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

ZerUns

Versão 1.0 (18/Outubro/2017)
Versão 2.0 (23/Outubro/2017) - eliminação de gralhas e clarificações

2017/2018
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 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 – ZerUns

Neste projecto pretende-se desenvolver um programa que seja capaz de resolver puzzles estáticos constituídos por zeros e uns apenas, que estejam circunscritos numa superfície rectangular, dispostos em linhas e colunas. O objectivo é preencher completamente uma matriz inicialmente preenchida parcialmente. O preenchimento completo necessita obedecer a um pequeno conjunto de restrições muito simples:

1. Não podem existir mais que dois uns ou dois zeros consecutivos em qualquer linha ou coluna;
2. Cada linha e cada coluna, depois de preenchida, possui exactamente o mesmo número de zeros e de uns;
3. Nenhuma linha pode ser repetida, assim como nenhuma coluna pode ser repetida.

Na Figura 1 ilustra-se um puzzle inicial (à esquerda) com a respectiva solução (à direita).

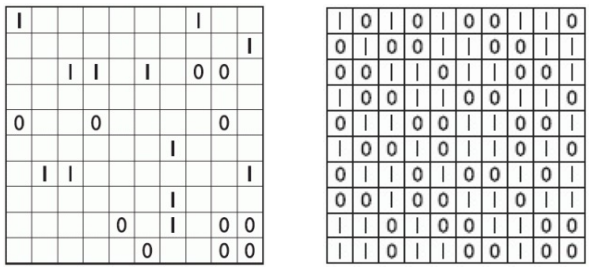

Figura 1: Exemplo de um puzzle e solução.

De acordo com as restrições do problema, existem situações em que o preenchimento de algumas células é imediato. Por exemplo, sempre que exista um par de uns ou um par de zeros consecutivos em alguma linha ou coluna, as células adjacentes admitem apenas uma solução. Na Figura 2, do lado esquerdo, as células azuis apenas podem ser preenchidas com 0.

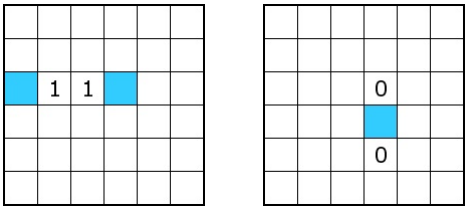

Figura 2: Procura de pares e evitar trios.

Outra situação de resolução imediata é a que está exemplificada na Figura 2 do lado direito, dado que não podem existir trios consecutivos com o mesmo valor binário. Neste caso, a célula azul terá de ser preenchida com 1.

A restrição de paridade entre zeros e uns em cada linha e coluna poderá permitir completar algumas linhas, como está ilustrado na Figura 3 do lado esquerdo. Falta um 1 na célula azul da linha e faltam dois 0 nas células azuis da coluna.

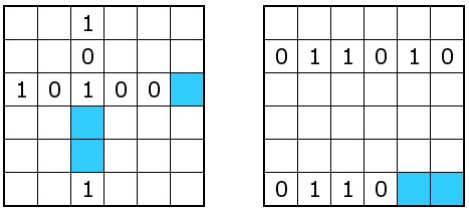

Figura 3: Finalização de linhas ou colunas (esq.). Eliminação de combinações impossíveis (dir.).

Poderão existir circunstâncias em que o facto de não poder haver repetição em linhas ou colunas permita a tomada de decisão relativamente a algumas das células. Na Figura 3, do lado direito, as duas células azuis terão de ter `01` dado que uma linha completa acima possui as primeiras 4 células iguais e as duas últimas com `10`.

Noutros casos, é possível decidir sobre uma célula para evitar a existência de trios, como está ilustrado na Figura 4. Nesse exemplo, a colocação do número 1 na célula azul forçaria a existência de um trio de zeros. Assim, naquela célula terá que ser forçosamente colocado um 0.

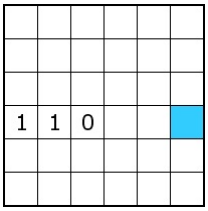

Figura 4: Eliminação de outras combinações impossíveis.

A maioria destes puzzles é resolúvel por aplicação sucessiva das regras acima descritas e o grau de dificuldade que poderão exibir prende-se com a maior ou menor dificuldade em determinar qual das células, não preenchidas, pode ser inequivocamente preenchida por aplicação de alguma das restrições do problema. No entanto, independentemente do grau de dificuldade específico, é sempre possível obter uma solução por procura no espaço

das soluções admissíveis. Naturalmente que uma metodologia de solução que intercale a aplicação directa das regras com procura poderá obter soluções em menos tempo, desde que adequadamente implementada.

