INSTITUTO FEDERAL
Rio Grande do Norte
Campus Pau dos Ferros

Aplicação de Pilha: *Trapped Mouse*

Estrutura de Dados

Prof. Demétrios Coutinho

23 de maio de 2016

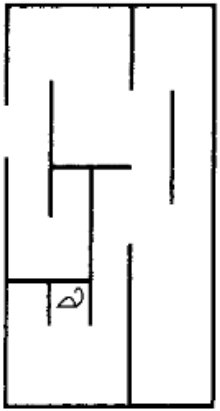
Trapped Mouse
Descrição do Problema

- Considere o problema de um rato preso em um labirinto e tenta encontrar a saída.

Trapped Mouse
Modelagem

- O labirinto é implementado como um array de caracteres bidimensional.
  - 0(zeros) : São os corredores.
  - 1 (ums) : São as paredes.
  - e : Representa a posição da **saída**.
  - m : Representa a posição do **rato**.

Trapped Mouse
Modelagem



```
11111111111
10000010001
10100010101
e0100000101
10111110101
10101000101
10001010001
11111010001
101m1010001
10000010001
11111111111
```

Trapped Mouse
Como fazer

- O programa usa duas pilhas:
  - Uma para inicializar o labirinto.
  - Outra para Implementar o **BackTracking**.

Trapped Mouse

Como fazer

- Inicializar o labirinto.
  - O programa deve permitir qualquer quantidade de linhas e colunas.

Trapped Mouse

Como fazer

- Inicializar o labirinto.
  - O programa deve permitir qualquer quantidade de linhas e colunas.
  - O programa deve ler as linhas e “deduzir” as dimensões do labirinto.

Trapped Mouse
Como fazer

- Inicializar o labirinto.
  - O programa deve permitir qualquer quantidade de linhas e colunas.
  - O programa deve ler as linhas e "deduzir" as dimensões do labirinto.
  - A entrada do programa pode ser feita via arquivos e direto do teclado.

Trapped Mouse

Como fazer

- Inicializar o labirinto.
  - O programa deve permitir qualquer quantidade de linhas e colunas.
  - O programa deve ler as linhas e "deduzir" as dimensões do labirinto.
  - A entrada do programa pode ser feita via arquivos e direto do teclado.
  - Somente serão aceitos os caracteres: `1`,`0`,`e`,`m`.

Trapped Mouse

Como fazer

- Inicializar o labirinto.
  - O programa deve permitir qualquer quantidade de linhas e colunas.
  - O programa deve ler as linhas e "deduzir" as dimensões do labirinto.
  - A entrada do programa pode ser feita via arquivos e direto do teclado.
  - Somente serão aceitos os caracteres: `1`,`0`,`e`,`m`.
  - A leitura de arquivo deve permitir a vários labirintos.

Trapped Mouse
Como fazer

- Inicializar o labirinto.
  - O programa deve permitir qualquer quantidade de linhas e colunas.
  - O programa deve ler as linhas e "deduzir" as dimensões do labirinto.
  - A entrada do programa pode ser feita via arquivos e direto do teclado.
  - Somente serão aceitos os caracteres: `1`,`0`,`e`,`m`.
  - A leitura de arquivo deve permitir a vários labirintos.
  - As paredes externas serão inseridas após leitura do labirinto.

Trapped Mouse
Como fazer

- Inicializar o labirinto.

```
1100
000e
00m1
```

Figura: Exemplo de um labirinto inserido pelo usuário.

Algoritmo 1: Inicializar Labirinto

```
1 Stack mazeRows;
2 Arrays de String maze;
3 enquanto Houver linhas para serem lidas faça
4     Adicionar paredes nas extremidades;
5     Fazer um push na pilha;
6 fim
7 Adicionar em maze a primeira linha de paredes;
8 Realizar pops na pilha adicionando em maze;
9 Adicionar em maze a última linha de paredes;
```

Trapped Mouse
Como fazer

- Implementar **BackTracking**:

BackTracking

- É um refinamento do algoritmo de busca por força bruta (ou enumeração exaustiva), no qual boa parte das soluções podem ser eliminadas sem serem explicitamente examinadas.

