Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação
Departamento de Ciências de Computação
SCC0202 – Algoritmos e Estrutura de Dados I - 2 o Sem /2023

Exercício: DFS com pilha

Professores: Dr. Marcelo Manzato (mmanzato@icmc.usp.br)
Dr. Rudinei Goularte (rudinei@icmc.usp.br)
PAE: Wan Song (wansong@usp.br)
Monitores: Gabriel Almeida, João Victor Sene, Otto Fernandes

1 Introdução

De modo geral, todas as estruturas de dados vistas ao longo da disciplina apresentam diversos usos em softwares difundidos, tanto na indústria quanto na academia.

Nesse exercício, vamos trabalhar com uma aplicação de pilha mais complexa e mais próxima da solução de problemas reais: a navegação em um grafo.

Apesar de o uso de grafos extrapolar o conteúdo da disciplina de Algoritmos, a aplicação nesse exercício se dará de forma simplificada. Assim, grafos não deve ser um problema para entender o exercício e desenvolver uma solução.

2 Descrição do problema

O problema que iremos resolver é o problema de **encontrar um caminho para o fim de um labirinto**. O labirinto em questão pode assumir várias formas: um tabuleiro de duas dimensões com posições discretas; um terreno contínuo; ou até mesmo um espaço de mais de três dimensões. O formato do labirinto pode variar de problema para problema, mas, no nosso caso, ele será como um tabuleiro de xadrez, de apenas duas dimensões.

Também podemos usar diferentes algoritmos para encontrar a saída do labirinto, mas, nesse caso, iremos utilizar o *Depth First Search*, ou **DFS**.

A DFS é um algoritmo que, entre os caminhos possíveis, escolhe um e continua nele até encontrar o fim; ou então esgotar os caminhos possíveis. Ao chegar num "beco sem saída", o algoritmo volta para o último ponto de escolha de caminhos, e escolhe o caminho seguinte. Ele continua seguindo esse processo até encontrar o final do labirinto ou não haver mais caminhos disponíveis.

O algoritmo de DFS recebe esse nome pois ele percorre o grafo, ou labirinto, navegando em profundidade - depth -, "ignorando"momentaneamente os outros caminhos. Ele se comporta semelhante a uma pessoa andando pelo labirinto.

Observe as imagens abaixo.

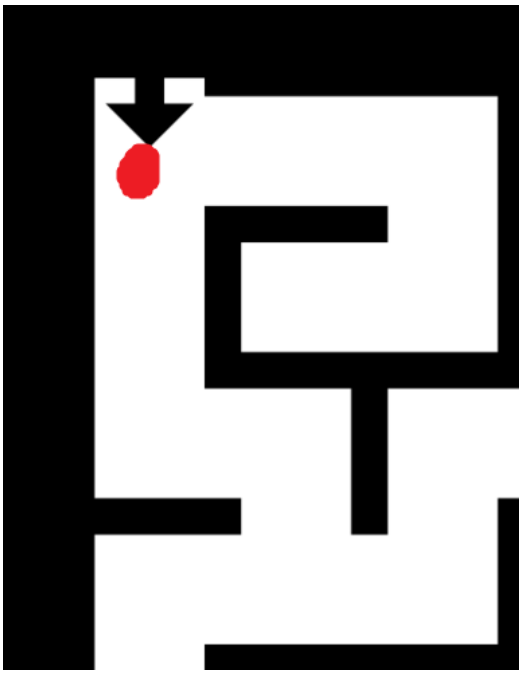

Figura 1: O algoritmo começa na posição inicial do labirinto

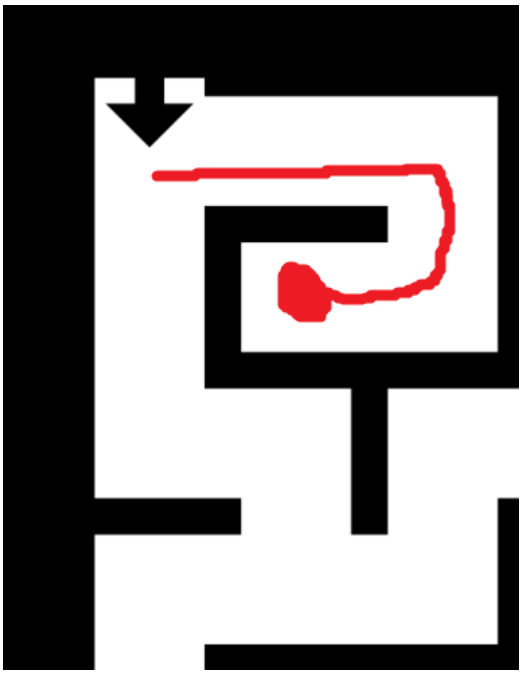

Figura 2: Após escolher um caminho, o lado direito, ele vai até o final, onde não houver mais caminhos para percorrer.

