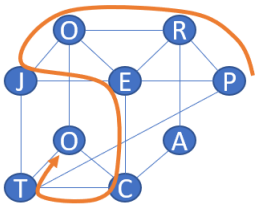
DEEC
DEPARTAMENTO DE ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES
TÉCNICO LISBOA

Mestrado Integrado em Engenharia Electrotécnica e de Computadores (MEEC)

Algoritmos e Estruturas de Dados

Enunciado do Projecto



WordZ

Versão 1.03 (11/Março/2020) – distribuição inicial

2019/2020
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.

O trabalho a realizar para a primeira fase de avaliação assenta no mesmo problema, mas consiste 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 – WordZ

Neste projecto pretende-se desenvolver um programa que seja capaz de produzir palavras com **três ou mais letras** a partir de um conjunto dado de caracteres. Cada palavra produzida terá que fazer parte de um dicionário fornecido para o efeito.

As palavras formadas têm de corresponder a um caminho de vértices adjacentes ao longo de um grafo, em que nenhum dos vértices do grafo pode ser usado mais que uma vez para cada palavra. Os vértices adjacentes obtêm-se por movimentos ao longo das arestas que os unem (Ver Figura 1).

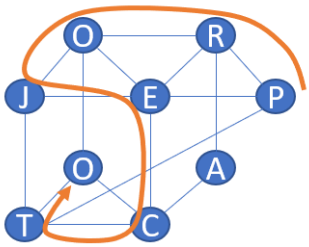


Figura 1: Exemplo de um caminho num grafo em que se produz a palavra "PROJECTO".

Em geral cada vértice contém apenas um carácter, como ilustrado na figura, mas qualquer dos vértices do grafo pode conter mais de um carácter. Isto permite definir prefixos, infixos, sufixos, e pares de caracteres que poderão ser usados em alternativa. Abaixo itemizam-se exemplos e descrição dos quatro casos.

- Um dado vértice pode conter um prefixo, por exemplo, `‘‘pro-’’`, significando que cada caminho que o use tem obrigatoriamente de iniciar-se nele – `‘‘pro-’’+‘v’+‘a’` corresponde à palavra `‘‘prova’’`;

- Um dado vértice pode conter, por exemplo, a cadeia `‘‘vas’’`, significando que cada caminho que use esse vértice produz uma palavra que possui aqueles três caracteres na ordem original – `‘p’+‘r’+‘o’+‘‘vas’’` corresponde à palavra `‘‘provas’’` ou `‘‘vas’’+‘a’` que corresponde à palavra `‘‘vasa’’`;

- Um dado vértice pode conter uma cadeia começada pelo carácter `‘-’`, por exemplo, `‘‘-das’’`, significando que cada caminho que use esse vértice não pode continuar depois dele. Isto é, a cadeia presente no vértice é um sufixo – `‘t’+‘o’+‘‘-das’’` corresponde à palavra `‘‘todas’’`;

- Um dado vértice pode conter, por exemplo, a cadeia `‘‘c/d’’`, significando que cada caminho que use esse vértice produz uma palavra usando o carácter `‘c’` ou o `‘d’` em alternativa – `‘a’+‘‘c/d’’+‘h’+‘a’`, corresponde à palavra `‘‘acha’’` e `‘‘c/d’’+‘a’+‘r’’` que corresponde à palavra `‘‘dar’’`.

Outra restrição do problema diz respeito à existência de mais que um percurso que produza a mesma palavra. Quando tal acontece, apenas é contabilizada a palavra uma vez. Quando dois, ou mais, percursos diferentes produzem a mesma palavra, as ocorrências posteriores à primeira não são contabilizadas. Na Figura 2 ilustra-se um caso de uma mesma palavra ser obtida com dois percursos diferentes.

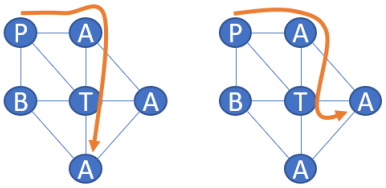


Figura 2: Exemplo de dois percursos diferentes que produzem a mesma palavra.

Cada vértice possui também um inteiro representando o valor de usar a letra ou cadeia de caracteres que possui, introduzindo uma segunda dimensão de avaliação da qualidade de um jogador. Ou seja, pode contar o número total de palavras formadas ou contar o valor total das palavras formadas.

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 restrições gerais do problema e as específicas 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 WordZ

O programa a desenvolver deverá ler e armazenar um dicionário, em qualquer língua de alfabeto latino, não usando acentos por questão de simplicidade.

Também lerá, de outro ficheiro, um ou mais grafos cujos vértices contêm caracteres e valores. Será com estes grafos que terá de formar palavras que pertençam ao dicionário utilizado, obedecendo às regras do problema.

