TÉCNICO LISBOA

Autores: António Terra, nº 84702
Miguel Viegas, nº 84747

Relatório ASA

Introdução breve

Neste segundo projeto de ASA, resolvemos o problema do Sr.João Caracol com recurso a grafos não-dirigidos. Cada cidade é representada por um vértice, e cada estrada é uma aresta que liga 2 vértices. Criou-se um vértice extra a que denominamos “Sky” que é ligado a todos os aeroportos. Em cada aresta foi guardado o custo de construção da estrada ou aeroporto. O algoritmo utilizado para percorrer o grafo e determinar a sua Minimum Spanning Tree (MST) foi uma versão modificada do Kruskal.

Descrição da solução

Foram utilizadas estruturas para representar conjuntos disjuntos (Vertex), necessários para o algoritmo Kruskal e para representar arestas do grafo (Edges). O nosso algoritmo funciona do seguinte modo:

- Começa por transcrever o input das estradas para um vetor de Vertex’s. Inicialmente cada Vertex é um conjunto disjunto por si só. A entrada ‘i’ do vetor contém o Vertex ‘i’, correspondente a uma cidade no grafo. Guarda também, num primeiro vetor de Edges, o conjunto de estradas e, num segundo vetor, o conjuntos de estradas e aeroportos (arestas que ligam os vértices ao ‘Sky’).

- Depois de recebido o input, é ordenado, por ordem não decrescente de custo, ambos os vetores de arestas acima referidos (QuickSort). Em caso de empate de pesos é favorecida a construção de estradas, como é referido no enunciado. Se ainda houver um caso de empate é escolhido a primeira aresta inserida.

- Depois é aplicado o Kruskal sobre o grafo que contém apenas as estradas. Isto porque queremos encontrar o melhor caminho possível só com estradas, se possível. Modificamos o Kruskal de modo a devolvesse os 3 parâmetros pedidos no enunciado (Custo total, número de aeroportos e número de estradas) em vez de um conjunto de arestas.

- Seguidamente é corrido o Kruskal sobre o grafo, mas que contém as estradas e os aeroportos.

- Por último são comparadas as tentativas de solução pelas 2 aplicações do algoritmo. Se a solução obtida no segundo Kruskal tiver melhor custo ou se for apenas possível fazer uma MST com o auxílio dos aeroportos, então a solução é o retorno desta aplicação do algoritmo. Caso contrário a solução é a solução da aplicação anterior do Kruskal. Se nenhum dos casos produzir uma MST, então é devolvida a respetiva mensagem de erro.

Análise teórica

Para tempos de execução o nosso algoritmo tem as seguintes etapas, sendo 'N' o número de cidades, 'A' o número de aeroportos e 'R' o número de potenciais estradas. A + R é o número de arestas no grafo, que iremos designar por 'E' daqui em diante.

- Guardar o input tem complexidade O(E) e ordenação dos vetores de Edges têm complexidade $O(E^2 + R^2)$, mas como R <= E, então pode-se simplificar a complexidade para $O(E^2)$.

- São realizadas duas aplicações do Kruskal, tendo cada uma complexidade $O(R\alpha(R,N))$ e $O(E\alpha(E,N))$ respetivamente para o grafo de apenas as estradas e o grafo completo.

- Estes tempos são só possíveis se as operações sobre os conjuntos disjuntos forem O(1). Isto é possível devido à compressão de caminhos associada à estrutura dos conjuntos disjuntos. O tempo total de O(N) é amortizado, ficando com a complexidade ideal de O(1).

