# Estruturas de Dados e Algoritmos II
# Trabalho Prático

Departamento de Informática
Universidade de Évora

2018/2019
v1.0

## Fly me to the moon...

Na maior parte das vezes em que se quer fazer uma viagem de avião, é necessário apanhar mais do que um voo. Isto deve-se a o número de voos directos ser relativamente pequeno, em relação ao número de aeroportos existentes.

Para se marcar uma viagem, pode ser, portanto, necessário conseguir conjugar vários voos. Isso implica conhecer os horários dos vários voos, a sua duração, e ter em conta os tempos de escala nos aeroportos intermédios. Por vezes, em localidades com mais do que um aeroporto, é-se obrigado a mudar de aeroporto, por exemplo, do aeroporto internacional para o utilizado pelos voos domésticos.

O objectivo deste trabalho é implementar um sistema que permita descobrir, a partir do conhecimento da rede de aeroportos e dos voos existentes, que voos apanhar para chegar ao destino desejado.

Por enquanto, a Lua não é um dos destinos possíveis.

### 1 Descrição

As entidades do sistema a implementar são os *aeroportos* e os *voos*. A partir destas entidades, o sistema calculará *viagens* entre dois aeroportos.

#### 1.1 Aeroportos

Um *aeroporto* é identificado pelo seu *código*, que é uma sequência de 3 ou 4 letras maiúsculas e é único. A cada aeroporto está, também, associado o *fuso horário* da região em que que se localiza.

Estima-se que existem, em todo o mundo, cerca de 200 000 aeroportos. De cada aeroporto partirão, no máximo, 150 voos.

##### Fusos horários

Um *fuso horário* é indicado como a diferença entre a hora local e a hora no meridiano de Greenwich (GMT). Essa diferença é positiva nos fusos horários a Este de Greenwich (Europa, África, Ásia e Oceânia) e negativa nos a Oeste (América e parte das ilhas do Pacífico).

Por exemplo, o fuso horário do aeroporto de Lisboa, cujo código é `LIS`, é o GMT, e a diferença entre a hora em Lisboa e a hora GMT é de 0 horas e 0 minutos. Madrid, cujo aeroporto tem o código `MAD`, está no fuso horário `+01:00`, o que significa que a hora local está adiantada uma hora em relação à hora GMT. Quando em Madrid a hora é 14h30, a hora GMT é 13h30, que também é a hora de Lisboa. O principal aeroporto do Rio de Janeiro, no Brasil, com código `GIG`, tem fuso horário `-03:00`, *i.e.*, a hora local está 3 horas atrasada em relação à hora GMT. Quando lá a hora é 14h30, a hora em Lisboa é 17h30 e em Madrid é 18h30.

Nem todos os fusos horários correspondem a diferenças de um número inteiro de horas. Por exemplo, o fuso horário da Índia e do Sri Lanka é `+05:30` (a hora está adiantada 5 horas e meia em relação à GMT), o do Nepal é `+05:45` e o do Irão é `+03:30`.

O fuso horário não está associado a um país. O aeroporto de Ponta Delgada (código `PDL`) está no fuso horário `-01:00`, por exemplo.

Para o trabalho, um fuso horário será denotado por uma expressão do tipo `hh:mm` (fusos horários não negativos, *e.g.*, `00:00`, `01:00` e `05:45`) ou do tipo `-hh:mm` (fusos horários negativos, *e.g.*, `-01:00`, `-09:20` e `-12:00`); os fusos horários têm valores entre `-12:00` e `14:00`. Não se considera a existência de Hora de Verão e Hora de Inverno; a diferença entre a hora em dois sítios diferentes é imutável.

### 1.2 Voos

Um *voo* é identificado por um *código* único, que consiste numa sequência de 2 letras maiúsculas, seguida por uma sequência de 1 a 4 algarismos decimais, em que o primeiro é diferente de `0` (zero). Além do código, um voo é caracterizado pelos aeroportos de *partida* e de *destino*, pela sua *hora de partida*, dada no fuso horário do aeroporto de partida, e pela sua *duração*, em minutos.

