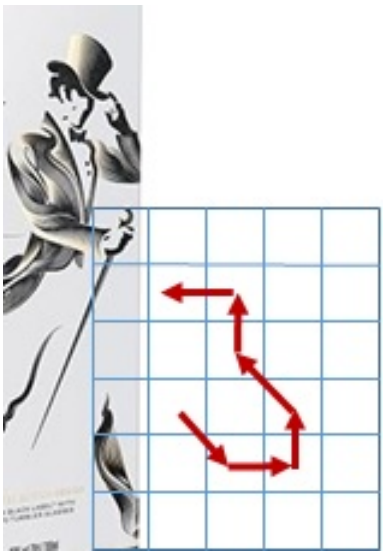
DEEC
DEPARTAMENTO DE ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES
TÉCNICO LISBOA

Mestrado Integrado em Engenharia Electrotécnica e de Computadores (MEEC)

Algoritmos e Estruturas de Dados

Enunciado do Projecto



Numeric Trails

Versão 2.0 (19/Março/2019)

2018/2019
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 trabalho 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 – Numeric Trails

Neste projecto pretende-se desenvolver um programa que seja capaz de produzir caminhos em mapas retangulares contendo apenas números inteiros, em que os deslocamentos admissíveis são oito direcções a partir de uma qualquer posição: verticais, horizontais e diagonais. O mapa é um conjunto de células, cada uma com um valor inteiro, e num qualquer caminho produzido nesse mapa uma dada célula não pode ser repetida, podendo naturalmente uma mesma célula ser utilizada em mais que um caminho. Cada movimento legal passa de uma célula para uma das imediatamente adjacentes que ainda não tenha sido usada no caminho em construção.

Na Figura 1 ilustra-se um mapa possível, omitindo-se apenas os valores de cada célula para simplificar o diagrama, onde está representado um caminho, em que cada seta corresponde a um passo desse caminho.

Figura 1: Um mapa com um caminho.

Para além do mapa existe um agente – Walker – que terá como objectivo realizar algum tipo de caminho nessa mapa. Cada passo desse caminho, que pode conter vários passos, deverá seguir a restrição de movimento associada com um de oito possíveis movimentos para uma das células adjacentes, sem nunca sair desse mapa. Um caminho tem de possuir, no mínimo, um passo.

Para além das restrições acima, o agente poderá possuir um de 6 objectivos, ou variantes, a atingir no final do caminho:

A. Dado um ponto de partida, x_0, determinar um caminho com exactamente k passos, tal que as células são visitadas por ordem crescente (no sentido estrito) dos seus valores, ou indicar que não existe;

B. Dado um ponto de partida, x_0, determinar um caminho com exactamente k passos, tal que as células visitadas têm todas valores pares (positivos ou negativos), ou indicar que não existe;

C. Determinar um caminho com exactamente k passos, tal que as células são visitadas por ordem crescente (no sentido estrito) dos seus valores, ou indicar que não existe;

D. Determinar um caminho com exactamente k passos, tal que as células visitadas têm todas valores pares (positivos ou negativos), ou indicar que não existe;

E. Determinar o caminho crescente (no sentido estrito) mais comprido;

F. Determinar o caminho mais comprido composto de números pares.

O que se pretende neste projecto é desenvolver uma aplicação em linguagem C que seja capaz de resolver automaticamente um qualquer mapa para um conjunto restrito de objectivos, descritos neste texto.

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 seja 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 "NumericTrails"

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

Este semestre 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 NumericTrails deverá ser invocado na linha de comandos da seguinte forma:

```
aed$ ./walker <nome>.puz
```

`walker` designa o nome do ficheiro executável contendo o programa NumericTrails;

`<nome>.puz` em que `<nome>` é variável, identificando o ficheiro contendo o(s) mapas(s) onde se realizam os caminhos.

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

- produzir uma solução que satisfaça a variante especificada;
- indicar que não existe solução.

Por exemplo em variante A, se todas as células adjacentes da célula inicial possuirem valores menores que o da célula de partida, o agente não pode dar o primeiro passo, ou o mesmo pode acontecer após alguns passos, impedindo-o de completar k passos. Nestas circunstâncias não existe solução, porque o agente ficará impedido de prosseguir o seu passeio em qualquer conjunto inicial de passos que escolher. Também é impossível produzir solução quando o ponto de partida estiver fora do mapa. Para caminhos de valor par, a célula de partida não tem que ter valor par, mas todas as restantes no caminho têm que satisfazer essa restrição. Também é impossível produzir caminhos em que o número de passos exceda o número de células do mapa, pois tal obrigaria a que alguma(s) célula(s) fosse(m) repetida(s), violando a restrição de não repetição.

