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

IST Navigate - The Infinite Loop of Doom

Versão 1.0 (05/Setembro/2024)

2024/2025
1º Trimestre

Conteúdo

1 Introdução

Este enunciado descreve as especificações do projecto da disciplina de Algoritmos e Estruturas de Dados e tem duas fases. A primeira fase consiste no desenvolvimento de algumas funcionalidades que, apesar de não determinarem em absoluto a solução final, podem ser usadas posteriormente na fase final. Em ambos os casos a aplicação deve ser escrita em linguagem C, cumprir todos objectivos descritos neste texto e ser computacionalmente eficiente (tempo e memória).

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 para algumas instâncias do problema.

Aconselha-se todos os alunos a lerem com a maior atenção o enunciado. 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.

2 O problema – NAVIGATE

Neste projecto pretende-se desenvolver um programa que seja capaz de navegar em mapas estáticos rectangulares em que os únicos movimentos admissíveis são deslocamentos horizontais e verticais. Cada célula desses mapas possui um atributo que representa a energia, que pode ser negativa, nula ou positiva. Na Figura 1 ilustra-se um mapa inicial.

-1	-1	0	0	0	0	20	0	0	-2	0	0	-1	-1	-1	-1
-1	-1	0	-3	-3	-3	0	-3	-1	-3	-3	-3	-3	-3	-3	-3
-9	-9	0	-3	-3	-3	0	-3	0	-3	-3	-3	-3	-3	-3	-3
-3	-3	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3
-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3
-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3
-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3
-3	-3	-3	-3	-3	-3	-3	-3	100	-3	-3	-3	-3	-3	-3	-3
-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3
-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3
-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3
-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3
-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3
-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3
-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3
-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3	-3

Figura 1: Exemplo de um mapa.

Para além do mapa existe um aluno (da LEEC, claro) colocado numa das células do mapa (que representam cadeiras, mas também outras atividades) e esse aluno possui um valor dado de energia inicial. Pretende-se que o aluno navegue ao longo do mapa gastando mas também recebendo energia ao entrar numa célula[1], com o objectivo de aumentar a sua energia, se tal for possível. As seguintes restrições aplicam-se aos caminhos realizados:

[1] A célula incial não pode entrar nestas contas porque o aluno já se encontra nela. Logo, qualquer que seja o seu valor deve assumir-se que já terá sido contabilizado na informação da energia inicial.

1. O aluno ao percorrer um caminho não pode usar uma qualquer célula mais que uma vez, incluindo a célula de partida.

2. O caminho realizado pelo aluno possui exactamente k passos, nem menos nem mais.

3. O aluno não pode prosseguir caminho se a sua energia alguma vez deixar de ser positiva.

Para além das restrições acima, são definidos dois objectivos para a fase final do projecto: no final do caminho, o aluno deve: 1) atingir ou superar um valor de energia específico ou 2) atingir a maior energia possível para o mapa em que é colocado.

Antes de mais, é necessário adoptar um sistema de coordenadas único. Na Figura 2, apresenta-se um possível mapa, omitindo os valores nas células, explicitando o sistema de coordenadas que todos os grupos deverão adoptar para ambas as fases de submissão.

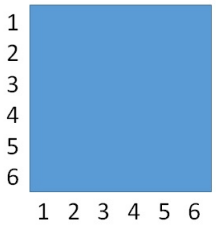

Figura 2: Sistema de coordenadas.

3 Fase intermédia do projecto

Nesta secção explicitam-se os objectivos, especificações e funcionalidades que deverão estar operacionais na primeira fase de submissões. Nesta fase devem ser implementadas as seguintes funcionalidades:

1. Tarefa #1: o programa deverá identificar o maior valor de energia positivo que no mapa dado diste k ou menos passos da posição inicial.

2. Tarefa #2: o programa deverá somar todos os valores de energia positivos das células dentro do mapa que estejam a uma distância menor ou igual a k da posição inicial.

