DEPARTAMENTO DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Informe Trabajo Practico 2: "Algoritmos sobre Árbol Generador Mínimo"

Análisis del Algoritmo del Árbol Generador Mínimo propuesto

26 de Mayo de 2023 — Algoritmos y Estructuras de Datos III

Integrante	LU	Correo electrónico
Jarabroviski, Tomas	325/21	jarabroviskit@gmail.com
Cravchik, Matias	720/21	cravchikm@gmail.com

Abstract

Presentaremos y haremos una descripción del problema planteado para luego explicar el algoritmo desarrollado, justificando la correctitud del mismo mediante el invariante del algoritmo seleccionado. Daremos su complejidad computacional justificada, dependiendo del caso de uso y experimentaremos en base la misma.

Keywords: Algortimos, Estructuras de Datos, Arbol Generador Mínimo, Recorridos, Pruebas empíricas, Experimentación, Complejidad, Técnicas Algorítmicas.

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2610 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (++54 +11) 4576-3300
http://www.exactas.uba.ar

1. Introducción

El problema a resolver es el siguiente:

Se quiere proveer de internet a N oficinas, representadas como puntos (x, y) en un eje cartesiano ($x, y \in \mathbb{N}$). Para esto, se cuenta con $W < N$ modems y se puede, además, instalar un cableado entre cada par de oficinas. Este cableado puede ser de UTP o de fibra óptica (FO) con un coste por centímetro de U y V respectivamente ($1 \leq U \leq V \leq 10$), aunque los cables UTP solo se podrán utilizar cuando la distancia es menor o igual a un natural R.

El objetivo será encontrar el coste por separado de cables UTP y FO a comprar de manera que se pueda proveer de internet a todas las oficinas, entendiendo que una oficina está proveída de internet cuando tiene un modem o hay un camino de cables hasta una oficina con uno. Cuando comprar un cable de UTP o FO dá lo mismo se debe elejir el de UTP.

Por ejemplo dadas tres oficinas en donde todas están a distancia 1 entre ellas, teniendo solamente un modem y con la distancia máxima para usar UTP siendo 1; la mejor solución es poner un modem en cualquiera de las 3 oficinas y unir esta con las otras dos mediante cables UTP. Este caso, al ser la distancia 1 entre las oficinas podemos utilizar cables UTP (más economicos) sin pasarnos de la restricción. De esta forma, el costo total de conectar las oficinas es 2 * 1 * (costo por unidad de UTP) ya que se necesitan 2 cables de longitud 1 de UTP. La solución sería entonces que se debe gastar 2 * U en UTP y 0 en FO.

2. Explicación del algortimo desarrollado

2.1. Modelado del problema

La primera parte del algoritmo consiste en ver como modelamos el problema de forma que la solución se obtenga a través de un algorítmo de árbol generador mínimo. Para esto que vamos a representar cada oficina como un vértice en un grafo. Luego por cada par de nodos distintos, calculamos la distancia entre ellos (pues sabemos su posición en un eje cartesiano) y nos fijamos si esta es menor o igual a la distancia máxima posible para utilizar los cables UTP. Si lo es, entonces agregamos una arista al grafo entre estos dos nodos guardandonos el peso de esta arista como *distancia * costo UTP*, con un identificador que nos diga que representa un cable UTP. Si la distancia es mayor a la del máximo posible para utilizar UTP, nos guardamos una arista de la misma manera pero el costo utilizado será el de fibra óptica, igualmente identificandola como tal. Observar que si el costo de UTP es igual al de la fibra óptica siempre vamos a priorizar utilizar cables UTP (mientras que se pueda) tal como indica la consigna del problema.

Por lo tanto al finalizar el modelado, nos queda un grafo de N vértices (donde N es la cantidad de oficinas) en el cual cada uno esta conectado con todos los demás mediante aristas que representan el coste más barato de conectar ambas oficinas mediante un cable según su distancia. Notar que de esta forma hay una única arista entre dos nodos cualquiera pues nunca convendrá unir dos oficinas con un cable de mayor coste al mínimo posible.

2.2. Algoritmo

Una vez que tenemos el modelado hecho, corremos Kruskal sobre este grafo y, por cada iteración del mismo, al elegir la arista menos costosa nos fijamos si es de tipo UTP o fibra optica y acumulamos el coste de estas por separado. Para finalizar, paramos el algoritmo cuando ya tenemos W comoponentes conexas, siendo W la cantidad de modems, y devolvemos los resultados por separado.

3. Justificación de Correctitud

Para justificar la correctitud del mismo primero recordemos el invariante de Kruskal: "Tenemos un bosque generador de i aristas que es mínimo entre todos los bosques de i aristas, siendo i la i-ésima iteración del ciclo del algoritmo de Kruskal". También recordemos que un bosque es un grafo sin circuitos simples, es decir que cada componente conexa de un bosque es un arbol.