Por exemplo, a linha abaixo descreve o voo `IB3111`, que parte do aeroporto `LIS` (Lisboa) às 9h50 e que aterra no aeroporto `MAD` (Madrid) 80 minutos depois:

```
IB3111 LIS MAD 09:50 80
```

Devido à diferença horária entre Lisboa e Madrid, o voo chega a Madrid às 12h10 locais (11h10 de Lisboa).

No trabalho, uma *indicação horária* terá a forma `hh:mm`, com valores entre `00:00` e `23:59`. A duração máxima de um voo será de 1 440 minutos.

O sistema conterá, no máximo, informação sobre 750 000 voos.

### 1.3 Viagens

Dados um aeroporto de partida, um aeroporto de destino e uma hora de chegada ao aeroporto de partida, uma *viagem* consistirá numa sequência de um ou mais voos, em que o aeroporto de partida do primeiro voo é o aeroporto de partida da viagem, e o aeroporto de destino do último voo é o aeroporto de destino da viagem. A hora de chegada ao aeroporto é dada no fuso horário do aeroporto de partida.

Se uma viagem for constituída por mais de um voo, o tempo que decorre entre a chegada a um aeroporto intermédio e a partida desse aeroporto é o *tempo de ligação*. O tempo de ligação mínimo num aeroporto intermédio é de 30 minutos.

A *duração* de uma viagem é o tempo que decorre entre a chegada ao aeroporto de partida e a chegada ao aeroporto de destino. Esse tempo corresponderá à soma do tempo de espera até à partida do primeiro voo (que pode ser de zero minutos), das durações dos voos efectuados e dos tempos de ligação nos aeroportos intermédios.

Neste trabalho, para cálculo de uma viagem, não serão consideradas mudanças de aeroporto, em localidades com mais do que um aeroporto.

## 2 Operações e comandos

São quatro as operações que o sistema deve suportar: a introdução de um novo aeroporto, a introdução de um novo voo, a eliminação de um voo, e o cálculo de uma viagem entre dois aeroportos. Estas operações serão invocadas recorrendo a quatro comandos, que são descritos a seguir. Além destes quatro comandos, haverá um outro comando para terminar a execução do programa que implementa o sistema.

A informação contida no sistema deverá manter-se entre execuções do programa.

Os vários elementos de um comando são separados, entre si, por um espaço. Nenhum comando começa ou termina com um espaço.

Pode assumir, sem testar, a correcção da entrada do programa, *i.e.*, que todos os comandos estão sintacticamente correctos e que não é violada nenhuma condição estabelecida no enunciado do trabalho, excepto nas situações mencionadas na descrição de cada comando.

Nos exemplos apresentados, o símbolo '$\Leftarrow$' assinala o comando introduzido e o símbolo '$\mapsto$' assinala a resposta do programa.

**Introdução de aeroporto** Introduz um novo aeroporto no sistema, com o código e o fuso horário dados.

Não tem efeito se o código corresponder ao de um aeroporto existente no sistema.

Entrada

```
AI <código> <fuso-horário>
```

Saída

- `+ novo aeroporto <código>`
- `+ aeroporto <código> existe`
  Se já existir um aeroporto com o código indicado.

Exemplos

```
⇐ AI LIS 00:00
↦ + novo aeroporto LIS
⇐ AI LIS 00:00
↦ + aeroporto LIS existe
⇐ AI MAD 01:00
↦ + novo aeroporto MAD
⇐ AI FAO 00:00
↦ + novo aeroporto FAO
```

**Introdução de voo** Introduz um novo voo no sistema, com o código, aeroportos de partida e de destino, hora de partida e duração dados.

Não tem efeito se o código corresponder ao de um voo existente ou se algum código de aeroporto não corresponder a nenhum aeroporto conhecido. Os aeroportos de partida e de destino serão sempre distintos.

