DEEC
DEPARTAMENTO DE ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES
TÉCNICO LISBOA

Mestrado Integrado em Engenharia Electrotécnica e de Computadores (MEEC)

Algoritmos e Estruturas de Dados

Enunciado do Projecto

AEDMaps

Versão 1.00 (23/Março/2021)

2020/2021
2º 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.

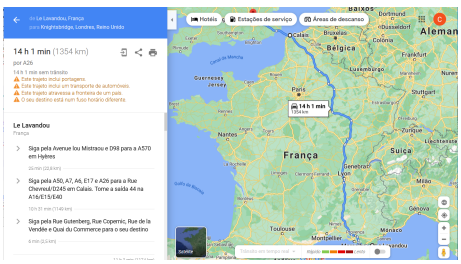
Para a primeira fase de avaliação o trabalho consiste apenas no desenvolvimento de algumas funcionalidades testáveis que podem ser usadas posteriormente para desenvolver a solução final.

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.

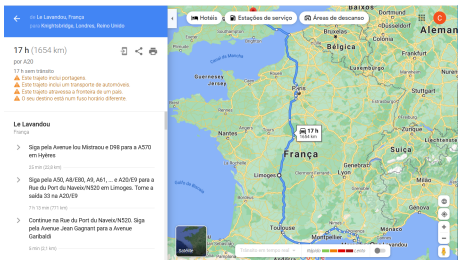
2 O problema – AEDMaps

Neste projecto pretende-se desenvolver um programa que seja capaz de produzir caminhos entre povoações de um mapa e que esses caminhos obedeçam a um determinado conjunto de restrições definidas previamente por algum viajante imaginário. Tipicamente o viajante pretenderá obter um percurso entre duas moradas e quer que o caminho produzido seja o mais barato possível, de acordo com alguma métrica pré-definida. No entanto, o viajante poderá pretender que o caminho produzido, para além de ser o mais barato possível, cumpra algum ou alguns requisitos culturais, estéticos, gastronómicos ou religiosos, só para dar alguns exemplos. Para além deste tipo de restrições definidas pelo viajante, podem também existir outras restrições que se prendam com a existência de interdição de entrada em determinadas povoações ou interdição de passagem em algumas estradas ou caminhos.

Num cenário normal é expectável que exista mais que uma forma de executar um qualquer percurso entre quaisquer duas povoações. No contexto deste projecto só interessam os percursos de menor custo. Quando mais que um deles tiver o custo mínimo, é indiferente qual deles escolher a menos que um deles não cumpra as especificações inicialmente definidas. Pode também acontecer que não exista caminho, porque as povoações de partida e chegada não possuem qualquer caminho que as una ou porque nenhum dos caminhos que as unem satisfaz as restrições impostas.



S



T

Figura 1: Caminho mais curto entre Le Lavandou, em França, e Knightsbridge, no Reino Unido, (S), e caminho mais curto entre os mesmos pontos com passagem obrigatória em Limoges, famosa pela sua loiça esmaltada, (T).

Pode também dar-se o caso que, uma vez iniciado o percurso, algum evento extraordinário e imprevisível faça com que alguma parte do caminho fique interditada. Assim, convém partir com informação de rotas alternativas para usar na eventualidade de o caminho originalmente calculado ficar comprometido.

Na Figura 1 apresenta-se um caso muito simples de um caminho entre uma povoação francesa e um bairro londrino (à esquerda) e uma outra solução para o mesmo problema que inclui passagem por uma zona industrial de produção de loiça esmaltada (à direita), como ilustração.

Após ser obtido um caminho que cumpra as especificações explicitadas, pode ser necessário identificar algumas características da solução obtida em termos da sua robustez/flexibilidade a interrupção de estradas e/ou inacessibilidade temporária de povoações.