Antes de empezar a correr Kruskal tenemos todos los nodos y ningúna arista, es decir, N componentes conexas (siendo N la cantidad de oficinas). En cada iteración, por el invariante, sabemos que vamos a tener un bosque pero cada vez con una arista más que la iteración anterior. Además, se puede decir que por cada iteración tenemos una componente conexa menos, pues se agrega una arista la cual irá de un arbol del bosque a otro (si fuera una arista entre dos vértices de un árbol este dejaría de serlo, por el invariante siempre debe haber un bosque). Por lo tanto, una vez que tenemos W componentes conexas (pues $W < N$), siendo W la cantidad de modems, paramos el algortimo de Kruskal ya que logramos tener W componentes conexas en donde cada una de ellas es un arbol y de costo mínimo. Entonces, cada modem iría en un vértice cualquiera de cada componente conexa ya que debe haber al menos uno en cada una (pues sino no habria internet y poner un modem no aumenta el costo), y de esta forma obtenemos lo que buscamos: que todos las oficinas estén conectadas a internet mediante un modem de forma directa o mediante cables que la conectan a una oficina con modem, gastando la menor cantidad posble.

4. Experimentación

4.1. Análisis teórico de las implementaciones de Kruskal

Para implementar el algoritmo Kruskal tenemos dos opciones: una versión para grafos densos y otra para ralos, es decir, una para grafos donde hay muchas aristas y otra para grafos con pocas respectivamente; entendiendo como "muchas" cuando superan la cantidad de vértices. La implementación de la versión rala tiene una complejidad que se puede acotar en O(M * N) al utilizar una estructura de DSU sin incluir optimizaciones y por O(M log(N)) cuando sí se incluyen, balanceando los árboles (siendo M la cantidad de aristas y N la cantidad de vertices), mientras que la implementación para grafos densos tiene complejidad O(N^2).

En nuestro caso, dada la forma en la que modelamos el problema, vamos a tener siempre la máxima cantidad de aristas que puede tener un grafo, pues cada vértice representa una oficina y este tendra una arista a todas las otras oficinas con un peso indicando el coste de poner un cableado que las conecte. Tendremos entonces exactamente $(N * (N - 1))/2$ aristas, siendo evidentemente un grafo denso. Podemos hipotetizar, por lo tanto, que convendrá utilizar la implementación densa, pues al haber $O(N^2)$ aristas la complejidad de la implementación rala será $O(N^2 log(N))$ (o peor sin optimizar) por lo que podemos esperar que su tiempo de ejecución crezca más rápido que la versión densa (que es $O(N^2)$) con el aumento de la cantidad de oficinas. Además, esperamos que la versión rala mejore al incluir en la estructura de DSU las optimizaciones de *Union By Rank* y *Path Compression*.

Para probar nuestra hipotesis, generamos casos de tests duplicando aproximadamente cada vez la cantidad de oficinas y randomizando la ubicacion de cada una, manteniendo un densidad más o menos similar. Luego corrimos diez veces cada test midiendo el tiempo que tardan en resolverlo las distintas implementaciones, y tomamos el promedio. Estos fueron los resultados:

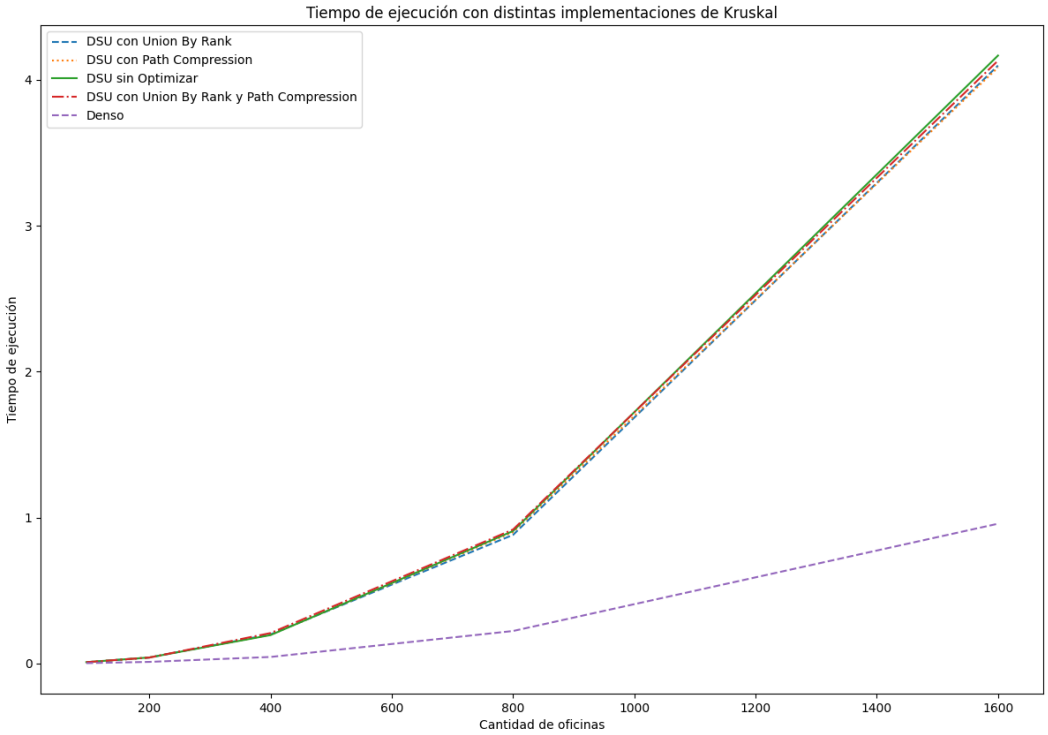

Figura 1: Tiempo de ejecucion (en segundos) por cantidad de oficinas de las distintas implementaciones de Kruskal

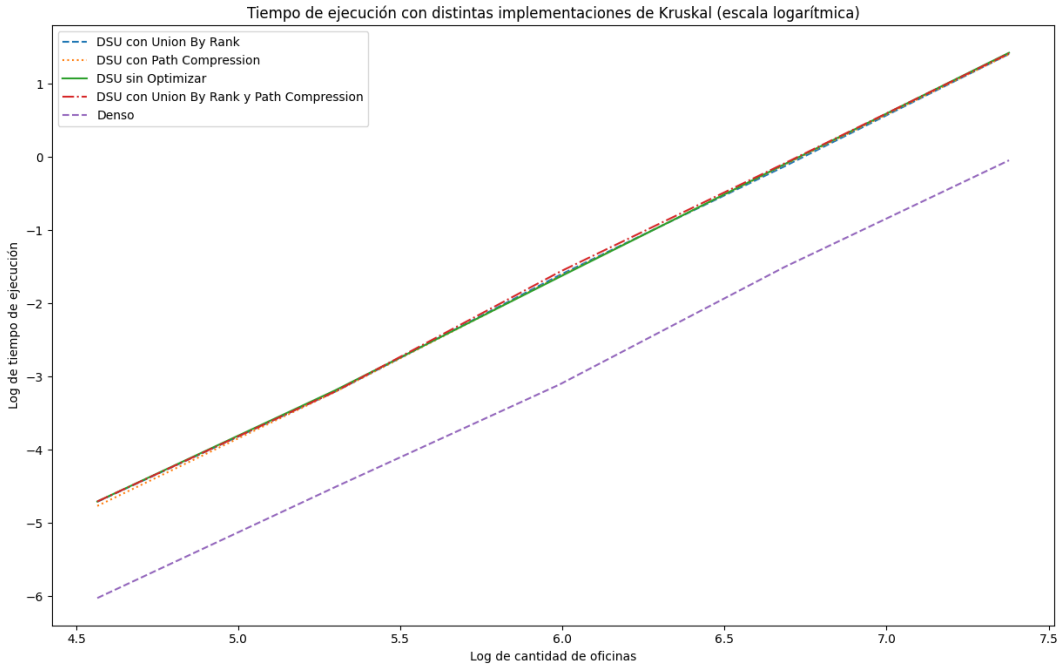

Figura 2: Tiempo de ejecucion (en segundos) por cantidad de oficinas de las distintas implementaciones de Kruskal es escala logarítmica

Como se puede observar, la implementación densa, efectivamente, es notablemente más rápida que la rala, con o sin sus optimizaciones. Sorprendentemente, no hay una diferencia apreciable entre las distintas implementaciones de DSU. Esto puede explicarse por la aleatoriedad de la muestra: los árboles del bosque tenderán a crecer en altura en una distribución normal, por lo cual se puede esperar que el find tome en promedio log(n) operaciones, acercandose a sus versiones optimizadas. Esto puede variar en muestras en las que pocos arboles del bosque crezcan con más frecuencia que el resto. Para revisar esto, realizamos algunas mediciones con un caso de test con un solo modem y en el cual el AGM del grafo será un camino de aristas crecientes en peso, de manera que sea un único árbol el que crezca en altura empeorando el tiempo de los siguientes find:

Cantidad de oficinas	DSU sin optimizar	DSU con Path Compression	DSU con Union By Rank	DSU con ambas
5000	31.954023104	24.291190018	24.44727271	24.148930819
10000	125.364185584	120.424494769	118.439933471	114.886296522

En este caso, es evidente que sí se aprecia una mejora con las optimizaciones del DSU.