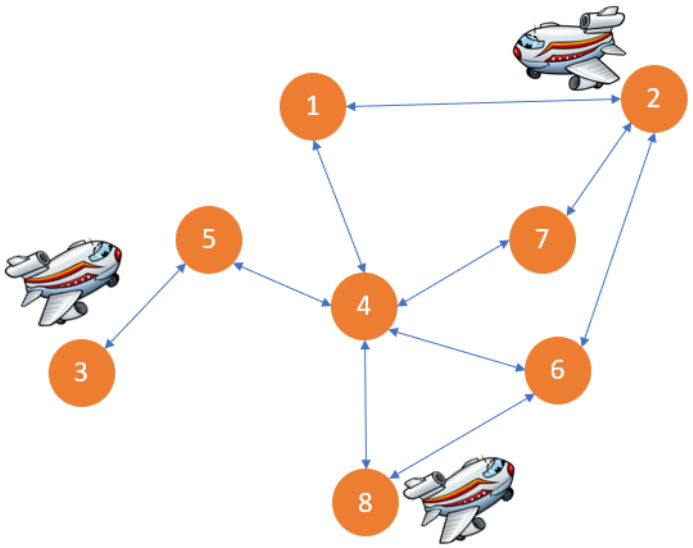
DEEC
DEPARTAMENTO DE ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES
TÉCNICO LISBOA

Licenciaturan em Engenharia Electrotécnica e de Computadores (LEEC)

# Algoritmos e Estruturas de Dados

## Enunciado do Projecto para Época Especial




# AirRoutes

**Versão 4.00 (10/Março/2023)** – Versão pública

2022/2023
Época Especial do 1º Semestre

# Conteúdo

# 1 Introdução

O trabalho que se descreve neste enunciado possui duas componentes, que correspondem às duas fases de avaliação de projecto para a disciplina de Algoritmos e Estruturas de Dados. A descrição geral do projecto que se propõe diz respeito ao trabalho a desenvolver para a última fase de avaliação.

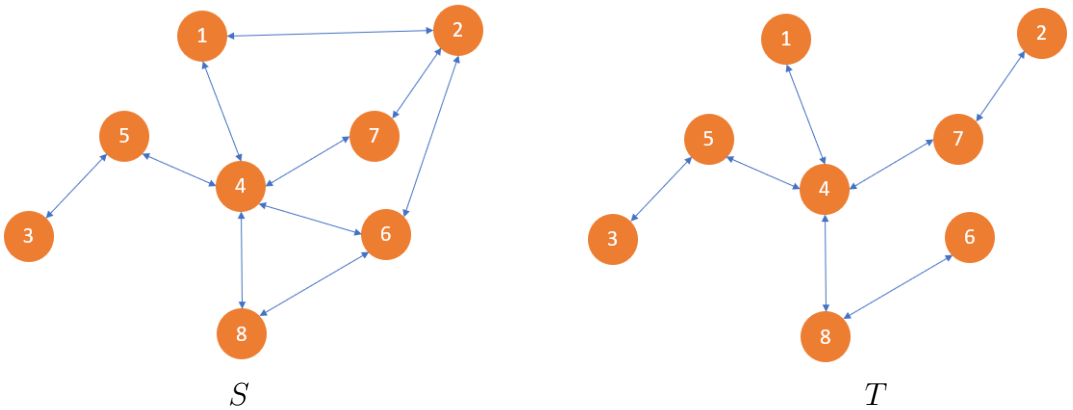
Para a primeira fase de avaliação o trabalho consiste apenas no desenvolvimento de algumas funcionalidades testáveis que podem ser usadas posteriormente para desenvolver a solução final.

A entrega do código fonte em ambas as fases é feita através de um sistema automático de submissão que verificará a sua correcção e testará a execução do programa em algumas instâncias do problema.

# 2 O problema – AirRoutes

Neste projecto pretende-se desenvolver um programa que seja capaz de produzir um conjunto de rotas ligando aeroportos que garanta a existência de um caminho entre cada par fonte/destino sem qualquer redundância. Num cenário normal é expectável que exista mais que uma forma de executar um qualquer percurso entre quaisquer duas cidades mas, num cenário de contracção de custos, a existência de múltiplas formas de atingir um qualquer destino, dado um qualquer ponto de partida, poderá ter de ser equacionada e questionada, substituindo-se algumas das rotas directas por conjuntos de vôos em sequência (multi-stop).

