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

RAIDERS OF ALL PYRAMIDS

Versão 2.0 (25/Setembro/2021) – Primeira versão pública

2021/2022
1º Período do 1º Semestre

Conteúdo

Revisões

Versão 0.1 (09 de Setembro de 2021)	Primeiro texto incompleto
Versão 0.2 (20 de Setembro de 2021)	Tudo menos primeira fase de submissões e contabilização do número de submissões na avaliação
Versão 0.3 (20 de Setembro de 2021)	Tudo menos primeira fase de submissões e gralhas e outros corrigidas
Versão 1.0 (21 de Setembro de 2021)	Primeira versão completa
Versão 1.1 (24 de Setembro de 2021)	Versão melhorada e clarificada
Versão 1.2 (25 de Setembro de 2021)	Revisão de gralhas e remoção de redundâncias
Versão 2.0 (25 de Setembro de 2021)	Primeira versão pública

1 Introdução

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

Para a primeira fase de avaliação o trabalho consiste apenas no desenvolvimento de algumas funcionalidades testáveis que podem ser usadas posteriormente para desenvolver a solução final.

A entrega do código fonte em ambas as fases é feita através de um sistema automático de submissão que verificará a sua correcção e testará a execução do programa em algumas instâncias do problema.

1.1 Enquadramento

Segundo rezam os mais respeitáveis livros de história, os egípcios tiveram uma enorme trabalheira no desenho e construção de túmulos para os seus mais importantes cidadãos, por forma a dificultar o acesso às riquezas neles guardadas para os tempos de segunda vi(n)da dos faraós. Consta que as pirâmides e outros túmulos possuem laboriosos labirintos, construídos com o objectivo de fazer com que se perdessem neles os que tentassem a sua ilegítima invasão.

No entanto, apesar de toda a ciência e arte colocadas ao serviço dessas fascinantes construções, praticamente todas as pirâmides foram saqueadas com grande facilidade. A razão de tal foi que os saqueadores, ao invés de andarem a perder tempo em labirintos manhosos, optaram por "ir a direito", derrubando paredes.

O projecto para o ano lectivo de 2021/22 inspira-se nesta estratégia de saque, mas fá-lo com mais inteligência e menos esforço. Pretende-se desenvolver um programa que resolva jogos, descritos sob a forma de problemas do tipo labirinto e que produza a solução ótima da forma mais eficiente possível. O objectivo do jogo é encontrar um caminho desde um ponto de partida até um ponto de chegada, que corresponda à solução de menor custo de acordo com as regras do problema.

2 O labirinto RAIDERS OF ALL PYRAMIDS

Um labirinto RAIDERS OF ALL PYRAMIDS – ROAP –, consiste numa grelha de dimensão variável com L linhas e C colunas. Cada uma das $L \times C$ células pode ser de um de três tipos: i) negras; ii) brancas; ou iii) cinzentas e cada *labirinto* pode conter um qualquer número de células de cada tipo (naturalmente entre 0 e $L \times C$). As células negras correspondem a obstáculos intransponíveis (paredes blindadas) e nenhum caminho pode passar por elas. As células brancas correspondem a espaços abertos sem qualquer impedimento ou custo e podem ser atravessadas tantas vezes quantas se pretender. As células cinzentas correspondem a "paredes" cujo rebentamento tem um custo que é indicado na própria célula. O custo de qualquer caminho é assim igual à soma do custo das paredes que forem demolidas nesse caminho, dado que o percurso nas células brancas não tem custo. A Figura 1 mostra um exemplo da representação gráfica de um labirinto.

Como foi indicado, o objectivo do jogo é encontrar o caminho ótimo, i.e. de menor custo, entre um ponto de partida e um ponto de chegada. No entanto, dado que atravessar as células brancas não tem custos, é muito simples definir caminhos ligeiramente distintos mas que têm o mesmo custo (por exemplo o caminho poderá num dado momento avançar, recuar e voltar a avançar de novo, não tendo esta operação qualquer "custo"). Desta forma o que define o custo de um caminho são as paredes rebentadas, pelo que a solução do problema passa simplesmente pela indicação das paredes que foram abertas. Note-se que é possível que num dado labirinto haja mais do que uma solução com

