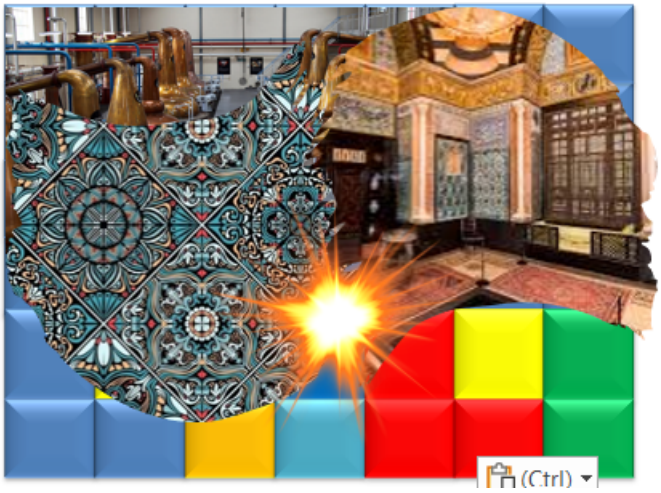
DEEC
DEPARTAMENTO DE ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES
TÉCNICO LISBOA

Licenciatura em Engenharia Electrotécnica e de Computadores (LEEC)

Algoritmos e Estruturas de Dados

Enunciado do Projecto



dISTiling

Versão 4.0 (17/Setembro/2023 – primeira versão pública)

2023/2024
1º Semestre

Conteúdo

1 Introdução

O trabalho descrito neste enunciado possui duas componentes, que correspondem às duas fases de avaliação de projecto para a disciplina de Algoritmos e Estruturas de Dados. A descrição geral do projecto que se propõe diz respeito ao trabalho a desenvolver para a última fase de avaliação.

Para a primeira fase de avaliação, o trabalho consiste 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 correção e testará a execução do programa em algumas instâncias do problema, sujeito a limitações de memória usada e tempo de execução.

2 O Problema – dISTiling

Este projecto pretende desenvolver um programa que seja capaz de remover azulejos (*tiles* na literatura anglo-saxónica) de uma matriz preenchida na totalidade ou em parte por azulejos de diversas cores. Pretende-se remover azulejos enquanto seja possível fazê-lo, gerando-se assim uma pontuação final que é uma função não linear no número de azulejos removidos.

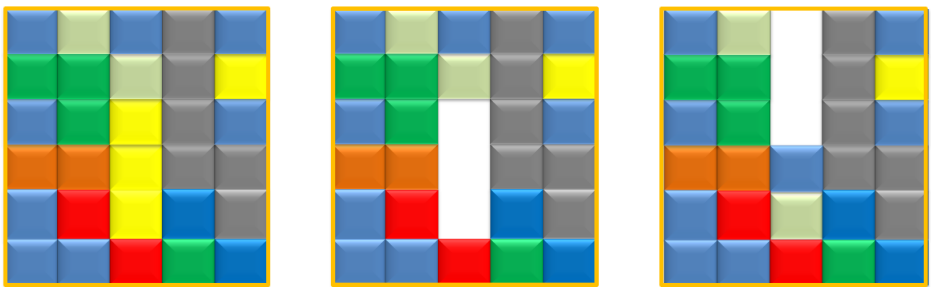
O objectivo é eliminar o maior número possível de azulejos, obtendo a pontuação pelo número de azulejos removidos. A remoção de azulejos é feita por selecção de um azulejo que pertença a um conjunto de azulejos adjacentes da mesma cor. Diz-se, neste contexto, que dois azulejos são adjacentes quando estiverem encostados um ao outro na vertical ou na horizontal. Dois azulejos adjacentes da mesma cor formam uma **mancha** de tamanho dois, se nenhum deles possuir outro da mesma cor que lhe seja adjacente. Uma **mancha** de tamanho superior a dois é tal que qualquer dos azulejos que lhe pertence possui, pelo menos outro da mesma cor que lhe é adjacente. Não há manchas de tamanho 1. Ou seja, apenas estamos interessados em agregados de azulejos adjacentes da mesma cor que constituam uma mancha.