Entrada

```
FI <código> <aeroporto-partida> <aeroporto-destino> <hora-partida> <duração>
```

Saída

- `+ novo voo <código>`
- `+ voo <código> existe`
  Se existir um voo com o código indicado.
- `+ aeroporto <aeroporto-partida> desconhecido`
  Se não existir um aeroporto com o código indicado para o aeroporto de partida.
- `+ aeroporto <aeroporto-destino> desconhecido`
  Se existir um aeroporto com o código indicado para o aeroporto de partida e não existir um aeroporto com o código indicado para o aeroporto de destino.

Exemplos

```
⇐ FI IB3111 PDL MAD 09:50 80
↦ + aeroporto PDL desconhecido
⇐ FI IB3111 MAD PDL 09:50 80
↦ + aeroporto PDL desconhecido
⇐ FI IB3111 OPO PDL 09:50 80
↦ + aeroporto OPO desconhecido
⇐ FI IB3111 LIS MAD 09:50 80
↦ + novo voo IB3111
⇐ FI IB3111 LIS MAD 09:50 80
↦ + voo IB3111 existe
⇐ FI IB3111 OPO PDL 09:50 80
↦ + voo IB3111 existe
⇐ FI TP1028 LIS MAD 14:15 75
↦ + novo voo TP1028
⇐ FI TP1904 FAO LIS 18:05 45
↦ + novo voo TP1904
```

**Eliminação de voo** Elimina um voo do sistema, a partir do seu código.

Não tem efeito se o código não corresponder ao de um voo existente.

Entrada

```
FD <código>
```

Saída

- `+ voo <código> removido`
- `+ voo <código> inexistente`

  Se o código não corresponder ao de um voo existente no sistema.

Exemplos

```
⇐ FD BA1
↦ + voo BA1 inexistente
⇐ FD IB3111
↦ + voo IB3111 removido
⇐ FD IB3111
↦ + voo IB3111 inexistente
```

**Cálculo de uma viagem** Calcula uma viagem entre dois aeroportos dados, se for possível voar de um para o outro, directamente ou não. A viagem calculada é uma cuja duração é a mínima possível.

Não provoca alterações no sistema. Não calcula nenhuma viagem se não houver voos que permitam fazer a viagem pretendida ou se algum dos códigos não corresponder ao de um aeroporto existente. Os aeroportos de partida e de destino serão sempre distintos.

Entrada

```
TR <aeroporto-partida> <aeroporto-destino> <hora-chegada-aeroporto>
```

Saída

- `Voo    De   Para Parte Chega`
  `====== ==== ==== ===== =====`
  `<cod`$_1$`> <p`$_1$`> <c`$_1$`> <hp`$_1$`> <hc`$_1$`>`
  `...`
  `<cod`$_n$`> <p`$_n$`> <c`$_n$`> <hp`$_n$`> <hc`$_n$`>`
  `Tempo de viagem: <duração> minutos`

  Onde `<cod`$_i$`>`, `<p`$_i$`>`, `<c`$_i$`>`, `<hp`$_i$`>` e `<hc`$_i$`>` são, respectivamente, os códigos do voo, do aeroporto de partida e do de destino, a hora de partida e a hora de chegada de cada um dos voos que constituem a viagem, pela ordem por que são efectuados, e `<duração>` é a duração total da viagem, em minutos.

  As horas de partida e de chegada dos voos são mostradas nos fusos horários dos aeroportos de onde o voo parte e onde o voo chega, respectivamente.
- `+ sem voos de <aeroporto-partida> para <aeroporto-destino>`

  Se os voos existentes não permitirem fazer a viagem pretendida.
- `+ aeroporto <aeroporto-partida> desconhecido`

  Se não existir um aeroporto com o código indicado para o aeroporto de partida.
- `+ aeroporto <aeroporto-destino> desconhecido`

  Se existir um aeroporto com o código indicado para o aeroporto de partida e não existir um aeroporto com o código indicado para o aeroporto de destino.