3. Tarefa #3: o programa deverá produzir o caminho de (l_1, c_1) até (l_2, c_2), percorrendo primeiro na vertical e depois na horizontal [2].

Na tarefa #1 e tarefa #2, a distância é calculada admitindo apenas deslocamentos horizontais e verticais.

[2]Por exemplo, se $l_1 = 12$, $c_1 = 2$, $l_2 = 8$ e $c_2 = 4$, os passos serão: (11, 2), (10, 2), (9, 2), (8, 2), (8, 3) e (8, 4). Se, em alternativa, fosse $l = 8$, $c = 4$, $l_2 = 12$ e $c_2 = 2$, os passos seriam: (9, 4), (10, 4), (11, 4), (12, 4), (12, 3) e (12, 2).

3.1 Formato de entrada

O ficheiro de entrada tem extensão `.1maps` e poderá conter mais do que um problema para resolver e cada um desses problemas poderá ter diferentes dimensões. Por cada problema existe uma primeira linha de cinco ou sete inteiros, referido como o cabeçalho do problema, que corresponde a:

1. Dimensão do mapa, L e C, sendo L o número de linhas e C o número de colunas do mapa.

2. Coordenadas de um ponto de partida, l e c que identificam uma célula do mapa. Ou seja, l, com $1 \leq l \leq L$ e c, com $1 \leq c \leq C$

3. Número de passos k do percurso que poderá assumir $k < 0$ (tarefa #1), $k > 0$ (tarefa #2) ou $k = 0$ (tarefa #3). Não existe limite superior nem inferior para o valor de k e para a tarefa #1 o número de passos a usar será $|k|$, isto é, necessita de converter o k lido para valor positivo.

4. Se $k = 0$, então existirão mais dois inteiros nessa linha, l_2 e c_2.

Após o cabeçalho existirão $L \times C$ inteiros que identificam a energia associada a cada célula do mapa. Sublinha-se que a polaridade de k identifica a tarefa e o seu valor absoluto o número de passos.

3.2 Formato de saída

O ficheiro de saída da primeira fase, tem o mesmo nome que o ficheiro de problemas, mas deverá ter extensão `.sol1maps` e deverá incluir os resultados de todos os problemas presentes no ficheiro de entrada. Existem duas possibilidades aqui apresentadas:

1. Tarefa #1 e Tarefa #2: ficheiro de saída deverá conter apenas uma linha por problema. Deverá repetir os cinco inteiros do cabeçalho do problema pela mesma ordem, seguidos do resultado obtido. Neste caso, o sexto inteiro especifica a resposta, que será sempre um número não negativo quando o problema esteja bem definido. Considera-se bem definido um problema em que as coordenadas de partida estejam dentro do mapa. Caso não existam células com energia positiva dentro do raio de procura, k, o valor de resposta deverá ser zero. Para problemas mal definidos, o valor de resposta está ausente (não deverá ser escrito qualquer valor no ficheiro).

2. Tarefa #3: o ficheiro de saída também deverá começar por repetir a primeira linha do problema. Após essa primeira linha deverá ser apresentado o caminho com um passo por linha, apresentando as coordenadas de chegada desse passo e a energia recebida. Se o problema estiver mal definido (coordenadas de partida e/ou de chegada fora das dimensões do mapa) nenhumas coordenadas deverão ser impressas, apenas repetir a primeira linha do problema, com todos os sete inteiros.

Note-se que se o ficheiro de entrada possuir mais que um problema, o ficheiro de saída será a concatenação das soluções de todos os problemas. Devem incluir uma linha em branco como separador das diferentes soluções. Também é **obrigatório** que entre cada inteiro exista **apenas** um espaço em branco.

