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

CAMP*ist*A

Versão 3.0 (15/Outubro/2019)

2019/2020
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. O trabalho a realizar para a primeira fase de avaliação assenta no mesmo problema, mas consiste no desenvolvimento de algumas funcionalidades que, apesar de não determinarem em absoluto a solução final, podem ser usadas posteriormente para ajudar na sua resolução. Assim, os alunos deverão encarar a primeira fase de avaliação como uma primeira etapa no trabalho de concepção e desenvolvimento da sua 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 – CAMP*ist*A

Neste projecto pretende-se desenvolver um programa que seja capaz de resolver problemas de colocação de tendas em parques de campismo de acordo com um conjunto de restrições. Os parques são mapas rectangulares de L linhas e C colunas e cada uma das $L \times C$ células pode estar vazia, ter uma árvore ou ter uma tenda. Cada tenda tem obrigatoriamente que ser colocada junto de uma árvore. Para cada árvore dentro do mapa apenas lhe pode ser atribuída uma tenda, no máximo, e esta terá de ocupar uma das quatro células adjacentes, não diagonais, àquela em que está a árvore (quando todas estejam dentro do mapa): norte, sul, este e oeste. Não é permitido colocar tendas em nenhuma das sete posições adjacentes, incluindo diagonais, de outra tenda já colocada (quando todas estejam dentro do mapa e estejam desocupadas)[1].

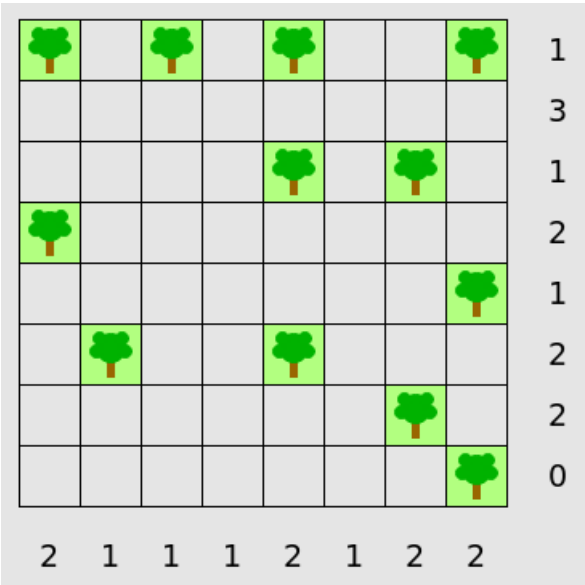

Figura 1: Um parque de campismo vazio.

[1]Faz-se notar que cada tenda possui um máximo de 8 células adjacentes, mas pelo menos uma delas possui necessariamente uma árvore.

Cada problema de colocação de tendas possui ainda dois vectores de inteiros indicando quantas tendas se devem colocar em cada linha e em cada coluna do mapa. A soma das tendas no vector das colunas e a soma das tendas no vector das linhas deverá ser igual ao número total de árvores no mapa.

Na Figura 1 ilustra-se um mapa inicial possível. O parque possui $L = 8$ linhas, $C = 8$ colunas e existem 12 árvores ao todo. Numerando as linhas de 0 a $L - 1$ e as colunas de 0 a $C - 1$, a solução deste problema obriga a colocar 3 tendas na linha 1 e 0 tendas na linha 7, assim como obriga a colocar 2 tendas na coluna 4.

Na Figura 2 apresenta-se a solução para o mapa da Figura 1, em que os triângulos representam as tendas colocadas. Deverá ser relativamente fácil verificar que as 12 tendas colocadas satisfazem todas as restrições do problema: adjacência não diagonal de cada tenda a uma árvore; não adjacência entre tendas; somatório de tendas em cada linha igual à restrição dessa linha; e o mesmo para as colunas.