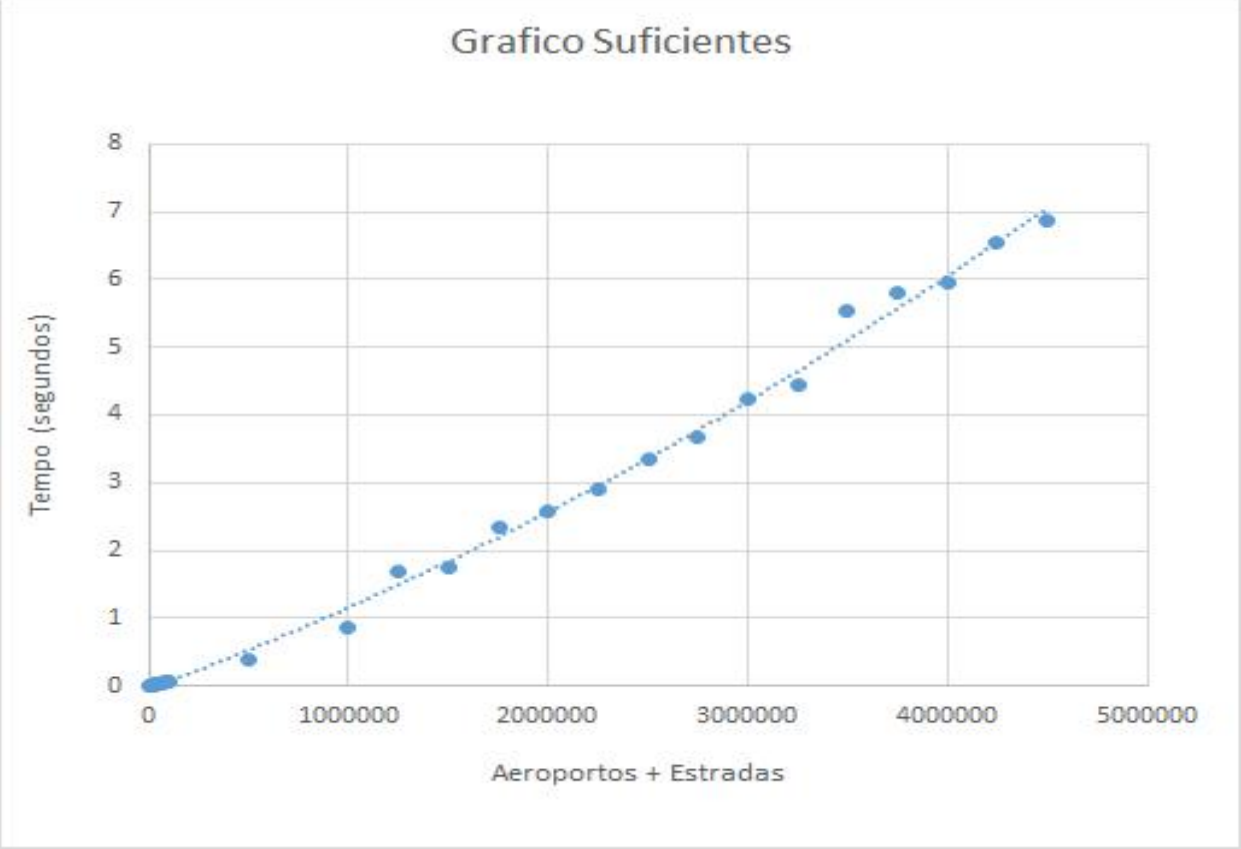
- A função α, caracterizada nas aulas teóricas, cresce tão devagar que, para qualquer efeito prático, é majorada pela constante 4, pelo que a complexidade de Kruskal = O(X). Como R <= E, então a complexidade das 2 aplicações é apenas O(E).

- Somadas todas as velocidades o programa escala com O(E²) devido à ordenação do vetor de Edges.

Em termos de uso de memória é apenas alocada o suficiente para o vetor de Vertex's e o vetor de todas as Edges, sendo o vetor de apenas as estradas um vetor auxiliar onde não é alocada memória para as estruturas, apenas é guardado o conjunto de ponteiros direcionados para o vetor das Edges. A complexidade é portanto O(N + E).

Avaliação experimental

Os resultados obtidos para testes de grafos suficientes e insuficientes são muito próximos, pois corremos o Kruskal para confirmar se o grafo é conexo, tendo assim a mesma complexidade. Não é verificado o pior caso, porque o QuickSort, no caso médio, tem como complexidade temporal O(nlog(n)), portanto, na prática, o algoritmo irá seguir esta complexidade. É muito raro o algoritmo atingir o pior caso do QuickSort.



Autores: António Terra, nº 84702
Miguel Viegas, nº 84747