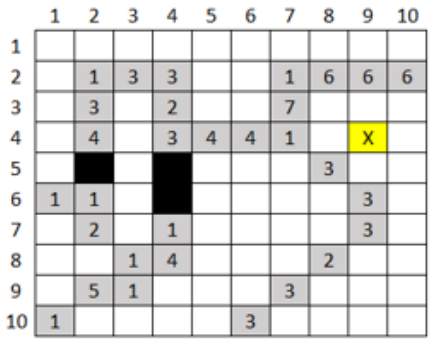

Figura 1: Exemplo de um labirinto RoaP.

o custo mínimo (abrindo uma sequência de paredes diferente). Nesse caso qualquer uma delas é aceite como solução do problema e essa solução seria dada pela indicação das células atravessadas no (ou num qualquer dos) caminho(s) ótimo(s).

As regras para percorrer o labirinto são relativamente simples e podem ser descritas facilmente e ilustradas através de um exemplo, como o do labirinto indicado na Figura 1 onde se assume que o ponto de partida é a célula $(1, 1)$ e o ponto de chegada é a célula $(4, 9)$ (na figura indicada a amarelo e preenchida com um "X"):

- Não é permitido atravessar diagonalmente as células, isto é, estando numa dada célula, as únicas células por onde se pode prosseguir caminho são as que estão na linha abaixo ou acima ou na coluna anterior ou posterior. Olhando para a Figura 1, se se estiver na célula $(3, 1)$, apenas são acessíveis as células $(3, 2)$ (uma parede cuja demolição tem custo 3) ou as células $(2, 1)$ e $(4, 1)$, com custo nulo. Também não se pode caminhar na diagonal, partindo uma parede. Por exemplo, a parede que está na célula $(2, 7)$ não pode ser quebrada para transitar entre $(1, 6)$ e $(3, 8)$.
- Não é possível demolir duas paredes de seguida. Na Figura 1, por exemplo, não é possível demolir a parede da célula $(8, 3)$ e depois a parede da célula $(8, 4)$ nem a $(9, 3)$. Ou seja, se se caminhar para uma célula onde está uma parede, a parede é demolida mas o caminho apenas pode prosseguir por células brancas (se estas estiverem disponíveis).
- Ao atravessar uma parede o caminho tem de prosseguir na mesma direcção. Na Figura 1, por exemplo, não é possível passar da célula $(9, 1)$ para a célula $(10, 2)$ abrindo a parede da célula $(10, 1)$. Esse caminho não é válido.
- Não é possível caminhar para fora dos limites do labirinto.

A solução do jogo descrito no labirinto da Figura 1 será abrir em sequência as portas $(3, 4)$, $(7, 4)$ e $(8, 8)$, com um custo total de 5.

Nas secções seguintes descreve-se o formato de utilização do programa a desenvolver, no que diz respeito aos ficheiros de entrada e saída; as regras de avaliação; e faz-se referência ao *Código de Conduta Académica*, que se espera ser zelosamente cumprido por todos os alunos que se submetam a esta avaliação.

Aconselha-se todos os alunos a lerem com a maior atenção todos os aspectos aqui descritos. Será obrigatória a adesão sem variações nem tonalidades a todas as especificações aqui descritas. A falha em cumprir alguma(s) especificação(ões) e/ou regra(s) constante(s) neste enunciado acarretará necessariamente alguma forma de desvalorização do trabalho apresentado.

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

3 O programa RAIDERS OF ALL PYRAMIDS

O programa a desenvolver deve conseguir ler a configuração de um labirinto ROAP, contendo informação sobre a dimensão da grelha, a localização da célula final e a posição das células negras e cinzentas, e gerar uma solução para o problema. Assume-se que o ponto de partida é sempre a célula $(1, 1)$ e que o ponto de chegada tem de ser uma célula branca. A solução consiste na identificação das paredes que devem ser demolidas no caminho entre o ponto inicial e final, bem como a indicação do custo final quando um caminho seja possível.