Assumindo que um qualquer mapa é identificado como um grafo em que as povoações são os vértices e as estradas ou caminhos que as unem são as arestas, 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 AEDMaps

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

A1 – O caminho de custo total menor entre um ponto de partida e um ponto de chegada;

B1 – O caminho de custo total menor entre um ponto de partida e um ponto de chegada com passagem por algum ponto de interesse com eventuais restrições de custo;

C1 – O mesmo que em A1 com indicação do custo adicional do percurso alternativo em caso de interdição de alguma das povoações no caminho original;

D1 – O mesmo que em A1 com indicação do custo adicional do percurso alternativo em caso de interdição de alguma das vias que o constituem;

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 de arestas, T, tal que não exista nenhum outro subconjunto de menor custo total e que forme um caminho unindo as duas povoações indicadas como argumento.

A variante $B1$ difere da variante $A1$ na medida em que pode ser especificado que o caminho tenha de conter, pelo menos, um ponto de passagem que satisfaça algum critério qualitativo[1]. De todos os caminhos que cumpram essa especificação o caminho produzido tem necessariamente que ser o de menor custo. Adicionalmente, o utilizador pode especificar um limite de custo percentual face ao custo do caminho absoluto mais curto. Se o melhor caminho que passa por uma das povoações de interesse exceder o custo original num percentual especificado, o utilizador optará pelo caminho mais curto absoluto sem passagem pelo ponto de interesse. Por exemplo, se o utilizador no seu caminho pretender visitar um monumento megalítico mas o desvio relativamente ao caminho mais curto acarretar um custo total que esteja acima de um limiar percentual pré-definido para um caminho que não contenha esse ponto de interesse, a opção é não visitar qualquer monumento megalítico. Nestes casos, quando a opção final acaba sendo pelo caminho mais curto absoluto, um problema de variante $B1$ não tem solução.

Na variante $C1$, especifica-se que alguma povoação está interdita a passagem e o programa tem de determinar um caminho alternativo que a não inclua e que seja o de menor custo que não passe por ela. Note-se que se a povoação interdita não fizer parte do caminho original não será necessário determinar o caminho alternativo.

Na variante $D1$, especifica-se alguma via ligando duas povoações que está interdita a passagem e o programa tem de determinar um caminho alternativo que a não inclua e que seja o de menor custo que não use essa via. Também aqui não é necessário determinar o caminho alternativo se a via interdita não estiver contida no conjunto de vias do caminho original.

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 AEDMaps deverá ser invocado na linha de comandos da seguinte forma:

```
aed$ ./aedmaps [OPTION] [PROBLEMS] [MAPS]
```

`aedmaps` é o nome do ficheiro executável contendo o programa AEDMaps;

`OPTION` é uma cadeia de caracteres iniciada sempre com o carácter `'-'` e explicita a existência de opções diversas de funcionamento do programa, descritas abaixo;

`PROBLEMS` está reservado a um ficheiro de entrada contendo apenas cabeçalhos de problemas a serem resolvidos;

`MAPS` está reservado a um ficheiro de entrada contendo apenas mapas;

[1] Optou-se por propor uma versão muito simples deste problema em que apenas exista intenção de passar por um ponto de interesse, ao invés de uma uma eventual colecção.

As opções admissíveis, sendo as únicas que o programa deve estar preparado para executar, são:

-oo – significa que se pretende apenas resolver o primeiro problema do ficheiro de problemas no primeiro mapa do ficheiro de mapas;

-oa – significa que se pretende resolver o primeiro problema do ficheiro de problemas para todos os mapas;

-ao – significa que se pretende resolver todos os problemas apenas no primeiro mapa do ficheiro de mapas;

-aa – significa que se pretende resolver todos os problemas em todos os mapas.

Um ficheiro de problemas terá sempre que ser o primeiro dos dois ficheiros na linha de comando, quando se invoca o programa e deverá ter extensão ``.prbs''. Um ficheiro de extensão ``.prbs'' pode conter um ou mais problemas diferentes. Um ficheiro de mapas terá sempre que ser o segundo dos dois ficheiros na linha de comando, quando se invoca o programa e deverá ter extensão ``.maps''. Um ficheiro de mapas pode conter um ou mais mapas diferentes.

