Algoritmos y Estructuras de Datos III

Trabajo Práctico 2

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

Recorridos y árbol generador mínimo

Integrante	LU	Correo electrónico
Fernández Spandau, Luciana	368/20	fernandezspandau@gmail.com
Chumacero, Carlos Nehemias	492/20	chumacero2013@gmail.com

Reservado para la cátedra

Instancia	Docente	Nota
Primera entrega		
Segunda entrega		

Módems

El problema apunta a proveer de internet a N oficinas. Para ello, se compraron W módems, los cuales pueden instalarse en cualquier oficina, brindándole acceso a internet. Como $W < N$, se deberán comprar también cables para conectar algunas oficinas entre si. Se pueden comprar cables UTP o de fibra óptica, los cuales tienen un costo por metro de U y V , respectivamente. Los cables UTP tienen la condición particular de que solo pueden usarse entre dos oficinas si estas están a menos de R centímetros. Dadas las posiciones de las oficinas en un eje cartesiano (expresadas en centímetros), la cantidad de módems adquiridos. el precio por metro de los cables UTP y de fibra óptica, y la distancia R, debemos encontrar cuál es la mínima cantidad de dinero que debe pagarse en cables para conectar todas las oficinas. Puntualmente, queremos saber cuánto debe gastarse en cables UTP y cuánto en cables de fibra óptica.

Algoritmo

El algoritmo planteado para solucionar este problema se baso primero en guardar todas las coordenadas en un vector C.
Luego armaremos todas las aristas posibles entre oficinas, por cada arista tendremos la siguiente información:

- El *costo* de esa arista, donde $costo = distancia * precio$. La *distancia* es la calculada entre las oficinas que conecta la arista y el *precio* es del cable más conveniente, siempre vamos a preferir usar cable UTP ya que siempre es el más barato o a lo sumo tiene el mismo precio que la fibra óptica, pero hay un limite de distancia donde ya no podremos usar cable UTP, en ese caso tendremos que usar fibra óptica.

- Los identificadores de las oficinas que conecta, que son la posición de la oficina en el vector C, estos identificadores van de 0 a $n - 1$.

- El tipo de cable que se uso.

Una vez que tenemos todas las aristas posibles, vamos a aplicar el algoritmo de Kruskal pero con algunas modificaciones en la parte del chequeo de cada arista segura. Además de unir las componentes de cada oficina, tendremos que:

- Guardarnos el costo de cada tipo de cable usado.

- Tener actualizado la cantidad de componentes que tenemos.

- La parte más importante, cortar la recorrida de las aristas cuando la cantidad de componentes sea igual a la cantidad de modems disponibles.

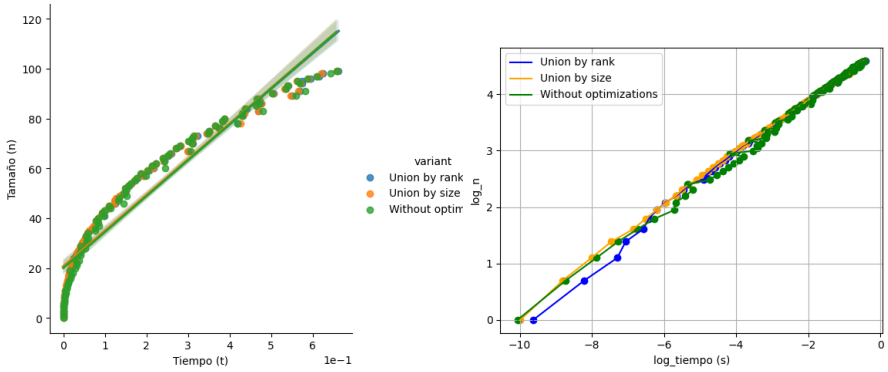
Finalmente se mostrará el costo de haber usado cable UTP y fibra óptica en el caso correspondiente.
Esto se realizará en todos los casos que se hayan ingresado.

Justificación del algoritmo

En esta sección daremos una justificación de nuestro algoritmo. Principalmente, la idea se basó en observar que al tener una cantidad limitada de W módems, no es necesario buscar el AGM, sino que al tener W módems, debemos ver la forma de poder encontrar W componentes conexas mínimas y así utilizar un módem por cada componente.
Al analizar el funcionamiento de Kruskal, pudimos notar que cuando se agrega una arista segura, la cantidad de componentes disminuye en 1. Es decir, conviene interrumpir el algoritmo de Kruskal cuando la cantidad de componentes sea igual a W, o, dicho de otra forma, cuando se hayan agregado k aristas seguras, donde $k = N - W$.
Estamos seguros de que este bosque generado es el mínimo, ya que, según el invariante de Kruskal, en el paso i, tenemos un bosque generador de i aristas que es el mínimo entre todos los bosques con i aristas. Por lo tanto, la solución es correcta.

Experimentación

Para experimentar con el algoritmo, generamos 100 conjuntos de modems aleatorios, con tamaños en un rango de 2 a 1.000. Experimentamos con las implementacions de Kruskal con DSU con rank, size y sin ninguna optimización. Obtuvimos resultados muy parecidos.



Estos resultados no nos parecieron correctos para la implementación sin optimizaciones (ni rank, ni size, ni path compression), porque en ese caso $find$ debería ser $O(n)$. Nuestra hipótesis fue que al estar generando aleatoriamente las posiciones de los módems, tal vez el árbol de DSU se estaba construyendo de forma balanceada. Entonces, decidimos "forzar" a que el árbol sea un camino.

Para ello, generamos a los conjuntos de módems de esta forma: todos los módems están en la posición 1 del eje Y y su posición en el eje X va aumentando su distancia en relación a la posición del nodo anterior.

De esta forma, el árbol de Kruskal va a elegir siempre al módem que tenga la menor distancia a otro y el árbol va a ser un camino.
Lamentablemente, los resultados que obtuvimos con esta idea, fueron los mismos que la experimentación anterior, no hubo ningún cambio significativo.