Uma nota suplementar: o programa deve estar também preparado para lidar com três tipos especiais de labirintos: i) os que não têm solução, isto é, aqueles em que não há nenhum caminho que permita ir da célula inicial à final de acordo com as regras do jogo; ii) aqueles em que a solução não tem qualquer custo, isto é, não é necessário partir qualquer parede para ir do ponto inicial ao final; iii) aqueles em que há mais do que uma solução - existe mais do que um caminho, atravessando paredes distintas, com o mesmo custo final (neste caso, como foi atrás indicado, qualquer solução é aceitável). Note-se que o número de paredes partidas não é relevante em termos da otimalidade de solução; apenas interessa o custo total.

Nas secções seguintes descreve-se a forma como o programa deve ser invocado, a forma como a configuração do jogo (labirinto) é descrita e o formato que têm de ter os dados de saída, ou seja, a forma de descrever a solução encontrada.

Na secção 4 descrever-se-ão os objectivos e especificações para a primeira fase de submissões.

3.1 Execução do Jogador

O programa de RAIDERS OF ALL PYRAMIDS deverá ser invocado na linha de comandos da seguinte forma:

```
aed$ roap maze.in
```

onde:

`roap` designa o nome do ficheiro executável contendo o programa de RAIDERS OF ALL PYRAMIDS;

`maze.in` é um exemplo do nome do ficheiro contendo a descrição da configuração do labirinto, de acordo com o formato indicado na Secção 3.2 (ver de seguida).

Os ficheiros utilizados para a descrição dos labirintos terão de ter a extensão `.in`, mas poderão ter qualquer nome, nada sendo assumido para esse efeito. Se se pedir a execução do programa indicando-se o nome de um ficheiro que não exista ou sem indicar qualquer nome, o programa deve simplesmente abortar a execução e concluir a sua execução de forma silenciosa, sem produzir qualquer mensagem de erro. Assume-se que os ficheiros contendo a descrição da configuração dos labirintos, quando existam, estão sempre correctos de acordo com o formato de entrada, pelo que o programa a desenvolver não tem de se preocupar com a detecção de erros na descrição dos labirintos.

3.2 Formato de Entrada

A descrição da configuração de um labirinto ROAP é feita num ficheiro, obedecendo às seguintes regras:

- cada ficheiro pode conter a descrição de um ou vários labirintos e quando tiver mais que um, existirá sempre, pelo menos, uma linha vazia de separação entre labirintos consecutivos. Isto serve apenas para humanos visualizarem.
- cada labirinto é iniciado com dois números inteiros, L e C. Estes números representam respetivamente o número de linhas e o número de colunas do tabuleiro do labirinto (por exemplo o labirinto ilustrado na Figura 1 tem 10 linhas e 10 colunas).
- após a indicação da dimensão do tabuleiro haverá dois números inteiros, L_t e C_t. Estes números indicam as coordenadas (linha e coluna) do ponto de chegada, onde se encontra o tesouro.
- o cabeçalho do problema conclui-se com a indicação do valor P de células negras e cinzentas no labirinto.
- terminado o cabeçalho haverá P trios de números inteiros que correspondem à indicação do posicionamento das células negras e cinzentas, bem como o custo associado às mesmas. O formato de cada trio é o mesmo, nomeadamente:
  - cada trio contém três números inteiros, L_i, C_i e V_i; os dois primeiros números L_i e C_i representam respectivamente as coordenadas (linha L_i e coluna C_i) da localização da célula preta ou cinzenta; o terceiro número, V_i, pode ter o valor -1, indicando uma célula negra, ou ser um inteiro positivo, indicando o custo de abrir a parede naquela localização, para $i = 1, 2, \ldots, P$.

A título de exemplo, o ficheiro que descreve o labirinto da Figura 1 pode ser o indicado na Figura 2.

Sublinha-se que não existe qualquer necessidade de ter as células negras e cinzentas apresentadas por qualquer ordem específica, assim como não é obrigatório que os ficheiros estejam compactados como se apresenta na figura. Nesta figura apenas se usou o varrimento de cima para baixo e da esquerda para a direita para facilitar a confirmação visual de que aquele ficheiro de entrada está correcto para o labirinto indicado.

3.3 Formato de Saída de Dados

O resultado da execução do programa de RAIDERS OF ALL PYRAMIDS consiste em determinar o caminho a percorrer entre o ponto de partida e o ponto de chegada indicado, tal que esse caminho tem o menor custo, isto é, destrói paredes tais que o custo total é o menor possível, para cada um dos labirintos presentes no ficheiro de entrada.

