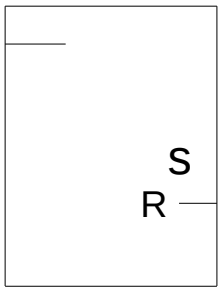
Universidade de Caxias do Sul
Centro de Ciências Exatas e das Tecnologias

Estruturas de Dados
Prof. André Zampieri

Trabalho de Implementação – Parte 2

Entrega: 25/10/2017
Realização: Individual ou dupla

Um labirinto pode ser implementado como uma matriz de números, na qual as passagens são marcadas com 0s, as paredes com 1s, a posição de saída do labirinto com 9 e a posição inicial do rato pelo número 5. A figura 1 mostra um exemplo de labirinto e como o mesmo é representado na matriz:




```
1 1 1 1 1 1
1 1 1 0 0 1
1 0 0 0 9 1
1 0 0 5 1 1
1 1 1 1 1 1
```

Figura 1 – Labirinto e Matriz

O rato tenta encontrar o caminho para a saída do labirinto. Para escapar ele tenta, sistematicamente, todas as rotas. Se atinge um corredor sem saída refaz as suas etapas até a última posição e começa pelo menos mais um caminho não tentado. Para cada posição, o rato pode ir em uma dentre quatro posições: para cima, para baixo, para esquerda e para a direita. Independentemente de quão perto está da saída, ele sempre tenta os caminhos abertos nessa ordem, o que pode levar a algumas voltas desnecessárias. Retendo a informação que permite retomar a procura depois que um corredor sem saída é atingido, o rato usa um método chamado de retrocesso.

O labirinto fornecido pelo usuário pode ter qualquer número de linhas e qualquer número de colunas. As únicas hipóteses que o programa considera são que todas as linhas são do mesmo comprimento e que ele usa somente estes números: qualquer número de 1s, qualquer número de 0s, um 5 (posição inicial do rato) e um 9 (posição de saída). O labirinto é circundado por 1s.

Uma **pilha** é utilizada no processo de escapar do labirinto. Para lembrar os caminhos não tentados para as próximas tentativas, as posições das vizinhanças não tentadas da posição corrente (se alguma) são armazenadas na pilha e sempre na mesma ordem, primeiro a vizinhança superior, depois a inferior, na sequência a esquerda e depois a direita. Depois de colocar na pilha as vizinhanças abertas, o rato tenta seguir a posição do topo da pilha. Primeiro armazena as vizinhanças não tentadas e tenta a posição do topo. Esse processo é repetido até que ele atinge a saída ou esgota todas as possibilidades e se encontra preso no labirinto. Para evitar de cair dentro de um laço infinito de caminhos tentados que já tenham sido investigados, cada posição visitada do labirinto é marcada com um 4.

Eis o algoritmo para sair do labirinto:

```
sairLabirinto()
   inicializa a pilha
   posicao_saida = coordenadas_saida
   posicao_entrada = coordenadas_entrada
   posicao_atual = coordenadas_entrada
   enquanto a posicao_atual não é posicao_saida faca
      marca posicao_atual como visitada
      empilha as vizinhancas não visitadas de posicao_atual
      se a pilha esta vazia
         falha
      senao
         retira uma posicao da pilha
         posicao_atual = posicao_pilha
      fimse
   fimenquanto
```

```
   fim_sairLabirinto
```

Exemplo

Considere a matriz abaixo, que representa o algoritmo da figura 1.

```
1 1 1 1 1 1
1 1 1 0 0 1
1 0 0 0 9 1
1 0 0 5 1 1
1 1 1 1 1 1
```

posicao_atual = (3, 3)
posicao_entrada = (3, 3)
posicao_saida = (2, 4)

Como a posicao_atual não é igual a posicao_saida, a posição atual é marcada como visitada (número 4) e, em seguida, todas as quatro posições da célula atual são testadas – testar a posição superior, a inferior, a esquerda e a direita, nessa ordem. Foram testadas as posições (2, 3) com conteúdo 0, posição (4, 3) com conteúdo 1, posição (3, 2) com conteúdo 0 e posição (3, 4) com conteúdo 1. Somente as posições (2, 3) e (3, 2) são candidatas ao processamento e colocadas na pilha – veja a figura abaixo.

```
(3, 2)
(2, 3)
```

Como a pilha não está vazia, a posição do topo se torna a posição atual. Veja abaixo como os dados estão agora.

```
1 1 1 1 1 1
1 1 1 0 0 1
1 0 0 0 9 1
1 0 0 4 1 1
1 1 1 1 1 1
```

```
(2 , 3)
```

```
posicao_atual = (3, 2)
posicao_entrada = (3, 3)
posicao_saida = (2, 4)
```

A posicao_atual ainda não é igual a posicao_saida; a posicao_atual é marcada como visitada e, em seguida, as duas opções viáveis a partir de (3 , 2) são colocadas na pilha: as posições (2, 2) e (3, 1).

```
1 1 1 1 1 1
1 1 1 0 0 1
1 0 0 0 9 1
1 0 0 4 1 1
1 1 1 1 1 1
```

```
(3 , 1)
(2 , 2)
(2 , 3)
```

```
posicao_atual = (3, 2)
posicao_entrada = (3, 3)
posicao_saida = (2, 4)
```