Para cada par problema/mapa 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 povoação que cumpra a especificação qualitativa em variante $B1$ ou o seu custo exceda o que o viajante está disposto a aceitar, não existe solução.

3.2 Formato de entrada

O ficheiro de extensão `.prbs` contém um ou mais problemas. Cada problema possui apenas um cabeçalho que o identifica e caracteriza. A variante $A1$ possui o cabeçalho mais simples: contém uma cadeia de caracteres de tamanho dois especificando a variante, (``A1''), seguida de dois inteiros, v_i e v_j, identificando o ponto de partida e o ponto de chegada.

O cabeçalho da variante $B1$ possui sempre 5 elementos: uma cadeia de caracteres de tamanho dois especificando a variante em causa, (``B1''), seguida de dois inteiros, v_i e v_j; após estes surge um carácter (pode ser lido tanto como `char` ou como cadeia de caracteres) e depois um número real especificando a percentagem de desvio aceitável. Este último elemento, quando for negativo significa que não existe limite para o custo adicional face ao caminho mais curto sem restrição de passagem.

O cabeçalho da variante $C1$ possui quatro elementos: os mesmos três que a variante $A1$ seguidos de um inteiro especificando qual a povoação interdita. Se k for o inteiro apresentado significa ser interdita a povoação que ocupa a k-ésima posição do caminho, contando o ponto de partida como estando na primeira. Os cabeçalhos da variante $D1$ são semelhantes aos da variante $C1$: os mesmos três que a variante $A1$ seguidos de um inteiro identificanto a posição no caminho da via interdita a passagem, contando como primeira a via que liga a povoação de partida à segunda povoação visitada. Por exemplo,

um cabeçalho para este problema poderia ser: D1 1 2 3. Significa que, contando por etapa o percurso entre duas povoações consecutivas, pretende-se determinar o caminho mais curto entre a povoação 1 e a 2 e ainda indicar qual o custo adicional incorrido se a terceira etapa for interdita.

No caso da variante $B1$ os objectivos de passagem estão codificados por caracteres minúsculos. Por exemplo, se o quarto elemento do cabeçalho for `''d''` significa que os únicos caminho entre v_i e v_j que são elegíveis têm forçosamente que passar por, pelo menos, uma povoação que possua classificação `''d''`. Caso o ponto de partida ou o ponto de chegada possuam a classificação de interesse, serve como solução o caminho mais curto absoluto, sem desvios.

O ficheiro de extensão `.maps` contém um ou mais mapas. Cada mapa é identificado por um cabeçalho e um grafo. O cabeçalho define as dimensões do grafo. O grafo consiste num conjunto de vértices, que representam as povoações, e um conjunto de arestas bidireccionais ligando pares de vértices, que representam as vias. Para além disso, cada uma das povoações possui um conjunto de classificadores. Uma mesma povoação pode possuir todos os classificadores, um subconjunto dos classificadores existentes ou mesmo nenhum. A maioria das povoações não possui qualquer classificador. Como os classificadores são caracteres alfabéticos minúsculos, pode assumir-se que nenhum mapa possuirá mais que 26 classificadores diferentes.

Após o cabeçalho, que identifica o mapa, o ficheiro contém a informação da classificação dos vértices, através de uma cadeia de caracteres para cada um deles. Após o vector de classificadores, surge 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;
- V linhas com v_i `[STRING]`, para todo o $1 \leq v_i \leq V$;
- E linhas com v_i v_j c_{ij}, para $v_i \neq v_j$, com $v_i, v_j = 1, 2, \ldots, V$ e com $c_{ij} \in \mathbb{R}^+$.

Por exemplo, um mapa poderia ser definido como se mostra abaixo.