A(s) solução(ões) deve(m) ser colocada(s) num ficheiro de saída, cujo nome deve ser o mesmo do ficheiro de descrição da configuração do labirinto mas **com extensão** `.sol`. Este ficheiro deve ser criado e aberto pelo programa. Por exemplo, se o ficheiro com a configuração do labirinto se chama `maze32.in`, o ficheiro de saída deve chamar-se `maze32.sol`.

Para cada problema a estrutura do ficheiro de saída é a seguinte:

- a primeira linha deve conter um único número que é o custo total da solução encontrada; se o labirinto não tiver solução, o valor indicado deve ser -1.
- quando o labirinto tiver solução estritamente positiva, a segunda linha deve indicar o número de paredes partidas, B. Se o custo tiver sido 0, nao é necessário indicar que se derrubaram 0 paredes.

```
10 10
4 9
33
2 2 1
2 3 3
2 4 3
2 7 1
2 8 6
2 9 6
2 10 6
3 2 3
3 4 2
3 7 7
4 2 4
4 4 3
4 5 4
4 6 4
4 7 1
5 2 -1
5 4 -1
5 8 3
6 1 1
6 2 1
6 4 -1
6 9 3
7 2 2
7 4 1
7 9 3
8 3 1
8 4 4
8 8 2
9 2 5
9 3 1
9 7 3
10 1 1
10 6 3
```

Figura 2: Exemplo do ficheiro de entrada para o labirinto apresentado na Figura 1.

- para $B > 0$, as B linhas subsequentes devem ter, cada uma delas, três números inteiros correspondentes às coordenadas, linha e coluna de cada parede partida no caminho encontrado, seguida do custo de a partir. A ordem pela qual as paredes são indicadas tem de ser a correspondente ao caminho utilizado para ir do ponto inicial ao ponto de chegada. Naturalmente se não houver solução para o labirinto ou se a solução não implicar a abertura de qualquer parede, não será indicada nenhuma linha adicional (parede).

```
5

3
3 4 2
7 4 1
8 8 2
```

Figura 3: Exemplo de um ficheiro de saída para o labirinto da Figura 1.

Como exemplo considere o labirinto apresentado na Figura 1. A solução apresentada para este labirinto, como já foi dito, passa por partir, em sequência, as paredes $(3, 4)$, $(7, 4)$ e $(8, 8)$ num caminho cujo custo total é 5. Neste caso, o ficheiro de saída tem o formato apresentado na Figura 3.

Nos casos em que o ficheiro de entrada possui mais que um labirinto, as soluções para cada um deles no ficheiro de saída devem estar separadas com uma linha em branco.

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 labirintos para extracção de informação a partir dos mesmos.

O formato de invocação do programa será o mesmo que o definido anteriormente, mas existem algumas diferenças de conteúdo.

```
aed$ ./roap -s maze.in1
```

O executável tem o mesmo nome, mas passarão a ser passados dois argumentos. O primeiro argumento explicita que o programa deverá executar funcionalidades associadas com a primeira fase. Os ficheiros de labirintos possuem extensão diferente da usada na fase final, porque o seu conteúdo é ligeiramente diferente.

Aqui também os sucessivos problemas, no ficheiro de extensão `.in1`, estarão separados por, pelo menos, uma linha em branco. Este ficheiro tem formato ligeiramente diferente do anteriormente definido, como se indicará mais à frente.

As variantes de funcionamento são:

A1 – para as coordenadas (l, c) identificar o tipo de célula;

A2 – para as coordenadas (l, c) indicar se existe algum vizinho que seja célula branca;

A3 – para as coordenadas (l, c) indicar se existe algum vizinho que seja célula cinzenta;

A4 – para as coordenadas (l, c) indicar se existe algum vizinho que seja célula negra;

A5 – para as coordenadas (l, c), se forem as coordenadas de uma célula cinzenta, indicar se é quebrável;

A6 – para as coordenadas (l_1, c_1) e (l_2, c_2), se ambas forem células brancas, indicar se pertencem à mesma sala.

```
10 10
2 5 A3
33
2 2 1
2 3 3
...
```

Figura 4: Primeiras linhas do ficheiro de entrada para o labirinto apresentado na Figura 1 em variante A3 da primeira fase.