Se cada rota possuir informação de custo de ser mantida, o objectivo poderá ser apenas manter as rotas mais baratas que assegurem a conectividade original. Ou seja, se no conjunto inicial de rotas disponíveis for possível realizar uma viagem aérea entre a cidade $A$ e a cidade $B$, no conjunto final de rotas a serem mantidas deverá também ser possível realizar essa viagem. Naturalmente que poderá dar-se o caso de o número de vôos agora necessário para concretizar essa viagem ser superior ao do caso original.




$S$ $T$

Figura 1: Exemplo de conjunto inicial de rotas aéreas ($S$) e de um conjunto final possível, ($T$), que assegura os mesmos serviços que o conjunto inicial sem rotas redundantes.

Na Figura 1 apresenta-se um caso muito simples de rotas iniciais (à esquerda) e rotas finais (à direita), como ilustração. Neste exemplo, que continha 10 rotas, mantiveram-se

apenas 7. Foram removidas as rotas: entre 1 e 2; entre 2 e 6; e entre 4 e 6. Apesar da eliminação destas três rotas, continua sendo possível realizar uma viagem aérea entre qualquer par de cidades, como inicialmente. Note-se, no entanto, que a remoção de mais alguma rota fará com que se perca a conectividade original.

Após ser concretizada uma redução de rotas, pode ser necessário identificar algumas características da solução obtida em termos da sua robustez/flexibilidade a interrupção de rotas e/ou inacessibilidade temporária de cidades/aeroportos.

Assumindo que um qualquer conjunto inicial de rotas é identificado como um grafo em que as cidades/aeroportos são os vértices e as rotas são as arestas que os ligam, o que se pretende neste projecto é desenvolver uma aplicação em linguagem C que seja capaz de resolver automaticamente um qualquer grafo, de acordo com as especificações gerais do problema e a natureza particular do grafo em questão.

Nas secções seguintes descreve-se o formato de utilização do programa a desenvolver, no que diz respeito aos ficheiros de entrada e saída; as regras de avaliação; e faz-se referência ao *Código de Conduta Académica*, que se espera ser zelosamente cumprido por todos os alunos que se submetam a esta avaliação.

Aconselha-se todos os alunos a lerem com a maior atenção todos os aspectos aqui descritos. Será obrigatória a adesão sem variações nem tonalidades a todas as especificações aqui descritas. A falha em cumprir alguma(s) especificação(ões) e/ou regra(s) constante(s) neste enunciado acarretará necessariamente alguma forma de desvalorização do trabalho apresentado.

Por esta razão, tão cedo quanto possível e para evitar contratempos tardios, deverão os alunos esclarecer com o corpo docente qualquer aspecto que esteja menos claro neste texto, ou qualquer dúvida que surja após uma leitura atenta e cuidada deste enunciado.

# 3 O programa AirRoutes

O programa a desenvolver deverá ler e armazenar um qualquer grafo representando um conjunto de cidades/aeroportos e rotas que as/os ligam e produzir uma solução para esse grafo. Nas possíveis variantes do problema a resolver quer-se obter:

A1 – O conjunto de rotas de custo total menor que assegure os mesmos serviços que o conjunto original – "backbone" do grafo original;

B1 – O "backbone" variacional em relação à solução obtida em $A1$ por exclusão da aresta entre os vértices $v_i$ e $v_j$;

D1 – O "backbone" variacional por exclusão do vértice $v_i$;

Em qualquer das variantes, o programa deverá escrever as soluções para um ficheiro de saída.

A variante $A1$ corresponde à descrição padrão inicialmente feita. Ou seja, a solução é um subconjunto das rotas originais, $T$, tal que não exista nenhum outro subconjunto de menor custo total, quando somados os custos de todas as rotas seleccionadas. Esse subconjunto tem de ser tal que qualquer par fonte/destino para o qual haja caminho no grafo original deverá também possuir caminho no subgrafo solução.

Na variante $B1$ pretende-se determinar que aresta substitui a aresta dada. Ou seja, sendo $T$ o conjunto de arestas da solução em variante $A1$ a remoção da aresta entre $v_i$ e $v_j$ faz com que algumas das ligações deixem de ser possíveis. A solução para este problema

terá de conter todas as arestas do conjunto $T \setminus \{v_i \leftrightarrow v_j\}$ mais a aresta mais barata que reponha a conectividade original. O termo variacional refere-se à restrição de a solução ter de conter todas as arestas em $T \setminus \{v_i \leftrightarrow v_j\}$.

