INSTITUTO SUPERIOR TÉCNICO

Introdução aos Algoritmos e Estruturas de Dados 2010/2011

Enunciado do 1^{o} Projecto

Data de entrega: 1 de Abril de 2011 às 23h00

1 Introdução

Quando se faz o planeamento de uma viagem de avião existem vários factores a considerar: hora de partida, hora de chegada, preço, número de escalas, etc. Cada pessoa tem prioridades diferentes, pelo que tendo como base o mesmo conjunto de voos disponível, a melhor solução depende das prioridades definidas por cada pessoa que viaja.

Neste projecto pretende-se desenvolver um programa que manipule informação relativa a voos entre cidades. Para tal, no *input* é fornecido um conjunto de voos que deverá ser considerado por forma a satisfazer os pedidos de viagem. Nesta fase do projecto apenas é pedido que guarde informação sobre os voos e aeroportos e responda a pedidos de informação simples.

2 Especificação do Programa

O programa recebe como *input* um conjunto de voos e um conjunto de pedidos de informação sobre os aeroportos. As linhas que especificam um voo começam com o caracter `v`, sendo que para cada voo é dada a seguinte informação:

- Código do voo
- Aeroporto de origem
- Aeroporto de destino
- Hora de partida no aeroporto de origem

- Hora de chegada no aeroporto de destino
- Preço

O código de um voo é definido como sendo uma sequência de caracteres composta por duas letras seguida por algarismos (no mínimo dois e no máximo quatro). Os aeroportos são identificados por códigos compostos por três letras.

Para além dos voos, o input contém linhas que começam com o caracter `i` que correspondem a pedidos de informação relativa a um aeroporto. Quando é lida uma linha de pedido de informação, o programa deverá fazer o *output* da informação relativa a esse aeroporto considerando apenas os voos especificados no *input* **antes** dessa linha.

No *output*, o programa deverá apresentar a informação relativa aos aeroportos. Por cada linha `i` do *input*, deverá ser apresentado o número de voos que se realizam com origem e destino num dado aeroporto, qual o destino (não considerando escalas) mais barato e qual o destino do voo com hora de partida mais tardia. Nas situações em que existe mais do que um voo com o mesmo preço ou com a mesma hora de partida, devem considerar o que aparece primeiro no input.

3 Dados de Entrada

O programa deverá ler os dados de entrada a partir do *standard input*. O formato dos dados de entrada será o seguinte:

- uma linha contendo um inteiro N ($N \geq 1$) que denota o número total de linhas a considerar no input; As linhas podem começar com um caracter `v` ou `i`.
- Uma linha que define um voo tem o formato que se segue, onde cada campo é separado por um espaço em branco:
  - um caracter `v`;
  - código de voo;
  - código do aeroporto de origem;
  - código do aeroporto de destino;
  - hora de partida do voo no aeroporto de origem no formato HH:MM onde HH denota um valor entre 00 e 23 e MM um valor entre 00 e 59;
  - hora de chegada do voo no aeroporto de destino no formato HH:MM onde HH denota um valor entre 00 e 23 e MM um valor entre 00 e 59;
  - preço do bilhete em euros definido por um número real.
- Uma linha que define um pedido de informação sobre um aeroporto tem o formato seguinte, sendo o separador um espaço em branco:

- um caracter `i`;
- código do aeroporto.

Assuma que os dados de entrada para o programa não contêm erros sintácticos, isto é, obedecem sempre ao formato descrito nesta secção. Pode assumir ainda que o **número máximo de voos é 1000 e que o número máximo de aeroportos é 100**.

4 Dados de Saída

Os dados de saída correspondem à informação relativa às linhas de tipo `i`. Por cada linha do tipo `i`, o programa deverá escrever no *standard output* um conjunto de estatísticas. A informação a escrever no output deverá ser a seguinte:

- Uma sequência de linhas onde cada linha corresponde a informação sobre um aeroporto pedida no *input*. A ordem das linhas do tipo `i` no *input* é respeitada no *output* e para cada linha deve ser considerada apenas a informação introduzida até essa altura. O output para cada linha é o descrito de seguida, onde o separador é um espaço em branco:
  - Código do aeroporto a;
  - Número de voos com origem no aeroporto a;
  - Número de voos com destino ao aeroporto a;
  - Código do aeroporto de destino com viagem mais barata (sem escalas) com origem em a. Se o número de voos com origem em a for 0, deverá colocar 000;
  - Código do aeroporto de destino com hora de partida mais tardia com origem em a; Se o número de voos com origem em a for 0, deverá colocar 000;
- Uma última linha deverá ser um inteiro M onde M denota o número de aeroportos diferentes usados na definição de voos no *input*.