Os ficheiros de extensão `.in1` possuem estrutura semelhante aos de extensão `.in`, mudando apenas o que nestes é a informação da célula de destino, para passar a interpretar-se como as coordenadas da célula cuja natureza se pretende investigar seguidas de uma das seis cadeias de caracteres acima identificadas (ver Fig. 4). No caso específico da variante `A6`, o segundo par de coordenadas é apresentado a seguir à cadeia de caracteres identificando a variante.

Em variante `A1` a resposta é uma de 3: 0, se for célula branca; valor da célula, se for célula não branca; -2 se for uma célula fora do labirinto. Para as restantes variantes a resposta é uma de 3: -2, se a (alguma das) célula estiver fora do labirinto; 0, se a resposta à pergunta for negativa; 1, se a resposta à pergunta for afirmativa. Sublinha-se que no caso concreto da variante `A5` é uma excepção a esta especificação. Quando as coordenadas não corresponderem à posição de uma célula cinzenta a resposta deverá ser -1, desde que dentro do labirinto. Para problemas de variante `A6` diz-se que duas células pertencem à mesma sala se não for necessário quebrar qualquer parede ao caminhar de uma para a outra.

Se tomarmos as coordenadas $(2, 5)$ da Figura 1 as respostas para cada uma das primeiras cinco variantes seriam, respectivamente: 0; 1; 1; 0; -1. Se tomássemos, em alternativa as coordenadas $(9, 2)$ as respostas seriam: 5; 1; 1; 0; 1. Tomando o par de coordenadas $(2, 5)$ e $(1, 10)$ a resposta em variante `A6` é 1, mas para $(2, 5)$ e $(5, 6)$ será 0.

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 mapas, mas deverá ter extensão `.sol1` e limita-se a apresentar a resposta a cada problema constante no ficheiro de entrada. Note-se que a solução de cada problema nesta fase é apenas um número inteiro.

Qualquer ficheiro de entrada poderá conter mais do que um problema para resolver e cada um desses problemas poderão estar associados a labirintos de diferentes dimensões.

O ficheiro de saída será a concatenação das soluções de todos os problemas resolvidos. Aqui também é **obrigatória** a inclusão de apenas uma linha em branco como separador das sucessivas soluções.

5 Visualizador de RAIDERS OF ALL PYRAMIDS

O corpo docente disponibilizará brevemente, na página da disciplina dedicada ao projecto, um programa **visualizador** que permite visualizar um labirinto RAIDERS OF ALL PYRAMIDS. Eventualmente poderão ser disponibilizados outros programas que auxiliem os alunos na visualização ou verificação das soluções e nesse caso será feito um aviso na página da disciplina.

O visualizador indicado deve ser invocado com um único argumento que é o nome do ficheiro contendo a descrição do labirinto. Por questões de ordem gráfica e computacional, o visualizador estará limitado a labirintos com um número máximo de linhas ou colunas. O programa a desenvolver deverá no entanto ser capaz de resolver labirintos de qualquer dimensão, sem limitações. O modo de invocação é o seguinte:

```
aed$ visroap maze.in
```

onde:

`visroap` designa o nome do ficheiro que contém o programa visualizador;

`maze.in` designa o nome do ficheiro que contém a descrição do labirinto;

O visualizador foi desenvolvido em *Tcl/Tk* e estará instalado nas máquinas do laboratório.

6 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.

Tabela 1: Datas importantes do Projecto

Data	Documentos a Entregar
24 de Setembro de 2021	Enunciado do projecto disponibilizado na página da disciplina.
até 30 de Setembro de 2021 (18h00)	Inscrição dos grupos no sistema Fénix.
20 de Outubro de 2021, (18h00)	Conclusão da primeira fase de submissões.
12 de Novembro de 2021 15h 18h	Conclusão da fase final de submissões. Fim da submissão electrónica. Entrega da ficha de auto-avaliação.
até algures de Fevereiro de 2022	Eventual discussão do trabalho (data combinada com cada grupo).