Exemplos

```
⇐ TR LIS PDL 00:00
↦ + aeroporto PDL desconhecido
⇐ TR PDL LIS 05:00
↦ + aeroporto PDL desconhecido
⇐ TR PDL OPO 10:00
↦ + aeroporto PDL desconhecido
```

```
⇐ TR LIS MAD 10:00
↦ Voo    De   Para Parte Chega
↦ ====== ==== ==== ===== =====
↦ TP1028 LIS  MAD  14:15 16:30
↦ Tempo de viagem: 330 minutos
⇐ TR LIS MAD 14:15
↦ Voo    De   Para Parte Chega
↦ ====== ==== ==== ===== =====
↦ TP1028 LIS  MAD  14:15 16:30
↦ Tempo de viagem: 75 minutos
⇐ TR MAD LIS 00:00
↦ + sem voos de MAD para LIS
⇐ TR FAO MAD 18:00
↦ Voo    De   Para Parte Chega
↦ ====== ==== ==== ===== =====
↦ TP1904 FAO  LIS  18:05 18:50
↦ TP1028 LIS  MAD  14:15 16:30
↦ Tempo de viagem: 1290 minutos
```

No último exemplo acima, o voo de Faro (`FAO`) chega a Lisboa depois da partida do voo para Madrid, pelo que será necessário esperar e apanhá-lo no dia seguinte.

**Terminar a execução** Termina a execução do programa.

Entrada

```
X
```

Saída

- *Nada*

Exemplos

```
⇐ X
```

## 3 Versão simplificada

Há uma versão simplificada do trabalho, em que não há códigos de voo e em que todas as indicações horárias se referem ao mesmo fuso horário.

Nesta versão, um voo é identificado através do triplo (origem, destino, hora de partida).

**Introdução de aeroporto** Introduz um novo aeroporto no sistema, com o código dado.

Não tem efeito se o código corresponder ao de um aeroporto existente.

Entrada

```
AI <código>
```

Saída

- `+ novo aeroporto <código>`
- `+ aeroporto <código> existe`
  Se já existir um aeroporto com o código indicado.

Exemplos

```
⇐ AI LIS
↦ + novo aeroporto LIS
⇐ AI LIS
↦ + aeroporto LIS existe
⇐ AI MAD
↦ + novo aeroporto MAD
⇐ AI FAO
↦ + novo aeroporto FAO
```

**Introdução de voo** Introduz um novo voo no sistema, com os aeroportos de partida e de destino, hora de partida e duração dados.

Não tem efeito se já existir um voo que parte e chega aos mesmos aeroportos, e que sai à mesma hora. Os aeroportos de partida e de destino serão sempre distintos.

Entrada

```
FI <aeroporto-partida> <aeroporto-destino> <hora-partida> <duração>
```

Saída

- `+ novo voo <aeroporto-partida> <aeroporto-destino> <hora-partida>`
- `+ voo <aeroporto-partida> <aeroporto-destino> <hora-partida> existe`
  Se já existir um voo com a mesma origem e o mesmo destino, à mesma hora.
- `+ aeroporto <aeroporto-partida> desconhecido`
  Se não existir um aeroporto com o código indicado para o aeroporto de partida.
- `+ aeroporto <aeroporto-destino> desconhecido`
  Se existir um aeroporto com o código indicado para o aeroporto de partida e não existir um aeroporto com o código indicado para o aeroporto de destino.

Exemplos

```
⇐ FI PDL MAD 09:50 80
↦ + aeroporto PDL desconhecido
⇐ FI MAD PDL 09:50 80
↦ + aeroporto PDL desconhecido
⇐ FI OPO PDL 09:50 80
↦ + aeroporto OPO desconhecido
⇐ FI LIS MAD 09:50 80
↦ + novo voo LIS MAD 09:50
⇐ FI LIS MAD 09:50 80
↦ + voo LIS MAD 09:50 existe
⇐ FI LIS MAD 09:50 65
↦ + voo LIS MAD 09:50 existe
⇐ FI LIS MAD 14:15 75
↦ + novo voo LIS MAD 14:15
⇐ FI FAO LIS 18:05 45
↦ + novo voo FAO LIS 18:05
```

**Eliminação de voo** Elimina, do sistema, o voo com os aeroportos de partida e de destino, e a hora de partida dados.

Não tem efeito se os dados não corresponderem a um voo existente.

