Trabalho Prático - Algoritmos e Técnicas de Programação

Integrantes:

Tulio Barros Palacine - matrícula: 791352

Alex Gonçalves Diniz - matrícula: 791489

Dayane Stefanne de Freitas Oliveira - matrícula: 793272

Algoritmo:

Primeiramente os arquivos dos mapas foram lidos e cada linha do arquivo foi passada para um array de strings chamado vetLinhas, onde cada elemento desse array continha cada linha do arquivo texto do mapa. Após isso, cada string no array de strings foi convertida para um vetor de caracteres e cada caractere foi atribuido na matriz MatrizLab, desta forma, formando a matriz do labirinto.

Após isso, foi usado o método procuraInicio para encontrar as coordenadas do ponto Start.

Chamada do método testaCaminhos e prints da matriz do labirinto após o mesmo ser resolvido, ou não:

```
/* ------------------------------------------------------------------------
 * Chamada do método que resolve o labirinto
   ------------------------------------------------------------------------*/

System.out.println(x: "Caminho percorrido: ");

if(!testaCaminhos(MatrizLab, coordLinha, coordColuna, direcao: "")){
    System.out.println(x: "Não existe um caminho até a saída!");
}

System.out.println(x: "\n\nprint do labirinto resolvido: ");
printaMatriz(MatrizLab);
```

Funcionamento dos métodos utilizados:

1. **Método printaMatriz:**

Recebe como parâmetro uma matriz de caracteres, a percorre, e printa cada elemento.

2. **Método contadorLinhas:**

Recebe o arquivo txt como parâmetro e conta a quantidade de linhas no arquivo e retorna um inteiro contendo a quantidade de linhas. A quantidade de linhas no arquivo foi usada para instanciar a matriz que armazena os labirintos.

3. **Método procuraInicio:**

Recebe como parâmetro a matriz de caracteres do labirinto, procura as coordenadas onde o caractere 'S' (start) se encontra e retorna um vetor de inteiros contendo as coordenadas i e j, onde o ponto 'S' se encontra.

4. **Método indexEValido:**

Recebe como parâmetro a matriz do labirinto, e dois inteiros que representam os índices das linhas e colunas da matriz. Checa se o elemento no índice é válido, primeiro testando se o elemento está nos limites de dentro da matriz, e depois checando se o elemento é um caractere vazio (caminho que pode ser percorrido), um caractere 'S' (ponto Start), ou um caractere 'E' (ponto End). Retorna um valor Booleano, retornando true caso o elemento possa ser percorrido ou false caso o elemento seja uma parede ou não esteja nos limites da matriz.

5. **Método testaCaminhos:**

Recebe como parâmetro a matriz do labirinto, dois inteiros que representam os índices das linhas e colunas da matriz e uma string que representa a direção que o algoritmo percorre para resolver o labirinto. Primeiramente, usa o método indexEValido para testar se o elemento atual é válido, após isso isso, checa se o elemento atual é equivalente a saída do labirinto (caractere 'E'), caso seja o labirinto está concluído, o método transforma a string direcao em um array, printa o array com as direções percorridas e retorna true. Caso não seja o método altera o elemento por um caractere '+', sinalizando que aquele elemento foi percorrido (testado), no caso, o primeiro elemento sempre será o caractere S (ponto Start), pois suas coordenadas são passadas como parâmetro inicial na chamada do método no main. Após isso, começam as chamadas recursivas dentro do método para testar cada direção que pode ser percorrida, criando uma pilha até o caractere 'E' (ponto End) ser encontrado. É criada a variável novoIndice que irá receber o valor booleano da chamada recursiva do método testaCaminhos, primeiramente é testado para a "direita", passando como parâmetro o valor atual da coluna + 1, e concatenando um 'D' na string direcao que representa que "andou para a direita" no caminho percorrido e retorna a variável booleana novoIndice que guarda a chamada recursiva do método, iterando a pilha. O mesmo teste é feito para as outras direções no sentido horário para criar a pilha, pois durante a criação da pilha o caminho pode "bater na parede". A última chamada do método ocorre quando o elemento atual é o caractere 'E' (ponto Exit), nesse caso, a direção percorrida para chegar no ponto Exit é printada, e o método retorna true, fazendo com que a pilha criada seja 'desempilhada' e todas as chamadas anteriores são concluídas. Caso o método não encontre a saída, retorna false.

Perguntas:

Perguntas:

A) Porque foi garantido que só existe um único caminho de Start até End, ou seja, qual o problema que isso pode gerar com algoritmo recursivo?

Não havendo garantia de uma única entrada e saída , o algoritmo recursivo poderá percorrer o mesmo caminho várias vezes, isso poderá causar um estouro de pilha, ou seja, ficar sem memória suficiente para continuar a execução do programa.

B) Como você trataria o problema de mais de um caminho de Start até End, caso ele existisse, utilizando um algoritmo recursivo?

Validaria os caminhos apresentados para determinar o caminho escolhido. Levando em conta a escolha de qualquer saída, desprezando qualquer entrada e não havendo esta opção, escolha de um caminho possível que tenha sido percorrido pelo menor número de vezes, permitindo a evolução dentro do labirinto.

C) Da mesma forma, como você trataria corredores com largura maior que 1 célula (ou salas), caso isso existisse?

Armazenaria o número de vezes de um caminho percorrido para que em uma validação que esse mesmo caminho fosse apresentado como opção, o programa deverá escolher o caminho com o menor número de vezes percorrido e assim garantir a sequência no labirinto apenas em caminhos não percorridos ou percorridos pelo menor número de vezes.

D) Por fim, caso o ponto End não fosse acessível, como você conseguiria identificar isso através do seu algoritmo iterativo/recursivo?

Uma vez armazenado o número de vezes que os caminhos foram percorridos e com base no tamanho do labirinto, seria possível considerar um número máximo de vezes que um caminho pode ser percorrido até que esgote todas as possibilidades de encontrar uma saída. Atingindo esse número para todos os caminhos possíveis, criaria uma condicional para verificação e retornaria então que não há saída possível.