O que se pretende neste projecto é desenvolver uma aplicação em linguagem C que seja capaz de resolver automaticamente um qualquer puzzle 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 "ZerUns"

O programa a desenvolver deverá ser capaz de ler puzzles 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 de ZerUns deverá ser invocado na linha de comandos da seguinte forma:

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

`zeruns` designa o nome do ficheiro executável contendo o programa ZerUns;

`<nome>.puz` em que `<nome>` é variável, identificando o ficheiro contendo o(s) puzzle(s) a resolver.

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

- produzir uma solução;
- indicar que não existe solução.

O programa deverá estar preparado para executar uma de duas variantes possíveis:

1. Restrições 1 e 2 activas;
2. Todas as restrições activas.

Tanto para a variante 1 como para a variante 2 é admissível que não haja solução para algum puzzle inicial.

3.2 Formato de entrada

O ficheiro de extensão `.puz` pode conter um ou mais puzzles para serem resolvidos. Cada puzzle é definido da seguinte forma: a primeira linha contém informação sobre as dimensões da superfície (linhas e colunas, através de um único inteiro, dado que os puzzles estão sempre definidos em superfícies quadradas) e qual o objectivo (variante); nas linhas seguintes identifica-se a composição do quadro, em que cada linha do ficheiro representa, através de números inteiros, de que forma está preenchida a superfície: 0, 1 ou 9. O inteiro 9 representa que a posição associada está vazia. A restrição 2 força a que as dimensões do número de linhas e colunas seja um sempre um número par.

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

```
10 2
1 9 9 9 9 9 9 1 9 9
9 9 9 9 9 9 9 9 9 1
9 9 1 1 9 1 9 0 0 9
9 9 9 9 9 9 9 9 9 9
0 9 9 0 9 9 9 9 0 9
9 9 9 9 9 9 1 9 9 9
9 1 1 9 9 9 9 9 9 1
9 9 9 9 9 9 1 9 9 9
9 9 9 9 0 9 1 9 0 0
9 9 9 9 9 0 9 9 0 0
```

O inteiro que identifica a variante é sempre positivo e apenas pode ser 1 ou 2. Qualquer outro valor não é admissível e se surgir algum puzzle com outro número de variante, tal significa que não há solução. No exemplo acima está-se a pedir a variante 2.

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, se a primeira linha indicar um puzzle de dimensão N, N será sempre par e existirão de certeza $N \times N$ inteiros abaixo dessa primeira linha e nenhum desses inteiros será diferente de 0, 1 ou 9. Adicionalmente, as células que estiverem preenchidas com 0 ou com 1 nunca violam qualquer das restrições do problema. Ou seja, não haverá puzzles iniciais com mais que dois 0 ou dois 1 consecutivos em linha ou coluna; nunca haverá puzzles iniciais em que uma linha ou coluna já possua mais que $N/2$ 0 ou 1; nem haverá puzzles iniciais em que, havendo linhas ou colunas preenchidas, se viole a terceira restrição do problema, quando se pretende resolver em variante 2.

Por esta razão, o programa não necessita fazer qualquer verificação daquele tipo de correcção. Apenas necessita de garantir que a extensão está correcta e que o ficheiro passado como argumento existe de facto.

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 2
1 9 9 9 9 9 9 1 9 9
9 9 9 9 9 9 9 9 9 1
9 9 1 1 9 1 9 0 0 9
9 9 9 9 9 9 9 9 9 9
0 9 9 0 9 9 9 9 0 9
9 9 9 9 9 9 1 9 9 9
9 1 1 9 9 9 9 9 9 1
9 9 9 9 9 9 1 9 9 9
9 9 9 9 0 9 1 9 0 0
9 9 9 9 9 0 9 9 0 0

