Relatório 2º projecto ASA 2022/2023

Grupo: AL042

Aluno(s): Mateus Spencer (100032)

Descrição do Problema e da Solução

Para resolver o Problema, calculei a Maximum Spanning Tree do Grafo dado, assim minimizando o número de arcos entre localidades para o mínimo e escolhendo os caminhos com maior valor económico. Utilizei o Algoritmo de Kruskal, com a estrutura de dados union-find para saber que vértices já estão unidos na criação da MST. Em concreto usei o quick-union, com weighting/ranking e compressão de caminhos.

O algoritmo de Kruskal começa por ordenar todos os arcos, e retira sequencialmente todos os arcos, começando pelo arco com maior peso neste caso. Todos os vértices são inicializados como membros de uma MST apenas com esse vértice. Quando retiramos um arco, verificamos se os dois vértices desse arco já pertencem a mesma MST, caso contrário, somamos esse peso ao peso total das MSTs e aplicamos o quick union para indicar que esses vértices estão unificados.

Para ordenar de forma a retirar primeiro o arco com o maior peso, guardo os arcos com pesos negativos e ordeno de forma crescente, ao adicionar o peso desse arco ao total da MST, subtraindo o valor negativo, na prática somando. Poderia ter sido alternativamente ordenando a lista ao contrário ou ter iterado a lista desde o fim.

Análise Teórica

A complexidade do algoritmo de Kruskal é de O(E log E), uma vez que começa por ordenar os arcos dados de forma crescente segundo os pesos, utilizando a função sort do c++ que é uma implementação do quicksort, que tem complexidade O(n log n).

Leitura dos Dados, negando pesos dos vértices

Ordenar Arcos em ordem crescente

for(iterar todos os arcos do início para o fim){

Encontrar as cabeças dos grupos a que ambos os vértices do arco pertencem

if(se nao sao do mesmo grupo, vamos unificar, se forem, ignorar e não adicionar peso){

    Adicionar peso deste vértice ao total

        (como está invertido para negativo, usar -= em vez de +=)

    Escolher o grupo com menos níveis para unificar ao maior

    Fazer compressão de caminhos nas duas árvores

}

}

Relatório 2º projecto ASA 2022/2023

Grupo: AL042

Aluno(s): Mateus Spencer (100032)

Display da solução

Exemplo:

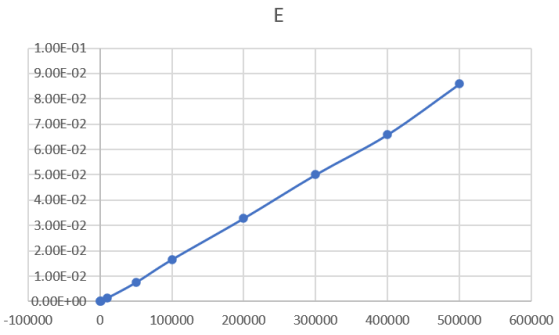
- Leitura dos dados de entrada: Leitura Simples dos dados de entrada, a depender do número de arcos logo O(E)
- Ordenamento dos Arcos com o algoritmo quicksort, logo em média O(E log E)
- Aplicação do algoritmo de kruskal para calcular o peso da Maximum Spanning Tree do Grafo, itera por todos os Arcos, logo O(E)
- A estrutura de dados usada é o union find, no pior caso a operação de find é O(N), mas com com weighting e path compression normalmente é mais próximo de O(1).
- Apresentação dos dados. O(1)

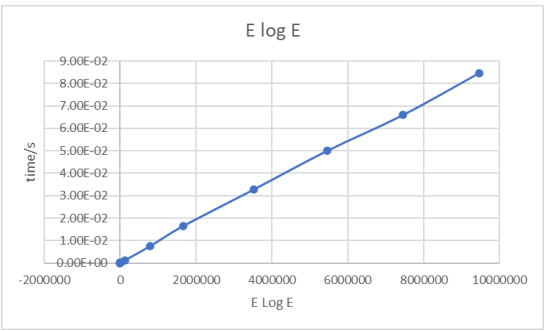
Complexidade global da solução: O(E log E) - limitado pelo sort

Avaliação Experimental dos Resultados

Grafos Gerados com o gerador de Gráficos Realísticos fornecidos na página do fénix.

10, 100, 1000, 10000, 50000,100000,200000,300000,400000,500000.





O gráfico da esquerda toma em conta no eixo x o número de Arcos, e seria esperado um crescimento logarítmico, mas o observado é mais próximo do linear, se bem que a diferença entre os dois é subtil.

No gráfico da direita ajustando o eixo dos X para ter E log E seria se esperar ver um crescimento linear o que confirmaria a complexidade esperada, e é sensivelmente isso que obtemos.

Para observar mais claramente o crescimento logarítmico teria de se correr o programa em grafos maiores, mas tal não foi possível uma vez que o programa gerador começa a demorar demasiado tempo a gerar gráficos maiores que estes utilizados.