Trapped Mouse
Como fazer

- Implementar **BackTracking**:

BackTracking

- É um refinamento do algoritmo de busca por força bruta (ou enumeração exaustiva), no qual boa parte das soluções podem ser eliminadas sem serem explicitamente examinadas.
- O algoritmo constroe uma árvore de ações, retornando para uma ação anterior, caso chegue em um caminho sem solução.

Trapped Mouse

Como fazer

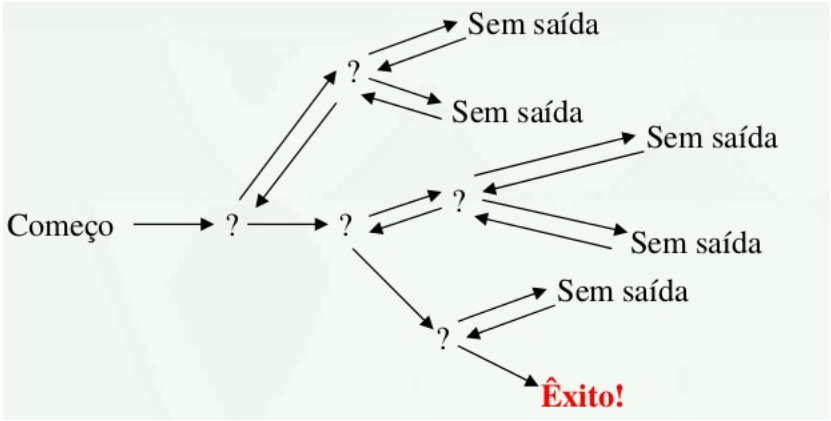


Figura : Exemplo do *BackTracking*

Trapped Mouse

Como fazer

- O rato esperar escapar do labirinto tentando, sistematicamente, todas as rotas.

Trapped Mouse

Como fazer

- O rato esperar escapar do labirinto tentando, sistematicamente, todas as rotas.
- Se chegar em um beco sem saída, ele retornar alguns passos até a última posição que tenha um ou mais caminhos não testados.

Trapped Mouse
Como fazer

- O rato esperar escapar do labirinto tentando, sistematicamente, todas as rotas.
- Se chegar em um beco sem saída, ele retornar alguns passos até a última posição que tenha um ou mais caminhos não testados.
- Mesmo próximo a saída, ele sempre testa os caminhos não percorridos na mesma ordem: **direita**, **esquerda**, **baixo** e **cima**.

