ALGORITMOS E ESTRUTURA DE DADOS
TRABALHO 1

ALUNA: Flávia Marcella Gonçalves Moreira MATRÍCULA: 2024.1.08.031
ALUNA: Larissa Rodrigues de Ávila MATRÍCULA: 2024.1.08.031

1. INTRODUÇÃO

O problema designado aos alunos trata-se de uma matriz 10x10 que simula o desenho de um labirinto. A matriz contém 100 posições, cujo conteúdo está distribuído em 4 tipos:

“E”, que representa a entrada do labirinto;
“S”, que representa a saída do labirinto;
“0”, que representa um caminho livre;
“X”, que representa uma parede.

Dessa forma, o objetivo do trabalho é criar um algoritmo que encontre um caminho para passar pelo labirinto. O código deve ter como saída todas as coordenadas necessárias para passar pelo labirinto e chegar até o final, imprimidas em sequência, sendo a primeira coordenada a da entrada, e a última coordenada a da saída.

Para resolver o problema, é necessário utilizar algumas das estruturas de listas que foram passadas em sala. Além disso, o trabalho com ponteiros e dimensões também favorece a criação do algoritmo.

2. ESTRUTURAS DE DADOS

As estruturas utilizadas para realizar o trabalho, foram basicamente alocação dinâmica e pilhas.

A alocação dinâmica foi utilizada para criar a matriz que conterá o labirinto, que foi elaborada a partir de ponteiros que apontam para a primeira posição de um “vetor” alocado. Não é exatamente um vetor, mas possui o mesmo funcionamento, e foi a melhor solução encontrada para alocar a matriz.

Além disso, também foi criada uma matriz de posições, que registra as posições que já foram verificadas pelo algoritmo, ajudando a identificar os caminhos já tentados, para que não sejam repetidos.

A estrutura pilha foi criada para salvar as posições cujo caminho é liberado. A escolha da pilha foi realizada pois ela permite que a última posição do caminho seja cancelada, caso o algoritmo tenha pegado o caminho errado. Isso facilita, pois o caminho de “volta” se torna mais fácil, já que é só remover a última posição ou as últimas posições percorridas até que ele encontre um novo caminho aberto.

Por fim, também foi criada a estrutura nomeada “posições”, que armazena as coordenadas (x,y), em relação a cada uma das posições da matriz do labirinto.

3. ALGORITMO

O algoritmo utilizado para resolver o problema do labirinto é uma variação da busca em profundidade (DFS - Depth-First Search). A seguir, descrevemos o algoritmo e sua complexidade:

Algoritmo de Busca em Profundidade (DFS)

Inicialização: Começamos na posição de entrada do labirinto.

Exploração: Utilizamos uma pilha para armazenar as posições a serem exploradas. A cada passo, retiramos uma posição da pilha e verificamos se é a posição de saída.

Movimentação: Se a posição atual não for a saída, marcamos como visitada e empilhamos todas as posições adjacentes que são caminhos livres e ainda não foram visitadas.

Terminação: O algoritmo termina quando encontramos a saída ou quando a pilha ficar vazia, indicando que não há caminho para a saída.

Complexidade

Tempo: O algoritmo DFS tem complexidade O(V + E), onde V é o número de vértices (células do labirinto) e E é o número de arestas (conexões entre as células).

Espaço: A complexidade espacial é O(V), devido ao armazenamento das posições na pilha e à matriz de visitados.

4. SOBRE O MAKEFILE

O makefile foi criado baseado nos exemplos da aula, especificamente baseado no segundo exemplo. No entanto, destacam-se as diferenças que o arquivo com as funções é .cpp, e não .c, portanto as devidas adaptações foram feitas no makefile para não haver conflito entre arquivos. A compilação é feita usando o comando make, e a execução é feita chamando o arquivo executável. O código pede uma entrada, na qual o usuário escolhe qual arquivo quer testar.