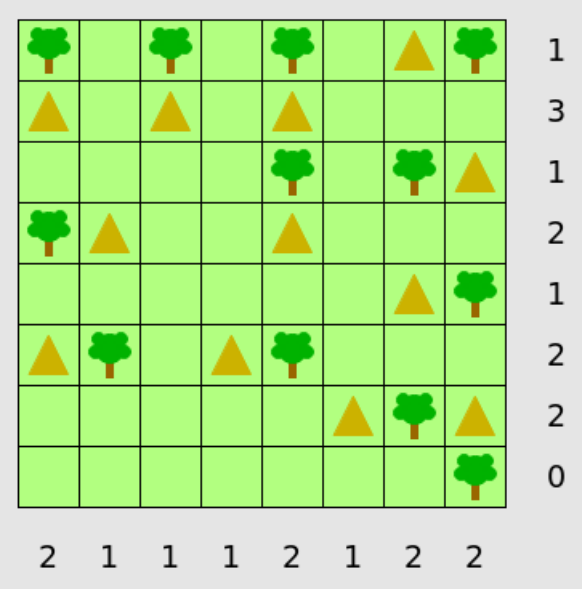

Figura 2: Um parque de campismo com a solução.

A definição original deste problema de satisfação de restrições – *(CSP - Constraint Satisfaction Problem)* – obriga a que na solução final exista sempre uma tenda por cada árvore no parque de campismo. Para o projecto de AED, com o objectivo de tornar o problema mais interessante, vamos considerar duas possibilidades: época alta e época baixa. Na época alta terá de ser colocada uma tenda por cada árvore (problema original). Na época baixa serão aceitáveis soluções com menos tendas que árvores (variante alternativa). A definição da época decorre dos valores apresentados nos dois vectores de inteiros já referidos. Quando a época é baixa o somatório do vector de linhas, sendo obrigatoriamente igual ao somatório do vector de colunas, é inferior ao número de árvores. Em época alta, estes somatórios serão sempre iguais ao número total de árvores, não podendo em caso algum ser superior.

O que se pretende neste projecto é desenvolver uma aplicação em linguagem C que seja capaz de resolver automaticamente um qualquer mapa, de acordo com as restrições gerais do problema e as específicas do mapa 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 "CAMP*ist*A"

O programa a desenvolver deverá ser capaz de ler mapas e produzir soluções para cada um deles ou indicar não haver solução, quando esse seja o caso.

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 CAMP*ist*A deverá ser invocado na linha de comandos da seguinte forma:

```
aed$ ./campista <nome>.camp
```

`campista` designa o nome do ficheiro executável contendo o programa CAMP*ist*A;

`<nome>.camp` em que `<nome>` é variável, identificando o ficheiro contendo o(s) mapas(s) onde se colocam tendas.

Para cada 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 o número de árvores de um mapa for inferior ao número de tendas pedidas, é impossível produzir uma solução. Também é possível, mesmo quando as tendas pedidas são iguais ao total de árvores, que o posicionamento de algumas árvores seja tal que impeça a colocação de todas as tendas. Neste tipo de circunstâncias não existe solução.

3.2 Formato de entrada

O ficheiro de extensão `.camp` pode conter um ou mais problemas para serem resolvidos. Cada problema e respectivo mapa são definidos com a indicação das dimensões do mapa, dois inteiros L(inhas) e C(olunas); dois vectores de inteiros $l_0\ l_1\ \ldots\ l_{L-1}$ com a indicação do número de tendas em cada uma das linhas e $c_0\ c_1\ \ldots\ c_{C-1}$; e uma matriz de $L \times C$ caracteres sem espaços para cada linha e em que os caracteres estão restringidos a ´A´, ´T´ e ´.´.

Por exemplo, o mapa da Figura 1 poderia ser definido da seguinte forma:

```
8 8
1 3 1 2 1 2 2 0
2 1 1 1 2 1 2 2
A.A.A..A
........
....A.A.
A.......
.......A
.A..A...
......A.
.......A
```

mas também poderia surgir do seguinte modo:

```
8
            8
1 3      1

 2 1    2 2 0
2
                      1 1 1            2 1 2 2


A.A.A..A ........
  ....A.A.
A.......
                                 .......A  .A..A...     ......A.
.......A
```

Sublinha-se aqui que os dados de cada problema não têm necessariamente que estar "arrumados" como se ilustra no primeiro exemplo. O segundo exemplo ilustra isso mesmo. 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 mapa inicia-se com dois inteiros, L e C; após estes dois inteiros, existirão **sempre** L inteiros seguidos de C inteiros; após esta informação existirão **sempre** L strings de comprimento C; todas as strings contêm apenas os caracteres ´A´, ´T´ e ´.´.