Na variante $D1$, ao remover o vértice $v_i$ todas as arestas em $T$ que sejam incidentes desse vértice deixam de poder ser usadas. Seja $T_i$ o subconjunto de arestas de $T$ que tenham $v_i$ como um dos vértices de que são incidentes. A solução da variante $D1$ deverá conter todas as arestas em $T \setminus T_i$ e as arestas mais baratas que reponham a conectividade original, quando e se tal for possível, exceptuando viagens começadas e terminadas em $v_i$.

Nas secções seguintes descreve-se a forma como o programa deve ser invocado, a forma como os dados de entrada são acedidos e o formato que têm de ter os dados de saída, ou seja, a forma de descrever a solução encontrada.

## 3.1 Execução do programa

Haverá duas fases de submissão do projecto. O que se descreve nas próximas secções diz respeito às especificações da versão final do projecto. Na secção 4 detalham-se as especificações relativas à primeira fase de submissões.

O programa AirRoutes deverá ser invocado na linha de comandos da seguinte forma:

```
aed$ backbone <nome>.routes
```

`backbone` é o nome do ficheiro executável contendo o programa AirRoutes;

`<nome>.routes` em que `<nome>` é variável, identificando o ficheiro contendo um ou mais grafos e respectivos problemas.

Para cada grafo o programa deverá fazer uma de duas coisas:

- produzir uma solução que satisfaça as especificações gerais e particulares do problema;
- indicar que não existe solução.

Por exemplo, se não existir nenhuma aresta alternativa em variante $B1$, não existe solução. Tomando o exemplo da Figura 1, não existe alternativa para a rota que liga 3 a 5. Se essa rota deixar de funcionar, então nenhuma outra fará com que seja reposta a conectividade original.

## 3.2 Formato de entrada

O ficheiro de extensão `routes` contém um ou mais grafos e, para cada grafo, especifica que variante se pretende resolver. Cada problema é identificado por um cabeçalho e um grafo. O cabeçalho define as dimensões do grafo e o tipo de problema que se pretende resolver. O grafo consiste num conjunto de vértices, que representam os aeroportos, e um conjunto de arestas bidireccionais ligando pares de vértices, que representam as rotas aéreas.

Cada uma das 3 variantes possui um cabeçalho único, mas todas possuem uma parte comum nesse cabeçalho: dois inteiros e uma cadeia de caracteres. Os dois inteiros identificam quantos vértices e arestas possui o grafo, $V$ e $E$. A cadeia será uma de 3 possibilidades: $A1$, $B1$ e $D1$.

Para a variante $A1$ não é necessária mais informação. Para a variante $B1$, após a cadeia identificadora da variante, o cabeçalho contém sempre dois inteiros. Os cabeçalhos da variante $D1$ só têm um inteiro após a identificação da variante.

Após o cabeçalho, que identifica o problema, o ficheiro contém a informação das arestas, identificadas por um par de inteiros que correspondem aos vértices que a aresta liga mais um terceiro número real (a ler como double) que especifica o custo da aresta.

Em síntese:

- uma linha com $V$ $E$ *Variante* $[v_i]$ $[v_j]$

- $E$ linhas com $v_i$ $v_j$ $c_{ij}$, para $i \neq j$ e $i, j = 1, 2, \ldots, V$ e com $c_{ij} \in \mathbb{R}^+$.

Por exemplo, um problema para o grafo da Figura 1, poderia ser definido como se mostra abaixo.

```
8 10 A1
1 2 20.5
1 4 9.22
2 6 11.7
2 7 3.68
3 5 4.0
4 5 3.96
4 6 5.3
4 7 5.2
4 8 4.8
6 8 5.1
```

Figura 2: Um ficheiro contendo um único problema.

No exemplo apresentado, o grafo possui 8 vértices/aeroportos e 10 arestas/rotas. O objectivo é produzir o "backbone" padrão. O vértice 4 possui uma aresta de peso 5.3 para o vértice 6. A existência de uma aresta entre estes dois vértices significa que tanto se pode viajar do vértice 4 ao 6 como se pode fazer o percurso inverso.

