Relatório

Projeto 2

Análise e Síntese de Algoritmos

Miguel Lourenço – 72588
Henrique Lourenço - 77459

Introdução

O objectivo deste projeto é, dado um conjunto de localidades , uma sede e o custo de transporte entre cada localidade, encontrar (i) a rota mais rentável que visita o maior número de localidades possível (ii) identificar, caso existam, subconjuntos de localidades que, quanto mais vezes por elas se passa, mais rentáveis ficam (iii) identificar, caso existam, localidades às quais não é possível chegar.

Estruturas utilizadas

Utilizámos um array de vértices (localidades), chamado "Graph", onde cada "vertex" tem armazenada a cor, custo, uma fila de arcos (caminhos entre localidades) e informação que mostra se o vértice está num ciclo de peso negativo (subconjunto de localidades que ficam mais rentáveis à medida que por elas se passa) e se este vértice pode ou não alterar o custo dos vértices adjacentes (ou seja, se passar por esta localidade irá afectar a rentabilidade das seguintes).
Esta Fila tem o nó (caminho entre localidades) inicial e final e cada nó tem um ponteiro para o vértice (localidade) e para o próximo nó.
Por último utilizámos uma fila semelhante, chamada "Priority Queue", que serve para saber a fila dos vértices sobre os quais ainda vamos iterar o algoritmo BFS (auxiliar ao Bellman-Ford que, sozinho, é incapaz de resolver o problema proposto neste projecto).

Algoritmo auxiliar (BFS)

O algoritmo BFS (Breadth-first search)
Este algoritmo explora sistematicamente os vértices de forma a descobrir todos os vértices atingíveis desde o vértice inicial.
A distância a cada vértice é calculada pelo menor número de arcos entre o vértice inicial e o vértice atingível.
Neste projecto o BFS é usado como auxiliar ao Bellman-Ford para quando encontrar uma localidade que está num ciclo de peso negativo, assinalar todas as localidades acessíveis a partir desta como localidades que estão num ciclo de peso negativo (em termos de output será um I)

Algoritmo principal (Bellman-Ford)

O algoritmo Bellman-Ford
Este algoritmo itera sobre um grafo sempre pela mesma ordem alterando os pesos de cada vértice para o menor possível sendo que os pesos começam todos a infinito (neste caso 2^15 – 1 devido à impossibilidade de ser realmente infinito) com excepção do vértice inicial que tem como peso 0. Após fazer número de vértices – 1 iterações faz mais uma iteração, na qual identifica a existência, ou não, de ciclos de peso negativo e vértices inalcançáveis partindo do vértice inicial.
A complexidade do algoritmo é, teoricamente, O(V*E).

Descrição do processo:

Inicialização:

- Começa por ler o número de vértices (localidades), arestas (caminhos entre localidades) e o vértice inicial (sede), do input.
- Cria um vector com todos os vértices (grafo / mapa)
- Lê todos os arcos do input e cria os arcos entre os vértices, na fila.

Ciclo principal:

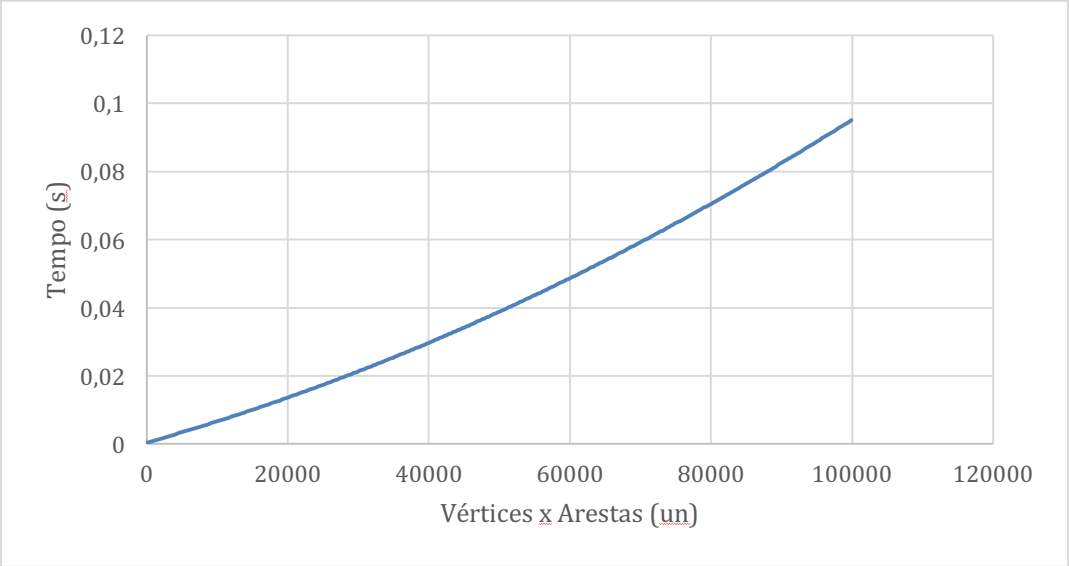
- O algoritmo começa com a estrutura dos vértices e arcos já criada, ou seja, o ficheiro já foi totalmente lido.
- Entra num ciclo que irá iterar de 1 até ao número de vértices – 1.
- Entra num ciclo que irá iterar no máximo de 0 até ao número de vértices.
- Caso o vértice actual tenha a informação de que pode alterar o peso dos vértices vizinhos, entra num terceiro ciclo que irá iterar sobre todos os arcos deste vértice.
- Neste terceiro ciclo relaxamos os arcos do vértice actual, ou seja, caso o peso do vértice actual somado ao custo do arco seja menor que o peso do vértice seguinte, o peso deste passa a ser o valor da soma.
- Caso o algoritmo não entre neste terceiro ciclo, pára. Caso isto nunca se verifique fará uma última iteração na qual fará novamente relaxamento aos arcos e caso o relaxamento seja bem sucedido marca o vértice actual como estando num ciclo de peso negativo e utilizando uma BFS marca todos os vértices alcançáveis a partir deste como estando também num ciclo de peso negativo.

Impressão:

- Faz uma impressão por cada vértice existente no grafo. Imprime “I” caso o vértice esteja num ciclo de peso negativo, “U” caso o vértice seja inalcançável a partir do vértice inicial ou, caso nenhum dos anteriores se verifique, imprime o peso do vértice.

Avaliação experimental:

Esta solução passa com sucesso aos 16 testes presentes no sistema Mooshak. Abaixo encontra-se um gráfico com o tempo de execução do programa (em segundos) com inputs com X vértices e X arcos.



Os valores acima apresentados são o resultado da execução do comando Unix time e correspondem ao valor real. Verifica-se um crescimento polinomial no tamanho do input.