CIÊNCIA DA COMPUTAÇÃO UNIFAL

TRABALHO PRÁTICO 1 - SAÍDA DO LABIRINTO

JOÃO PEDRO PEREIRA DE FREITAS 2024.1.08.029
GABRIEL RIBEIRO 2024.1.08.010
CARLOS AUGUSTO REGIS LORIM 2024.1.08.004

AEDS II - PRÁTICA
PROFESSOR IAGO CARVALHO

2024

1. INTRODUÇÃO

O presente trabalho tem como objetivo desenvolver uma solução para o problema de resolução de labirintos utilizando a estrutura de dados Pilha, abordada na disciplina de Algoritmos e Estruturas de Dados II (AEDS 2). A resolução de labirintos é um clássico problema computacional que envolve encontrar um caminho viável entre um ponto de entrada e um ponto de saída dentro de um espaço com obstáculos. Neste contexto, a Pilha se mostra uma estrutura apropriada para implementar algoritmos de busca.

Este relatório descreve o desenvolvimento da aplicação, detalhando os princípios da Pilha, a lógica do algoritmo implementado e os resultados obtidos. A escolha pela Pilha como principal estrutura de dados deve-se à sua capacidade de armazenar e manipular dados de maneira LIFO (Last In, First Out), o que facilita a exploração e o retrocesso (backtracking) no labirinto.

2. ESTRUTURA DE DADOS

Para a implementação do algoritmo de resolução do labirinto, foram utilizadas as seguintes estruturas de dados:

2.1 ESTRUTURA Pilha

```
struct pilha
{
   int topo;
   int tamanho;
   int *valori;
   int *valorj;
};
```

2.2 Matriz do Labirinto

```
char M[10][10];
```

3. ALGORITMOS

Este código implementa a solução de um **labirinto** utilizando o algoritmo de **backtracking** e a estrutura de dados de uma **pilha** para armazenar o caminho percorrido. O labirinto é representado por uma matriz 10x10 lida de um arquivo de texto, onde:

- **'E'** representa a entrada do labirinto;
- **'S'** representa a saída;
- **'X'** são paredes;
- O objetivo é encontrar o caminho da entrada até a saída.

Estrutura do Algoritmo:

3.1 - Criação e Gerenciamento da Pilha: A pilha armazena as coordenadas (i, j) do caminho percorrido. Há funções auxiliares como push (inserir coordenadas na pilha), pop (remover coordenadas) e funções para verificar se a pilha está cheia ou vazia.

3.2 - Leitura da Matriz: A função lerMatriz lê o labirinto de um arquivo e armazena na matriz 10x10.

3.3 - Verificação de Posições Válidas: A função posicao_valida verifica se uma determinada posição na matriz está dentro dos limites e se não é uma parede ('X').

3.4 - Backtracking: A função resolver_labirinto usa backtracking para explorar o labirinto recursivamente. A cada movimento, verifica se a saída ('S') foi encontrada. Caso contrário, tenta mover-se para as quatro direções possíveis (cima, baixo, esquerda e direita). Se uma direção não for válida, a função volta para a posição anterior (backtrack), removendo a última coordenada inserida na pilha.

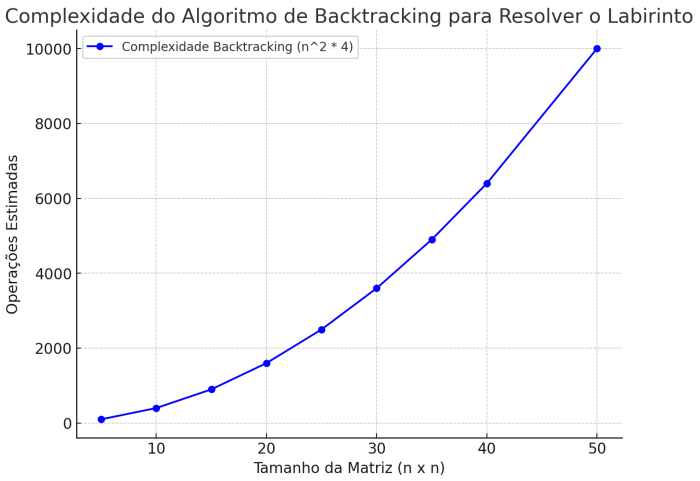
3.5 - Impressão do Caminho: Se a função encontrar a saída, a pilha contendo o caminho é impressa, mostrando todas as coordenadas percorridas desde a entrada até a saída.

Fluxo Geral:

- O algoritmo começa lendo a matriz do arquivo e encontrando a posição inicial ('E').
- A função de backtracking tenta encontrar a saída ('S').
- Se o caminho for encontrado, ele é exibido; caso contrário, uma mensagem indicando a ausência de caminho é exibida.

Em resumo, o código resolve um labirinto utilizando a técnica de backtracking e pilha, e imprime o caminho se uma solução for encontrada.

4. NÍVEL DE COMPLEXIDADE DO ALGORITMO



5. DESCRIÇÃO MAKEFILE

Variáveis:

- **CC=gcc:** Define o compilador (gcc)**.**
- **CFLAGS=-c -Wall:** Flags de compilação:
- **-c:** Gera arquivos objeto (.o), sem linkar.
- **-Wall:** Ativa todos os avisos de compilação.
- **LFLAGS=:** Flags de linkagem (vazio neste caso).

Regra principal:

- **all:** labirinto: Cria o executável labirinto, verificando a dependência de trabalho.o.

Regra para labirinto:

- **labirinto:** trabalho.o: O executável depende de trabalho.o.
- **$(CC) $(LFLAGS) -o labirinto trabalho.o:** Usa o compilador gcc para linkar o arquivo objeto e gerar o executável labirinto.

Regra para gerar main.o:

- **main.o:** trabalho.c: Gera o objeto main.o a partir de trabalho.c.
- **$(CC) $(CFLAGS) trabalho.c:** Compila o arquivo-fonte trabalho.c com os flags -c e -Wall.

Regra de limpeza:

- **clean:** Remove os arquivos objeto (.o) e o executável labirinto com rm -f.