Sublinha-se aqui que os dados de cada problema não têm necessariamente que estar "arrumados" como se ilustra no exemplo da Figura 2[1]. Os procedimentos de leitura a adoptar pelos alunos deverão ser robustos a qualquer tipo de desarrumação, sendo que existirá sempre, pelo menos, um espaço em branco[2] após cada dado de cada problema.

Independentemente da forma como os dados estão distribuídos no ficheiro de entrada, garante-se o seguinte: cada novo grafo inicia-se com dois inteiros positivos, $V$ e $E$; após estes dois inteiros, existirá **sempre** uma cadeia de caracteres especificando a variante a resolver; depois da cadeia especificando a variante poderão existir dois, um ou nenhum inteiro de acordo com a variante em causa; após esta informação existirão **sempre** $E$ pares de inteiros seguidos de um real positivo não nulo e em que ambos os inteiros são diferentes (ambos restringidos a valores entre 1 e $V$).

Os ficheiros com um ou mais problemas poderão ter qualquer nome, mas têm obrigatoriamente a extensão `.routes`. Assume-se que todos os ficheiros de extensão `.routes`

[1]Ver exemplo alternativo no apêndice, que clarifica o que se entende por não estar "arrumado".

[2]Ou mudança de linha ou tabular.

estão correctos e no formato especificado anteriormente, no que à sequência e tipologia dos dados de cada problema diz respeito.

O programa não necessita fazer qualquer verificação de correcção do formato dos ficheiros de entrada. Apenas necessita de garantir que a extensão está correcta, que o ficheiro passado como argumento existe de facto e interpretar correctamente o seu conteúdo semântico.

Finalmente, se se pretendesse resolver dois grafos a partir de um só ficheiro de entrada, o seu formato seria a justaposição desses dois grafos e respectivos cabeçalhos, definidores do problema a resolver para cada um deles.

Apenas por razões estéticas, haverá sempre, pelo menos, uma linha em branco entre dois grafos/problemas sucessivos. Em geral, o número de linhas em branco entre dois problemas não tem de ser fixo, pelo que o procedimento de leitura dos ficheiros de entrada deverá ser suficientemente geral, para acomodar oscilações nesta formatação[3].

## 3.3 Formato de saída

O resultado da execução do programa AirRoutes consiste em apresentar primeiro o cabeçalho da solução, baseado no cabeçalho do problema acrescentando-lhe mais informação. Após o cabeçalho da solução, deverá ser apresentada a solução do problema. Cada um dos três problemas possui um formato específico de construção de cabeçalho e apresentação de solução. Estes formatos descrevem-se abaixo.

```
8 10 A1 7 35.96
1 4 9.22
2 7 3.68
3 5 4.00
4 5 3.96
4 7 5.20
4 8 4.80
6 8 5.10
```

Figura 3: Um ficheiro contendo a solução em variante $A1$.

(A1)- O cabeçalho da solução desta variante deve repetir o cabeçalho do problema após o que apresenta o número de rotas preservadas e o seu custo. Após o cabeçalho deverão ser apresentadas todas as rotas preservadas. Ver Figura 3 para a solução do grafo da Figura 1.

(B1)- O cabeçalho da solução desta variante repete o cabeçalho do problema seguido do número de rotas preservadas e seu custo para o "backbone" original, como em variante $A1$, e apresenta um de três inteiros: se o inteiro apresentado no final do cabeçalho for -1, significa que não existe rota que reponha a conectividade do "backbone" original ou que algum dos dois vértices não pertence ao grafo; se o inteiro apresentado for 0, significa que a rota interditada não faz parte do "backbone"; e se for 1 significa que existe uma rota de substituição. Após o cabeçalho deverão ser

[3]Não é objectivo que os alunos se dediquem à mui nobre e pouco edificante tarefa de andar a contar linhas vazias.

apresentadas todas as rotas do "backbone" original seguidas da rota de substituição apenas se o último inteiro no cabeçalho tiver sido 1. Ver Figura 4 para a solução do grafo da Figura 1 se tiver sido interditada a rota $8 \leftrightarrow 6$. A aresta substituta está a negrito para facilidade de leitura. Se a rota interditada tivesse sido a $4 \leftrightarrow 5$ a resposta seria -1 no cabeçalho e apenas se apresentaria o "backbone" original.

```
8 10 B1 8 6 7 35.96 1
1 4 9.22
2 7 3.68
3 5 4.00
4 5 3.96
4 7 5.20
4 8 4.80
6 8 5.10
4 6 5.30
```

Figura 4: Um ficheiro contendo a solução em variante $B1$.

