Relatório 1º projecto ASA 2022/2023

Grupo: AL039/TP11

Aluno(s): Gonçalo Rua (102604) e João Gouveia (102611)

Descrição do Problema e da Solução

A solução que encontrámos passa por utilizar uma variação do algoritmo de Kruskal. Em vez de querermos a Minimum Spanning Tree, no contexto deste problema pretendemos encontrar a Maximum Spanning Tree, i.e., a spanning tree cuja soma dos pesos é máxima. Para isto basta ordenarmos os arcos por ordem decrescente, em vez de crescente, e aplicar o algoritmo normalmente.

Análise Teórica

A complexidade total do algoritmo é O(ElogV), sendo E o número de arcos e V o número de vértices do grafo. Abaixo encontra-se uma análise mais pormenorizada de cada função.

- **Leitura de dados de entrada** (read_input): Simples leitura do input, com um ciclo que depende do número de arcos do grafo, Θ(E).

- **Implementação do algoritmo de kruskal** (get_maximum_cost_spanning_tree): Para ordenar os arcos do grafo foi utilizada a função sort, da Standard Template Library do c++, que tem complexidade O(ElogE) [1]. A estrutura de dados utilizada para representar os conjuntos disjuntos foi uma árvore e fazemos uso das heurísticas “compressão de caminhos” e “união por categoria”. Atendendo a isto, sabemos que é possível assegurar que a complexidade desta implementação é O(ElogV) [2].

- **Operações sobre conjuntos disjuntos** (make_set, find_set, link, node_union): Tanto a função make_set como a função link são O(1). Quanto ás funções find_set e node_union, a utilização das heurísticas “compressão de caminhos” e “união por categoria” garante-nos que ambas são O(V) [3].

Notas:

1. https://en.cppreference.com/w/cpp/algorithm/sort
2. CLRS, pág. 633
3. CLRS, pág. 572

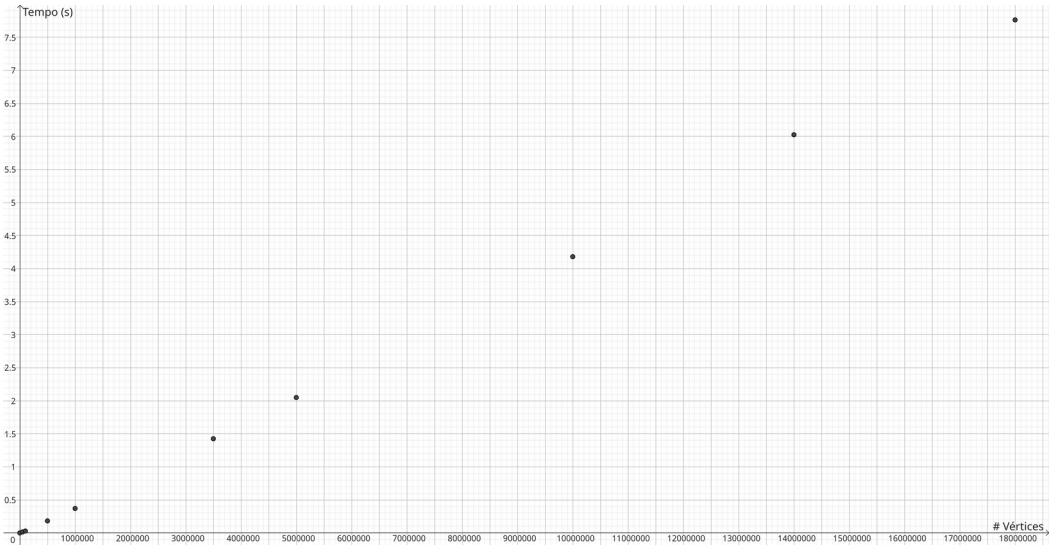
Relatório 1º projecto ASA 2022/2023

Grupo: AL039/TP11

Aluno(s): Gonçalo Rua (102604) e João Gouveia (102611)

Avaliação Experimental dos Resultados

Após gerados testes com número de vértices crescente, computou-se o seguinte gráfico, onde o eixo das abcissas representa o número vértices e o eixo das ordenadas representa o tempo, em segundos, que demorou a ser obtido o resultado com o algoritmo final.



Este resultado indica-nos, na prática, que a complexidade do algoritmo é linear para problemas relacionados com o objetivo do projeto, ou seja, que respeitam a triangulação de Delaunay. No entanto, a análise teórica acima realizada diz respeito à complexidade real do algoritmo e pressupõe grafos aleatórios. Após a realização de mais testes, desta vez com grafos aleatórios e sem restrições, e da elaboração de um novo gráfico, verifica-se a validade do resultado obtido na análise teórica.