Relatório 2º projecto ASA 2022/2023

Grupo: AL030

Aluno(s): Fábio Mata (102802) e Nuno Gonçalves (103392)

Descrição do Problema e da Solução

Dado um grafo, em que os vértices representam capitais e os arcos ligações entre as mesmas, sendo que o peso de cada arco corresponde ao valor em M€ de trocas comerciais que é possível realizar entre a sua origem e destino, pretende-se calcular o valor máximo de trocas comerciais minimizando os custos da infraestrutura (*i.e.*, número de arcos).

A nossa solução consiste em encontrar uma Maximum Spanning Tree, isto é, uma árvore de arcos que garante a existência de um caminho entre quaisquer vértices, utilizando o menor número de arcos possível e maximizando o peso total dos respetivos arcos. Para tal, decidimos utilizar o algoritmo de *Kruskal*, alterando a ordem pela qual os arcos são ordenados (non-ascending order). Quanto à representação de conjuntos disjuntos, utilizámos árvores com compressão de caminhos e união por categorias.

Análise Teórica

Seja *V* o número de vértices e *E* o número de arcos:

- Leitura dos dados de entrada: simples leitura através de um ciclo que depende do número de arcos. Logo Θ(E);
- Operações sobre conjuntos disjuntos:
    - O(V) operações de Make-Set;
    - O(E) operações de Find-Set e Union;
    - Devido ao uso de árvores com compressão de caminhos e união por categorias, m operações de Make-set, Find-Set e Union: O(m α(V)) em que α(V) ≤ 4, pelo que podemos estabelecer: O((V + E) α(V)) ≈ O(V + E). [1]
- Ordenação dos arcos: O($E \times log_2 E$); [2]
- Apresentação do resultado: O(1).

Complexidade global da solução: O($E \times log_2 E + V$).

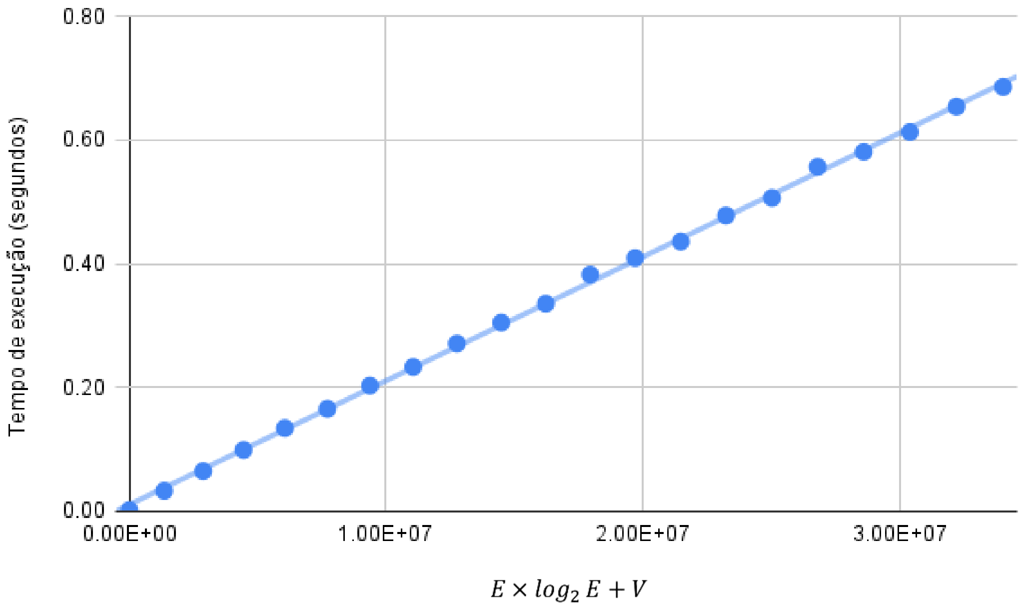
Relatório 2º projecto ASA 2022/2023

Grupo: AL030

Aluno(s): Fábio Mata (102802) e Nuno Gonçalves (103392)

Avaliação Experimental dos Resultados

Utilizámos o gerador de grafos fornecido para obter casos de teste e assim cronometrar o tempo de execução da nossa solução. Gerámos grafos com número de vértices entre 0 e 1M. Os resultados obtidos estão descritos no gráfico abaixo:



O gráfico acima representa o tempo de execução do programa em função de $E \times log_2 E + V$. Podemos observar que existe uma relação linear entre ambos, o que está de acordo com o que foi teoricamente previsto. Concluímos assim que a nossa implementação ∈ O($E \times log_2 E + V$).

Referências

1. Slides da aula teórica nº 12
2. https://en.cppreference.com/w/cpp/algorithm/sort