(D1)- O cabeçalho da solução em variante $D1$, depois de repetir o cabeçalho do problema, apresenta a informação relativa ao "backbone" original seguida do número de rotas que necessita/pode adicionar para substituir a perda de conectividade por interdição de um aeroporto. Depois do cabeçalho, apresenta o "backbone" original seguido das rotas de reposição, como em $B1$. Se o vértice não pertencer ao grafo o número de rotas a adicionar é -1.

Se o ficheiro de extensão `.routes` possuir mais do que um grafo, o ficheiro de saída deverá conter uma solução para cada um dos problemas indicados e pela mesma ordem em que surgem no ficheiro de entrada. Para facilitar a interpretação das várias soluções num mesmo ficheiro de saída, é **obrigatório** que entre cada duas soluções exista uma, e não mais, linha vazia de separação.

A(s) solução(ões) deve(m) ser colocada(s) num único ficheiro de saída, cujo nome deve ser o mesmo do ficheiro de problemas mas **com extensão** `.bbones`. Este ficheiro deve ser criado e aberto pelo programa. Por exemplo, se o ficheiro com problemas se chama `teste231.routes`, o ficheiro de saída deve-se chamar `teste231.bbones`. Note-se que em situações em que haja erro na passagem de argumentos ao programa não faz qualquer sentido criar um ficheiro de saída.

Todas as rotas apresentadas no ficheiro de saída deverão estar ordenadas por ordem crescente do vértice de partida e por ordem crescente do vértice de chegada quando os vértices de partida são iguais. Quando é necessário apresentar dois conjuntos de rotas ("backbone" original e rotas novas, por exemplo) cada conjunto segue a regra de apresentação ordenada. A aresta $u \leftrightarrow v$ deverá ser apresentada usando o vértice de menor índice como ponto de partida e o de maior índice como ponto de chegada (ver exemplos apresentados neste texto), após o que se indica o seu custo.

Se o programa for invocado com ficheiros inexistentes, que não possuam a extensão `.routes`, sem qualquer argumento ou com argumentos a mais, deverá sair silenciosamente. Ou seja, sem escrever qualquer mensagem de erro, nem criar qualquer ficheiro de saída.

Sublinha-se aqui que a única forma admissível para produção de resultados do programa é para ficheiro de saída, quando tal for possível. Qualquer escrita para *stdout* ou qualquer escrita em ficheiro que não siga o formato aqui descrito constitui erro.

Todas as execuções do programa deverão sempre retornar o inteiro 0 quando este termina. Qualquer execução que, ao terminar o programa, retorne (através da instrução `return`, na função `main`, ou da invocação da função `exit`), em qualquer função, um valor diferente de 0, será interpretada pelo site de submissões como "Erro de Execução"se alguma vez o programa terminar por essa "porta de saída".

# 4 Primeira fase de submissões

Nesta secção explicitam-se os objectivos, especificações e funcionalidades que deverão estar operacionais na data da primeira fase de submissões. Todas as funcionalidades desta fase de submissão dizem exclusivamente respeito ao processamento de grafos para extracção de informação a partir dos mesmos.

O formato de invocação do programa será o mesmo que o definido anteriormente. Ou seja, o executável tem o mesmo nome e deverá ser passado apenas um argumento: um ficheiro de extensão `.routes0`. Este contém um ou mais grafos para um ou mais problemas diferentes. Aqui também os sucessivos problemas, no ficheiro de extensão `.routes0` estarão separados por, pelo menos, uma linha em branco. Este ficheiro tem formato ligeiramente diferente do anteriormente definido, como se indicará mais à frente.

As variantes de funcionamento são:

A0 – determinação de grau;

B0 – determinação de adjacência;

C0 – determinação de cliques de três vértices;

D0 – contagem de cliques de três vértices.

Em variante $A0$ pretende-se determinar o **grau** de um vértice/aeroporto passado como argumento do problema no seu cabeçalho. Define-se como grau de um vértice o número de arestas que o têm como um dos extremos. Em variante $B0$ pretende-se saber se dois vértices/aeroportos passados como argumento no cabeçalho do problema possuem uma rota que os una. Em variante $C0$ pretende-se saber se o vértice/aeroporto passado como argumento do problema no seu cabeçalho faz parte de alguma **clique** de três vértices. Define-se como clique de um grafo um seu subgrafo que seja completo. Ou seja, em que quaisquer dois vértices desse subgrafo possuem uma aresta directa entre si. No grafo da Figura 1 existe uma clique de três vértices constituida pelos vértices 4, 6 e 8. É a única daquele grafo. Em variante $D0$ pretende-se contar em quantas cliques diferentes de três vértices está o vértice/aeroporto passado como argumento no cabeçalho que define o problema.

