Relatório 2º projecto ASA 2022/2023

Grupo: AL049

Aluno(s): Pedro Manuel Afonso Sousa (102664) e Pedro Alexandre Fatia Curto (103091)

Descrição do Problema e da Solução

O problema descrito neste projeto é, dado um grafo G = (V,E), calcular o valor máximo de trocas comerciais (valores das arestas), minimizando-se os custos de infraestrutura, ou seja, de arestas utilizadas.

Para resolver tal problema, começámos naturalmente por ler os dados de entrada; para processar as arestas, criámos uma estrutura *edge*, e um vetor onde guardamos todos os vértices lidos. Para questões de otimização de tempo, incluiu-se uma instrução (*sync_with_stdio(false)*). De seguida, resolve-se efetivamente o problema através da utilização de uma árvore abrangente que maximiza os pesos das arestas; para tal, recorremos ao algoritmo de Kruskal, suportado pela utilização de conjuntos disjuntos. Como tal, começámos por ordenar o vetor de arestas com base no seu peso. Posteriormente, inicializou-se o *parent* e o *rank* de cada vértice, guardando essa informação em dois vetores separados. Por fim, aplicam-se operações sucessivas de *findset* e *union* sobre cada aresta no vetor de arestas previamente ordenadas, somando-se a uma variável o peso de cada aresta em que se efetivamente aplica uma operação de *union*. Após decorrer este ciclo, imprime-se para o *stdout* o resultado da variável soma, com a solução encontrada.

Análise Teórica

A complexidade apresentada será expressa em função de vértices (V) e arestas (E).

- Leitura dos dados de entrada: loop que depende da quantidade de arestas: Θ(E)
- Ordenação do vetor de arestas utilizando o sort do *header* <algorithm>: O(E log E), uma vez que no pior caso a sua complexidade é O(n log n), e neste caso os elementos comparados são arestas.
- Inicialização do *parent* e *rank* de cada vértice: loop com duas operações O(1) que passa por todos os vértices, logo, Θ(V).
- Loop sobre todas as arestas com operações sucessivas de findset e union: uma vez que as operações de findset e union são ambas O(1), a complexidade deste procedimento é O(E).

Complexidade global da solução: O(E log E). Para a calcular, teremos de considerar: as operações de utilização de conjuntos disjuntos com árvores, e a operação de ordenação. Com base nos *slides* teóricos da utilização de conjuntos disjuntos com árvores, a complexidade de todas estas operações somadas é O(m α (n)), em que se executam m operações sobre n elementos. Com base na análise acima, verifica-se que as m operações são V+E, sobre V elementos. Uma vez que o α (n) é descartável por ser <= 4, temos O(V+E). Comparando com a complexidade da operação de ordenação, O(E log E), verificamos que esta última é a que majora o tempo de execução. Logo, pode-se afirmar que a complexidade global da solução é majorada por O(E log E).

Relatório 2º projecto ASA 2022/2023

Grupo: AL049

Aluno(s): Pedro Manuel Afonso Sousa (102664) e Pedro Alexandre Fatia Curto (103091)

Avaliação Experimental dos Resultados

De modo a gerar o gráfico apresentado abaixo, foram geradas 15 instâncias de tamanho progressivamente maior (desde 10 mil vértices até 4 milhões), e a quantidade de arestas de cada uma dessas instâncias foi multiplicada pelo logaritmo de base 2 dessas mesmas arestas, para se relacionar o tempo que cada instância demora a executar com E log E (em que E representa *edges*, ou arestas).

Os dois conjuntos estão apresentados na tabela que se encontra à direita.

O gráfico obtido experimentalmente relaciona, então, a grandeza descrita acima (E log E, no eixo dos X) com o tempo que essa instância demora a ser executada (no eixo dos Y). Como se pode observar, a relação entre os dois conjuntos de dados é linear, concluindo-se assim que o algoritmo que implementámos para a resolução do problema está em conformidade com a análise teórica previamente realizada, ou seja, O(E log E).

Time(s)	Edges*log(Edges)
0,019	337m
0,077	1.97M
0,149	4.17M
0,311	8.78M
0,48	13.59M
0,639	18.51M
0,823	23.52M
1,174	33.70M
1,511	44.12M
1,651	49.35M
2,218	62.67M
2,569	76.09M
3,639	103.42M
5,535	159.20M
7,586	216.07M