Relatório 2º projecto ASA 2022/2023

Grupo: TP021

Aluno(s): Henrique Costa (103663)

Descrição do Problema e da Solução

O problema apresentado tem como objetivo, dado um grafo G = (V, E) calcular o valor máximo de trocas comerciais, minimizando os custos infraestrutura (número de troços).

Primeiro são pedidos o números de vértices e de arcos do grafo. Consequentemente utilizou-se um ciclo for para obter todos os arcos que formam o grafo. Para armazenar os arcos e adicionar os arcos a um vetor, utilizou-se uma struct Graph. Esta struct também tem a função que iremos utilizar após o processamento dos arcos, chamada computeKruskal. Como disse na linha anterior o algoritmo escolhido foi então o de kruskal, para isso foram implementadas funções adicionais para auxiliar na aplicação do algoritmo e manter assim o código mais limpo, tais funções foram makeSet, findSet, link e Union.

O makeSet vai colocar no inicio do algoritmo todos os vértices como pais de si próprios e com rank 0. O findSet é utlizado para descobrir a root de uma determinada árvore formada pelos vértices do grafo. A função link vai ser responsável por juntar duas árvores, estando escrita de forma que os valores esperados sejam as roots das próprias árvores, tal acontece devido à função Union. Sempre que um arco é adiconado a uma árvore, a variável max, inicializada a 0, é incrementada com o valor desse arco. No final do algoritmo é printado o valor da variável max.

Análise Teórica

- As funções makeSet, findSet e link têm todas O(1).
- A função Union tem complexidade O(logV), em que V é o número de vértices do grafo.
- Em computeKruskal a função sort é a que domina com complexidade O(E logE), onde E é o número de arcos do grafo. Logo a complexidade da função computeKruskal é O(E logE + E log V).
- Na função main as funcões scanf e addToGraph, ambas de complexidade O(1) são chamadas E vezes, logo temos O(E).
- A função main também chama a função computeKruskal, que tem a complexidade referida anteriormente.

Complexidade global da solução: O(E logE + E logV).

Relatório 2º projecto ASA 2022/2023

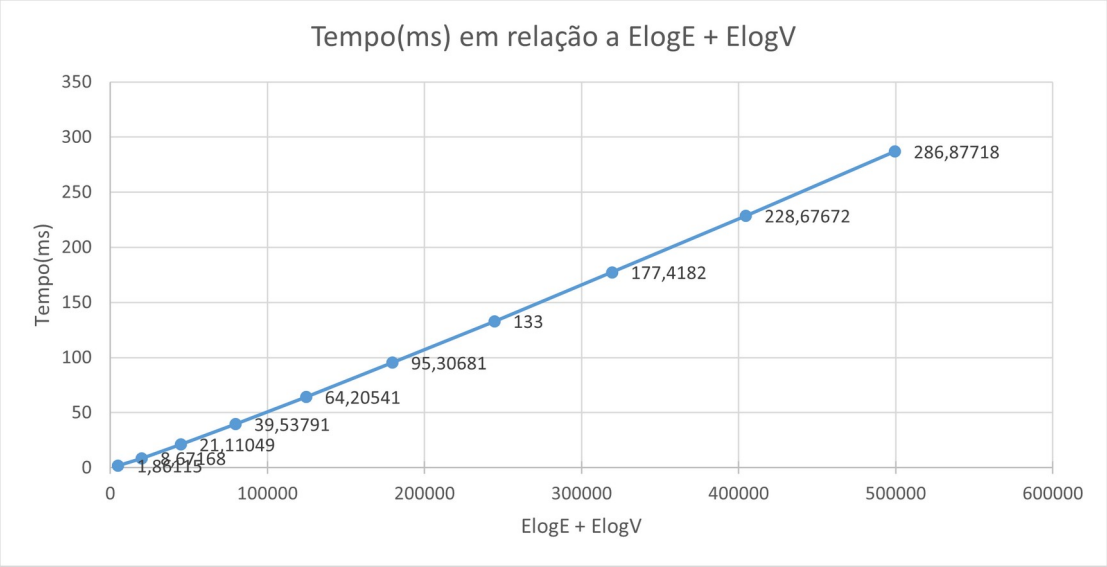
Grupo: TP021

Aluno(s): Henrique Costa (103663)

Avaliação Experimental dos Resultados

Utilizando o gerador de instâncias fornecido na página da cadeira foram feitos 11 testes em que o número de vértices foi sendo incrementado de 100 em 100.

Como podemos verificar esta linha não é completamente linear, e aqui o eixo dos X está a variar linearmente com o número de vértices. Assim, vamos pôr o eixo dos X a variar com o previsto pela análise teórica (neste caso, O(ElogE + ElogV)).



Ao mudarmos o eixo dos X para ElogE + ElogV, vemos que temos uma relação linear com os tempos no eixo dos Y. Assim, podemos concluir que a nossa implementação está de acordo com a análise teórica de O(ElogE + ElogV).