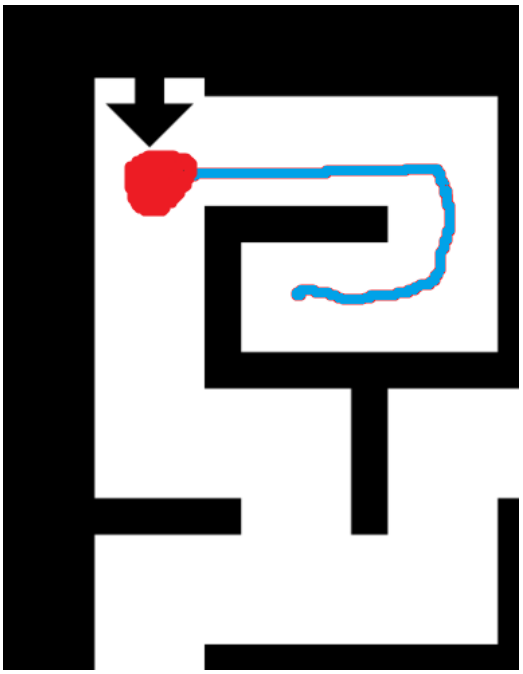

Figura 3: Ao chegar numa posição onde não há mais caminhos, o algoritmo volta até e busca o próximo caminho possível. Repare que o caminho já percorrido fica marcado.

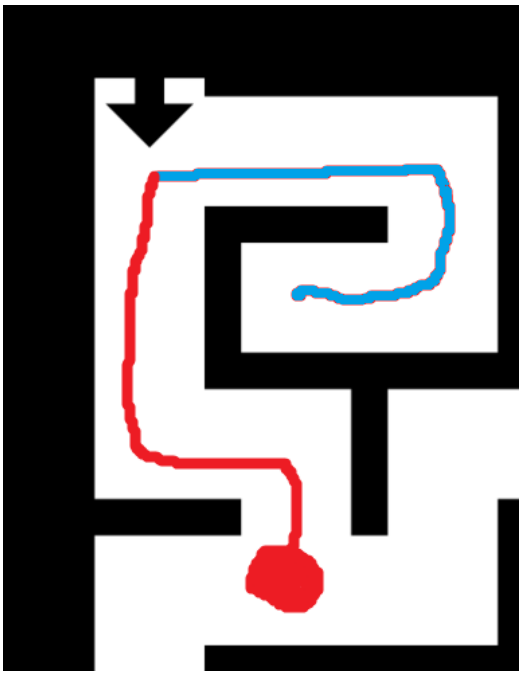

Figura 4: O algoritmo continua procurando o final do labirinto pelos caminhos possíveis.

Dado esse problema, você deve implementar um programa para encontrar o caminho em um labirinto utilizando uma DFS.

3 Como o algoritmo funciona (com pilhas)

A implementação do algoritmo utilizando pilhas é bem simples.

1. Iniciamos nossa execução no início do labirinto e verificamos todas as células que podemos nos movimentar (células adjacentes).

2. Para cada uma delas, colocamos a célula dentro de uma pilha.

3. Para descobrir a próxima célula a ser visitada, basta desempilhar uma célula da pilha e usá-la.

Esse algoritmo apresenta algumas características interessantes:

- Como o comportamento da pilha é LIFO - *last in, first out* - o último a entrar é o primeiro a sair -, então a última célula empilhada será o caminho que o algoritmo irá seguir.

- Quando a posição atual só tiver uma célula adjacente, ela será a única a ser empilhada e, portanto, a única a ser desempilhada. Então o algoritmo continua seguindo o caminho, como na Figura 2.