4 1
9 9 1 9
9 9 1 9
0 0 9 9
9 1 9 9
```

3.3 Formato de saída

O resultado da execução do programa ZerUns consiste em apresentar o quadro completamente preenchido para cada problema resolvido ou a indicação de que o problema não admite solução.

Para qualquer problema, a primeira linha da solução deverá sempre repetir a primeira linha do problema, tal como apresentada no ficheiro de entrada, à qual se adiciona um inteiro. Este terceiro inteiro deverá ser 1, indicando que o problema possui solução, ou -1, indicando que o problema não possui solução. Se o terceiro inteiro for 1, as linhas seguintes deverão conter a solução do problema. Ou seja, deverão conter $N \times N$ inteiros que deverão apenas ser 0 ou 1. Se o terceiro inteiro da primeira linha for -1, a solução desse problema limita-se a esta linha. Quando se pedir alguma variante diferente de 1 ou de 2, como mais nenhuma está prevista, a solução deverá ser apenas repetir a primeira linha original, apenas com os dois inteiros.

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 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** `.sol`. 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.sol`. Note-se que, em situações em que haja erro na passagem de argumentos ao programa, não faz qualquer sentido criar um ficheiro de saída.

Considere, por exemplo, o seguinte ficheiro de entrada apresentado na secção anterior, com dois problemas. O ficheiro de soluções seria

```
10 2 1
1 0 1 0 1 0 0 1 1 0
0 1 0 0 1 1 0 0 1 1
0 0 1 1 0 1 1 0 0 1
1 0 0 1 1 0 0 1 1 0
0 1 1 0 0 1 1 0 0 1
1 0 0 1 0 1 1 0 1 0
0 1 1 0 1 0 0 1 0 1
0 0 1 0 0 1 1 0 1 1
1 1 0 1 0 0 1 1 0 0
1 1 0 1 1 0 0 1 0 0

4 1 -1
```

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 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".

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 puzzles 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 `.puz`, contendo um ou mais problemas. Este ficheiro tem um formato ligeiramente diferente do que foi anteriormente definido. Por cada problema tem uma primeira linha de quatro inteiros, que correspondem a: dimensão do puzzle, linha, coluna, e um valor binário. Os dois inteiros linha e coluna identificam uma célula do puzzle. Ou seja, l, com $1 \leq l \leq N$ e c, com $1 \leq c \leq N$, sendo N o número de linhas e colunas do puzzle. O valor binário será 0 ou 1. Após esta primeira linha existirão $N \times N$ inteiros que serão 0, 1 ou 9.

O que se pretende é saber se, para o puzzle dado, na célula dada se pode dizer que o binário é de certeza solução, não é de certeza solução, ou se não existe informação para afirmar nenhuma das situações anteriores. Para todos os problemas nesta fase, e para simplificar as implementações, apenas se pretende a verificação das duas primeiras restrições para responder a cada um dos "queries".

Antes de mais, é necessário adoptar um sistema de coordenadas único. Na Figura 5, apresenta-se um possível puzzle explicitando o sistema de coordenadas que todos os grupos deverão adoptar.

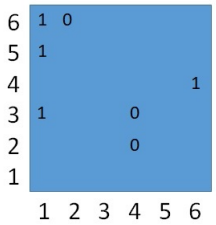

Figura 5: Sistema de coordenadas.

Por exemplo, para o puzzle da Figura 1 poder-se-ia ter a seguinte linha `10 8 2 1` cuja resposta é não, porque em (8, 3) e (8, 4) estão dois 1 e ficar-se-ia com três 1 consecutivos na linha 8. Já a linha `10 8 2 0` tem resposta sim, exactamente por existir um 1 em (8, 3) e outro em (8, 4). A linha `10 10 3 0` não tem resposta sim nem não, com base exclusivamente no estado inicial daquele puzzle, porque não existe nenhuma informação local contígua que permita alguma conclusão definitiva, nem aquelas cordenadas pertencem a alguma linha ou coluna à qual falte apenas preencher aquela célula.

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 `.query` 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 os quatro inteiros do problema, seguidos de -1, 0 ou 1. O quinto inteiro especifica a resposta; -1 significa não, 0 significa talvez sim ou talvez não e 1 significa sim.

O ficheiro de entrada poderá conter mais do que um problema para resolver e cada um desses problemas poderá ter diferentes dimensões. Abaixo apresentam-se dois exemplos de problemas, para a matriz da Figura 1.