No exemplo apresentado, o grafo possui 8 vértices/povoações e 10 arestas/vias. O vértice 7 não possui classificação e o vértice 5 possui classificação do tipo `a` e do tipo `c`. 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 usando aquela aresta 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[2]. Os procedimentos de leitura a adoptar pelos alunos deverão ser robustos a qualquer tipo de desarrumação.

Independentemente da forma como os dados estão distribuídos no ficheiro de mapas, garante-se o seguinte: cada novo grafo inicia-se com dois inteiros positivos, V e E; após estes dois inteiros, existirão **sempre** V pares com um inteiro e uma cadeia de caracteres, em que os inteiros são garantidamente maiores ou iguais a 1 e menores ou iguais a V. 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).

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

```
8 10
1 -
2 a
3 d
4 -
5 ca
6 b
7 -
8 bd
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: Informação de um único mapa.

De igual forma, os ficheiros de extensão `.prbs` estão sempre correctos do ponto de vista do seu formato. Isto é, cabeçalhos de problemas da variante $A1$ são sempre constituídos por três elementos com a tipologia anteriormente descrita; os cabeçalhos de problemas de variante $B1$ possuem sempre cinco elementos com a tipologia e ordem descritas; os cabeçalhos de problemas de variante $C1$ possuem sempre quatro elementos com a tipologia descrita; e os cabeçalhos de problemas de variante $D1$ possuem sempre cinco elementos com a tipologia descrita anteriormente.

Os ficheiros com um ou mais problemas poderão ter um qualquer nome mas têm obrigatoriamente a extensão `.prbs` Os ficheiros com um ou mais mapas poderão ter qualquer nome, mas têm obrigatoriamente a extensão `.maps`. Assume-se que todos os ficheiros de extensão `.maps` estão correctos e no formato especificado anteriormente, no que à sequência e tipologia dos dados de cada mapa diz respeito.

O programa não necessita fazer qualquer verificação de correcção do formato dos ficheiros de entrada. Apenas necessita de garantir que apenas é executado quando as extensões estão correctas, que os ficheiros passados como argumento existem de facto, interpretando correctamente o seu conteúdo semântico.

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

3.3 Formato de saída

O resultado da execução do programa AEDMaps consiste em apresentar primeiro um cabeçalho do ficheiro seguido de um cabeçalho de cada solução produzida, baseado no cabeçalho do problema e do mapa a que se aplica, acrescentando-lhe mais informação. Após o cabeçalho da solução, deverá ser apresentada a solução do problema. Cada uma das quatro variantes possui um formato específico de construção de cabeçalho e apresentação de solução, quando uma existe. Estes formatos descrevem-se abaixo.

O cabeçalho do ficheiro deverá ser colocado na primeira linha do ficheiro de saída e consiste na indicação dos argumentos usados na invocação do programa. Por exemplo, se um utilizador escrever

```
aed$ ./aedmaps -oa alguns.problemas.prbs alguns.mapas.maps
```

a primeira linha do ficheiro conterá `-oa alguns.problemas.prbs alguns.mapas.maps`. Os cabeçalhos das soluções deverão ser escritos numa mesma linha. Quando os problemas a resolver possuam solução, esta caracteriza-se por um conjunto de arestas que definem o caminho. Na apresentação dos caminhos é obrigatório que cada linha possua apenas uma das arestas, pela ordem em que devem ser percorridas.

