Algoritmos e Estruturas de Dados II
2º semestre de 2013 - Turma 02

Primeiro Exercício Programa

Busca de Rotas entre Aeroportos

1 Introdução

Neste exercício programa, vocês devem implementar um pequeno programa que realiza busca de rotas entre aeroportos. O programa deve receber como entrada: (i) uma base de dados de aeroportos e rotas; (ii) um critério a ser usado para realizar a busca das rotas; (iii) um aeroporto de partida; e (iv) um ou mais aeroportos de destino. Seu programa deve, para cada aeroporto destino, sugerir uma rota que satisfaça o critério selecionado. Para cada rota sugerida seu programa deve fornecer na saída: a quantidade de aeroportos presentes na rota, a distância total percorrida pela rota, e a lista de aeroportos que fazem parte da rota (maiores detalhes em relação à formatação de saída são dados na seção 3).

Seu programa deve ser capaz de trabalhar com dois critérios distintos para realizar o cálculo das rotas. O primeiro tem por objetivo encontrar uma rota que possui o menor número de escalas. Já o objetivo do segundo critério é encontrar uma rota que corresponda a viagem mais curta em termos de distância percorrida.

As buscas pelas rotas devem ser feitas a partir de um grafo direcionado onde os vértice representam aeroportos e as arestas representam voos ligando dois aeroportos. Como um dos critérios que pode ser especificado para a busca leva em conta a distância total percorrida pela rota, cada aresta também deve possuir um peso associado, que no caso corresponderá à distância (em km) entre dois aeroportos.

Para que o programa seja capaz de lidar com os dois critérios especificados, vocês deverão implementar os seguintes algoritmos: busca em largura (para encontrar rota com menor número de escalas) e algoritmo de *Dijkstra* (para encontrar rota que percorre a menor distância).

2 Base de dados utilizada

Para tornar este exercício mais interessante usaremos a base de dados utilizada pelo projeto OpenFlights (http://openflights.org), que é dividida em 3 porções: base da dados de aeroportos, base de dados de companhias aéreas e base de dados de voos. A fim de garantir que todos usem a mesma versão da base de dados, os arquivos específicos que contem a base serão disponibilizados no TIDIA

juntamente com este enunciado. Como o objetivo deste trabalho é realizar a pesquisa de rotas com base apenas nos critérios de distância viajada e número de escalas, a base de dados das companhias não será necessária para este EP. Assim, os arquivos de dados a serem utilizados são os seguintes: **airports.dat** e **routes.dat**. Nestes arquivos cada linha corresponde a um registro e os campos de cada registro são separados por virgulas. A página http://openflights.org/data.html descreve em maiores detalhes como estes arquivos de dados estão organizados.

2.1 Observações

Nem todos os campos presentes nos dois arquivos de dados são necessários para a realização do EP. Os campos importantes na base de dados de aeroportos são os seguintes:

- Airport ID (id aeroporto).
- City (nome da cidade).
- Country (nome do país).
- Código IATA/FAA (código de 3 letras para o aeroporto, como CGH e GRU, por exemplo).
- Latitude.
- Longitude.

Já para os registros de voos os seguintes campos são importantes:

- Source airport ID (id do aeroporto de origem).
- Destination airport ID (id do aeroporto de destino).

Observe que existe um campo "stops" nos registros de voos que indica se o voo em questão possui escalas mas, para fins de simplificação, tal parâmetro deve ser ignorado (ou seja, assumiremos que todos os voos definidos no arquivo **routes.dat** são voos diretos).

Observe também que os registros de voos não incluem a distância percorrida por cada voo. Para obter este valor (e assim poder associar um peso na aresta do grafo), é preciso calcular a distância entre o aeroporto de origem e o aeroporto de destino do voo. Este cálculo pode ser feito a partir das coordenadas de latitude e longitude associadas a cada aeroporto (um código em C que faz tal cálculo será disponibilizado junto com este enunciado).

Ao fazer a carga de dados, se algum dos campos mencionados anteriormente (com exceção dos nomes da cidade e país) possuir o valor "\N" (representando o valor NULL) todo o registro associado deve ser ignorado.

Note ainda que, como diferentes companhias podem oferecer o mesmo voo, é possível (e provável) que voos repetidos estejam definidos no arquivo **routes.dat**. Assim, durante a carga dos dados, no instante de adicionar uma nova aresta (representando um voo) ao grafo, deve-se verificar se já não existe no grafo uma aresta que representa o mesmo voo.