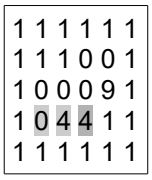
Como a pilha não está vazia, a posicao_atual se torna o valor do topo da pilha. Veja abaixo como os dados se encontram:

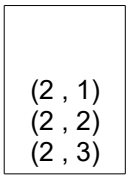
```
1 1 1 1 1 1
1 1 1 0 0 1
1 0 0 0 9 1
1 0 4 4 1 1
1 1 1 1 1 1
```

```
(2 , 2)
(2 , 3)
```

```
posicao_atual = (3, 1)
posicao_entrada = (3, 3)
posicao_saida = (2, 4)
```

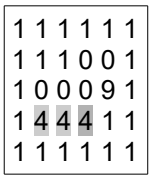
A posicao_atual ainda não é igual a posicao_saida; a posicao_atual é marcada como visitada e, em seguida, a única opção viável a partir de (3 , 1) é colocada na pilha: a posição (2, 1).

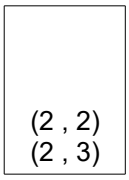





```
posicao_atual = (3, 1)
posicao_entrada = (3, 3)
posicao_saida = (2, 4)
```

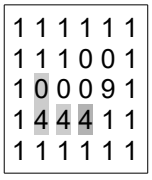
Como a pilha não está vazia, a posicao_atual se torna o valor do topo da pilha – (2, 1). Veja abaixo como os dados se encontram:

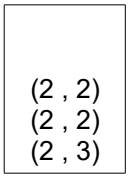





```
posicao_atual = (2, 1)
posicao_entrada = (3, 3)
posicao_saida = (2, 4)
```

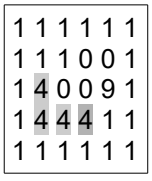
A posicao_atual ainda não é igual a posicao_saida; a posicao_atual é marcada como visitada e, em seguida, a única opção viável a partir de (2 , 1) é colocada na pilha: a posição (2, 2).

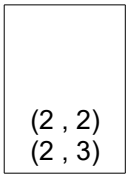





```
posicao_atual = (2, 1)
posicao_entrada = (3, 3)
posicao_saida = (2, 4)
```

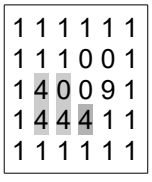
Como a pilha não está vazia, a posicao_atual se torna o valor do topo da pilha – (2, 2). Veja abaixo como os dados se encontram:

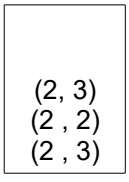





```
posicao_atual = (2, 2)
posicao_entrada = (3, 3)
posicao_saida = (2, 4)
```

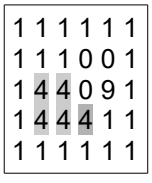
A posicao_atual ainda não é igual a posicao_saida; a posicao_atual é marcada como visitada e, em seguida, a única opção viável a partir de (2 , 2) é colocada na pilha: a posição (2, 3).

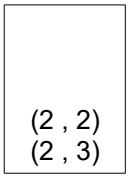





```
posicao_atual = (2, 2)
posicao_entrada = (3, 3)
posicao_saida = (2, 4)
```

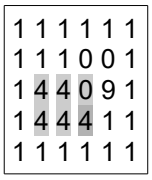
Como a pilha não está vazia, a posicao_atual se torna o valor do topo da pilha – (2, 3). Veja abaixo como os dados se encontram:

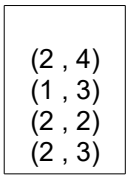





```
posicao_atual = (2, 3)
posicao_entrada = (3, 3)
posicao_saida = (2, 4)
```

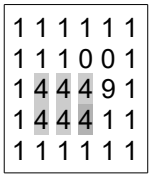
A posicao_atual ainda não é igual a posicao_saida; a posicao_atual é marcada como visitada e, em seguida, as opções viáveis a partir de (2 , 3) são colocadas na pilha: a posição (1, 3) e a posição (2, 4).

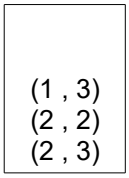





```
posicao_atual = (2, 3)
posicao_entrada = (3, 3)
posicao_saida = (2, 4)
```

Como a pilha não está vazia, a posicao_atual se torna o valor do topo da pilha – (2, 4). Veja abaixo como os dados se encontram:

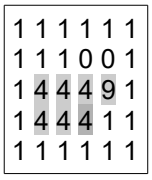


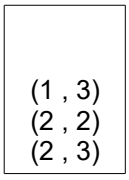



```
posicao_atual = (2, 4)
posicao_entrada = (3, 3)
posicao_saida = (2, 4)
```

Como a pilha não está vazia, a posicao_atual se torna o valor do topo da pilha – (2, 4).

Neste momento, o valor da posicao_atual é o mesmo da posicao_saida. O processo de busca é interrompido. Veja abaixo como os dados se encontram:






```
posicao_atual = (2, 4)
posicao_entrada = (3, 3)
posicao_saida = (2, 4)
```