```
10 8 2 1
1 9 9 9 9 9 9 1 9 9
9 9 9 9 9 9 9 9 9 1
9 9 1 1 9 1 9 0 0 9
9 9 9 9 9 9 9 9 9 9
0 9 9 0 9 9 9 9 0 9
9 9 9 9 9 9 1 9 9 9
9 1 1 9 9 9 9 9 9 1
9 9 9 9 9 9 1 9 9 9
9 9 9 9 0 9 1 9 0 0
9 9 9 9 9 0 9 9 0 0
```

```
10 8 2 0
1 9 9 9 9 9 9 1 9 9
9 9 9 9 9 9 9 9 9 1
9 9 1 1 9 1 9 0 0 9
9 9 9 9 9 9 9 9 9 9
0 9 9 0 9 9 9 9 0 9
9 9 9 9 9 9 1 9 9 9
9 1 1 9 9 9 9 9 9 1
9 9 9 9 9 9 1 9 9 9
9 9 9 9 0 9 1 9 0 0
9 9 9 9 9 0 9 9 0 0
```

```
10 1 3 0
1 9 9 9 9 9 9 1 9 9
9 9 9 9 9 9 9 9 9 1
9 9 1 1 9 1 9 0 0 9
9 9 9 9 9 9 9 9 9 9
0 9 9 0 9 9 9 9 0 9
9 9 9 9 9 9 1 9 9 9
9 1 1 9 9 9 9 9 9 1
9 9 9 9 9 9 1 9 9 9
9 9 9 9 0 9 1 9 0 0
9 9 9 9 9 0 9 9 0 0
```

As soluções respectivas estão indicadas abaixo.

```
10 8 2 1 -1
```

```
10 8 2 0 1
```

```
10 1 3 0 0
```

Se, por hipótese, o ficheiro de entrada fosse a concatenação dos três problemas acima, o ficheiro de saída seria a concatenação das três soluções apresentadas. 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.

Poderão existir problemas mal definidos: seja porque a não é fornecido um binário (0 ou 1), seja porque as coordenadas dadas estão fora das dimensões da matriz. Nesses casos, a resposta correcta será simplesmente repetir a primeira linha que define o problema, sem mais informação. Por exemplo, se a primeira linha fosse 10 4 5 3, a solução seria apenas repetir essa linha.

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 do projecto, 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. Ainda relativamente à última fase de submissões, prevê-se que o número de testes seja distribuído da seguinte forma: 60% para variante 1; 40% para variante 2.

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,

Tabela 1: Datas importantes do Projecto

Data	Documentos a Entregar
23 de Outubro de 2017, 2ª feira	Enunciado do projecto disponibilizado na página da disciplina.
até 10 de Novembro de 2017 (19h)	Inscrição dos grupos no sistema Fenix.
17 de Novembro de 2017, 6ª feira	Primeira submissão.
13 de Dezembro de 2017, 4ª feira 12h 15h	**1ª Data de entrega do projecto:** Submissão electrónica do projecto. Entrega do relatório do projecto em papel.
14 de Dezembro de 2017, 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.
15 de Dezembro de 2017, 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 15 de Dezembro têm penalização de 20 valores.
a partir de 22 de Janeiro de 2018	Eventual discussão do trabalho (data combinada com cada grupo).

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. Ao contrário de anos anteriores, neste semestre **não haverá qualquer extensão nos prazos de entrega**, pelo que os alunos 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.

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 contemplar os aspectos seguidamente indicados. A apresentação do mesmo deverá iniciar-se por aspectos de ordem geral, seguindo-se descrições mais detalhadas dos módulos e estruturas de dados utilizados. Sugere-se a seguinte estrutura para o relatório:

- 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 do problema que foi resolvido, com indicação clara das especificações do mesmo tal como foram entendidas;
- Um texto simples que indique como os alunos abordaram e resolveram o problema;
- 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 descrição detalhada dos tipos de dados utilizadas e justificação dos mesmos (tabelas, listas, filas, pilhas, árvores, grafos, acervos, etc.);
- Descrição dos algoritmos usados (por exemplo, na manipulação dos tipos de dados);
- Uma descrição dos subsistemas funcionais que existam e, para cada um
  - a descrição dos objectivos do subsistema (até 5 linhas);
  - o nome do módulo onde estão definidos os tipos de dados abstractos (ficheiros `.h`) a utilizar no subsistema;

  - o nome do módulo `C` onde estão as funções do respectivo subsistema;
  - listagem das funções implementadas no subsistema, indicando para cada uma a respectiva assinatura e os objectivos da função (descrição sumária, sem código);
- 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;
- Uma análise crítica do funcionamento do programa e a avaliação do desempenho do projecto implementado;
- 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/` .