O cabeçalho de cada problema é definido por: $V$ $E$ *Variante* $v_i$ $[v_j]$. $V$ identifica quantos aeroportos existem, $E$ especifica qual o número total de rotas, *Variante* é uma cadeia de caracteres que assume o valor de uma das quatro variantes acima descritas, $v_i$ é um vértice e $v_j$ é um segundo vértice, cuja ocorrência só se dá em variante $B0$. O resto do problema segue o mesmo formato dos ficheiros de extensão `.routes` descrito anteriormente.

Para problemas de variante $A0$ a resposta será um inteiro não negativo quando o vértice passado como argumento pertence ao grafo e será -1 no caso contrário. Para problemas de variante $B0$ a resposta será o custo da rota quando exista (escrito com duas casas decimais apenas), e -1 quando não exista rota ou algum dos vértices não pertencer ao grafo. Para problemas de variante $C0$ a resposta será -1 se o vértice não pertencer ao grafo, 0 se o vértice não estiver incluído em nenhuma clique de três e 1 se estiver em alguma. Para problemas de variante $D0$ a resposta será -1 se o vértice não pertencer ao grafo, 0 se não estiver incluído em nenhuma clique de três ou algum $k > 0$ se existirem $k$ conjuntos diferentes de dois vértices com os quais forme uma clique de três.

Por exemplo, para o grafo da Figura 1 poder-se-ia ter o ficheiro de entrada apresentado na Figura 5.

```
8 10 A0 4
1 2 20.5
1 4 9.22
2 6 11.7
2 7 3.68
3 5 4.0
4 5 3.96
4 6 5.3
4 7 5.2
4 8 4.8
6 8 5.1
```

Figura 5: Exemplo de ficheiro contendo um problema em variante $A0$.

em que se pergunta qual o grau do vértice 4. A resposta será:

```
8 10 A0 4 5
```

Se, em vez de $A0$, tivesse sido pedida a variante $C0$, o resultado seria

```
8 10 C0 4 1
```

## 4.1 Formato de saída da primeira fase de submissões

O ficheiro de saída da primeira fase, tem o mesmo nome que o ficheiro de problemas, mas deverá ter extensão `.queries` e deverá incluir todos os resultados associados com cada um dos problemas presentes no ficheiro de entrada. O ficheiro de saída deverá conter apenas uma linha por problema: repete o cabeçalho do problema seguido do resultado obtido, de acordo com a descrição feita na secção anterior.

O ficheiro de entrada poderá conter mais do que um problema para resolver e cada um desses problemas poderá ter diferentes dimensões.

Se, por hipótese, o ficheiro de entrada possuir mais que um problema, o ficheiro de saída será a concatenação das soluções de todos os problemas. Aqui também é **obrigatória** a inclusão de uma linha em branco como separador das diferentes soluções. Também é **obrigatório** que entre cada inteiro(ou string) exista **apenas** um espaço em branco, tal como ilustrado nos exemplos para dados de saída.

# 5 Avaliação do Projecto

A presente versão deste projecto, uma vez que se destina à componente de avaliação prática para Época Especial deverá ser realizado individualmente, não se aceitando a formação de grupos.

Do ponto de vista do planeamento, os alunos deverão ter em consideração que o tempo de execução e a memória usada serão tidos em conta na avaliação do projecto submetido. Por essas razões, a representação dos dados necessários à resolução dos problemas deverá ser criteriosamente escolhida tendo em conta o espaço necessário, mas também a sua adequação às operações necessárias sobre eles.

Serão admissíveis para avaliação versões do programa que não possuam todas as funcionalidades, seja no que à primeira fase de submissões diz respeito, como para a fase final. Naturalmente que um menor número de funcionalidades operacionais acarretará penalização na avaliação final.