Quando os grupos de projecto estiverem constituídos, os alunos devem obrigatoriamente inscrever-se no sistema Fénix, no agrupamento Projecto, que já foi criado e terá as inscrições abertas, em princípio, até 30 de Setembro. Esta inscrição é necessária para que se crie um ficheiro de grupos para obtenção da password de acesso ao site de submissões.

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 auto-avaliação em moldes a definir. Esta entrega ratifica e lacra a submissão electrónica anteriormente realizada.

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 dentro dos prazos indicados.

Note-se que, na versão final, o projecto só é considerado entregue aquando da entrega da ficha de auto-avaliação. As submissões electrónicas do código não são suficientes para concretizar a entrega. A ausência de ficha de auto-avaliação é a forma que os alunos têm de indicar que não pretendem que a sua implementação seja avaliada.

6.1 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.

6.2 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. As instruções específicas de submissão serão disponibilizadas na página da disciplina, constituindo uma extensão deste enunciado.

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.

6.3 Critérios de Avaliação

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

- Testes passados na primeira submissão electrónica – 10% a 15%
- Número de submissões na primeira fase – ver descrição abaixo
- Testes passados na última submissão electrónica – 65% a 60%
- Número de submissões na fase final – ver descrição abaixo
- Estruturação do código e comentários – 5%
- Gestão de memória e tipos abstractos – 5%
- Ficha de auto-avaliação – 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 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 a sua complexidade. Um teste considera-se errado se, pelo menos, um dos problemas do ficheiro de entrada correspondente for incorrectamente resolvido. Nesse caso, a pontuação recebida será zero.

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.

Com o objectivo de se contribuir para que os alunos façam melhor uso do seu tempo no desenvolvimento das suas implementações, este ano lectivo vai-se introduzir um factor correctivo associado ao 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 serem capazes de validar as suas implementações fora do contexto do site de submissões. Por isso, dois projectos que resolvam todos os testes no site de submissões não valem o mesmo se um deles atingir essa marca em, por exemplo, 5 submissões e o outro atingir a mesma marca em 18.

[1] O limite de tempo para os testes da primeira fase será inferior, $y < x$. Ambos serão definidos posteriormente nas instruções de submissão.

[2] Não existe diferença nos limites de memória nas duas fases, mas pode dar-se o caso de ser necessário alterar o limite padrão.

Na Figura 5 apresenta-se um gráfico com o multiplicador associado ao número de submissões. Cada grupo dispõe de 10 submissões livres em cada uma das duas fases. As submissões livres servem para pequenos percalços como 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 submissão, cada submissão adicional é penalizada em 3%. Por exemplo, suponha-se um grupo que atinge a pontuação ilíquida de 180 pontos na primeira fase: recebe esses 180 pontos por inteiro se fizer 10 ou menos submissões; recebe 162 pontos se tiver feito 15 submissões; e recebe 117 pontos se tiver feito 25 submissões.

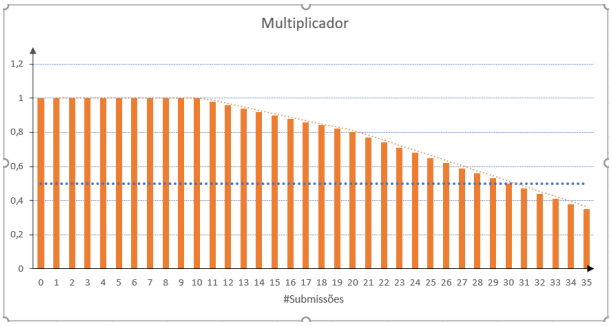

Figura 5: Multiplicador para cada uma das duas fases de submissão.

Outra alteração que irá ser implementada neste projecto é a não obrigatoriedade de se avaliar a última submissão feita dentro dos prazos, para cada uma das duas fases de submissão. Compete aos alunos indicar na ficha de auto-avaliação quais as suas duas submissões (uma por fase) que pretendem ver contabilizadas para produção da pontuação, sendo que a submissão da fase final que indicarem será a única a ser avaliada pelos docentes.

No que à avaliação da ficha de auto-avaliação diz respeito, os elementos de avaliação incluem: rigor do 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 correcta e eficazmente os problemas a que for submetido e que os alunos sejam também eficientes no seu desenvolvimento. 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 elevado de submissões para melhorar a sua correcção, dificilmente terá uma nota positiva.

7 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/` .