- Quando o algoritmo encontrar um "beco sem saída", ele não empilha nenhuma célula. Portanto, ele volta para a última célula que ele empilhou. No entanto, a última célula empilhada equivale a um caminho que o algoritmo não escolheu. Isso é equivalente a voltar pelo caminho que já percorremos, até encontrar um outro caminho possível.

Graças a isso, o algoritmo funciona. Veja com as imagens abaixo.

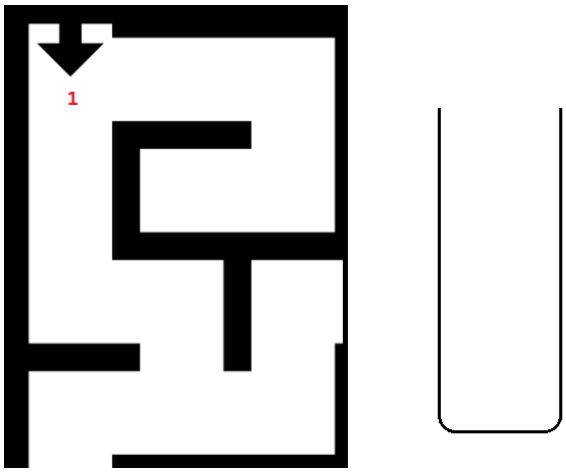

Figura 5: O algoritmo começa na posição inicial do labirinto, a célula 1. Repare que a pilha começa vazia.

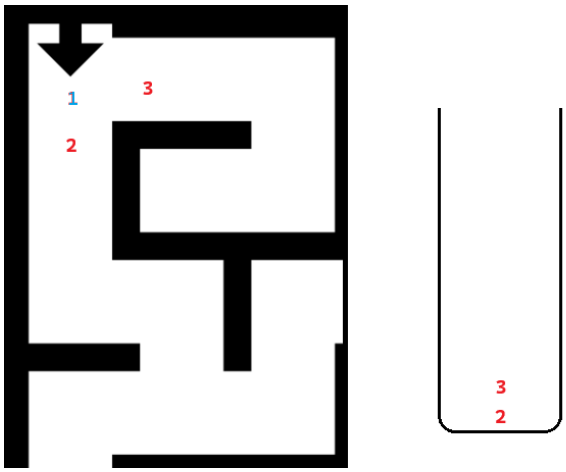

Figura 6: Após marcar a primeira célula como visitada, o algoritmo coloca as duas células adjacentes na pilha, a de baixo e a da direita.

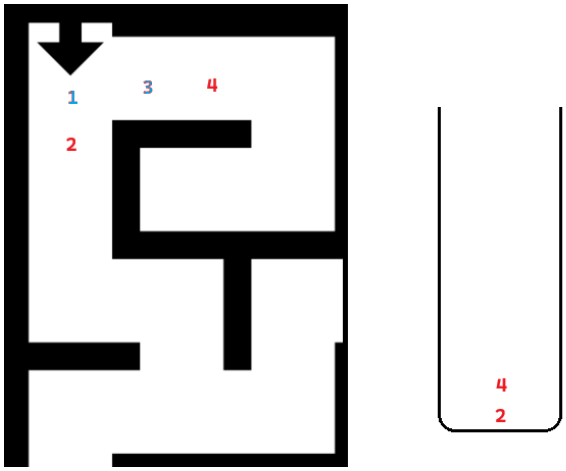

Figura 7: Como a célula 3 foi empilhada por último, será esse o caminho seguido pelo algoritmo. A célula 3 é marcada como visitada.

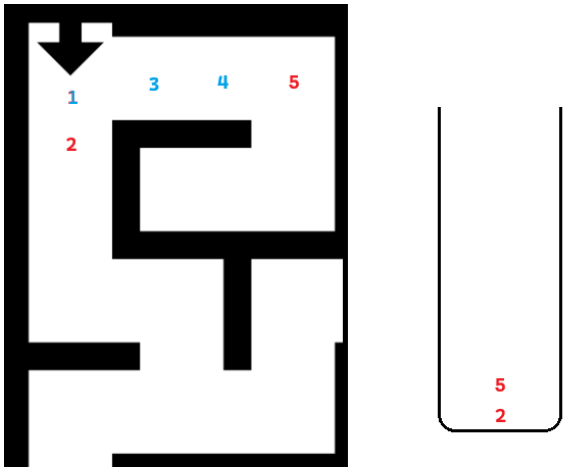

Figura 8: O processo continua com a célula 4, adicionando a célula 5 na pilha.