Nas possíveis variantes do problema a resolver quer-se obter:

A1 – Uma palavra qualquer de tamanho k cujo caminho se inicia no vértice v_0;

B1 – Todas as palavras diferentes de tamanho k com início no vértice v_0;

C1 – Todas as palavras diferentes com início no vértice v_0;

D1 – A palavra mais comprida com início no vértice v_0;

E1 – A mais valiosa palavra de tamanho k com início no vértice v_0;

F1 – A mais curta palavra de valor k com início no vértice v_0;

A2 – Uma palavra qualquer de tamanho k;

B2 – Todas as palavras diferentes de tamanho k;

C2 – Todas as palavras diferentes;

D2 – A palavra mais comprida;

E2 – A mais valiosa palavra de tamanho k;

F2 – A mais curta palavra de valor k.

Em qualquer das variantes, o programa deverá escrever um ficheiro de saída com todas as palavras construídas, o caminho que as produz e sua respectiva pontuação. 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.

No momento em que se publica este enunciado está previsto que nem todas as 12 variantes acima explicitadas tenham de estar implementadas na submissão final. A decisão de quais daquelas 12 variantes irão ser avaliadas será tomada quando se concluir a primeira fase de submissão. Nessa altura será afixado anúncio na página da disciplina, sob o separador Projecto, indicando quais as variantes que deverão estar implementadas para a cotação máxima.

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

```
aed$ wordz <nome1>.dict <nome2>.graph
```

`wordz` designa o nome do ficheiro executável contendo o programa WordZ;

`<nome1>.dict` em que `<nome1>` é variável, identificando o ficheiro dicionário.

`<nome2>.graph` em que `<nome2>` é variável, identificando o ficheiro contendo um ou mais grafos para resolver, de acordo com a informação que os precede.

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 palavra no tamanho pedido não existe solução.

3.2 Formato de entrada

O ficheiro dicionário contém palavras separadas por espaços em branco ou mudanças de linha. Por exemplo, um ficheiro dicionário em português poderá ser

```
aba
ababalhar
            abacate
abertura          alvo                amora


alto
...
zurzir
...
```

O ficheiro dicionário só contém palavras com caracteres minúsculos, não contendo caracteres acentuados ou especiais, como cedilhas, nem palavra hifenadas. Os ficheiros contendo dicionários não possuem palavras repetidas e não têm necessariamente que estar alfabeticamente ordenados.

O ficheiro de grafos pode conter mais que um problema. 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 e um conjunto de arestas bidireccionais ligando pares de vértices.

Qualquer das 12 variantes possui um cabeçalho com o mesmo formato: dois inteiros, seguidos de uma string, seguida de dois inteiros: os primeiros dois inteiros correspondem ao número de vértices, V, e ao número de arestas, E; a string, O, define a opção de processamento; os últimos dois inteiros correspondem ao valor objectivo, k, (em tamanho ou valor) e ao vértice de partida, v_0. A string pode ser uma de doze: `''A`i`''`, `''B`i`''`, `''C`i`''`, `''D`i`''`, `''E`i`''`, `''F`i`''`, com $i = 1, 2$. Excepto para os casos em que a string seja `''C`i`''` ou `''D`i`''` a definição completa do problema requer que se indique o valor de k. Para simplificar o procedimento de leitura nestes dois casos, todos os problemas possuem sempre dois inteiros na quarta e na quinta posição, antes da apresentação dos vértices e arestas. Para as variantes `''C`i`''` e `''D`i`''` o valor do quarto inteiro não carece qualquer validação. Nas opções em que o ponto de partida é irrelevante, $v_0 = 0$. Para todos os outros casos $v_0 = 1, 2, \ldots, V$. Se o vértice de partida for negativo ou maior que V o problema está mal definido, pelo que o seu resultado é indicar que não possui solução.

Após o cabeçalho, que identifica o problema, o ficheiro contém a informação dos vértices: identificador do vértice – um número entre 1 e V; a string contida no vértice; e o seu valor. Após a informação de todos os vértices, o ficheiro contém as arestas, identificadas por um par de inteiros que correspondem aos vértices que a aresta liga.

Em síntese:

- uma linha com V E O k v_0
- V linhas com v_i S_i w_i, para $i = 1, 2, \ldots, V$
- E linhas com v_i v_j, para $i \neq j$ e $i, j = 1, 2, \ldots, V$

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

```
9 16 A 4 0
1 o 1
2 r 3
3 j 5
4 e 1
5 p 4
6 o 1
7 a 1
8 t 6
9 c 3
1 2
1 3
1 4
1 6
2 4
2 5
2 7
3 4
3 8
4 5
4 9
5 8
6 8
6 9
7 9
8 9
```