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

[2]Ou mudança de linha.

o ficheiro passado como argumento existe de facto e interpretar correctamente o seu conteúdo semântico.

Finalmente, se se pretendesse resolver dois puzzles a partir de um só ficheiro de entrada, o seu formato seria a justaposição desses dois puzzles, como se apresenta abaixo:

```
8 8
1 3 1 2 1 2 2 0
2 1 1 1 2 1 2 2
A.A.A..A
........
....A.A.
A.......
.......A
.A..A...
......A.
.......A

4 6
1 1 1 1
1 0 1 0 1 1
...A..
A..A..
......
....A.
```

Neste exemplo os dois problemas estão separados por uma linha em branco. Haverá sempre, pelo menos, uma linha em branco entre dois 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 na formatação.

3.3 Formato de saída

O resultado da execução do programa CAMP*ist*A consiste em apresentar um mapa com uma colocação para todas as tendas pedidas ou a indicação de que o problema não admite solução.

A primeira linha da solução deverá sempre repetir a caracterização do problema, à qual se adiciona um inteiro. Esse inteiro indica se todas as tendas foram colocadas, valendo 1, ou -1 quando o problema não admite solução. As linhas seguintes deverão conter o mapa completo com árvores e tendas, quando existe solução. Ou seja, se este inteiro for, 1, deverão existir L linhas com uma string por linha e cada uma das strings deverá ter tamanho C. Se o problema não admitir solução, a resposta faz-se apenas com os três inteiros já indicados: L, C e -1.

A solução completa para o mapa da Figura 1, de acordo com que está apresentado na Figura 2 seria descrita no ficheiro de saída na forma seguinte:

```
8 8 1
A.A.A.TA
T.T.T...
....A.AT
AT..T...
......TA
TA.TA...
.....TAT
.......A
```

Se o ficheiro de extensão `.camp` possuir mais do que um problema, 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** `.tents`. Este ficheiro deve ser criado e aberto pelo programa. Por exemplo, se o ficheiro com problemas se chama `teste231.camp`, o ficheiro de saída deve-se chamar `teste231.tents`. 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 a extensão `.camp`, 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 mapas e 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 um argumento: o nome de um ficheiro, de extensão **.camp0**, contendo um ou mais problemas. Aqui também os sucessivos problemas estarão separados por, pelo menos, uma linha em branco. Este ficheiro tem formato ligeiramente diferente do anteriormente definido.

Existirão apenas três variantes de funcionamento: variante A, B e C. Para cada problema, a indicação da variante a resolver surgirá sempre a seguir à informação das dimensões do problema.

Em problemas de variante A pretende-se determinar se o mapa dado é admissível. Ou seja, se não existe logo à partida informação que permita concluir sem tentar resolver o mapa que este não admite solução. Por exemplo, se o somatório de tendas nas linhas for diferente do somatório nas colunas o problema não é admissível. Outra possibilidade é o número de árvores ser insuficiente para as tendas pedidas. Faz-se notar que o teste de admissibilidade não garante que exista solução quando um mapa seja admissível. Não se pretende, na primeira fase de submissões, que se determine da existência ou não existência de solução.

Para problemas de variante B pretende-se determinar se uma dada célula (l_0, c_0) com certeza não terá tenda. Para que não haja tenda de certeza é preciso que não existam árvores adjacentes a essa célula ou, havendo, que na linha ou coluna em que se insira, não podem ser colocadas mais tendas. A resposta a problemas de variante B é 1 quando a célula não terá tenda de certeza e 0 quando não se possa afirmar 1.