```
8 10 A1 3 1 3 17.18
3 5 4.00
5 4 3.96
4 1 9.22
```

Figura 3: A solução da variante $A1$ para o mapa da Figura 2 quando se pede o caminho mais curto entre o vértice 3 e o vértice 1.

(A1)- O cabeçalho da solução desta variante concatena o cabeçalho do mapa e do problema que a ele se aplicou após o que apresenta o número de passos do caminho e o seu custo. Após o cabeçalho deverão ser apresentadas todas as arestas que fazem parte do caminho e o seu custo individual. Ver Figura 3 para a solução do grafo da Figura 1 para o qual se pediu a determinação do caminho mais curto entre o vértice 3 e o vértice 1. A solução possui três passos e o custo total é de 17.18. Se não houver solução para o problema pedido, depois da concatenação dos dois cabeçalhos apresenta-se apenas o inteiro -1.

(B1)- O cabeçalho da solução desta variante concatena o cabeçalho do mapa com o cabeçalho do problema que a ele se aplicou seguido da mesma informação para problemas de variante $A1$. Ver Figura 4 para a solução do grafo da Figura 1 quando se pretende viajar do vértice 3 ao vértice 1 passando por um vértice que tenha a classificação `''b''` sem restrições no valor adicional de custo. O vértice que cumpre esta especificação está a negrito para facilidade de leitura. Note-se que se houvesse um limite de 50% para o excesso associado ao desvio, o cabeçalho do problema teria o número 0.5 em vez do número -1 e o cabeçalho da solução teria -1 onde está agora o número 5 e a solução estaria concluída apenas com a apresentação do cabeçalho que terminaria com este último -1, por não haver caminho que satisfaça as restrições definidas.

```
8 10 B1 3 1 b -1 5 26.78
3 5 4.00
5 4 3.96
4 8 4.80
8 4 4.80
4 1 9.22
```

Figura 4: Apresentação da solução de um problema de variante $B1$ para o mapa da Figura 2.

(C1)- O cabeçalho da solução de problemas em variante $C1$ concatena o cabeçalho do mapa com o cabeçalho do problema que lhe foi aplicado, seguido do número de passos e custo do caminho original sem restrições, como em $A1$. Após esta informação deverá ter a informação do caminho alternativo apenas em termos do seu custo adicional. Se o caminho original custar 20.34 e o caminho que não usa o vértice interditado custar 24.40, o último elemento do cabeçalho da solução é 4.06. Caso o vértice interditado não faça parte do caminho original ou a sua interdição, quando faça, impede que se atinja o vértice destino, o último elemento do cabeçalho será -1. De igual modo, se o caminho sem restrições não possuir solução, após o -1 indicando que não há solução não é necessário indicar que não existe caminho alternativo. Após o cabeçalho, sempre que o problema original possua solução, deverão ser apresentados os passos desse caminho, tal como na variante $A1$ ou $B1$.

(D1)- O cabeçalho da solução de problemas em variante $D1$, concatena o cabeçalho do mapa com o cabeçalho do problema que lhe foi aplicado, seguido da mesma informação já referida na variante $C1$.

Quando o utilizador especifica que se deverão resolver todos os problemas do ficheiro de problemas em todos os mapas do ficheiro de mapas, a ordem de apresentação das soluções deverá ser todos os problemas aplicados ao primeiro mapa seguidos de todos os problemas aplicados ao segundo mapa, etc..

Sublinha-se que é possível que um determinado ficheiro de problemas possua mais que um problema e que o de mapas também possua mais que um mapa, mas o ficheiro de soluções só contém uma solução, caso o utilizador tenha usado a opção `-oo`.

Para facilitar a interpretação visual 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 mapas mas **com extensão substituída por** `.routes`. Este ficheiro deve ser criado e aberto pelo programa. Por exemplo, se o ficheiro com problemas se chama `teste231.prbs` e o ficheiro de mapas se chamar `aqui.tenho.mapas.maps`, o ficheiro de saída deve-se chamar `aqui.tenho.mapas.routes`. Note-se que em situações em que haja erro na passagem de argumentos ao programa não faz qualquer sentido criar um ficheiro de saída.

Se o programa for invocado com ficheiros inexistentes, que não possuam as extensões `.prbs` ou `.maps`, sem qualquer argumento, ou com argumentos a menos ou 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 mapas para extracção de informação a partir dos mesmos.

O formato de invocação do programa será o mesmo que o definido anteriormente, mas existem algumas diferenças de conteúdo.