No exemplo apresentado, o grafo possui 9 vértices e 16 arestas. O objectivo é produzir um caminho que possua uma palavra de tamanho 4 sem ponto de partida obrigatório. O vértice 4 possui a string ‘‘e’’ que vale 1 ponto e existe uma aresta entre o vértice 4 e o vértice 9. O vértice 9 contém a string ‘‘c’’ de valor 3. A existência de uma aresta entre estes dois vértices significa que tanto se pode passar do vértice 4 ao 9 como se pode fazer o percurso inverso. Por exemplo, se o dicionário fosse de língua inglesa o percurso 2-7-9-4 produziria a palavra ‘‘race’’ e o percurso 9-7-2-4 produziria a palavra ‘‘care’’, em que ambas pertencem a um dicionário completo de língua inglesa. Ambas

seriam soluções correctas para um problema de variante `A` sem ponto de partida fixo, mas apenas a primeira seria solução aceitável para ponto de partida $v_0 = 2$.

Sublinha-se aqui que os dados de cada problema não têm necessariamente que estar "arrumados" como se ilustra no exemplo do ficheiro de dicionário. 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[1] 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 string especificando a variante a resolver; depois da string especificando a variante existirão sempre dois inteiros: o valor do objectivo e o ponto de partida; após esta informação existirão **sempre** V conjuntos de um inteiro (sempre entre 1 e V), uma string e um segundo inteiro (sempre não negativo); depois destes V conjuntos haverá sempre E pares de inteiros 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 `.graph`. Assume-se que todos os ficheiros de extensão `.graph` 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.

Apenas por razões estéticas, haverá sempre, pelo menos, uma linha em branco entre dois grafos 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 Formato de saída

O resultado da execução do programa WordZ consiste em apresentar primeiro o cabeçalho do problema a que se acrescenta um inteiro, que deverá ser -1 quando o problema não tem solução ou um inteiro positivo indicando quantas palavras foram encontradas. Após o cabeçalho da solução, seguem-se todas as palavras que constituem solução do problema, uma por linha: primeiro a palavra; depois uma sequência de inteiros, representando os vértices pela ordem em que a palavra é produzida; e no final um inteiro. O inteiro final é o valor da palavra, obtido por somatório dos valores individuais das *strings* que a compõem.

Uma solução completa para o grafo da Figura 1, de acordo com o exemplo de ficheiro de entrada apresentado anteriormente, seria descrita no ficheiro de saída na forma seguinte:

```
9 16 A 4 0 1
care 9 7 2 4 8
```

[1] Ou mudança de linha ou tabular.

Neste exemplo indica-se que o grafo de entrada possui 9 vértices, 16 arestas, resolveu-se a variante `A` com $k = 4$, sem ponto de partida obrigatório e o problema tem solução com apenas uma palavra. Na segunda linha apresenta-se a palavra `‘‘care’’` obtida no caminho 9-7-2-4, com valor 8.

Se o ficheiro de extensão `.graph` 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** `.wordz`. Este ficheiro deve ser criado e aberto pelo programa. Por exemplo, se o ficheiro com problemas se chama `teste231.graph`, o ficheiro de saída deve-se chamar `teste231.wordz`. Note-se que, em situações em que haja erro na passagem de argumentos ao programa, não faz qualquer sentido criar um ficheiro de saída.

Para grafos e variantes em que a solução possua mais que uma palavra, como nas variantes `B`i e `C`i, estas devem ser apresentadas por ordem alfabética crescente e uma por linha.

Para um grafo em que nenhum caminho produza uma palavra de acordo com a especificação do problema a solução produzida deverá conter o inteiro -1 no final do cabeçalho da solução. Nestes casos a solução só possui cabeçalho. Por exemplo, para o grafo da Figura 1 se se pedisse variante `F`i com $k = 100$, a resposta seria `9 16 F 100` v_0 `-1`. A resposta deverá ser -1 para todos casos em que a definição do problema não corresponda a nenhuma das doze possibilidades. Se, por hipótese, a definição do problema fosse `9 16 a 4 1`, a resposta teria de ser `9 16 a 4 1 -1`. Outro caso possível sem solução é quando o vértice de partida é negativo ou superior a V.

Se o programa for invocado com ficheiros inexistentes, que não possuam a extensão `.graph` ou extensão `.dict`, 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` ou da invocação da função `exit`) 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 dicionários e 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 deverão ser passados também dois argumentos: um ficheiro de extensão **.dict** e um ficheiro de extensão `.graph0`. O primeiro contém um

dicionário e o segundo contém um ou mais grafos para um ou mais problemas diferentes. Aqui também os sucessivos problemas, no ficheiro de extensão `.graph0` 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:

A – validação de caminhos;

B – determinação de valor;

C – determinação de palavra;

D – validação de palavra.