3 Funcionamento do programa e formatos de entrada e saída

Seu EP deve receber dois parâmetros pela linha de comando, devendo ser executado da seguinte forma:

```
>./busca_rotas entrada.txt saida.txt
```

sendo que no arquivo de entrada devem estar especificados: a base de dados a ser utilizada, o critério a ser usado para calculo das rotas, o aeroporto de partida e uma lista de aeroportos de destino para os quais serão buscadas as rotas (os aeroportos devem ser especificados através do código IATA/FAA). O conteúdo deste arquivo deve estar organizado como no exemplo abaixo:

```
airports.dat
routes.dat
DISTANCIA
GIG
3
JFK
NRT
GPH
```

onde: **airports.dat** e **routes.dat** são os arquivos referentes à base de dados; **DISTANCIA** indica qual o critério a ser utilizado (as únicas opções válidas são **ESCALAS** e **DISTANCIA**); **GIG** é o aeroporto de origem; o valor **3** indica quantos aeroportos de destino vem listados a seguir; e, finalmente, **JFK**, **NRT** e **CPH** são aeroportos de destino para os quais rotas serão pesquisadas a partir de **GIG**. O arquivo de saída deve vir formatado conforme modelo abaixo:

```
<origem> <critério utilizado> <num.  total de vértices> <num.  total de arestas>
<destino 1> <quantidade de aeroportos> <distância> <lista de aeroportos>
<destino 2> <quantidade de aeroportos> <distância> <lista de aeroportos>
...
<destino N> <quantidade de aeroportos> <distância> <lista de aeroportos>
```

onde: <critério utilizado> indica qual o critério que foi utilizado na busca pelas rotas; <origem> e <destino X> correspondem ao código IATA/FAA do aeroporto de origem e destino, respectivamente; <lista de aeroportos> deve ser composta por uma sequência de códigos IATA/FAA, todos na mesma linha, correspondente aos aeroportos presentes na rota encontrada; e todos os demais elementos da saída (número de vértices, número de arestas, quantidade de aeroportos presentes na rota e distância total percorrida pela rota) correspondem a valores numéricos inteiros.

É permitido colocar linhas com informações extras na saída de seu programa, desde que elas sejam iniciadas pelo caractere "#". Dessa forma é possível complementar a saída com informações de depuração ou para também oferecer uma saída mais clara para usuários. O seguinte exemplo, que mostra um trecho de saída, seria valido:

```
GIG DISTANCIA 5671 38400
# O grafo possui 5671 vértices e 38400 arestas.
```

```
# buscas de rotas feitas a partir do aeroporto GIG (Rio de Janeiro - Brazil).
JFK 4 3500 GIG AAA BBB JFK
# A rota mais curta para JFK passa por 4 aeroportos e percorre 3500 km:
# GIG - Rio de Janeiro - Brazil
# AAA - Cidade A - País A
# BBB - Cidade B - País B
# JFK - New York - United States
...
```

4 Entrega e datas

Este exercício programa deve ser entregue até as 23h59 do dia 23/11 através do TIDIA. Entregue um arquivo .zip contendo todos os fontes utilizados (arquivos .h e .c) e um arquivo README com instruções de como compilar seu EP e outras informações que julgar relevante. Não se esqueça de colocar seu nome e número USP no cabeçalho de todos os arquivos (incluindo o README). Lembre-se que seu EP deve ser compilável pela linha de comando usando o GCC e que, como será corrigido em ambiente Linux, não é permitido usar qualquer biblioteca exclusiva de outras plataformas.

5 Bônus

Receberá um bônus de 1.0 ponto na nota do EP quem, além de implementar o algoritmo de *Dijkstra*, também implementar o algoritmo de *Bellman-Ford* e apresentar uma breve comparação de desempenho entre os dois (medida do tempo que cada um leva para executar). Observe que é possível que ambos os algoritmos encontrem duas rotas distintas para um mesmo destino, mas nestes casos a distância percorrida por ambas as rotas devem ser a mesma.

Caso opte por implementar a parte bônus, deixe clara sua opção no arquivo README e escreva 1 ou 2 parágrafos para discutir alguns testes realizados e os resultados obtidos. Neste caso seu EP deve aceitar como critério, além das opções **ESCALAS** e **DISTANCIA**, a opção adicional **DIST_BELLMAN_FORD**. O formato de saída permanece inalterado, mas recomenda-se adicionar informações extras (linhas iniciadas com "#") sobre o desempenho dos algoritmos.