Quando se decide remover um azulejo, removem-se todos os azulejos contidos na mancha a que pertence. Isto é, remove-se toda a mancha. A resolução de um determinado problema desta natureza consiste numa sequência de remoções de manchas, enquanto existirem. Não se pode parar havendo, pelo menos, uma mancha na matriz. Pela descrição anterior, deverá ser claro que apenas se consideram como lances válidos aqueles em que o azulejo indicado para remoção fizer parte de uma mancha.

Cada vez que se elimina uma mancha, os azulejos que estiverem acima dos removidos são sujeitos a efeito gravítico vertical. Desta forma, devem deslizar (para baixo) até tocarem outros azulejos, ou no limite inferior da superfície que os contém. Alguns lances podem deixar uma coluna vazia. Nessas circunstâncias, todas as colunas que estiverem à sua esquerda são sujeitas a efeito gravítico horizontal, deslocando-se para a direita.

A aplicação dos dois efeitos gravíticos obedece a prioridade fixa: primeiro aplica-se a gravidade vertical, sempre que a remoção da mancha produzir alguma(s) coluna(s) com azulejos não adjacentes na vertical ou não vazia(s) sem azulejos na linha inferior; em segundo lugar aplica-se a gravidade horizontal, quando alguma(s) das colunas ficar(em) vazia(s) por efeito da remoção de uma mancha.

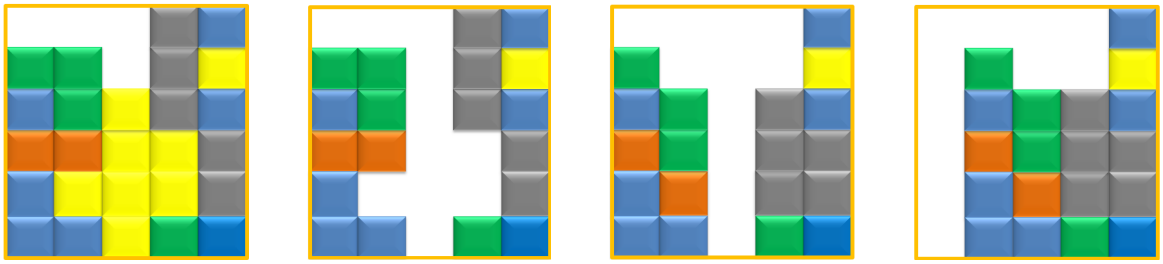
A Figura 1 ilustra uma parede possível (à esquerda) e o respectivo resultado (à direita) da selecção da mancha de três azulejos amarelos. A Figura 2 ilustra o resultado de um



(a) (b) (c)

Figura 1: Efeito gravítico vertical como resultado da remoção da mancha amarela presente em (a). Em (b) a remoção simples da mancha e em (c) a subsequente aplicação do efeito gravítico vertical.

lance que esvazia uma coluna, e a respectiva deslocação para a direita das restantes colunas, também neste exemplo por remoção da mancha amarela.



(a) (b) (c) (d)

Figura 2: Exemplo de um lance em que uma das colunas fica vazia. Em (b) a simples remoção da mancha amarela presente em (a), em (c) a aplicação do efeito gravítico vertical, seguida da aplicação do efeito gravítico horizontal em (d).

Cada lance realizado recebe uma pontuação, que depende do número de azulejos removidos. Seja $k \geq 2$ o número de azulejos pertencentes à mancha seleccionada para remoção. Então, a pontuação recebida é de $k(k-1)$ pontos.