Entrada

```
FD <aeroporto-partida> <aeroporto-destino> <hora-partida>
```

Saída

- `+ voo <aeroporto-partida> <aeroporto-destino> <hora-partida> removido`
- `+ voo <aeroporto-partida> <aeroporto-destino> <hora-partida> inexistente`
  Se algum dos códigos não corresponder a um aeroporto existente ou se não existir um voo que viaja entre os aeroportos dados, àquela hora.

Exemplos

```
⇐ FD OPO PDL 11:20
↦ + voo OPO PDL 11:20 inexistente
⇐ FD LIS MAD 07:18
↦ + voo LIS MAD 07:18 inexistente
⇐ FD LIS MAD 09:50
↦ + voo LIS MAD 09:50 removido
⇐ FD LIS MAD 09:50
↦ + voo LIS MAD 09:50 inexistente
```

**Cálculo de uma viagem** Calcula uma viagem entre dois aeroportos dados, se for possível voar de um para o outro, directamente ou não. A viagem calculada é uma cuja duração é a mínima possível.

Não provoca alterações no sistema. Não calcula nenhuma viagem se não houver voos que permitam fazer a viagem pretendida ou se algum dos códigos não corresponder ao de um aeroporto existente. Os aeroportos de partida e de destino serão sempre distintos.

Entrada

```
TR <aeroporto-partida> <aeroporto-destino> <hora-chegada-aeroporto>
```

Saída

- ```
  De   Para Parte Chega
  ==== ==== ===== =====
  ```
  <p$_1$> <c$_1$> <hp$_1$> <hc$_1$>
  ...
  <p$_n$> <c$_n$> <hp$_n$> <hc$_n$>
  `Tempo de viagem: <duração> minutos`

  Onde <p$_i$>, <c$_i$>, <hp$_i$> e <hc$_i$> são, respectivamente, os códigos do aeroporto de partida e do de destino, a hora de partida e a hora de chegada de cada um dos voos que constituem a viagem, pela ordem por que são efectuados, e `&lt;duração&gt;` é a duração total da viagem, em minutos.
- `+ sem voos de <aeroporto-partida> para <aeroporto-destino>`

  Se os voos existentes não permitirem fazer a viagem pretendida.
- `+ aeroporto <aeroporto-partida> desconhecido`

  Se não existir um aeroporto com o código indicado para o aeroporto de partida.
- `+ aeroporto <aeroporto-destino> desconhecido`

  Se existir um aeroporto com o código indicado para o aeroporto de partida e não existir um aeroporto com o código indicado para o aeroporto de destino.

Exemplos

```
⇐ TR LIS PDL 00:00
↦ + aeroporto PDL desconhecido
⇐ TR OPO LIS 05:00
↦ + aeroporto OPO desconhecido
⇐ TR PDL OPO 10:00
↦ + aeroporto PDL desconhecido
⇐ TR LIS MAD 10:00
↦ De   Para Parte Chega
↦ ==== ==== ===== =====
↦ LIS  MAD  14:15 15:30
↦ Tempo de viagem: 330 minutos
⇐ TR LIS MAD 14:15
↦ De   Para Parte Chega
↦ ==== ==== ===== =====
↦ LIS  MAD  14:15 15:30
↦ Tempo de viagem: 75 minutos
⇐ TR MAD LIS 00:00
↦ + sem voos de MAD para LIS
⇐ TR FAO MAD 18:00
↦ De   Para Parte Chega
↦ ==== ==== ===== =====
↦ FAO  LIS  18:05 18:50
↦ LIS  MAD  14:15 15:30
↦ Tempo de viagem: 1290 minutos
```
```

**Terminar a execução** Termina a execução do programa.

Entrada

`X`

Saída

- *Nada*

Exemplos

⇐ `X`

## 4 Potencial de expansão

Se a implementação for apropriada, põe-se a hipótese de a aproveitar para a aplicar a outro tipo de viagens, como viagens de autocarro e de comboio, ou mesmo de integrar todos os meios de transporte (interurbanos) com horários regulares no mesmo sistema. Em qualquer destas aplicações, o volume de dados com que o programa deverá trabalhar é muito maior,

## 5 Realização e entrega

### 5.1 Realização

A realização do trabalho consiste na criação de um programa, em C, que implemente o sistema descrito. O programa deverá ler os comandos a executar da sua entrada normal (*standard input*) e escrever as respostas na sua saída normal (*standard output*).

O trabalho será realizado em grupos de um ou dois elementos. Só serão considerados os trabalhos de grupos cujos elementos tenham todos obtido a pré-qualificação para o trabalho.

O código C entregue deverá estar de acordo com o *standard* C99, poder ser compilado com o GCC e executado em Linux. O código será compilado, com as opções `-std=gnu99 -Wall -lm`, através do comando

