Estrutura de dados

2019/2 - **Trabalho 2**

Enunciado

O trabalho consiste na implementação, em C, de um programa para encontrar soluções de labirintos usando os algoritmos da Busca em Profundidade (DFS) e da Busca em Largura (BFS).

O labirinto é descrito por uma matriz de N linhas e M colunas. Cada posição da matriz pode ser `.` (ponto), indicando uma posição vazia, ou `#` (*hashtag*), indicando uma parede. Inicialmente você está na primeira linha da primeira coluna (posição (0,0)), e a saída do labirinto está na última linha da última coluna (posição ($N-1$, $M-1$)). Em um passo, você pode andar para a esquerda, para a direita, para cima ou para baixo (desde que a posição de destino esteja vazia).

Entrada

A primeira linha da entrada contém dois inteiros N e M ($2 \leq N, M \leq 100$), indicando o número de linhas e de colunas do labirinto. As próximas N linhas contém M caracteres cada, indicando o labirinto em si. É garantido que tanto a posição (0,0) quanto a posição ($N-1$,$M-1$) estarão vazias.

Saída

Caso o labirinto não tenha solução, imprima apenas `voce esta preso!`. Caso contrário, imprima dois blocos de informações: um para a Busca em Profundidade (DFS) e outro para a Busca em Largura (BFS).

Cada bloco inicia com o labirinto em si, com a indicação do caminho encontrado pela busca (imprima `x` para as posições que fazem parte do caminho). Em seguida, imprima o número de passos necessários para percorrer o caminho encontrado (conforme exemplo). Por fim, imprima o vetor de pai gerado pela busca (em uma linha contendo o pai do vértice 0, o pai do vértice 1, etc. (veja a seção *Implementação* para a numeração dos vértices) Imprima -1 para vértices sem pai).

Verifique os exemplos abaixo:

Exemplo de entrada	Exemplo de saída
3 3 ... .#. ...	x.. x#. xxx 4 passo(s) -1 2 5 0 -1 8 3 6 7 x.. x#. xxx 4 passo(s) -1 0 1 0 -1 2 3 6 7

Exemplo de entrada	Exemplo de saída
4 5 ...## ..... .#.#.	x..## x.xxx x#x#x xxx.x 11 passo(s) -1 2 7 -1 -1 0 1 12 7 8 5 -1 17 -1 9 10 15 16 19 14 x..## x.... x#.#. xxxxx 7 passo(s) -1 0 1 -1 -1 0 5 6 7 8 5 -1 7 -1 9 10 15 16 17 18

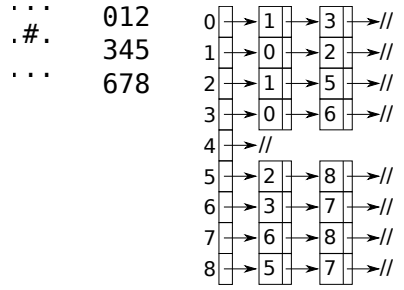
Exemplo de entrada	Exemplo de saída
3 3 ... .## #..	voce esta preso!

Note que o caminho encontrado pela busca depende da ordem em que os vértices são visitados. Logo, seu programa não precisa imprimir exatamente os caminhos dados para os exemplos acima: é suficiente que seu programa imprima algum caminho gerado pela DFS e algum caminho gerado pela BFS (note, entretanto, que a BFS deve gerar um caminho de tamanho mínimo).

Implementação

Após a leitura, construa um grafo equivalente ao labirinto lido. Para tal, crie um grafo de $N \times M$ vértices numerados de 0 a $N \times M - 1$. O vértice 0 representa a posição $(0, 0)$ da matriz; o vértice 1 representa a posição $(0, 1)$ da matriz; e assim por diante. Note que o vértice 0 representa a entrada do labirinto, enquanto o vértice $N \times M - 1$ representa sua saída.

O grafo deve necessariamente ser representado por sua *lista de adjacência*. Como exemplo, a figura abaixo apresenta um labirinto com $N = 3$ e $M = 3$, como suas posições devem ser enumeradas, e a lista de adjacência de seu grafo equivalente. Em cada lista em si, os vértices podem ocorrer em qualquer ordem.



Uma vez construída a lista de adjacência, execute uma DFS e uma BFS e imprima os resultados obtidos conforme especificado. Você é livre para escolher entre implementar a DFS recursiva ou iterativa (e, se iterativa, com a pilha estática ou dinâmica). Você também é livre para, na BFS, escolher entre usar uma fila estática ou dinâmica.

Por fim, certifique-se que toda memória alocada por seu programa é desalocada ao final do mesmo.

Orientações

- O trabalho pode ser feito por equipes de *até* 2 (dois) estudantes;
- Submeta, via *Moodle*, um pacote (zip ou tar.gz) contendo todos os arquivos necessários para compilar e executar seu programa, além de um arquivo de texto (txt) onde conste:
  - O nome de todos os integrantes da equipe;
  - Toda informação que a equipe julgar relevante para a correção (como *bugs* conhecidos, detalhes de implementação, escolhas de projeto, etc.)
- Comente adequadamente seus códigos para facilitar a correção.
- Atenção: a correção será parcialmente automatizada, e a saída do programa será testada com outras entradas além das fornecidas como exemplo. *Siga* ***fielmente*** *o formato de saída dado nos exemplos*, sob pena de grande redução da nota;
- Certifique-se que seu programa funciona antes de submetê-lo;
- O trabalho deve ser entregue até **7 de Dezembro de 2019 (sábado), 23:59**, via *Moodle*. É suficiente que o trabalho seja submetido por apenas um estudante da equipe;
- Trabalhos copiados ou plagiados receberão **todos** a nota 0 (**ZERO**).