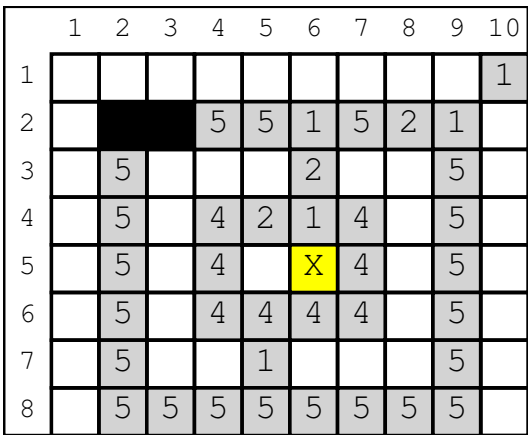
Instituto Superior Técnico

MEEC - Mestrado em Engenharia Eletrotécnica e de Computadores

Algoritmos e Estruturas de Dados

Enunciado do Projecto



DOORS MAZE

Versão 1.3 (1/11/2013)

2013/2014

1º Semestre

Conteúdo

Revisões

Versão 1.0 (18 de outubro de 2013)	Versão inicial
Versão 1.1 (21 de outubro de 2013)	Clarificação do formato de saída quando não há solução.
Versão 1.2 (28 de outubro de 2013)	Correção do exemplo da Figura 2.
Versão 1.3 (1 de novembro de 2013)	Correção da data final para inscrições dos grupos.

1 Introdução

Neste projecto 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 DOORS MAZE

O DOORS MAZE 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 e nenhum caminho pode passar por elas. As células brancas correspondem a caminhos sem qualquer impedimento ou custo e podem ser atravessadas tantas vezes quantas se pretender. As células cinzentas correspondem a "portas" cuja abertura tem um custo que é indicado na própria célula. O custo de qualquer caminho é assim igual à soma do custo das portas que forem abertas nesse caminho, dado que o percurso nas células brancas não tem custo. A Figura 1 mostra um exemplo da representação gráfica de um labirinto.

	1	2	3	4	5	6	7	8	9	10
1										1
2				5	5	1	5	2	1	
3		5				2			5	
4		5		4	2	1	4		5	
5		5		4		X	4		5	
6		5		4	4	4	4		5	
7		5			1				5	
8		5	5	5	5	5	5	5	5	

Figura 1: Exemplo de um labirinto DOORS MAZE.

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 portas que atravessa, pelo que a solução do problema passa simplesmente pela indicação das portas que foram abertas. Note-se que é possível que num dado labirinto haja mais do que uma solução com o custo mínimo (abrindo uma sequência de portas 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 $(5, 6)$ (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 porta cujo acesso tem custo 5) ou as células $(2, 1)$ e $(4, 1)$, com custo nulo. Não é possível, por exemplo, abrir a porta que está na célula $(2, 9)$ para aceder à célula $(3, 8)$.
- não é possível abrir duas portas de seguida. Na Figura 1, por exemplo, não é possível abrir a porta da célula $(2, 6)$ e depois a porta da célula $(3, 6)$. Ou seja, se se caminhar para uma célula onde está uma porta, a porta é aberta mas o caminho apenas pode prosseguir por células brancas (se estas estiverem disponíveis).
- ao atravessar uma porta o caminho tem de prosseguir na mesma direcção. Na Figura 1, por exemplo, não é possível passar da célula $(1, 9)$ para a célula $(2, 10)$ abrindo a porta da célula $(1, 10)$. 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 $(2, 8)$, $(7, 5)$ e $(4, 5)$, com um custo total de 5.

3 O programa DOORS MAZE

O programa a desenvolver deve conseguir ler a configuração de um labirinto DOORS MAZE, 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 portas que devem ser abertas no caminho entre o ponto inicial bem como a indicação do custo final.

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 abrir qualquer porta para ir do ponto inicial ao final; iii) aqueles em que há mais do que uma solução - existe mais do que um caminho, atravessando portas distintas, com o mesmo custo final (neste caso, como foi atrás indicado, qualquer solução é aceitável). Note-se que o número de portas abertas 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.