4 Fase final do projecto

Nesta secção explicitam-se os objectivos, especificações e funcionalidades que deverão estar operacionais na fase final de submissões. Nesta fase devem ser implementadas as seguintes funcionalidades para um dado mapa:

1. Tarefa #1: Atingir ou superar um valor de energia dado com exactamente k passos, isto é, o aluno terá de ter energia superior a um dado valor após k passos.

2. Tarefa #2: Atingir a energia máxima possível com exactamente k passos, isto é, o aluno deve terminar com a máxima energia após k passos.

O resultado da execução do programa NAVIGATE consiste em apresentar os passos que constituem o caminho produzido ou a indicação de que o problema não admite solução. Por exemplo, pode dar-se o caso de a energia inicial ser de tal modo baixa que é impossível atingir alguma célula com energia positiva em menos que k passos antes daquela se esgotar completamente. Neste caso não existe solução, porque o aluno "desiste"para qualquer caminho. Alternativamente, pode acontecer que seja impossível atingir a energia definida como limiar mínimo em k passos.

4.1 Formato de entrada

O ficheiro de extensão `.maps` pode conter um ou mais mapas para serem resolvidos. O cabeçalho de cada problema (primeira linha) tem sempre 7 inteiros e é definido da seguinte forma:

1. Dimensões do mapa: linhas L e colunas C com $L > 0$ e $C > 0$.

2. Tarefa/energia: representado por um inteiro positivo para a energia mínima a atingir, tarefa #1, ou -2 para a tarefa #2. Qualquer outro valor não é admissível e se surgir algum mapa com outro valor, tal significa que não há solução.

3. Coordenadas da posição de partida do aluno: As coordenadas do ponto de partida podem estar dentro ou fora da matriz – neste segundo caso o problema não tem solução.

4. Número de passos, k, do percurso: o inteiro que especifica o número de passos, k, tem de ser tal que $k \in \mathbb{N}$ e $0 \leq k < L \times C$, para que o problema admita solução.

5. Energia inicial: a energia inicial tem de ser positiva para que o problema possa admitir solução.

Nas linhas seguintes identifica-se a composição do mapa com $L \times C$ inteiros que representam a energia em cada célula dessa linha. O programa não necessita fazer qualquer verificação se os valores presentes no ficheiro (tanto no cabeçalho como o mapa) são válidos. Apenas necessita de garantir que a extensão está correcta e que o ficheiro passado como argumento existe de facto. Depois, para cada problema deverá ser capaz de responder adequadamente, tanto para problemas bem definidos como para problemas mal definidos ou indefinidos.

4.2 Formato de saída

O ficheiro de saída da fase final tem o mesmo nome que o ficheiro de problemas, mas deverá ter extensão `.solmaps`.

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 oitavo inteiro deverá indicar a energia final, para problemas que possuem solução, ou -1, indicando que o problema não possui solução. Se o oitavo inteiro for positivo, as linhas seguintes deverão conter a solução do problema. Ou seja, deverão existir k linhas de três inteiros cada: coordenadas da posição para que se avança e a energia, e_i, com $i = 1, \ldots, k$, dessa posição[3]. Se o oitavo inteiro for -1 apenas essa primeira linha constitui a resposta adequada para o problema em causa.

5 Execução do programa

O programa tem um executável com o nome **navigate** e deverá ser passado um argumento: o nome de um ficheiro, de extensão **.1maps** para a fase intermédia e **.maps** para a fase final, contendo um ou mais problemas. Os sucessivos problemas estarão separados por, pelo menos, uma linha em branco.

Por exemplo para a fase final, o programa NAVIGATE deverá ser invocado na linha de comandos da seguinte forma:

```
aed$ ./navigate <nome>.maps
```

`navigate` designa o nome do ficheiro executável contendo o programa NAVIGATE;

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

Todas a(s) solução(ões) deve(m) ser colocada(s) num único ficheiro de saída pela mesma ordem em que surgem no ficheiro de entrada, cujo nome deve ser o mesmo do ficheiro de problemas mas **com extensão** `.solmaps`[4]. Este ficheiro deve ser criado e aberto pelo programa. 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.