Em problemas de variante C pretende-se determinar se alguma das tendas colocadas no mapa viola alguma das restrições do problema. Ou seja, se houver mais tendas nalguma linha ou coluna do que o limite para essa linha ou coluna, se alguma tenda não tiver qualquer árvore adjacente, ou se alguma tenda tiver como adjacente alguma árvore que esteja associada com alguma outra tenda.

A forma de invocar as três variantes é:

A. Cada problema desta variante inicia-se com dois inteiros e um caracter: L C A

B. Cada problema desta variante inicia-se com dois inteiros, um caracter e mais dois inteiros: L C B l_0 c_0

C. Cada problema desta variante inicia-se com dois inteiros e um caracter: L C C

Por exemplo, para o exemplo da Figura 1 se se pedisse

```
8 8 B 2 2
1 3 1 2 1 2 2 0
2 1 1 1 2 1 2 2
A.A.A..A
........
....A.A.
A.......
.......A
.A..A...
......A.
.......A
```

como não existe nenhuma árvore adjacente do ponto (2, 2) a resposta deveria ser:

```
8 8 B 2 2 1
```

Mas, se se pedisse para $l_0 = 0$ e $c_0 = 1$ a resposta deveria ser

```
8 8 B 0 1 0
```

dado que não se pode à partida saber, sem resolver o mapa, se aquela célula ficará vazia.

Para problemas de variante A a resposta será 1 quando o problema for admissível e 0 caso contrário. Para problemas de variante C a resposta deverá ser 1 quando existe violação de alguma(s) restrição(ões) para alguma das tendas e 0 caso contrário. Por exemplo, se o ficheiro de entrada fosse:

```
8 8 A
1 3 1 2 1 2 2 0
2 1 1 1 2 1 2 2
A.A.A..A
........
....A.A.
A.......
.......A
.A..A...
......A.
.......A
```

a resposta deveria ser

```
8 8 A 1
```

Para o seguinte teste

```
8 8 C
1 2 2 2 1 2 2 0
2 1 1 2 1 1 2 2
A.A....A
........
.TAT..A.
A..A...
...T...A
.A..A...
......A.
.......A
```

a resposta deverá ser

```
8 8 C 1
```

porque se assumirmos que a tenda em (2, 1) está associada à árvore em (2, 2), por essa razão, a tenda em (2, 3) deveria estar associada com a árvore em (3, 3), o que faz com que a tenda em (4, 3) não tenha qualquer árvore que se lhe possa atribuir. Sublinha-se que, se ao contrário assumirmos estar a tenda em (4, 3) associada com a árvore em (3, 3), tal obriga a que a tenda em (2, 3) esteja associada com a árvore em (2, 2), deixando sem árvore a tenda em (2, 1). Isto é, a conclusão seria exactamente a mesma: existe uma tenda colocada que viola alguma das restrições do problema.

Qualquer outra variante só admite resposta de que não existe solução. Isto é, repete-se a informação de caracterização do problema seguida de -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 `.tents0` 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 a informação que caracteriza o problema, excepto os vectores de tendas e o mapa, seguida do resultado obtido. O resultado, para qualquer das variantes será sempre 1 ou 0 para cada uma das três variantes, de acordo com a descrição acima e -1 para qualquer outra variante.

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 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 nem C, ou a célula a inspeccionar estar fora do mapa (em variante B).

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.

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.

Tabela 1: Datas importantes do Projecto

Data	Documentos a Entregar
14 a 18 de Outubro de 2019	Enunciado do projecto disponibilizado na página da disciplina.
até 08 de Novembro de 2019 (18h00)	Inscrição dos grupos no sistema Fenix.
15 de Novembro de 2019, (18h00)	Conclusão da primeira submissão.
11 de Dezembro de 2019, 4ª feira 12h 15h	**1ª Data de entrega do projecto:** Fim da submissão electrónica em primeira data. Entrega do relatório do projecto em papel.
12 de Dezembro de 2019, 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.
13 de Dezembro de 2019, 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 13 de Dezembro têm penalização de 20 valores.
até 31 de Janeiro de 2020	Eventual discussão do trabalho (data combinada com cada grupo).

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/` .