3.1 Execução do Jogador

O programa de DOORS MAZE deverá ser invocado na linha de comandos da seguinte forma:

```
aed$ doorsmaze maze.in
```

onde:

`doorsmaze` designa o nome do ficheiro executável contendo o programa de DOORS MAZE;

`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 dar uma mensagem de erro apropriada. Assume-se que os ficheiros contendo a descrição da configuração dos labirintos 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 DOORS MAZE é feita num ficheiro, obedecendo às seguintes regras:

- cada ficheiro contém a descrição de um único labirinto.
- a 1ª linha contém dois números inteiros, NL e NC. 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 8 linhas e 10 colunas).
- a 2ª linha contém dois números inteiros, CL e CC. Estes números indicam as coordenadas (linha e coluna) do ponto de chegada.
- a 3ª linha contém um número inteiro P, que indica o número de células negras e cinzentas no labirinto.
- as seguintes P linhas correspondem à indicação do posicionamento das células negras e cinzentas, bem como o custo associado às mesmas. O formato de cada linha é o mesmo, nomeadamente:
  - cada linha contém três números inteiros, PL, PC e PV; os dois primeiros números PL e PC representam respectivamente as coordenadas (linha PL e coluna PC) da localização da porta ou célula negra; o terceiro número, PV, pode ter o valor -1, indicando uma célula negra, ou ser um inteiro positivo, indicando o custo de abrir uma porta naquela localização.

  Estas P linhas estão ordenadas de forma não decrescente nos valores de PL e para o mesmo valor de PL estão ordenadas de forma crescente nos valores de PC.

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

3.3 Formato de Saída de Dados

O resultado da execução do programa de DOORS MAZE 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, abre portas cujo custo total é o menor possível.

A solução deve ser colocada 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`.

A estrutura do ficheiro de saída é a seguinte:

```
8 10
5 6
39
1 10 1
2 2 -1
2 3 -1
2 4 5
2 5 5
2 6 1
2 7 5
2 8 2
2 9 1
3 2 5
3 6 2
3 9 5
4 2 5
4 4 4
4 5 2
4 6 1
4 7 4
4 9 5
5 2 5
5 4 4
5 7 4
5 9 5
6 2 5
6 4 4
6 5 4
6 6 4
6 7 4
6 9 5
7 2 5
7 5 1
7 9 5
8 2 5
8 3 5
8 4 5
8 5 5
8 6 5
8 7 5
8 8 5
8 9 5
```

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

```
5
2 8 2
7 5 1
4 5 2
```

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

- 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.
- as linhas subsequentes devem ter, cada uma delas, três números inteiros correspondentes às coordenadas, linha e coluna de cada porta aberta no caminho encontrado, seguida do custo de abrir a mesma. A ordem pela qual as portas são indicadas deve 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 porta, não será indicada nenhuma linha adicional (porta).

Como exemplo considere o labirinto apresentado na Figura 1. A solução apresentada para este labirinto, como já foi dito, passa por abrir, em sequência, as portas $(2, 8)$, $(7, 5)$ e $(4, 5)$ num caminho cujo custo total é 5. Neste caso, o ficheiro de saída tem o formato apresentado na Figura 3.