5 Exemplo

5.1 Dados de Entrada

Vamos admitir que o ficheiro de entrada, que designaremos por `in.txt`, tem o seguinte conteúdo:

```
13
v TP001 LIS CDG 08:00 11:30 247.56
v IB002 MAD CDG 08:15 10:30 190.67
i LIS
i LHR
```

```
v BA375 LIS LHR 07:00 10:15 185.73
i LIS
v TP002 CDG LIS 10:00 11:30 220.00
v BA836 CDG LHR 10:00 10:40 99.99
i LIS
i CDG
i MAD
i LHR
i BOS
```

5.2 Dados de Saída

Para o ficheiro de entrada `in.txt` descrito na secção anterior, o programa deverá escrever na saída a seguinte informação:

```
LIS 1 0 CDG CDG
LHR 0 0 000 000
LIS 2 0 LHR CDG
LIS 2 1 LHR CDG
CDG 2 2 LHR LIS
MAD 1 0 CDG CDG
LHR 0 2 000 000
BOS 0 0 000 000
4
```

6 Compilação do Programa

Deve concretizar o ficheiro `Makefile` onde deverá definir o processo de geração do executável correspondente ao programa realizado. **O compilador a utilizar é o `gcc` com as seguintes opções de compilação: `-ansi -Wall -pedantic`. Este processo deve ser definido na regra `all` do ficheiro `Makefile`.** O executável gerado deve ter o nome `proj1`. Para compilar o programa deve executar o seguinte comando:

```
$ make -s all
```

o qual deve ter como resultado a geração do ficheiro executável `proj1`, caso não haja erros de compilação. A execução deste comando não deverá escrever qualquer resultado no terminal (daí a utilização da opção `-s` do comando `make`). Caso a execução deste comando escreva algum resultado no terminal, considera-se que o programa não compilou com sucesso. Por exemplo, durante a compilação do programa, o compilador não deve escrever mensagens de *warning*.

7 Execução do Programa

O programa deve ser executado da forma seguinte:

```
$ ./proj1 < in.txt > out.txt
```

8 Entrega do Projecto

A entrega do projecto deverá respeitar o procedimento seguinte:

- Na página da disciplina aceda ao sistema para entrega de projectos. O sistema será activado uma semana antes da data limite de entrega. Instruções acerca da forma de acesso ao sistema serão oportunamente fornecidas.
- Efectue o upload de um ficheiro arquivo com extensão `.tgz` que inclua o seguinte:
  - Ficheiros fonte (`.c`) e cabeçalho (`.h`) que constituem o programa.
  - Ficheiro `Makefile` que permita criar o executável `proj1`.

  Para criar um ficheiro arquivo com a extensão `.tgz` deve executar o seguinte comando na directoria onde se encontram o ficheiro `Makefile` e os ficheiros com extensão `.c` e `.h`, criados durante o desenvolvimento do projecto:

  ```
  $ tar cfz proj1.tgz Makefile *.c *.h
  ```
- Como resultado do processo de upload será informado se a resolução entregue apresenta a resposta esperada num conjunto de casos de teste.
- **O sistema não permite submissões com menos de 30 minutos de intervalo para o mesmo grupo.** Exemplos de casos de teste serão oportunamente fornecidos.
- Data limite de entrega do projecto: **23h00 do dia 01 de Abril de 2011**. Até à data limite poderá efectuar o número de entregas que desejar, sendo utilizada para efeitos de avaliação a **última** entrega efectuada. Deverá portanto verificar cuidadosamente que a última entrega realizada corresponde à versão do projecto que pretende que seja avaliada. Não serão abertas excepções.

9 Avaliação do Projecto

9.1 Componentes da Avaliação

Na avaliação do projecto serão consideradas as seguintes componentes:

1. A primeira componente avalia o desempenho da funcionalidade do programa realizado. Esta componente é avaliada entre 0 e 16 valores.