Se a resolução de um dado mapa for feita através da remoção de M manchas, em que cada mancha possui $k_m \geq 2$ azulejos, com $m = 1, 2, \ldots M$, a pontuação obtida é dada pela expressão

$$P = \sum_{m=1}^{M} k_m(k_m - 1). \tag{1}$$

Pode-se definir diferentes objectivos para a resolução deste tipo de problemas. Por exemplo, pode ter-se como objectivo remover todos os azulejos, quando tal for possível, ou dizer que não existe tal solução; pode-se querer minimizar o número de azulejos sobrantes, quando já não existam mais manchas; pode-se querer atingir uma dada pontuação; identificar a sequência de lances que produza a maior pontuação possível; etc..

Pretende-se com este projecto desenvolver uma aplicação em linguagem C que seja capaz de resolver (automaticamente) qualquer parede de azulejos, para um conjunto restrito de objectivos apresentados na próxima secção.

Nas secções seguintes descreve-se o formato de utilização do programa a desenvolver no que diz respeito aos ficheiros de entrada e saída; os objectivos a implementar; as regras de avaliação e datas de entrega; e faz-se referência ao *Código de Conduta Académica*, que se espera ser zelosamente cumprido por todos os alunos que se submetam a 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 ou interpretações pessoais a todas as especificações aqui explicitadas. A falha em cumprir alguma(s) especificação(ões) e/ou regra(s) constantes neste enunciado acarretará necessariamente alguma forma de desvalorização do trabalho apresentado.

Por estas razões, tão cedo quanto possível e para evitar contratempos tardios, deverão os alunos esclarecer com o corpo docente qualquer aspecto que esteja menos claro, ou omisso, neste texto, ou qualquer dúvida que surja após uma leitura atenta e cuidada deste enunciado.

3 O programa "dISTiling"

O programa a desenvolver deve ser capaz de ler mapas ou paredes de azulejos e produzir soluções, para cada uma delas, ou indicar não haver solução (quando esse seja o caso).

3.1 Execução do programa

Neste semestre haverá duas fases de submissão. O que se descreve nas próximas secções, diz respeito às especificações da versão final do projecto. Na secção 4, detalham-se as especificações de projecto relativas à primeira fase de submissões.

O programa de dISTiling deve ser invocado na linha de comandos, da seguinte forma

```
somedirectoryinyourmachine$ ./tileblaster <nome>.tilewalls
```

tal que:

`somedirectoryinyourmachine`: é a pasta ou directoria onde se encontra o executável;

`tileblaster`: designa o nome do ficheiro executável contendo o programa dISTiling; e

`<nome>.tilewalls`: identifica o ficheiro contendo a(s) parede(s) de azulejos a resolver, em que `<nome>` é uma cadeia de caracteres de tamanho e conteúdo variável e a extensão tem obrigatoriamente que ser `.tilewalls`

Para cada problema contido no ficheiro de entrada, o programa deve fazer uma de duas coisas: produzir uma solução; ou indicar que não existe solução.

O programa deve estar preparado para executar uma de três variantes possíveis, ordenadas aqui de acordo com complexidade crescente

1. remoção simples de manchas;
2. atingir um limiar mínimo de pontuação;
3. atingir a máxima pontuação possível.

Em qualquer das três variantes, havendo solução, a parede final tem de ser tal que não subsistam mais manchas. Ou seja, enquanto existir, pelo menos, uma mancha o programa terá de produzir, pelo menos, mais um lance.

Só é admissível não haver solução para a variante dois, por não haver sequência de lances que atinja a pontuação objectivo. Neste caso, apesar de poderem existir lances na parede inicial, como não existe sequência de lances que cumpra o objectivo, o programa não pode apresentar nenhum lance. Todas as restantes variantes admitem sempre alguma solução que, em certos casos, pode ser não haver qualquer lance possível, se a parede inicial não possuir qualquer mancha.

