(MlogM)$ o en $O(N^2)$. La versión de $O(MlogM)$ se usa para grafos ralos, mientras que preferimos usar la de $O(N^2)$ para grafos densos.
Para la implementación en $O(MlogM)$ usamos la estructura `DSU`, Disjoint Set Union, que nos permite determinar a qué componente conexa pertenece cada vértice y unir componentes si estamos agregando una arista segura, esto optimiza la complejidad de `Kruskal` para ralos.

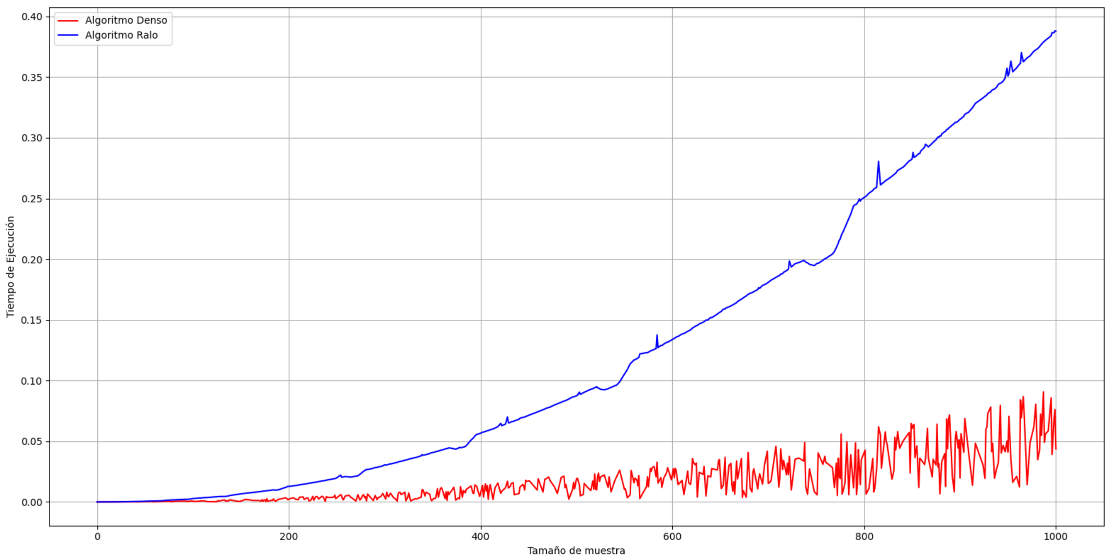
6. Experimentación

Generamos aleatoriamente 500 muestras de entradas para el problema, y comparamos los tiempos de computo con ambas implementaciones de `Kruskal` es estos 2 casos de muestras.

<u>Caso 1:</u> Tomamos $W = 1$ en todas los tests, es decir en cada instancia el algoritmo deberá generar un `AGM` completo de una componente conexa.



Caso 2: Tomamos $W < N$ aleatorio en todos los tests. El algoritmo tiene $N - W$ instancias.



7. Conclusiones

En particular para este problema como la entrada es un grafo de $M = N^2$ aristas, usar la implementación para `ralos` es $O(N^2 log N^2)$ mientras que la implementación para `densos` es $O(N^2)$. En el gráfico se ven reflejadas estas cotas y comprobamos que evidentemente es mejor usar la implementación para `densos` en este problema.

Otra observación es que hay instancias mas fáciles de resolver en la implementación de `densos`, por esto se ven picos de bajada en el gráfico. Esto se debe a que en esta versión de `Kruskal` no estamos ordenando las aristas, sino que directamente buscamos la mínima arista segura ($O(N)$) en las $N - W$ iteraciones. Esto determina que la complejidad en esta implementación termina siendo $\Theta(N * (N - W))$, y al ser W aleatorio para cada N, ocurren estas variaciones en el tiempo, cuanto más grande es W menos tiempo tarda y la complejidad es más lineal. Mientras que la implementación para ralos que sí ordena todas las aristas, en todos los casos se mantiene en $O(N^2 log N^2)$