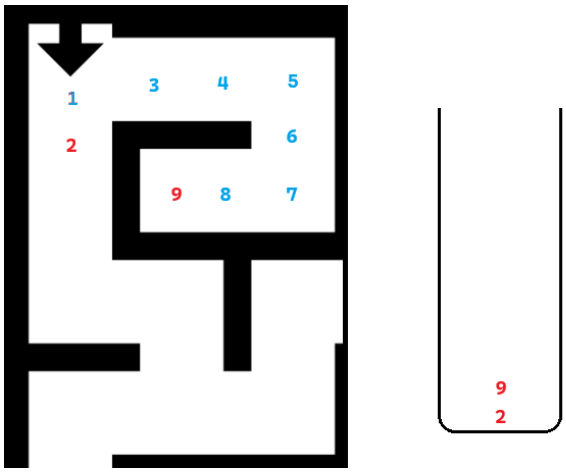

Figura 9: O algoritmo continua até não ter novas posições para empilhar.

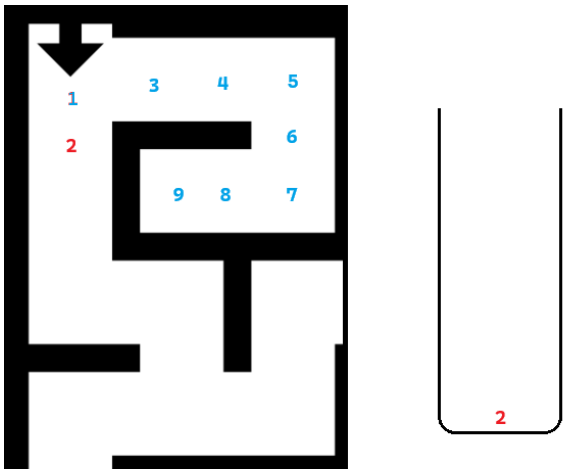

Figura 10: Quando chega a célula 9, ela é desempilhada e a próxima célula a ser visitada é a 2, que estava na pilha o tempo todo.

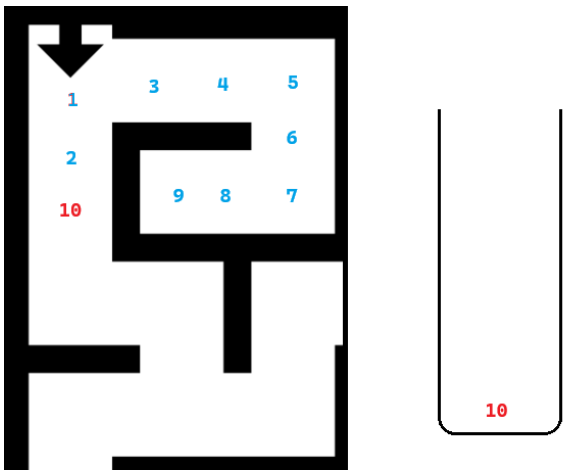

Figura 11: O processo continua com a célula 10.

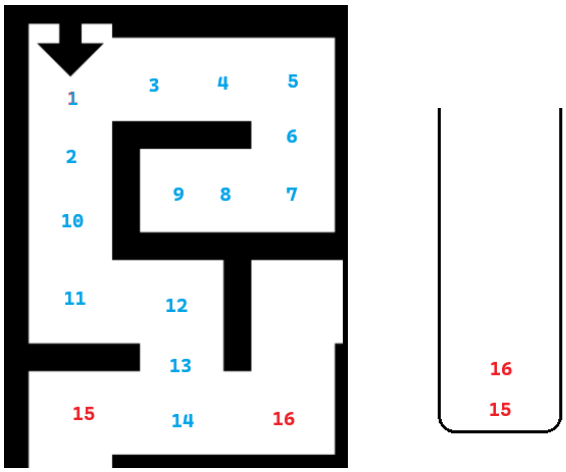

Figura 12: Quando uma célula com mais de um vizinho é visitada, ambas são empilhadas. Repare que a última célula a ser empilhada é a o caminho que será seguido.