3.2 Formato de entrada

O ficheiro de extensão `.tilewalls` pode conter um ou mais problemas para serem resolvidos. Cada problema é definido da seguinte forma:

- um cabeçalho com três inteiros, L, C e v, contendo informação sobre as dimensões da parede que define a envolvente externa onde se encontram os azulejos coloridos (L para as linhas e C para as colunas), e qual a variante a aplicar, v.
- nas linhas seguintes identifica-se qual o preenchimento da parede, através de números inteiros, com os azulejos que estejam presentes em cada uma das linhas da parede.
- o conteúdo após o cabeçalho conterá sempre de certeza $L \times C$ inteiros, em que nenhum deles será zero.

Algumas paredes podem não ter todas as posições preenchidas com azulejos. Nesse caso, a representação de cor faz-se com o inteiro -1 (ver exemplo de ficheiro de entrada relativo à parede inicial apresentada na Fig. 2). A existência de azulejo numa posição faz-se com um inteiro positivo, não nulo. Por exemplo, o problema original da Figura 1 pode ser definido da seguinte forma:

```
6 5 40
1 2 1 3 1
4 4 2 3 5
1 4 5 3 1
6 6 5 3 3
1 7 5 1 3
1 1 7 4 1
```

O inteiro, v, que identifica a variante a aplicar pode ser positivo ou negativo. Quando positivo ou nulo, diz respeito à variante 2 e indica qual a pontuação que deve ser atingida, ou ultrapassada, para se considerar o problema adequadamente resolvido. Quando negativo, pode ser: -1 para a variante 1; ou -3 para a variante 3.

Qualquer outro valor negativo não é admissível. Se surgir algum problema em que a variante seja algum outro número negativo, significa tratar-se um problema não admissível. Quando se pretende aplicar a variante 2, o terceiro número da primeira linha pode ser também igual a zero, não podendo nunca ser negativo, naturalmente. Quando tal acontecer (se acontecer ser zero) significa, para todos os efeitos, que a variante 2 admite uma qualquer solução obtenível através do procedimento associado com a variante 1.

No exemplo acima, pede-se a variante 2 e indica-se que qualquer solução com $P \geq 40$ é admissível como válida, desde que todos os lances identifiquem manchas e a parede não contenha qualquer mancha no final da sequência de lances.

No exemplo abaixo, ilustra-se a descrição de um problema em que a superfície rectangular não está completamente preenchida, e em que se pretende aplicar a variante 1. Este exemplo corresponde à parede na Figura 2(a).

```
6
5
-1
-1 -1 -1 3 1 4 4 -1 3 5
1 4 5
3 1 6 6 5      5 3 1 5
5
5 3



          1
1 5
4 1
```

Um ficheiro de entrada preenchido como ilustrado acima está tão correcto como no primeiro exemplo dado, uma vez que começa por conter 3 inteiros, com os primeiros dois estritamente maiores que zero e contém exactamente 30 inteiros após o cabeçalho, descrevendo completamente como está preenchida a parede de azulejos.

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

Assume-se que todos os ficheiros de extensão `.tilewalls` estão correctos e no formato especificado anteriormente. Ou seja, se o cabeçalho de um problema indicar uma parede com L linhas e C colunas, existem de certeza $L \times C$ inteiros depois do cabeçalho e nenhum desses inteiros será zero. Apenas positivos ou -1. Nenhum ficheiro de entrada possui uma descrição em que haja colunas em que os seus azulejos não estejam adjacentes e apoiados na linha inferior. Quando existam colunas completamente vazias, estarão sempre encostadas à esquerda.

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

Por fim, caso se pretendesse resolver os dois problemas apresentados nesta secção, usando um ficheiro de entrada, o seu formato seria a justaposição dos dois problemas, como se apresenta na Figura 3.

```
6 5 40
1 2 1 3 1
4 4 2 3 5
1 4 6 3 5
7 7 6 3 3
1 8 6 1 3
1 1 8 9 1

6
5
-1
-1 -1 -1 3 1 4 4 -1 3 5
1 4 5
3 1 6 6 5      5 3 1 5
5
5 3


         1
1 5
4 1
```

Figura 3: Ficheiro de entrada contendo dois problemas válidos.