3.2 Formato de entrada

O ficheiro de extensão `.puz` pode conter um ou mais mapas para serem resolvidos. Cada problema e respectivo mapa são definidos com a indicação das dimensões do mapa, dois inteiros L(inhas) e C(olunas); qual a variante aplicável, um caracter `A, B, ..., F`; as coordenadas do ponto de partida, dois inteiros l_0 e c_0, mesmo para as variantes que não necessitam de um ponto de partida[1]; e o número de passos, k do caminho[2].

Se o objectivo for como o A, deverão existir três inteiros após o caracter que o define – coordenadas do ponto de partida e número de passos. Para objectivos como o C, também existem três inteiros, mas os dois primeiros podendo ser quaisquer, os seus valores não podem ter qualquer influência na definição da existência ou não existência de solução.

Após esta informação de caracterização do problema haverá uma matriz de inteiros, sem restrições aos seus valores, representando o mapa. Se for indicado que existem L linhas e C colunas, então a matriz apresentará sempre $L \times C$ inteiros.

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

```
6 5 A 3 1 6
0 20 3 2 3
1 2 13 2 1
4 3 12 1 2
3 4 5 10 3
1 1 6 8 3
2 3 2 3 2
```

[1]Para variantes em que não é necessário definir qual o ponto de partida, estas coordenadas dadas não são relevantes, devendo o seu valor ser ignorado para efeitos de tratamento dos dados. Ou seja, lido mas não tido em conta.

[2]Para variantes em que não é necessário definir o número de passos, este inteiro não é relevante, devendo o seu valor ser ignorado para efeitos de tratamento dos dados. Ou seja, lido mas não tido em conta.

Neste exemplo, fornece-se um mapa de 6 linhas e 5 colunas; pretende-se produzir um passeio de valores crescentes de 6 passos – variante A; o ponto de partida é linha 3 e coluna 1, identificado aqui a negrito para efeitos de ilustração do sistema de coordenadas a adoptar.

Os ficheiros com um ou mais problemas poderão ter qualquer nome, mas têm obrigatoriamente a extensão `.puz`.

Assume-se que todos os ficheiros de extensão `.puz` estão correctos e no formato especificado anteriormente. Ou seja, cada problema define-se sempre com 2 inteiros e um caracter seguidos sempre de mais três inteiros; se se indicar um puzzle de dimensão $L \times C$, L e C serão sempre positivos e existirão de certeza $L \times C$ inteiros, após a definição do problema; o identificador do objectivo/variante deverá ser A, B, ..., F – qualquer outro valor não é admissível e se surgir algum puzzle com outro objectivo, tal significa que a resposta terá de ser que não há solução; as coordenadas do ponto de partida podem estar dentro ou fora das dimensões do mapa – neste segundo caso o problema não tem solução se a variante necessitar um ponto de partida; o número de passos deverá permitir tentativa de produção de solução se o seu valor for maior ou igual a 1 e menor ou igual a $L \times C - 1$, excepto para variantes em que este valor seja irrelevante.

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

```
10 11 C 0 0 12
1 0 2 3 2 3 2 3 2 3 1
3 2 1 2 3 3 1 2 3 2 1
4 3 2 1 2 3 4 5 0 3 1
1 6 2 3 2 3 2 3 2 3 1
3 2 1 2 3 0 1 2 3 2 1
4 3 2 1 2 3 4 5 3 3 1
1 2 0 3 2 3 2 3 2 3 1
3 2 1 2 3 3 1 2 3 2 1
4 3 2 1 2 3 4 5 3 0 1
1 7 2 3 2 3 2 3 2 3 1

6 7
A 0 0
4
7 0 1 1 0 1 0
1 1 0 2 1 0 1
1 4 7 1



9 4 1
0 1 3 0 8 1 1
  2 1 5 1 0 0 1
1 6 1 0 4 3 1
```

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.

O primeiro problema é de variante C e o segundo problema é de variante A. Sublinha-se aqui que os dados de cada problema não têm necessariamente que estar "arrumados"como se ilustra no primeiro problema. O segundo problema ilustra isso mesmo. Os procedimentos de leitura a adoptar pelos alunos deverão ser robustos a qualquer tipo de desarrumação.

3.3 Formato de saída

O resultado da execução do programa NumericTrails consiste em apresentar os passos que constituem o caminho produzido ou a indicaçao de que o problema não admite solução.

Para problemas das variantes A e B a primeira linha da solução deverá sempre repetir a caracterização do problema, à qual se adiciona um inteiro. Esse inteiro indica quantos passos o caminho possui ou é -1 quando o problema não admite solução. As linhas seguintes deverão conter a solução do problema, quando existe. Ou seja, se este inteiro for, p, deverão existir p linhas de três inteiros cada: coordenadas da posição para que se avança e o valor dessa posição.