```
aed$ ./aedmaps [OPTION] [PROBLEMS] [MAPS]
```

O executável tem o mesmo nome e continuarão ser passados três argumentos. As cadeias de caracteres especificando a opção de funcionamento possuem um carácter extra para as distinguir das opções da fase final. Por exemplo, `-oo` é substituída por `-1oo`, tendo o mesmo significado. Todas as opções da primeira fase têm o carácter `'1'` após `'-'`. Os ficheiros de problemas possuem extensão `.prbs1` para os distinguir dos ficheiros de problemas da fase final. Já os ficheiros de mapas não têm qualquer alteração entre as duas fases, possuindo a mesma extensão.

Aqui também os sucessivos problemas, no ficheiro de extensão `.prbs1`, 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 – existência de vizinhos a k etapas;

D0 – contagem de todos os vizinhos a k etapas.

Em variante $A0$ pretende-se determinar o **grau** de um vértice/povoação passado como argumento do problema no seu cabeçalho. Define-se como grau de um vértice o número de arestas/vias que o têm como um dos extremos. Em variante $B0$ pretende-se saber se dois vértices/povoações passados como argumento no cabeçalho do problema possuem uma via directa que os una. Em variante $C0$ pretende-se saber se o vértice/povoação passado como argumento do problema no seu cabeçalho possui algum outro vértice/povoação à distância exacta de k etapas. Define-se como distância exacta entre dois vértices o menor número de arestas que é necessário percorrer para ir de um a outro. Para $k > 1$ se dois vértices/povoações forem adjacentes é falso que possam estar à distância k um do outro,

uma vez que estão à distância 1, e $1 < k$. Na variante $D0$ pretende-se contar todos os vértices que estejam à distância exacta de k etapas de um dado vértice.

O cabeçalho de cada problema é definido por: *Variante* v_i $[v_j]$ $[k]$. *Variante* é uma cadeia de caracteres que assume o valor de uma das quatro variantes acima descritas, v_i é um vértice, v_j é um segundo vértice, cuja ocorrência só se dá em variante $B0$ e k é um inteiro que apenas ocorre nos cabeçalhos de problemas de variantes $C0$ e $D0$.

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 via quando exista (escrito com duas casas decimais apenas), e -1 quando não exista via 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 não existir nenhum vértice à distância exacta de k etapas e 1 se existir, pelo menos, um. Para problemas de variante $D0$ a resposta será -1 se o vértice não pertencer ao grafo, 0 se não existir qualquer vértice à distância exacta de k etapas ou algum $p > 0$ se existirem p vértices distintos que estejam à distância exacta de k etapas.

Por exemplo, para o mapa da Figura 2, poder-se-ia ter o ficheiro de problemas apresentado na Figura 5.

```
A0 4
```

Figura 5: Exemplo de 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 $C0$ com $k = 3$, o resultado seria