3.3 Formato de saída

O resultado da execução do programa dISTiling consiste em listar os lances realizados e a pontuação atingida para cada problema resolvido. Para tal, é necessário adoptar um sistema de coordenadas único. A Figura 4 apresenta a parede da Figura 1, explicitando o sistema de coordenadas que todos os grupos devem adoptar. De acordo com este sistema de coordenadas, o lance realizado na Figura 1 pode ser identificado de uma de três formas: `2 3`; `3 3`; ou `4 3`. O primeiro inteiro indica a linha e o segundo a coluna. De acordo com este sistema de coordenadas os ficheiros de entrada apresentam cada linha em ordem decrescente do seu índice e, em cada linha, surgem as colunas por ordem crescente do índice. Assim, no exemplo da Figura 1 os primeiros 5 inteiros no ficheiro de entrada correspondem à linha 6 e os últimos 5 correspondem à linha 1.

Para qualquer problema, a primeira linha da solução deve incluir o cabeçalho do problema numa só linha. A segunda linha deve indicar o número de lances realizados e a pontuação obtida. As linhas seguintes devem indicar as coordenadas de todos os lances, pela ordem em que devem ser realizados, com um lance por linha.

Se o ficheiro de extensão `.tilewalls` possuir mais do que um problema, o ficheiro de saída deve conter uma solução para cada um dos problemas indicados e pela mesma ordem em que surgem no ficheiro de entrada. Para facilitar a interpretação visual 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 **com extensão** `.tileblasts` (em substituição de

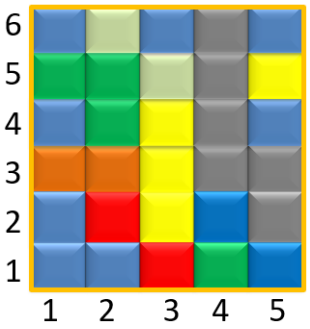


Figura 4: Sistema de coordenadas.

`.tilewalls`). Este ficheiro deve ser criado e aberto pelo programa. Por exemplo, se o ficheiro com problemas se chama `teste231.tilewalls`, o ficheiro de saída deve ter o nome `teste231.tileblasts`. Nas situações em que ocorra erro na passagem de argumentos ao programa, não faz qualquer sentido criar um ficheiro de saída.

Considere, por exemplo, o seguinte ficheiro de entrada:

```
6 5 40
1 2 1 3 1
4 4 2 3 5
1 4 6 3 5
7 7 6 3 3
1 8 6 1 3
1 1 8 9 1
```

Este ficheiro contém apenas um problema, cuja solução poderia hipoteticamente ser a seguinte (omite-se aqui a apresentação de uma solução completa):

```
6 5 40
10 56
3 2
5 2
4 1
...
```

Dado que a segunda linha da solução indica que existem 10 lances, o número de pares de inteiros que explicita a solução tem de ser 10, em que cada par ocupa uma linha.

Quando, em variante 2, um problema não possui solução – para o exemplo anterior se se pedir 900 como pontuação não existe solução de certeza –, o output deve ser:

```
6 5 900
0 -1
```

Ou seja, para um problema sem solução o número de lances é zero e a pontuação obtida é -1, significando tratar-se de um problema sem solução. Existem outros casos em que a solução é zero lances com pontuação zero. A resposta -1 deve exclusivamente ser usada

para variante 2, explicitando ausência de solução que cumpra a pontuação objectivo. Se em variante 1 não existir qualquer mancha inicial, zero lances com pontuação nula cumprem o objectivo de deixar a parede sem qualquer mancha.

Para os problemas não admissíveis o programa deve apenas escrever no ficheiro de saída, numa só linha, todo o cabeçalho presente no ficheiro de entrada.

Se o programa for invocado com ficheiros inexistentes, que não possuam a extensão `.tilewalls`, sem qualquer argumento ou com argumentos a mais, deve sair silenciosamente. Isto é, sem escrever qualquer mensagem de erro e/ou criar qualquer ficheiro de saída.