Trapped Mouse

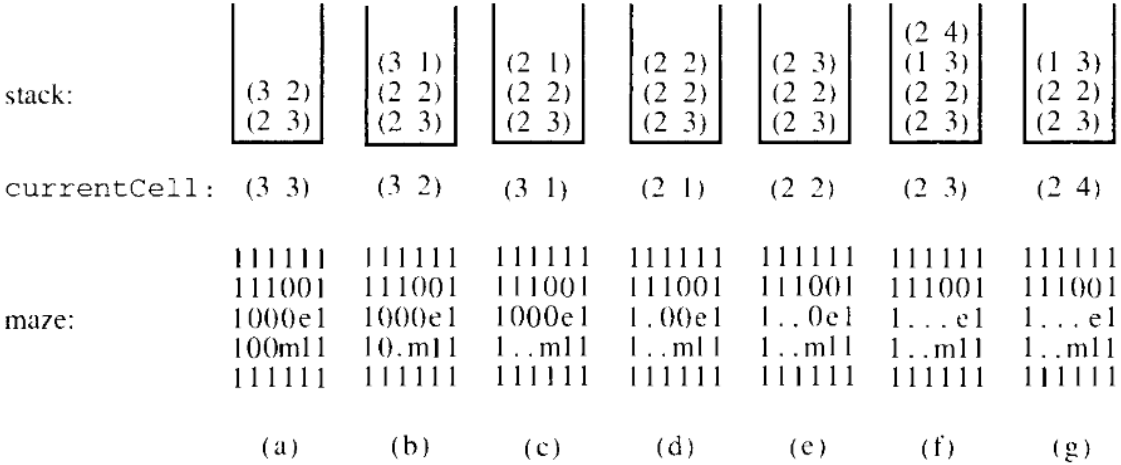
Como fazer

- O rato esperar escapar do labirinto tentando, sistematicamente, todas as rotas.
- Se chegar em um beco sem saída, ele retornar alguns passos até a última posição que tenha um ou mais caminhos não testados.
- Mesmo próximo a saída, ele sempre testa os caminhos não percorridos na mesma ordem: **direita**, **esquerda**, **baixo** e **cima**.
- Para evitar cair em um loop infinito de testar todos os caminhos, os quais já foram investigados, cada posição **visitada** deve ser **macarda**.

Trapped Mouse

Como fazer

- O rato esperar escapar do labirinto tentando, sistematicamente, todas as rotas.
- Se chegar em um beco sem saída, ele retornar alguns passos até a última posição que tenha um ou mais caminhos não testados.
- Mesmo próximo a saída, ele sempre testa os caminhos não percorridos na mesma ordem: **direita**, **esquerda**, **baixo** e **cima**.
- Para evitar cair em um loop infinito de testar todos os caminhos, os quais já foram investigados, cada posição **visitada** deve ser **macarda**.
- Vamos utilizar o caracter **. (ponto)** para marcar como visitado.


stack:
(3 2)
(2 3)
(3 1)
(2 2)
(2 3)
(2 1)
(2 2)
(2 3)
(2 2)
(2 3)
(2 3)
(2 4)
(1 3)
(2 2)
(2 3)
currentCell: (3 3) (3 2) (3 1) (2 1) (2 2) (2 3) (2 4)
maze:
111111
111001
1000e1
100m11
111111
1..m11
1...e1
(a) (b) (c) (d) (e) (f) (g)

Algoritmo 2: Sair do Labirinto

```
1  Stack mazeStack;
2  currentCell = posição inicial do rato;
3  enquanto currentCell não for exitCell faça
4      Marque currentCell como visitado;
5      Coloque na pilha os vizinhos não-visitados de currentCell;
6      se A pilha mazeStack estiver vazia então
7          caminho não encontrado;
8      senão
9          Faça um pop na pilha e o faça currentCell;
10     fim
11 fim
```

Trapped Mouse

A tarefa

- Deve ser utilizado a pilha feita da atividade anterior.
- Criar uma classe, chamada `Cell`, contendo:
  - Privado: Posição `x` e `y`.
  - C++: sobrecarregar o perador ==.
  - Java: sobrecarregar o método `equals`.

Trapped Mouse

A tarefa

- Criar uma classe, chamada `Maze`, contendo:
  - Privado:
    - `Cell` currentCell, exitCell, entryCell;
    - `const char` exitMarker, entryMarker, visited, passage, wall;
    - `Pilha de Cell` mazeStack;
    - `Arrays de String` maze;
    - C++ : Sobrecarregar o operador <<.
    - Java : Sobrecarregar o método `toString()`.
  - Método Público: void exitMaze().

Trapped Mouse

A tarefa

- Pode ser feito em duplas.
- O projeto deve manter o nome das variáveis, métodos e classes.
- Pode-se acrescentar novas coisas no projeto, mas o que foi pedido não pode ser modificado.
- **Ponto extra** para quem fizer o programa com a animação do rato procurando a saída do labirinto.
- Prazo de entrega e apresentação dia 31/05/2016.

Dúvidas, Perguntas?