USP - Universidade de São Paulo
Instituto de Ciências Matemáticas e Computação

INTRODUÇÃO À CIÊNCIA DA COMPUTAÇÃO I
SCC0221

MAZE RUNNER

Exercício destinado aos alunos da recuperação da disciplina de 2021/1.
Dificuldades fazem parte do nosso dia a dia, não deixe que elas te amedrontem! Boa sorte! :)

1 Introdução

Labirintos são estruturas constituídas de um vasto conjunto de redes de salas e galerias que se entrecruzam de tal maneira a confundir o percursor e dificultá-lo de encontrar a saída. Muito comum em diversos tipos de jogos e passatempos, muitas vezes a estruturação de um labirinto pode conter uma solução desafiadora.

A computação, no entanto, permite que automatizemos o processo de solução de provavelmente muitos dos tipos de labirintos que podem ser construídos a partir de algoritmos. Sua tarefa, nesse caso, será construir um algoritmo que permita encontrar o final (ou saída, como chamaremos) de um labirinto dado como entrada. A grosso modo, você deve **construir um algoritmo que seja capaz de resolver um labirinto dado de entrada**.

1.1 Notações comuns e estruturação de um arquivo

Nesse exercício, estamos considerando as seguintes notações:

- `'#'` representa uma parede no labirinto.
- `'.'` representa um ponto não percorrido no labirinto.
- `'*'` representa um ponto já percorrido no labirinto.

Além disso, o labirinto será fornecido em um arquivo escrito em texto plano, contendo todas as informações de interesse e seguindo o molde do tipo:o

- Na primeira linha, existem dois números m e n, inteiros e separados por um espaço, que representam as **dimensões do labirinto (linhas e colunas, respectivamente)**. Normalmente, o labirinto será representado por uma matriz quadrada (i.e. possuirá o mesmo número de linhas e colunas).
- Na segunda linha, existem dois números inteiros separados por um espaço, que representam o **ponto inicial da matriz que o labirinto começará a ser percorrido (linha e coluna, respectivamente)**.

- Na terceira linha, existem dois números inteiros separados por um espaço, que representam o **ponto final da matriz em que o labirinto deve parar de ser percorrido (linha e coluna, respectivamente)**. Isto é, esse ponto indica a **saída** do labirinto.
- Por fim, haverão m outras linhas, com n caracteres cada uma (fora a quebra de linha), podendo estes ser `'#'`, `'.'` ou `'*'`. Esse conjunto de linhas representa a formação do labirinto propriamente dito.

Veja a seguir uma representação real de um exemplo de arquivo que será usado nos casos de teste:

Exemplo de arquivo de labirinto a ser fornecido

```
5 5
1 1
3 3
#####
#.###
#..##
##..#
#####
```

1.2 Instruções para a construção do algoritmo

Seu algoritmo deve ser **recursivo**, ou seja, a função deve chamar ela mesma repetidas vezes para prosseguir com o que deve ser feito. Além disso, estabeleceremos um pequeno padrão para o percurso:

- O algoritmo deve sempre priorizar a seguinte ordem de percurso: ESQUERDA → BAIXO → DIREITA → CIMA. Ou seja, o algoritmo **sempre** deve começar tentando ir para esquerda no labirinto; se não for possível, ele deve tentar ir para baixo, e assim sucessivamente.

Considere também que muitos labirintos haverão **bifurcações**; isto é, 'nós' que possuem mais de um caminho disponível. É necessário pensar em como administrá-los da melhor forma (mesmo nas bifurcações, a ordem de percurso dita acima também vale). Após percorrer um ponto no labirinto, você deve marcar esse ponto como "visitado" através do símbolo `'*'` para, assim, ser possível acompanhar o caminho que o algoritmo está fazendo.

Acompanhe abaixo uma breve simulação de como seu algoritmo deve funcionar, em um percurso de labirinto simples, onde o ponto de início é o ponto (3,3) e o ponto de saída é o ponto (1,1):

Exemplo simples de labirinto sendo percorrido

```
Inicio      1          2         3         4       Fim
#####     #####      #####     #####     #####     #####
#.###     #.###      #.###     #.###     #.###     #*###
#..## ->  #..## ->   #..## ->  #.*## ->  #**## ->  #**##
##..#     ##.*#      #***#     #***#     #***#     #***#
#####     #####      #####     #####     #####     #####

```

2 Orientações para confecção do código

- Seu código deve interpretar o labirinto como uma **matriz dinamicamente alocada**, uma vez que seu tamanho depende dos valores dados no arquivo de entrada. Aloque dinamicamente também todos os tipos de dados que julgar necessário (i.e. que podem potencialmente ser grandes, como vetores, matrizes e *structs*).
- É responsabilidade do programador desalocar toda a memória que foi alocada no final da execução do programa. Isso inclui o arquivo aberto com `fopen(...)`.
- Modularização, estruturação e documentação do código também são critérios considerados na avaliação. Utilize de funções para dividir seu código em etapas, inclusive criando a função recursiva para percorrer o labirinto.

3 Entrada

Como entrada, será dado apenas o **nome de um arquivo**, este contendo todas as informações necessárias para a construção do labirinto nos padrões da seção 1.1: dimensões do labirinto, ponto de início, ponto de saída (todos em linhas separadas, respectivamente) e, finalmente, o labirinto.

- Interprete o labirinto na memória RAM como uma **matriz**.
- O nome do arquivo contém apenas caracteres inclusos na tabela ASCII.
- Todo o conteúdo do arquivo está escrito em texto plano (*plain text*), podendo ser lido com `fscanf`.
- Nesse exercício, inexiste a necessidade de realizar tratamento de erros de entrada. A validação das entradas é garantia pelos arquivos fornecidos.

4 Saída

Seu programa deve exibir, na saída padrão, uma cópia do labirinto percorrido a partir das instruções dadas anteriormente (do ponto inicial ao ponto de saída), incluindo quantos passos foram dados até encontrar a saída. Confira a próxima seção para um exemplo concreto da formatação das saídas.

- O labirinto deve ser impresso com o caminho percorrido atualizado.
- Cada linha do labirinto deve ser separada por uma quebra de linha (`'\n'`).
- Certifique-se de formatar a saída de acordo com o desejado.

5 Exemplos de entrada e saída

- Exemplo 1

Entrada

```
caso1.in
```

Saída

```
Caminho final:
#####
#*###
#**##
##**#
#####
Passos percorridos: 5
```

- Exemplo 2

Entrada

```
caso2.in
```

Saída

```
Caminho final:
##########
#**#.##...#
#***.###.#
###*####.#
#***.#.#.#
#*##*##.#.#
#*#*******#
###*###*#*#
#*****#*#*#
##########
Passos percorridos: 26
```

Bom trabalho e boas férias! :)