Sublinha-se que a única forma admissível para produção de output do programa é para ficheiro de saída, quando tal for possível. Qualquer escrita para `stdio` ou qualquer escrita em ficheiro que não siga o formato descrito constitui erro.

4 Primeira fase de submissões

Nesta secção apresentam-se os objectivos, especificações e funcionalidades que devem estar operacionais na data da primeira fase de submissões. Todas as funcionalidades, desta fase de submissão, dizem exclusivamente respeito ao processamento de paredes e extracção de informação a partir das mesmas.

O formato de invocação do programa é o mesmo que o definido anteriormente. Ou seja, o executável tem o mesmo nome e deve ser passado um argumento: o nome de um ficheiro, de extensão `.tilewalls1`, contendo um ou mais problemas. Este ficheiro tem um formato ligeiramente diferente do anterior. Por cada problema o cabeçalho necessita de cinco inteiros, que correspondem a: L(inhas), C(olunas), v(ariante), l(inha) e c(oluna). Os últimos dois inteiros são as coordenadas de um azulejo dentro da parede (ou fora dela). Ou seja, l, com $1 \leq l \leq L$ e c, com $1 \leq c \leq C$. Quando o par (l, c) estiver fora da parede estaremos em presença de um problema não admissível. Existem duas variantes, identificadas pelo inteiro 1 ou pelo inteiro 2. Qualquer outro valor para a variante define um problema não admissível. Após o cabeçalho, as linhas seguintes apresentam a parede de azulejos, tal como já descrito anteriormente para a fase final.

O terceiro inteiro, a variante, define a funcionalidade, que deve ser executada para a parede de azulejos a ele associada:

1. Indicar o valor da mancha associada às coordenadas (l, c) nessa matriz;

2. Produzir a matriz resultante da remoção da mancha existente na posição (l, c).

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 deve ter extensão `.singlestep` (a substituir `.tilewalls1`) e deve incluir todos os resultados associados com cada um dos problemas presentes no ficheiro de entrada. Para problemas de variante 1, o ficheiro de saída deve conter apenas duas linhas: a primeira repete o cabeçalho do problema – L, C, 1, l e c; a segunda indica o valor da mancha. Ou seja, a solução de um problema de variante 1 repete o cabeçalho desse problema, e acrescenta-lhe o valor da mancha que exista na posição pedida. Para problemas de variante 2, o ficheiro de saída deve conter o cabeçalho do problema na primeira linha, seguida da matriz resultante da aplicação do lance pedido, em que cada linha da matriz é apresentada numa só linha do ficheiro de saída.

O ficheiro de entrada pode conter mais do que um problema para resolver, e cada um destes pode ter qualquer uma das duas opções para a variante. Abaixo apresentam-se dois exemplos de problemas, para a matriz da Figura 1.

```
6 5 1 3 4
1 2 1 3 1
4 4 2 3 5
1 4 5 3 1
6 6 5 3 3
1 7 5 1 3
1 1 7 4 1
```

```
6 5 2 1 2
1 2 1 3 1
4 4 2 3 5
1 4 5 3 1
6 6 5 3 3
1 7 5 1 3
1 1 7 4 1
```

As soluções respectivas estão indicadas abaixo.

```
6 5 1 3 4
30
```

```
6 5 2 1 2
-1 -1 1 3 1
-1 2 2 3 5
1 4 5 3 1
4 4 5 3 3
1 6 5 1 3
6 7 7 4 1
```

Se, por hipótese, o ficheiro de entrada fosse a concatenação dos dois problemas acima, o ficheiro de saída seria também a concatenação das duas soluções apresentadas. Desta forma, é **obrigatória** a inclusão de uma linha em branco, como separador das diferentes soluções, assim como entre cada dois inteiros escritos em ficheiro de saída na mesma linha exista **apenas** um espaço em branco (como ilustrado nos exemplos).