2. A segunda componente avalia a qualidade do código entregue, nomeadamente os seguintes aspectos: comentários, identação, estruturação, modularidade, abstracção, entre outros. Esta componente poderá variar entre -4 valores e +4 valores relativamente à classificação calculada no item anterior e será atribuída na discussão final do projecto.

3. **Grupos que não utilizem as flags correctas de compilação têm penalização de 50% na primeira componente**.

Durante a discussão final do projecto será averiguada a participação de cada elemento do grupo na realização do projecto, bem como a sua compreensão do trabalho realizado, sendo a respectiva classificação ponderada em conformidade, isto é, elementos do mesmo grupo podem ter classificações diferentes. **Elementos do grupo que se verifique não terem participado na realização do respectivo projecto terão a classificação de 0 (zero) valores.**

9.2 Atribuição Automática da Classificação

A classificação da primeira componente da avaliação do projecto é obtida automaticamente através da execução de um conjunto de testes executados num computador com o sistema operativo **GNU/Linux**. Torna-se portanto essencial que o código compile correctamente e que respeite o formato de entrada e saída dos dados descrito anteriormente. Projectos que não obedeçam ao formato indicado no enunciado serão penalizados na avaliação automática, podendo, no limite, ter 0 (zero) valores se falharem todos os testes. Os testes considerados para efeitos de avaliação podem incluir (ou não) os disponibilizados na página da disciplina, além de um conjunto de testes adicionais. A execução de cada programa em cada teste é limitada na quantidade de memória que pode utilizar, até um máximo de 32 MBytes, e no tempo total disponível para execução, sendo o tempo limite distinto para cada teste.

Note-se que o facto de um projecto passar com sucesso o conjunto de testes disponibilizado na página da disciplina não implica que esse projecto esteja totalmente correcto. Apenas indica que passou alguns testes com sucesso, mas este conjunto de testes não é exaustivo. É da responsabilidade dos alunos garantir que o código produzido está correcto.

Em caso algum será disponibilizada qualquer tipo de informação sobre os casos de teste utilizados pelo sistema de avaliação automática. A totalidade de ficheiros de teste usados na avaliação do projecto serão disponibilizados na página da disciplina após a data de entrega.

9.3 Detecção de Cópias

A avaliação dos projectos inclui a utilização de um sistema para detecção de situações de cópia entre projectos. A submissão de um projecto pressupõe o compromisso de honra que o trabalho incluso foi realizado única e exclusivamente pelos alunos referenciados nos ficheiros submetidos para avaliação. A quebra deste compromisso, ou seja a tentativa de um grupo se apropriar

de trabalho de outrém (sejam colegas ou outra pessoa), tem como consequência a reprovação de todos os alunos envolvidos (incluindo quem possibilitar a ocorrência de cópias) à disciplina de IAED.

Toda e qualquer situação de fraude ou facilitação de fraude terá então como consequência a reprovação imediata à disciplina de IAED neste semestre, assim como a comunicação da ocorrência ao respectivo Coordenador de curso e ao Conselho Pedagógico do IST para sanções adicionais de acordo com as regras aprovadas pela UTL e publicadas em Diário da República.

9.4 Considerações adicionais

Todos os programas são avaliados do modo seguinte:

```
$ ./proj1 < in.txt > out.txt; diff out.txt exp.txt
```

em que o ficheiro `exp.txt` representa o resultado esperado da execução do programa para os dados de entrada definidos pelo ficheiro `in.txt`. A impossibilidade de verificar automaticamente o resultado da execução de um dado programa implica uma penalização de **100%** na componente de avaliação automática. Considera-se que um programa passou um teste com sucesso se o resultado produzido por esse programa for exactamente igual ao resultado esperado, o que significa que o comando `diff` não deverá encontrar diferenças entre o resultado produzido pelo programa submetido e o esperado. Para poder ser avaliado, um projecto deverá compilar correctamente num computador com o sistema operativo **GNU/Linux**, sendo o utilizado o compilador **gcc** da **GNU**. A entrega de código não compilável, ou a não inclusão de qualquer dos ficheiros requeridos, ou a utilização de nomes diferentes para o ficheiro executável conduz a uma classificação de 0 (zero) valores. Não serão aceites quaisquer justificações.