IF61C—Fundamentos de Programação 1

Atividade Prática Supervisionada - Labirinto

1. Ajude um rato a encontrar um pedaço de queijo num labirinto como o do desenho abaixo (`R` denota a posição do rato e `Q` a do queijo):

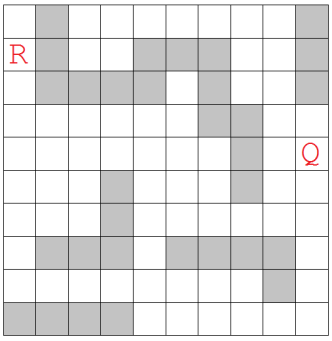

Figura 1: Labirinto

Uma possível representação computacional se dá por uma matriz retangular `L`, cujo elemento L_{ij} vale `0`, se corresponder a uma passagem livre, ou `-1`, caso contrário.

Neste contexto, diferentes abordagens poderiam ser exploradas para fazer com que o rato encontre o caminho até o queijo. Contudo, como discutido em sala, algumas só são eficientes em cenários específicos. Outras demandam o teste de muitas condições em cada passo, deixando a implementação pouco eficiente.

Uma abordagem para resolver esse problema consiste em marcar com o número `k`, `k= 1,2,...`, todas as casas livres que estejam a exatamente `k` passos de distância do queijo (pelo caminho mais curto possível). Suponha que, a cada passo, o rato possa se deslocar apenas uma casa na vertical ou na horizontal. Então, rotula-se inicialmente a posição do queijo com `k=0` e, para cada `k` subsequente, examinam-se todas as casas livres do labirinto, marcando-se com `k` aquelas ainda não marcadas E que sejam adjacentes a alguma casa marcada com `k-1`. Para o exemplo acima, teríamos configuração ilustrada na Figura 2(a). Neste contexto, o caminho delineado na Figura 2(b) poderia ser escolhido.

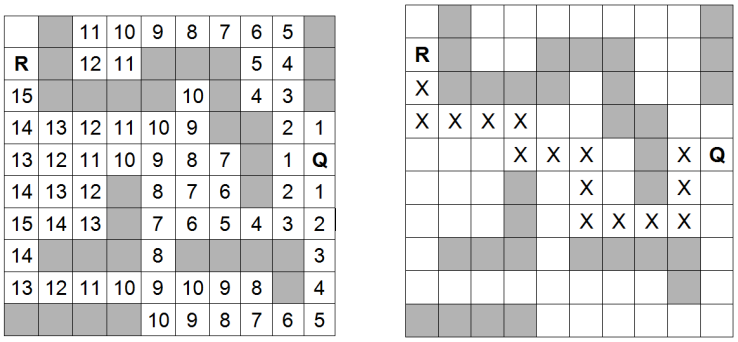

Figura 2: Exemplos de configuração e caminho.

TAREFAS

Escreva um programa com as seguintes funções/procedimentos.

(a) `int ** alocaMatriz(int tam)`: função que aloca dinamicamente uma matriz de tamanho `tam` × `tam`;

(b) `void inicializaLabirinto(int **m, int tam)`: inicializa a matriz `m`, de dimensão `tam`×`tam`, com `0` nas posições livres e `-1` nas demais (correspondentes às paredes). OBS.: você escolhe o tipo da inicialização: ler do teclado os índices correspondentes à paredes, iniciar com uma configuração padrão, etc (este último caso é mais adequado para a fase de testes).

(c) `void criaMatrizCusto(int **m, int tam`: cria a matriz de custo, segundo o processo descrito acima.

(d) `void calculaCaminho (int **m, int tam, int *indicesLinha, int *indicesColuna, int *nroPassos)`: este procedimento recebe a matriz de custo já preenchida (denotada por `m` e de tamanho `tam` × `tam`) e preenche os vetores `indicesLinha` e `indicesColuna` com os índices de linha e coluna do caminho a ser percorrido entre o rato e o queijo. Além disso, retorna na variável `nroPassos` (passada por referência) o número de passos do caminho.

(e) `void criaCaminho(int *indicesLinha, int *indicesColuna, int nroPassos, int tamLabirinto)`: dados os índices de linha e coluna do caminho a ser percorrido entre o rato e o queijo (os vetores terão `nroPassos` posições), este procedimento cria e preenche uma matriz de caracteres de dimensão `tamLabirinto`×`tamLabirinto` de tal forma a possibilitar a visualização do caminho percorrido na tela. Para tal impressão, use o procedimento definido a seguir.

(f) `void mostraCaminho(char **caminho, int tam)`: imprime a matriz de caracteres na tela.

(g) `main()`: na função principal, você deverá ler do teclado a posição dos alvos (ie., do rato e do queijo) e o tamanho do labirinto (`tam` - note que esse valor irá determinar o tamanho das matrizes utilizadas, as quais deverão ser alocadas dinamicamente). Depois, usando as funções/procedimentos acima, deverá calcular e imprimir na tela o menor caminho percorrido entre o rato e o queijo.