Importa sublinhar que podem ocorrer problemas, em que a parede de azulejos não esteja completamente preenchida. No entanto, os azulejos presentes estão sempre correctamente arrumados. Isto é, a existirem colunas vazias, são sempre as mais à esquerda na matriz. Da mesma forma, a existirem colunas preenchidas de forma incompleta, apenas as posições superiores estão vazias. Também é possível que a matriz não contenha mancha nas coordenadas pedidas, seja em variante 1 ou 2. Nesses casos, a resposta é trivial. Na variante 1, não havendo mancha, o valor é zero. Na variante 2, não havendo mancha nas coordenadas indicadas, a matriz resultante é igual à de entrada.

Podem existir problemas mal definidos ou inadmissíveis: 1) porque a variante pedida é diferente de 1 ou de 2; ou 2) porque as coordenadas dadas estão fora das dimensões da matriz. Nesses casos, a resposta correcta é simplesmente repetir o cabeçalho que define o problema, sem mais informação.

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.

Quando os grupos de projecto estiverem constituídos, os alunos devem obrigatoriamente inscrever-se no sistema Fénix, no grupo designado como "Projecto de AED", que será criado em simultâneo com a publicação deste enunciado.

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 de uma ficha de autoavaliação em moldes a definir. A entrega desta ficha ratifica e lacra a submissão electrónica realizada. Só serão avaliadas as submissões que possuam ficha. Assim, caso um grupo não obtenha pontuação suficiente no conjunto das duas submissões para que a nota mínima seja atingível com significativa probabilidade deverá evitar submeter-se a avaliação. A forma de o fazer é não entregando a ficha de autoavaliação.

Tabela 1: Datas importantes do Projecto

Data	Documentos a Entregar
17 de Setembro de 2023	Enunciado do projecto disponibilizado na página da disciplina.
até 22 de Setembro de 2023 (18h00)	Inscrição dos grupos no sistema Fenix.
06 de Outubro de 2023 (18h00)	**Conclusão da primeira fase de submissões.**
27 de Outubro de 2023 (18h00)	**Conclusão da fase final de submissões.**
28 de Outubro de 2023 (18h00)	Entrega da ficha de autoavaliação (via Fénix).
Até 19 de Janeiro de 2024	Comunicação de notas e/ou convocatória para oral.

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. A falta de um ou de ambos os alunos à oral convocada pelos docentes acarreta reprovação automática de quem faltar, pese embora ser necessário justificar a(s) razão(ões) dessa falta.

As datas de entrega referentes aos vários passos da avaliação do projecto estão indicadas na Tabela 1. O facto de a comunicação de notas poder ser tão tardia face ao prazo final de submissões pode gerar algum desconforto entre os alunos. Sublinha-se que, assim que se concluir o processo de submissões, os alunos possuem alguma ideia ou estimativa de qual possa vir a ser a sua nota a projecto. Esta situação não é comparável a contextos em que a nota toda só é conhecida após a avaliação feita pelos docentes.

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 no sentido de estarem em condições de entregar a versão final até aos instantes finais, embora devam tentar realizar submissões com alguma antecedência para que surpresas, como erros de compilação infantis ou outros, não atrapalhem a sua capacidade de submeter implementações funcionais.

Note-se que, na versão final, o projecto só é considerado entregue aquando da entrega da ficha de autoavaliação. As submissões electrónicas do código não são suficientes para concretizar a entrega. A ausência de ficha de autoavaliação é a forma que os alunos têm de informar os docentes que não pretendem que a sua implementação seja avaliada. Não se consideram admissíveis a avaliação implementações sem submissões na fase final ou que nela obtenham pontuação nula.