Se o programa for invocado com ficheiros inexistentes, que não possuam a extensão `.maps`, sem qualquer argumento ou com argumentos a mais, deverá sair silenciosamente. Ou seja, sem escrever qualquer mensagem de erro, nem criar qualquer ficheiro de saída.

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

Todas as execuções do programa deverão sempre retornar o inteiro 0. Qualquer execução que retorne (através da instrução `return` ou da invocação da função `exit`) um valor diferente de 0, será interpretada pelo site de submissões como "Erro de Execução".

[3]Por exemplo, se as coordenadas iniciais forem $l = 3$ e $c = 7$, a primeira linha indicando o primeiro passo do caminho poderá ser uma das seguintes possibilidades: 3 8 e_1; 3 6 e_2; 2 7 e_3; ou 1 7 e_4.

[4]Por exemplo, se o ficheiro com problemas se chama `teste231.maps`, o ficheiro de saída deve-se chamar `teste231.solmaps`.

6 Exemplos de entrada/saída

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

```
17 16 -2 15 5 9 100
-1 -1 0 0 0 0 20 0 0 -2 0 0 -1 -1 -1 -1
-1 -1 0 -3 -3 -3 0 -3 -1 -3 -3 -3 -3 -3 -3 -3
-9 -9 0 -3 -3 -3 0 -3 0 -3 -3 -3 -3 -3 -3 -3
-3 -3 0 0 0 0 0 0 0 0 0 0 0 0 0 0
-3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3
-3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3
-3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3
-3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3
-3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3
-3 -3 -3 -3 -3 -3 -3 -3 100 -3 -3 -3 -3 -3 -3 -3
-3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3
-3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3
-3 -3 -3 -3 -3 -3 -1 -3 -3 -3 -3 -3 -3 -3 -3 -3
-3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3
-3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3
-3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3
-3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3
```

Neste exemplo, fornece-se um mapa de 17 linhas e 16 colunas; pretende-se obter a máxima energia; o ponto de partida é linha 15 e coluna 5, identificado aqui a encarnado para facilitar a visualização; o número de passos é 9; e a energia inicial é 100.

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:

```
17 16 -2 15 5 9 100
-1 -1 0 0 0 0 20 0 0 -2 0 0 -1 -1 -1 -1
-1 -1 0 -3 -3 -3 0 -3 -1 -3 -3 -3 -3 -3 -3 -3
-9 -9 0 -3 -3 -3 0 -3 0 -3 -3 -3 -3 -3 -3 -3
-3 -3 0 0 0 0 0 0 0 0 0 0 0 0 0 0
-3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3
-3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3
-3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3
-3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3
-3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3
-3 -3 -3 -3 -3 -3 -3 -3 100 -3 -3 -3 -3 -3 -3 -3
-3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3
-3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3
-3 -3 -3 -3 -3 -3 -1 -3 -3 -3 -3 -3 -3 -3 -3 -3
-3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3
-3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3
-3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3
-3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3
```

```
 9 16 150 2 8 10 90
-3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3
-3 -3 -3 -3 -3 -3 -3 -3 100 -3 -3 -3 -3 -3 -3 -3
-3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3
-3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3
-3 -3 -3 -3 -3 -3 -1 -3 -3 -3 -3 -3 -3 -3 -3 -3
-3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3
-3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3
-3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3
-3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3 -3
```

Neste exemplo os dois problemas estão separados por uma linha em branco. Haverá sempre, pelo menos, uma linha em branco entre dois problemas sucessivos. Em geral, o número de linhas em branco entre dois problemas não é fixo.

Em relação à solução (fase final), esta poderia ser para um dado mapa:

```
10 8 -2 1 1 5 100 128
1 2 -1
1 3 -1
1 4 27
1 5 -1
1 6 4
```

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/` .