Algoritmos e Estruturas de Dados 1

Trabalho 3

Entrega

09/10

Solução de um labirinto

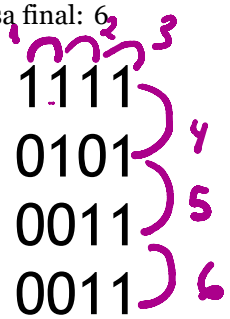
Nesse trabalho iremos encontrar um caminho para sair de um labirinto.

O labirinto será um tabuleiro quadriculado com m x n casas distribuídas em m linhas e n colunas. As casas livres (caminhos livres) são marcadas por `1` e as casas ocupadas (paredes do labirinto) são marcadas por `0`.

Na casa (0, 0), que sempre é livre, existe um robô que deve encontrar a saída do labirinto, que é a posição (m−1, n −1), e pode se mover nas seguintes direções: **direita e para baixo**, uma casa por vez, desde que a casa a ser alcançada seja livre. Você deve descobrir se existe um caminho para a saída do labirinto e, se o caminho existir, imprimir a distância entre a casa (1, 1) e a casa ($m - 1, n - 1$).

Exemplo:

Labirinto (4x4):
Distância da casa inicial à casa final: 6



Ao descobrir uma solução você deve armazenar os nós do caminho em uma fila e imprimir o mesmo. Ao imprimir ,a fila deve ser esvaziada.

O programa deve funcionar para matrizes de tamanho 4x4. Por exemplo, no programa main ao chamar

int labirinto3 [N][N] = { { 1, 1, 1, 0 },

resolveLabirinto(labirinto3);

A saida deve ser a impressão da matriz com 1 no caminho seguido. Em seguida é impressa a fila com as posições que foram percorridas.

```
1 1 0 0
0 1 0 0
0 1 0 0
0 1 1 1
(0,0)->(0,1)->(1,1)->(2,1)->(3,1)->(3,2)->(3,3)->
```

Para fazer esse programa vocês devem usar programação por retrocesso (backtracking). Esse estilo de programação consiste em passos incrementais, que quando não é possível prosseguir retornam aos passos anteriores e tentam novos caminhos.

Nosso algoritmo terá o tamanho do labirinto fixo em N = 4. O código do programa main já é fornecido também, assim como a função de impressão da solução.

Os passos para achar a solução serão sempre para frente ao procurar possíveis soluções. Um pseudocodigo da função recursiva (chamada pela resolveLabirinto) para resolver esse problema é apresentada abaixo.

//pseudocódigo

```
int procuraLab(int ** lab, int x, int y, int **sol){

se x e y estiverem em n-1, retorne 1, atingiu o fim da matriz

se x e y estão livres e são uma posição valida (nao ultrapassou os
limites da matriz)
        sol[x][y] = 1;

        /* movimente-se para frente na direção do x*/
        if  procuraLab (lab,x+1,y,sol)
            retorne 1;

        /* se movimentar para frente o x não deu certo, tente
movimentar o y*/
        if  procuraLab (lab,x,y+1,sol)
            retorne 1;

        /* Se nenhuma das soluções deu certo, a posição x,y não
```

```
faz parte da solução então retroceda e desmarque x,y como sendo
parte da solução, impeça ele de ser testado de novo*/

      sol[x][y] = 0;
        return 0;
    }
```

(1) (1) (1) (0)
A distancia da casa inicial a final: 6 Caminho percorrido: (0, 0) -> (0, 1) -> (0, 2) -> (0, 3) -> (1, 3) -> (2, 3) -> (3, 3)