4 Visualizador de DOORS MAZE

O corpo docente disponibilizará brevemente, na página da disciplina dedicada ao projecto, um programa **visualizador** que permite visualizar um labirinto DOORS MAZE. 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$ visdm maze.in
```

onde:

`vismd` 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.

5 Avaliação do Projecto

O projecto está dimensionado para ser feito por grupos de dois alunos, não se aceitando grupos de outra dimensão. 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.

Tabela 1: Datas importantes do Projecto

Data	Documentos a Entregar
até 22 de outubro de 2013, 3ª feira	Enunciado do projecto disponibilizado na página da disciplina.
até 15 de novembro de 2013 (24h)	Inscrição dos grupos no sistema Fénix.
9 de dezembro de 2013, 2ª feira 10h 15h	**1ª Data de entrega do projecto:** Submissão eletrónica do projecto. Entrega do relatório do projecto em papel.
10 de dezembro de 2013, 3ª feira 10h 15h	**2ª Data de entrega do projecto: penalização de um (1) valor** Submissão eletrónica do projecto. Entrega do relatório do projecto em papel.
11 de dezembro de 2013, 4ª feira 10h 15h	**3ª Data de entrega do projecto: penalização de dois (2) valores** Submissão eletrónica do projecto. Entrega do relatório do projecto em papel.
	Submissões posteriores a 11 de dezembro têm penalização de 20 valores.
16 a 20 de dezembro de 2013	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 grupo de Projecto correspondente, que será criado oportunamente.

A avaliação do projecto envolve duas datas distintas: numa primeira data é entregue o relatório (em papel) e o código do projecto (eletronicamente, em moldes a definir posteriormente) e numa segunda data é feita a discussão e defesa do mesmo. No entanto, em princípio só terão que realizar a discussão os grupos em que se verifique uma diferença superior a três valores entre a classificação do laboratório e a classificação provisória do projecto e/ou para grupos em que a diferença de desempenho no laboratório entre os dois alunos seja também superior a três valores. O corpo docente reserva-se no entanto o direito de convocar para discussão qualquer grupo de alunos. As datas relevantes referentes aos vários passos da avaliação do projecto estão indicadas na Tabela 1.

Note-se que o projecto só é considerado entregue aquando da entrega do relatório em papel. As submissões eletrónicas do código não são suficientes para concretizar a entrega. Um grupo que faça a submissão eletrónica 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.

A avaliação do projecto basear-se-á no funcionamento do trabalho, qualidade do código, qualidade do relatório e, eventualmente, na discussão.

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 eletrónica o código do programa (ficheiros `.h` e `.c`) e uma `Makefile` para gerar o executável. Todos os ficheiros (`*.c`, `*.h`, `Makefile`, etc.) 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 2 e dirigidos ao docente do laboratório respectivo. Os grupos compostos por alunos que frequentam laboratórios distintos devem indicar um dos docentes desses laboratórios. Os alunos que não estão a frequentar o laboratório devem mencionar essa situação.

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 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, incluído um fluxograma detalhado e um texto claro, mas sucinto, indicando a divisão lógica e funcional dos módulos desenvolvidos para a resolução do problema, indicando os respectivos objectivos, as funções utilizadas e as estruturas de dados de suporte;
- Uma descrição detalhada das estruturas de dados utilizadas e justificação das mesmas;
- Descrição dos algoritmos usados (por exemplo, na manipulação das estruturas de dados);
- Uma descrição dos subsistemas funcionais que existam e, para cada um
  - a descrição dos objectivos do subsistema (até 5 linhas);

Tabela 2: Grelha de Avaliação do Projecto

Situação	**Nota Máxima**
Programa não compila, termina sistematicamente de formas ilegais com erros de execução, etc.	Trabalho não é aceite
Programa funciona de forma algo incompleta ou incorrecta, i.e., determina alguns valores mas apresenta resultados total ou parcialmente errados, não obtém a solução completa para o problema ou tem qualquer outra limitação considerada grave.	12 valores
Programa determina a solução do problema mas funciona de forma ineficiente (má gestão de memória ou computacionalmente ineficaz).	16 valores
Programa funciona de forma eficiente e completa, com código correctamente comentado e estruturado e relatório bem escrito, claro e completo.	20 valores

  - o nome do módulo onde estão definidas as estruturas de dados (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 dos requisitos computacionais do programa desenvolvido, tanto em termos da memória que utiliza como da complexidade temporal, com particular ênfase no custo das operações de processamento sobre as estruturas de dados;
- Uma análise crítica do funcionamento do programa e a avaliação do desempenho do projecto implementado;
- Pelo menos, um pequeno exemplo completo de aplicação, com descrição da utilização das estruturas de dados em cada passo.

A nota final do projecto será calculada com base nas diferentes componentes de avaliação mas estará condicionada ao funcionamento do programa de acordo com os valores máximos indicados na Tabela 2.

5.4 Discussão

Só terão de realizar discussão os grupos em que se verifique uma diferença entre a classificação provisória do projecto e a obtida em laboratório superior a três valores, ou grupos em que os dois elementos possuam uma diferença na classificação do laboratório superior a três valores. O corpo docente reserva-se no entanto o direito de convocar para discussão qualquer grupo de alunos. As orais marcadas pelo corpo docente realizar-se-ão na última semana de aulas de dezembro. Os alunos têm direito a uma oral de recurso caso estejam insatisfeitos com a nota atribuída. Estas orais realizar-se-ão no período entre o final das aulas e o primeiro exame. As discussões serão feitas em grupo embora possam ser colocadas questões distintas aos elementos de cada grupo e os mesmos possam vir a obter nota diferente no projecto, quando tal se justificar. Se um grupo for convocado para discussão, ambos os elementos do grupo têm de comparecer à mesma; qualquer elemento que falte à discussão do grupo está automaticamente reprovado.

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. 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/` .