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 Critérios de Avaliação

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

- Testes passados na primeira submissão electrónica – 10% a 20%
- Testes passados na última submissão electrónica – 65% a 55%
- Ficha de autoavaliação – 25%
  - Opções de projecto – 15%
  - Estruturação do código e comentários – 5%
  - Gestão de memória e tipos abstractos – 5%

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 x segundos[1]. Para o limite de memória, cada um dos testes não poderá exceder 100MB como pico de memória usada[2]. Cada teste resolvido dentro dos orçamentos temporal e de memória que produza soluções correctas recebe uma pontuação de acordo com o grau de dificuldade desse mesmo teste.

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.

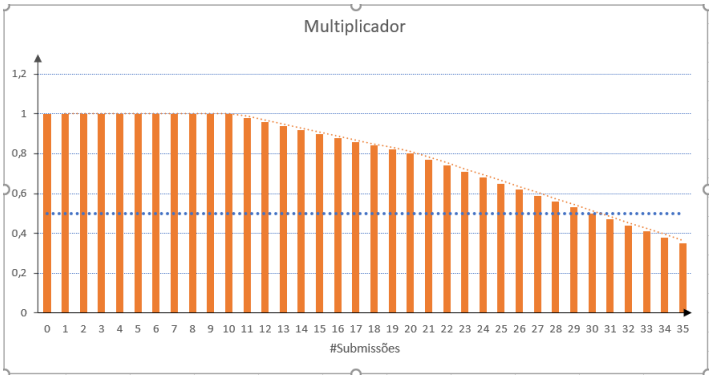


Figura 5: Multiplicador para cada uma das duas fases de submissão, como função do seu número.

Com o objectivo de se contribuir para que os alunos façam melhor uso do seu tempo no desenvolvimento das suas implementações, introduz-se um factor correctivo associado com o número de submissões de cada grupo. O site de submissões não pode, nem deve, ser encarado como um mecanismo de apoio ao desenvolvimento do projecto, na medida em que o *feedback* que fornece é muito limitado. Por outro lado, é entendimento da equipa docente que se deve valorizar a capacidade dos grupos em validarem as suas

[1] O limite de tempo para os testes da primeira fase será inferior, $y < x$, a definir.

[2] Não existe diferença nos limites de memória nas duas fases.

implementações fora do contexto do site de submissões. Por isso, dois projectos que resolvam todos os testes não valem o mesmo se um deles atingir essa marca em, por exemplo, 5 submissões e o outro atingir a mesma marca em 18.

Na Figura 5 apresenta-se um gráfico com o multiplicador associado ao número de submissões. Cada grupo dispõe de 10 submissões livres de penalização em cada uma das duas fases. As submissões livres servem para acomodar ligeiros percalços com erros de compilação ou erros na composição dos ficheiros a submeter. Se um grupo fizer mais que 10 submissões em alguma das fases, a pontuação que recebe nessa fase é penalizada em 2% por cada submissão adicional até à vigésima submissão. A partir da vigésima, cada submissão adicional é penalizada em 3%. Por exemplo, suponha-se que um grupo atinge a pontuação ilíquida de 180 pontos na primeira fase: recebe 180 pontos se tiver feito 10 ou menos submissões; recebe 162 pontos se tiver realizado 15 submissões; e recebe 117 pontos caso concretize 25 submissões.

Qualquer que venha a ser o número de submissões, avaliar-se-á apenas a que tiver obtido a melhor pontuação, independentemente de ser a última, a primeira ou uma intermédia, a menos que o grupo indique pretender que outra seja avaliada. Na eventualidade de existir empate na pontuação, também cabe ao grupo indicar, na ficha de autoavaliação, quais as suas duas submissões (uma por fase) pretende que sejam contabilizadas, sendo que a submissão da fase final que indicarem será a única a ser avaliada pelo corpo docente.

No que à avaliação da ficha de autoavaliação diz respeito, os elementos importantes são: rigor no seu preenchimento; apreciação da qualidade da abordagem geral ao problema e respectiva implementação, face às alternativas disponíveis; e clareza e suficiência do texto, na sua capacidade de descrever e justificar com precisão algumas das opções.

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 ou se necessitar de um número substancial de submissões para demonstrar a sua correcção, 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/` .