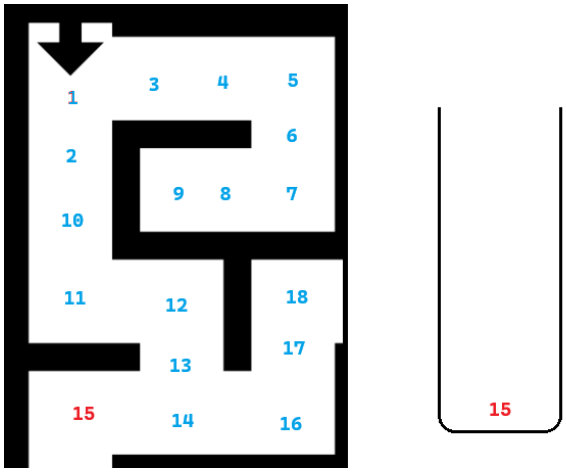

Figura 13: O caminho da direita é esgotado, então o algoritmo seguirá pela esquerda.

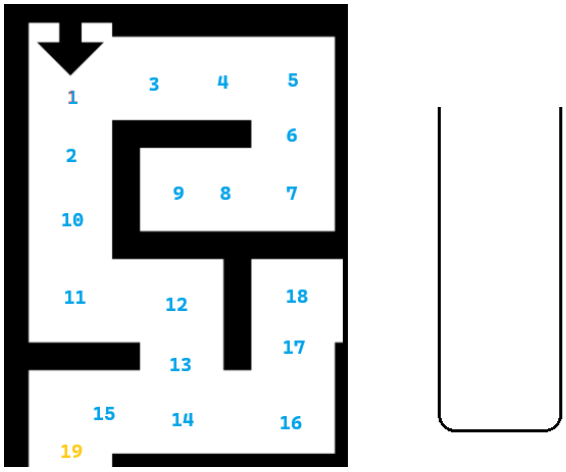

Figura 14: Ao seguir pela esquerda, o fim do labirinto é encontrado, e a execução acaba.

Repare duas coisas nessa execução:

- O percuso pelo labirinto depende da ordem em que empilhamos as células;

- Nesse exemplo, empilhamos as células como números para facilitar a representação. Se estivermos usando uma matriz, podemos empilhar as coordenadas da célula, isto é, os índices dela. Por exemplo, empilhamos (0, 0), depois (1, 0), depois (0, 1), (0, 2) e assim por diante.

4 Instruções Complementares

- A primeira linha da entrada contém 2 inteiros: n, m, o número de linhas e colunas do labirinto.
- As n linhas seguintes contém m inteiros separados por espaços. Esse é o labirinto. Valores 1 representam caminhos possíveis. Valores 0 representam paredes, e não são acessíveis. O valor 2 é a saída do labirinto.
- Todos os labirintos começam na posição (0, 0), isto é, na primeira posição da matriz. É garantindo que essa célula sempre será acessível.
- A saída do seu programa deverá ser o caminho percorrido pela DFS. Cada célula visitada deverá ser impressa, no formato (i, j), uma célula por linha.
- Caso o programa não encontre uma saída para o labirinto, deverá ser impresso "Saída não encontrada.".
- Seu código **deve** utilizar a estrutura de Pilha.
- A ordem de empilhagem das células adjacentes do seu código deve ser: baixo, esquerda, cima e direita.
- Faça o upload de um arquivo compactado .zip com um Makefile. O seu programa deverá funcionar utilizando o *target* "make run", e compilado com o *target* "make all". Submeta o .zip no http://runcodes.icmc.usp.br.

5 Exemplos de Entrada e Saída

A seguir são apresentados exemplos de entrada e saída para que você teste seu código enquanto desenvolve o exercício.

Entrada

```
3 3
1 1 1
1 0 1
1 1 2
```

Saída

```
(0, 0)
(0, 1)
(0, 2)
(1, 2)
(2, 2)
```

Entrada

```
3 3
1 1 1
1 0 0
1 1 2
```

Saída

```
(0, 0)
(0, 1)
(0, 2)
(1, 0)
(2, 0)
(2, 1)
(2, 2)
```

Entrada

```
1 1
2
```

Saída

```
(0, 0)
```