A avaliação do projecto decorre em três ou quatro instantes distintos. O primeiro instante coincide com a primeira submissão electrónica, onde os projectos serão avaliados automaticamente com base na sua capacidade de cumprir as especificações e funcionalidades definidas na Secção 4. Para esta fase existe apenas uma data limite de submissão (veja a Tabela 1) e não há qualquer entrega de relatório. O segundo instante corresponde à submissão electrónica do código na sua versão final e à entrega do relatório em versáo electrónica, por e-mail. Essa entrega ratifica e lacra a submissão electrónica anteriormente realizada.

Num terceiro instante há uma proposta enviada pelo corpo docente que pode conter a indicação de convocatória para a discussão e defesa do trabalho ou uma proposta de nota para a componente de projecto. Caso o aluno aceite a nota proposta pelo docente avaliador, a discussão não é necessária e a nota torna-se final. Se, pelo contrário, o aluno decidir recorrer da nota proposta, será marcada uma discussão de recurso em data posterior. O quarto instante acontece apenas caso haja marcação de uma discussão, seja por convocatória do docente, seja por solicitação do aluno.

As datas de entrega referentes aos vários passos da avaliação do projecto estão indicadas na Tabela 1.

As submissões electrónicas estarão disponíveis em datas e condições a indicar posteriormente na página da disciplina e serão aceites trabalhos entregues até aos instantes finais indicados.

Os alunos não devem esperar qualquer **extensão nos prazos de entrega**, pelo que devem organizar o seu tempo de forma a estarem em condições de entregar a versão final na data indicada.

Tabela 1: Datas importantes do Projecto

| Data | Documentos a Entregar |
|---|---|
| 10 de Março de 2023 | Enunciado do projecto disponibilizado por e-mail. |
| 05 de Maio de 2023, (18h00) | Conclusão da primeira fase de submissões. |
| 07 de Julho de 2023 (18h00) | Conclusão da fase final de submissões. |
| 21 de Julho de 2023 (18h00) | Entrega do relatório do projecto por e-mail. |

Note-se que, na versão final, o projecto só é considerado entregue aquando da entrega do relatório. As submissões electrónicas do código não são suficientes para concretizar a entrega.

## 5.1 Funcionamento

A verificação do funcionamento do código a desenvolver no âmbito do projecto será exclusivamente efectuada nas máquinas do laboratório da disciplina, embora o desenvolvimento possa ser efectuado em qualquer plataforma ou sistema que os alunos escolham. Esta regra será estritamente seguida, não se aceitando quaisquer excepções. Por esta razão, é essencial que os alunos, independentemente do local e ambiente em que desenvolvam os seus trabalhos, os verifiquem no laboratório antes de os submeterem, de forma a evitar problemas de última hora. Uma vez que os laboratórios estão abertos e disponíveis para os alunos em largos períodos fora do horário das aulas, este facto não deverá causar qualquer tipo de problemas.

## 5.2 Código

**Não deve ser entregue código em papel**. Os alunos devem entregar por via electrónica o código do programa (ficheiros `.h` e `.c`) e uma `Makefile` para gerar o executável. Todos os ficheiros (`*.c`, `*.h` e `Makefile`) devem estar localizados na directoria raiz.

O código deve ser estruturado de forma lógica em vários ficheiros (`*.c` e `*.h`). As funções devem ter um cabeçalho curto mas explicativo e o código deve estar correctamente indentado e com comentários que facilitem a sua legibilidade.

## 5.3 Relatório

O relatório deve ser entregue na altura indicada na Tabela 1 e não deverá exceder as 15 páginas.

O relatório do projecto deverá ser claro e conciso e deverá permitir que se fique a saber como o aluno desenhou e implementou a solução apresentada e as razões que justificam as opções tomadas. Ou seja, uma leitura do relatório deverá dispensar a necessidade de se inspeccionar o código para que fiquem claras as opções tomadas, justificações das mesmas e respectivas implementações.

Por exemplo, se um dado aluno necessitar usar pilhas como uma das componentes do projecto, deverá ser claro pela leitura do relatório que usa pilhas, onde as usa, porque as usa e qual a implementação que adoptou (tabela, lista simples, outra...), com respectiva justificação. Qualquer das principais componentes algorítmicas e/ou de estruturas de dados implementada deverá merecer este tipo de atenção no relatório.