Em todas as 4 variantes da primeira fase um problema define-se através de uma sequência de vértices do grafo dado. Em variante A pretende-se verificar se a sequência de vértices constitui um caminho válido. Ou seja, se não há repetição de vértices, se todos os vértices pertencem ao grafo e se cada dois vértices sucessivos possuem uma aresta que os una. Em variante B pretende-se calcular o valor associado com a sequência de vértices. Ou seja, pretende-se somar os valores de cada vértice na sequência. A única restrição para sequências nesta variante é que todos os vértices pertençam ao grafo, não se pretendendo verificar da existência de arestas ligando os vértices sucessivos. Em variante C pretende-se compor a palavra produzida pela sequência de vértices. As restrições para este problema são: todos os vértices da sequência pertencem ao grafo; nenhum prefixo pode estar noutra posição que não seja a inicial; nenhum sufixo pode estar noutra posição que não seja a final; em letras alternativas usar a primeira do par. Em variante D pretende-se validar a palavra produzida. Ou seja, verificar se o dicionário contém a palavra produzida pela sequência dada. As restrições para este problema são as mesmas que para a variante C. Note-se que nesta última variante, se um ou mais vértices possuirem caracteres para serem usados em alternativa o programa deverá determinar se alguma escolha produz uma palavra legal, só devendo responder com -1 se todas as possibilidades alternativas produzirem palavras ilegais.

O cabeçalho de cada problema é definido por

- V, número de vértices;
- E, número de arestas;
- O, variante;
- k, comprimento do caminho;

e o resto do problema é definido por

- v_1, v_2, ..., v_k, vértices do caminho;
- Informação dos vértices;
- Informação das arestas.

Por exemplo, para o exemplo da Figura 1 poder-se-ia ter

```
9 16 A 5
1 2 5 4 9
1 o 1
2 r 3
3 j 5
4 e 1
5 p 4
6 o 1
7 a 1
8 t 6
9 c 3
1 2
1 3
1 4
1 6
2 4
2 5
2 7
3 4
3 8
4 5
4 9
5 8
6 8
6 9
7 9
8 9
```

em que se pergunta se o caminho de cinco vértices 1-2-5-4-9 é um caminho válido para o grafo. Como este caminho é válido, a resposta será:

```
9 16 A 5 1
```

Se, em vez de A, tivesse sido pedida a variante B, o resultado seria

```
9 16 B 5 12
```

Para a variante C seria

```
9 16 C 5 1 orpec
```

e para um dicionário português, a variante D produziria

```
9 16 D 5 -1
```

Se se pedisse para validar o caminho 1-2-5-7-9 a resposta deveria ser

```
9 16 A 5 -1
```

dado que não existe aresta entre os vértices 5 e 7.

Para problemas de variante A a resposta será 1 quando o caminho é válido e -1 no caso contrário. Para problemas de variante B a resposta será o inteiro correspondente ao valor da sequência de vértices (que pode ou não ser um caminho no grafo) ou -1 se algum dos vértices da sequência não pertencer ao grafo. Para problemas de variante C a resposta será 1 seguido da palavra produzida ou -1 se houver alguma violação da restrições desta variante. Para problemas de variante D a resposta será 1 se a sequência de vértices contiver uma palavra legal ou -1 no caso contrário.

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 `.wordz0` 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. O resultado, para qualquer das variantes será sempre 1 ou -1 para cada uma das quatro variantes, excepto para a variante C em que se tem que apresentar a palavra, de acordo com a descrição acima. Qualquer variante diferente daquelas 4 tem de produzir -1 como resultado após o cabeçalho.

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.

Se algum problema estiver mal definido, deverá ser interpretado como um problema sem solução. Exemplos de problemas mal definidos são a variante pedida não ser A, B, C nem D.

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.

Note-se que, na versão final, o projecto só é considerado entregue aquando da entrega do relatório em papel. 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 em papel, 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.

Tabela 1: Datas importantes do Projecto

Data	Documentos a Entregar
16 a 20 de Março de 2020	Enunciado do projecto disponibilizado na página da disciplina.
até 10 de Abril de 2020 (18h00)	Inscrição dos grupos no sistema Fenix.
17 de Abril de 2020, (18h00)	Conclusão da primeira submissão.
20 de Maio de 2020, 4ª feira 12h 15h	**1ª Data de entrega do projecto:** Fim da submissão electrónica em primeira data. Entrega do relatório do projecto em papel.
21 de Maio de 2020, 5ª feira 12h 15h	**2ª Data de entrega do projecto: penalização de um (1) valor** Fim da submissão electrónica em segunda data. Entrega do relatório do projecto em papel.
22 de Maio de 2020, 6ª feira 12h 15h	**3ª Data de entrega do projecto: penalização de dois (2) valores** Fim da submissão electrónica em terceira data. Entrega do relatório do projecto em papel.
	Submissões posteriores a 22 de Maio têm penalização de 20 valores.
até 29 de Junho de 2020	Eventual discussão do trabalho (data combinada com cada grupo).

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.

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/` .