**Análise e Síntese de Algoritmos**

2º Projeto 2015/2016 | Grupo 022 | 78974 | 82448

# Descrição do Projeto

O problema abordado neste projeto corresponde a encontrar a localidade, num mapa com **N** localidades e **E** caminhos com um determinado valor de custo, onde uma empresa que tem **F** filiais sediadas em localidades distintas irá realizar o seu encontro, sendo que a localidade escolhida será aquela em que a soma das deslocações das filiais até ela tenha o menor custo. A empresa em questão faz transporte de mercadorias. Cada rota, com origem nas filiais, tem um determinado custo: se o custo for negativo, a rota trás lucro. Sabendo o custo de cada ligação entre localidades, pretendemos resolver o problema, que pode ser abordado em parte como um “All-Pair-Shortest-Path Problem”.

# Implementação do Programa

O programa foi implementado na linguagem C, apenas com a utilização das bibliotecas <stdio.h>, <stdlib.h> e <limits.h>. Ao longo do programa, toma-se como infinito o valor SHRT\_MAX, uma vez que os pesos são implementados como short ints.

## Implementação da estrutura de dados

O problema acima descrito foi implementado com a utilização de grafos não dirigidos pesados, representados por listas de adjacências. A implementação da lista de adjacências foi feita com base num vetor primário para representar cada um dos vértices, e uma lista simplesmente ligada para representar os arcos. São definidas as operações de inicialização do grafo, criação de arcos, e inserção dos arcos.

São também criadas estruturas de dados auxiliares à resolução do problema. Uma Min-Heap, onde cada nó tem uma chave (correspondente ao número da localidade) e um peso (que corresponde ao custo de um determinado vértice até ele). São definidas, para a Heap, operações de inicialização, inserção de um elemento, remoção do elemento mínimo, troca de pesos, e, naturalmente, rearranjo da Heap.

É também criada uma FIFO com operações de inicialização, inserção e remoção de elementos (inteiros a representar localidades).

## Algoritmo de Johnson

Seja **V** o número de vértices (localidades) e **F** o número de filiais. A ideia base para a resolução do problema é a utilização do algoritmo de Johnson. Sabendo que o número de filiais é significativamente menor que o número de localidades e que o grafo é esparso, este algoritmo toma uma complexidade menor que, por exemplo, o algoritmo de Floyd-Warshall. O algoritmo de Johnson precisa, por sua vez, de recorrer às rotinas de Bellman-Ford e Dijkstra.

Começamos, então, por adicionar um vértice “dummy”, com ligação a todos os outros de peso 0. Ao corrermos o algoritmo de Bellman-Ford com base neste vértice e, depois, repesarmos cada arco do grafo com a função **w(u,v) = w(u,v)+h(u)-h(v)**, todos os valores de arcos passam agora a ser positivos. Neste momento, podemos correr o algoritmo de Dijkstra a partir de cada uma das filiais, preenchendo uma matriz de **V x F** com o custo (modificado) das filiais a todas as localidades.

Após termos esta matriz, podemos voltar a repesar os arcos, sendo que **w(u,v) = w(u,v) - h(u) + h(v)**. Nestas condições, temos já ao nosso dispor o custo real de cada uma das filiais a cada uma das localidades.

## Função main

A função main começa por ler do stdin o número de vértices com que é necessário inicializar o grafo, o número de filiais e o número de arestas.

O número de filiais corresponde ao número de valores que vão ser lidos na segunda linha de input, guardando estes valores num vetor de dimensão **F + 1**, onde se guarda o número de filiais na posição 0. Já o número de arestas corresponde ao número restante de linhas de input que serão lidas. Cada uma destas linhas é então transformada numa aresta e adicionada ao grafo. Terminado este passo, inicializam-se as estruturas de dados auxiliares e corre-se o algoritmo de Johnson.

Após o retorno do algoritmo, temos uma matriz de **V x F** preenchida com o menor dos custos de cada uma das filiais a cada uma das localidades. Tendo esses valores, é feito o seguinte:

* + Percorremos cada uma das colunas da matriz, onde cada coluna corresponde a uma localidade, somando todos os valores das suas linhas. Este valor corresponde ao custo de se realizar o encontro nesta localidade. A soma é considerada a soma mínima se for menor que a soma anterior e se o valor de uma das filiais a esta localidade não for infinito.
  + Sabendo o valor da soma mínima, se este for infinito, então não há nenhuma localidade que ligue todas as filiais, e não se pode realizar o encontro. Caso não o seja, devolvemos o seu custo, a localidade onde acontece, e, de seguida, cada um dos custos do caminho entre as filiais e essa localidade.

Note-se que os valores enviados para a estrutura de dados para os valores das arestas **(u,v)** são **u-1** e **v-1**. Isto deve-se ao facto de o programa esperar como input **N** vértices de valores de **1** a **N**, e a estrutura de dados interna funcionar com **N** vértices de valores **0** a **N-1**.

# Análise Teórica

Foi escolhida como estrutura de dados para a implementação deste problema um grafo representado como lista de adjacências. Tomando **V** como o número de vértices e **E** como o número de arcos, numa estrutura deste tipo, a complexidade de inicializar um grafo vazio é de **O(|V|)**, sendo que a posterior inserção de todas as arestas tem um custo de **O(|E|)**. Assim sendo, a complexidade total da construção de um grafo dado como input é de **O(|V|+|E|)**.

Sendo que o número de vértices corresponde ao o número de localidades e **F** é o número de filiais, a posterior inicialização da estrutura de dados que armazena os valores de custo é de **O(|V||F|)**.

Os passos principais no algoritmo são a chamada a Bellman-Ford, que tem uma complexidade estudada de **O(|V||E|)**, e F chamadas ao algoritmo de Dijkstra, que tem uma complexidade, devido à implementação com uma Min-Heap, de **O(|V|log|V| + |E|)**. Assim, a complexidade total do algoritmo é de **O(VE + F(Vlog(V) + E))**, ou seja, **O(VE + FVlog(V))**.

# Avaliação Experimental dos Resultados

Para proceder à avaliação da eficiência do programa...

# Referências bibliográficas

SEDGEWICK, Robert; **Algorithms in C;** 1997; Addison-Wesley Publishing Company