O relatório deverá incluir os seguintes elementos:

- Uma capa identificando a disciplina, o projecto e o seu autor;
- Uma página com o índice das secções em que o relatório se divide;
- Uma descrição completa da arquitectura do programa, incluindo fluxogramas detalhados e um texto claro, mas sucinto, indicando a divisão lógica e funcional dos módulos desenvolvidos para a resolução do problema, explicitando os respectivos objectivos, as funções utilizadas e as estruturas de dados de suporte;

- Uma análise, formal e/ou empírica, dos requisitos computacionais do programa desenvolvido, tanto em termos da memória que utiliza como da complexidade temporal, com particular ênfase no custo das operações de processamento sobre os tipos de dados usados e/ou criados.

## 5.4 Critérios de Avaliação

Os projectos submetidos serão avaliados de acordo com a seguinte grelha:

- Testes passados na primeira submissão electrónica – 20%
- Testes passados na última submissão electrónica – 55%
- Estruturação do código e comentários – 5%
- Gestão de memória e adequada utilização de tipos abstractos – 5%
- Relatório escrito – 15%

Tanto na primeira como na submissão electrónica final, cada projeto será testado com vários ficheiros de problemas de diferentes graus de complexidade, onde se avaliará a capacidade de produzir soluções correctas dentro de limites de tempo e memória. Para o limite de tempo, cada um dos testes terá de ser resolvido em menos de 60 segundos. Para o limite de memória, cada um dos testes não poderá exceder 100MB como pico de memória usada. Cada teste resolvido dentro dos orçamentos temporal e de memória que produza soluções correctas recebe um ponto. Na primeira fase de submissão, embora se mantenha o mesmo limite de memória, o tempo limite para resolver qualquer teste será muito mais baixo do que os 60 segundos (por exemplo, na faixa dos 10 a 15 segundos), que ficam reservados apenas para os testes da fase final.

Um teste considera-se errado se, pelo menos, um dos problemas do ficheiro de entrada correspondente for incorrectamente resolvido.

Se o corpo docente entender necessário, face à complexidade dos problemas a resolver, poderão os limites de tempo e/ou memória ser alterados.

Caso o desempenho de alguma submissão electrónica não seja suficientemente conclusivo, poderá ser sujeita a testes adicionais fora do contexto da submissão electrónica. O desempenho nesses testes adicionais poderá contribuir para subir ou baixar a pontuação obtida na submissão electrónica.

No que à avaliação do relatório diz respeito, os elementos de avaliação incluem: apreciação da abordagem geral ao problema e respectiva implementação; análise de complexidade temporal e de memória; clareza e suficiência do texto, na sua capacidade de descrever e justificar com precisão o que está feito; e qualidade do texto escrito e estruturação do relatório.

Pela análise da grelha de avaliação aqui descrita, deverá ficar claro que a ênfase da avaliação se coloca na capacidade de um programa resolver correctamente os problemas a que for submetido. Ou seja, o código de uma submissão até pode ser muito bonito e bem estruturado e o aluno até pode ter dispendido muitas horas no seu desenvolvimento. No entanto, se esse código não resolver um número substancial de testes na submissão electrónica dificilmente terá uma nota positiva.

# 6 Código de Honestidade Académica

Espera-se que os alunos conheçam e respeitem o Código de Honestidade Académica que rege esta disciplina e que pode ser consultado na página da cadeira. O projecto é para ser planeado e executado individualmente e é nessa base que será avaliado. Quaisquer associações de grupos ou outras, que eventualmente venham a ocorrer, serão obviamente interpretadas como violação do Código de Honestidade Académica e terão como consequência a anulação do projecto aos elementos envolvidos.

Lembra-se igualmente que a verificação de potenciais violações a este código é feita de forma automática com recurso a sofisticados métodos de comparação de código, que envolvem não apenas a comparação directa do código mas também da estrutura do mesmo. Esta verificação é feita com recurso ao software disponibilizado em

`http://moss.stanford.edu/` .

# A Ficheiro de entrada desarrumado

Apresenta-se aqui um exemplo de ficheiro de entrada, contendo exactamente a mesma informação que o ficheiro da Figura 2, pelo qual terá passado alguma ventania como ilustração do que será um ficheiro que não esteja "arrumado".

```
                          8 10

     A1
1 2 20.5



                    1
4
                                        9.22 2 6 11.7
2 7 3.68

3 5 4.0 4 5 3.96 4
                                              6 5.3 4

                                       7

                     5.2 4

              8 4.8 6 8 5.1
```

Figura 6: Um ficheiro contendo um único problema.