```
8 10 C0 4 3 0
```

porque todos os vértices estão à distância exacta de 1 ou de 2 relativamente ao vértice 4.

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 mapas, mas deverá ter extensão `.queries` e deverá incluir um cabeçalho de ficheiro, igual ao já descrito para a fase final, e todos os resultados associados com cada um dos problemas resolvidos. Este ficheiro deverá conter apenas uma linha por problema: concatena o cabeçalho do mapa com o cabeçalho do problema seguido do resultado obtido, de acordo com a descrição feita na secção anterior.

Qualquer dos ficheiros de entrada (de problemas e de mapas) poderá conter mais do que um problema e mais que um mapa para resolver e cada um desses problemas e mapas poderão ter diferentes dimensões. O número de problemas a emparelhar com os mapas existentes é condicionado pela opção escolhida na linha de comando.

O ficheiro de saída será a concatenação das soluções de todos os problemas resolvidos. 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

O projecto está dimensionado para ser feito por grupos de dois alunos, não se aceitando grupos de dimensão superior nem inferior. Para os alunos que frequentam o laboratório, o grupo de projecto não tem de ser o mesmo do laboratório, mas é aconselhável que assim seja.

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.

Quando os grupos de projecto estiverem constituídos, os alunos devem obrigatoriamente inscrever-se no sistema Fenix, no grupo de Projecto correspondente, que será criado oportunamente.

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 mãos aos docentes, entrega essa que ratifica e lacra a submissão electrónica anteriormente realizada. Na submissão final é possível submeter o projecto e entregar o relatório durante três dias consecutivos. No entanto, entregas depois da primeira data sofrerão uma penalização (veja a Tabela 1).

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 os alunos aceitem a nota proposta pelo docente avaliador, a discussão não é necessária e a nota torna-se final. Se, pelo contrário, os alunos decidirem 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 dos alunos. Nestas circunstâncias, a discussão é obrigatoriamente feita a todo o grupo, sem prejuízo de as classificações dos elementos do grupo poderem vir a ser distintas.

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 primeira data indicada. As restantes datas devem ser encaradas como soluções de recurso, para a eventualidade de alguma coisa correr menos bem durante o processo de submissão. O relatório final deverá ser entregue em mão aos docentes no laboratório no dia indicado na Tabela 1.

Tabela 1: Datas importantes do Projecto

Data	Documentos a Entregar
23 de Março de 2021	Enunciado do projecto disponibilizado na página da disciplina.
até 21 de Abril de 2021 (18h00)	Inscrição dos grupos no sistema Fenix.
28 de Abril de 2021, (18h00)	Conclusão da primeira fase de submissões.
24 de Maio de 2021, 2ª feira 15h 18h	**1ª Data de entrega do projecto:** Fim da submissão electrónica em primeira data. Entrega do relatório do projecto via Fénix.
25 de Maio de 2021, 3ª feira 15h 18h	**2ª Data de entrega do projecto: penalização de um (1) valor** Fim da submissão electrónica em segunda data. Entrega do relatório do projecto via Fénix.
26 de Maio de 2021, 4ª feira 15h 18h	**3ª Data de entrega do projecto: penalização de dois (2) valores** Fim da submissão electrónica em terceira data. Entrega do relatório do projecto via Fénix.
	Submissões posteriores à terceira data têm penalização de 20 valores.
até 5 de Julho de 2021	Eventual discussão do trabalho (data combinada com cada grupo).

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. Um grupo que faça a submissão electrónica do código e a entrega do relatório, por exemplo, na 1ª data de entrega pode fazer submissões nas datas seguintes, mas se fizer a entrega de um novo relatório em papel, será este, e as respectivas submissões, o considerado para avaliação, com a penalização indicada. De modo semelhante, aos grupos que façam a sua última submissão electrónica na primeira data, mas entreguem o relatório numa das outras duas datas posteriores, será contada como data de submissão aquela em que o relatório for apresentado.

Face à actual situação relativa à epidemia do COVID-19 importa sublinhar que os prazos definidos na Tabela 1 poderão vir a sofrer alterações, pendente de decisões dos órgãos executivos da escola relativas ao funcionamento do semestre.

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

Os relatórios devem ser entregues na altura indicada na Tabela 1. O relatório do projecto deverá ser claro e conciso e deverá permitir que se fique a saber como o grupo desenhou e implementou a solução apresentada. 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 grupo 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 com os dados dos membros do grupo, incluindo nome, número e e-mail. Esta capa deverá seguir o formato indicado na página da disciplina (oportunamente será disponibilizado);
- 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 computacional, com particular ênfase no custo das operações de processamento sobre os tipos de dados usados e/ou criados;
- Pelo menos, um pequeno exemplo completo e detalhado de aplicação, com descrição da utilização das estruturas de dados em cada passo e de como são tomadas as decisões.

5.4 Critérios de Avaliação

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

- Testes passados na primeira submissão electrónica – 10% a 15%
- Testes passados na última submissão electrónica – 60% a 65%
- Estruturação do código e comentários – 5%
- Gestão de memória e 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; exemplo de aplicação; 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 grupo 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 por grupos de dois alunos 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.

Lembramos 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


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: Exemplo de um ficheiro contendo um único mapa.