```
gcc -std=gnu99 -Wall -lm *.c
```

A opção `-lm` indica ao compilador que deve incluir no executável criado uma biblioteca com funções matemáticas, como a função `sqrt`, que calcula a raiz quadrada de um número. Para usar estas funções, o programa deverá conter uma directiva

```
#include <math.h>
```

O estado do sistema deverá persistir para além da execução do programa. Toda a informação existente no sistema no fim de uma execução do programa deverá estar disponível quando o programa voltar a ser executado.

Todas as escritas e leituras de informação em disco deverão ser controladas explicitamente pelo programa.

### 5.2 Ambiente de execução e restrições

Na entrega, os programas serão testados numa máquina com sistema operativo Linux, de 64 bits, com um disco magnético normal (HD, e não SSD), e com páginas (de disco) de 4096 *bytes*.

A memória RAM utilizada pelo programa estará limitada a 64 MB e o espaço disponível no disco para dados é de 2 GiB.

### 5.3 Entrega

O trabalho será submetido através do Mooshak, uma aplicação que compilará o código e executará e testará os programas criados, no concurso "EDA2 2018 (Trabalho)". O endereço para acesso ao concurso estará disponível na página Moodle de EDA2.

O acesso ao Mooshak faz-se através da identificação do grupo e de uma *password*, que serão fornecidas após o envio da constituição do grupo ao docente de EDA2.

As submissões poderão ter uma de três formas:

- Um arquivo `tar` comprimido, num ficheiro com extensão `.tgz`, contendo *somente* os ficheiros com o código do programa (com extensão `.c` ou `.h`).

  Os ficheiros deverão ser extraídos para a directoria corrente.

  Um arquivo com estas características pode ser criado através de um comando como o seguinte:

  ```
  tar cvzf nome-do-arquivo.tgz *.[ch]
  ```

- Um arquivo `zip`, num ficheiro com extensão `.zip`, contendo *somente* os ficheiros com o código do programa (com extensão `.c` ou `.h`).

  Também neste caso, os ficheiros deverão ser extraídos para a directoria corrente.

- Um ficheiro com todo o código do programa e com extensão `.c`. (Não aconselhado.)

No concurso, estarão definidos quatro problemas:

C-T (Completo – Teste)

Este problema serve para testar o programa (versão não simplificada) com os testes que estarão disponíveis em:

http://www.di.uevora.pt/~vp/eda2/testes/.

C-E (Completo – Entrega)

Este é o problema em que é feita a entrega do código.

Um trabalho (versão não simplificada) só será aceite se o programa passar todos os testes deste problema.

S-T (Simplificada – Teste)

Este problema serve para testar o programa (versão simplificada) com os testes que estarão disponíveis em:

http://www.di.uevora.pt/~vp/eda2/testes/.

S-E (Simplificada – Entrega)

Este é o problema em que é feita a entrega do código para a versão simplificada do trabalho.

A versão simplificada do trabalho só será aceite se o programa passar todos os testes deste problema.

Se cumprir todos os restantes requisitos, a classificação de um trabalho aceite neste problema poderá ir de 7 a 15 valores.

**Importante** Qualquer mensagem do compilador e qualquer diferença entre o que o programa escreve e o que devia escrever (e tal como descrito nas secções anteriores) levará à sua não aceitação. Também poderá impedir a sua aceitação os programas terminarem sem ser por `return 0` (ou equivalente).

**Testar o programa** Se o executável que contém o programa se chamar `fly-me`, podem verificar se a resposta a um teste — por exemplo, o `teste-C-A-1` — está correcta através do comando:

```
$ ./fly-me < teste-C-A-1.in | diff teste-C-A-1.out -
```

Se o funcionamento do programa estiver de acordo com o enunciado, o comando terminará sem nada ser escrito no terminal, caso contrário, mostrará as diferenças entre a resposta correcta e a dada pelo programa.

Também é possível redireccionar a saída do programa para um ficheiro e, depois, recorrer ao comando `diff` para comparar o ficheiro obtido e o ficheiro com o resultado esperado:

```
$ ./fly-me < teste-C-A-1.in > teste-C-A-1.o-meu-out
$ diff teste-C-A-1.out teste-C-A-1.o-meu-out
```

(Em vez do comando `diff`, pode ser usado qualquer comando que permita comparar o conteúdo de dois ficheiros. No Linux, o comando `meld` permite comparar o conteúdo de dois ficheiros de modo mais amigável.)

### 5.4 Relatório

Além do código submetido e aceite pelo Mooshak, deverá ser entregue um relatório, em papel, com:

- a identificação dos elementos do grupo;
- uma descrição pormenorizada das estruturas de dados usadas (bonecos, também conhecidos como diagramas, são algo que dá muito jeito para descrever estruturas de dados);
- a descrição pormenorizada do formato do(s) ficheiro(s) de dados;
- a descrição do funcionamento do programa, nomeadamente no que toca ao funcionamento das várias operações (devem explicar o que acontece, sem apresentar código nem nomes de funções);
- a justificação de todas as escolhas feitas (suportada na complexidade dos algoritmos usados e na análise dos acessos a disco feitos pelo programa);
- a análise de como a implementação feita se poderá adaptar para lidar com um volume superior de dados;
- o código dos programas, exactamente como submetido no Mooshak;
- a identificação das fontes consultadas para a realização do trabalho.

Encontrarão um modelo para o relatório na página Moodle de EDA2.

### 5.5 Avaliação do trabalho

A um trabalho aceite pelo Mooshak corresponderá uma nota igual ou superior a 7, desde que acompanhado de um relatório que cumpra os objectivos.

Na classificação do trabalho serão avaliadas a qualidade das estruturas de dados e dos algoritmos utilizados, a qualidade do código (incluindo a sua clareza, a sua legibilidade, e os comentários), o potencial de expansão da implementação (10% da cotação), a qualidade do relatório e a qualidade do *trabalho* efectuado.

### 5.6 Datas

A data limite de submissão do **programa** no Mooshak é 4ª-feira, dia 26 de Junho de 2019.

A data limite de entrega do **relatório** é 5ª-feira, dia 27 de Junho de 2019, até às 19h00.

A **discussão** do trabalho será realizada na semana seguinte, em dia(s) a combinar.

### 5.7 Não há Fs

*Todos* os elementos entregues pelo grupo deverão ser da autoria de *todos* os membros do grupo. Qualquer elemento usado no trabalho que não seja da autoria dos elementos do grupo, deverá estar devidamente assinalado e a sua origem indicada. (Este uso só é admissível em situações *muito pontuais*, como, por exemplo, na escolha de uma função de *hash* e na utilização de algoritmos conhecidos, e não se aplica à partilha de código entre grupos.) Também deverá ser assinalado o código que se tenha inspirado em código cuja autoria não pertence aos membros do grupo (excepto código fornecido em EDA2).

Durante a discussão, *todos* os elementos do grupo deverão demonstrar conhecimento da estruturação do trabalho (estruturas de dados usadas, algoritmos aplicados, etc.) e do código entregue.

O código contido no relatório deverá corresponder *exactamente* ao código submetido no Mooshak, incluindo a formatação e os comentários.

A inobservância destas regras terá como consequência a anulação do(s) trabalho(s) e a reprovação de *todos* os elementos do grupo ou grupos envolvidos.

Bom Trabalho