A solução completa para um dado mapa poderia ser, por exemplo:

```
6 5 A 3 1 6 6
4 2 6
4 3 8
3 3 10
2 2 12
1 2 13
1 1 20
```

Para as restantes variantes, a primeira linha da solução substitui o par (l_0, c_0) dados originalmente pelo ponto de partida usado quando esses problemas admitam solução e repete a informação original quando não haja solução. No restante a primeira linha da solução segue o mesmo formato que o usado nas variantes A e B.

Se o ficheiro de extensão `.puz` 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** `.path`. Este ficheiro deve ser criado e aberto pelo programa. Por exemplo, se o ficheiro com problemas se chama `teste231.puz`, o ficheiro de saída deve-se chamar `teste231.path`. 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 `.puz`, 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 output 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. Qualquer execução que 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 **.puz0**, 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.

Um problema de variante A define-se com L, C, l_0, c_0, k, e pretende-se determinar qual o valor mais elevado das células vizinhas que seja superior ao valor da célula (l_0, c_0). A resposta deverá ser apenas repetir a informação do problema seguida de -1 ou seguida 1 e outro inteiro. Quando a resposta for -1 tal significa que nenhuma das, no máximo 8, células vizinhas possui valor mais alto. Quando a resposta for afirmativa, a resposta deverá ser 1 seguido do valor da célula de maior valor que satisfaça a restrição.

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

```
6 5 A 3 1 6
0 20 3 2 3
1 2 13 2 1
4 3 12 1 2
3 4 5 10 3
1 1 6 8 3
2 3 2 3 2
```

a resposta deveria ser

```
6 5 A 3 1 6 1 12
```

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

```
6 5 A 0 1 6 -1
```

Para problemas de variante B, pretende-se indicar qual o menor valor par de todos os seus vizinhos, ou -1 quando nenhum dos vizinhos for par. Por exemplo, para os dados abaixo

```
6 5 B 3 1 6
0 20 3 2 3
1 2 13 2 1
4 3 12 1 2
3 4 5 10 3
1 1 6 8 3
2 3 2 3 2
```

a resposta terá de ser

```
6 5 B 3 1 6 1 4
```

Mas para $l_0 = 5$ e $c_0 = 0$ a resposta é

```
6 5 B 5 0 6 -1
```

Finalmente, para variante C pretende-se validar um caminho. Ou seja, sendo k o último inteiro que define o problema, seguir-se-ão, antes do mapa, exactamente k conjuntos de três inteiros, no formato linha, coluna, valor. O programa deverá indicar se o caminho dado é válido para valores crescentes ou válido para valores pares. Se for válido para ambos, tal deverá ser indicado. Assim, -1 para caminho inválido; 1 para caminho de valores crescentes válido; 2 para caminho de valores pares válido; 3 quando o caminho é valido para ambos os formatos. Para que o caminho seja válido, cada passo tem de mover para uma célula adjacente, não se pode sair do mapa, o valor indicado para cada c elula tem de corresponder ao seu verdadeiro valor, não se podem repetir células e uma ou ambas as restrições têm de se verificar.

Por exemplo, para este problema

```
6 5 C 3 1 6
4 2 6
4 3 8
3 3 10
2 2 12
1 2 13
1 1 20
0 20 3 2 3
1 2 13 2 1
4 3 12 1 2
3 4 5 10 3
1 1 6 8 3
2 3 2 3 2
```

a resposta deverá ser

```
6 5 C 3 1 6 1
```

Qualquer outra variante só admite resposta de que não existe solução.

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 `.vrf` 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 o mapa e/ou passos, seguida do resultado obtido. O resultado, para qualquer das variantes será sempre um ou um par de inteiros, de acordo com a descrição acima

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, o ponto de partida estar fora do mapa, ou $k < 1$ para variante C.

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
18 a 22 de Março de 2019	Enunciado do projecto disponibilizado na página da disciplina.
até 05 de Abril de 2019 (18h00)	Inscrição dos grupos no sistema Fenix.
12 de Abril de 2019, (18h00)	Primeira submissão.
22 de Maio de 2019, 4ª feira 12h 15h	**1ª Data de entrega do projecto:** Submissão electrónica do projecto. Entrega do relatório do projecto em papel.
23 de Maio de 2019, 5ª feira 12h 15h	**2ª Data de entrega do projecto: penalização de um (1) valor** Submissão electrónica do projecto. Entrega do relatório do projecto em papel.
24 de Maio de 2019, 6ª feira 12h 15h	**3ª Data de entrega do projecto: penalização de dois (2) valores** Submissão electrónica do projecto. Entrega do relatório do projecto em papel.
	Submissões posteriores a 24 de Maio têm penalização de 20 valores.
até 1 de Julho de 2019	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 substancial.

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/` .