Trabalho Prático 1 – Projeto e Análise de Algoritmos

Universidade Federal de Viçosa – Campus UFV-Florestal
Curso: Ciência da Computação
Disciplina: Projeto e Análise de Algoritmos
Professor: Daniel Mendes Barbosa
Alunos: Lucas Garcia- 5362, Matheus Silva- 5382,Marcus Melo- 5779

Sumário

1. Introdução

Este documento descreve o desenvolvimento de um programa em C para resolver o problema de escape de um labirinto utilizando o paradigma de **backtracking**. O objetivo é implementar um algoritmo eficiente que encontra a saída do labirinto fornecido por meio de um arquivo de texto, respeitando as limitações e possibilidades do problema.

2. Desenvolvimento

2.1 Estrutura do Algoritmo

O programa utiliza estruturas de dados dinâmicas para representar o labirinto e uma abordagem recursiva para explorar as possibilidades de movimento do estudante. O algoritmo verifica todas as opções viáveis até encontrar a saída ou concluir que o labirinto não tem solução.

2.2 Funcionamento do Backtracking

O **backtracking** é utilizado para retroceder em caminhos que não levam à solução. O estudante marca as células visitadas e retorna ao estado anterior se encontrar uma barreira

ou um beco sem saída. Essa abordagem garante a exploração de todas as opções possíveis.

2.3 Implementação

Estruturas de Dados

- `Student`: Representa o estado atual do estudante (posição e número de chaves).
- `maze`: Matriz alocada dinamicamente para armazenar o labirinto.

Principais Funções

1. **`load_file`:** Carrega o labirinto a partir de um arquivo de texto.
2. **`valid_position`:** Verifica se uma posição é válida para movimentação.
3. **`move_student`:** Realiza a exploração recursiva do labirinto.
4. **`free_memory`:** Libera memória alocada dinamicamente.

Trechos de Código Relevante:

1.

```
bool valid_position(int row, int col, int keys) {
   return (row >= 0 && row < num_rows &&
           col >= 0 && col < num_cols &&
           (maze[row][col] == 1 || maze[row][col] == 0 ||
            (maze[row][col] == 3 && keys > 0)));
}
```

Este trecho verifica se a posição está dentro dos limites do labirinto e se é acessível, considerando o número de chaves disponíveis.

2.

```
void move_student(Student *student, int recursion_level){}
```

Essa é a função mais importante do programa, ela realiza o backtracking para tentar encontrar a saída do labirinto.

3. Testes e Resultados

Arquivo de Entrada:

10 10 1
1111111111
1111111111
2222212222
1111111111
1111111111
3222222111
1111112111
1222222111
1111111111
1111011111

Cenário de Teste

Leitura do arquivo test.txt

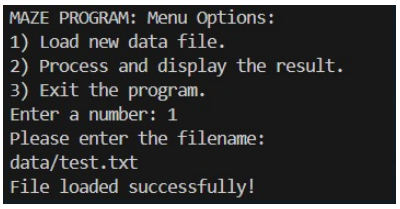

Figura 1: Leitura do arquivo.

Resultado

Saída gerada:

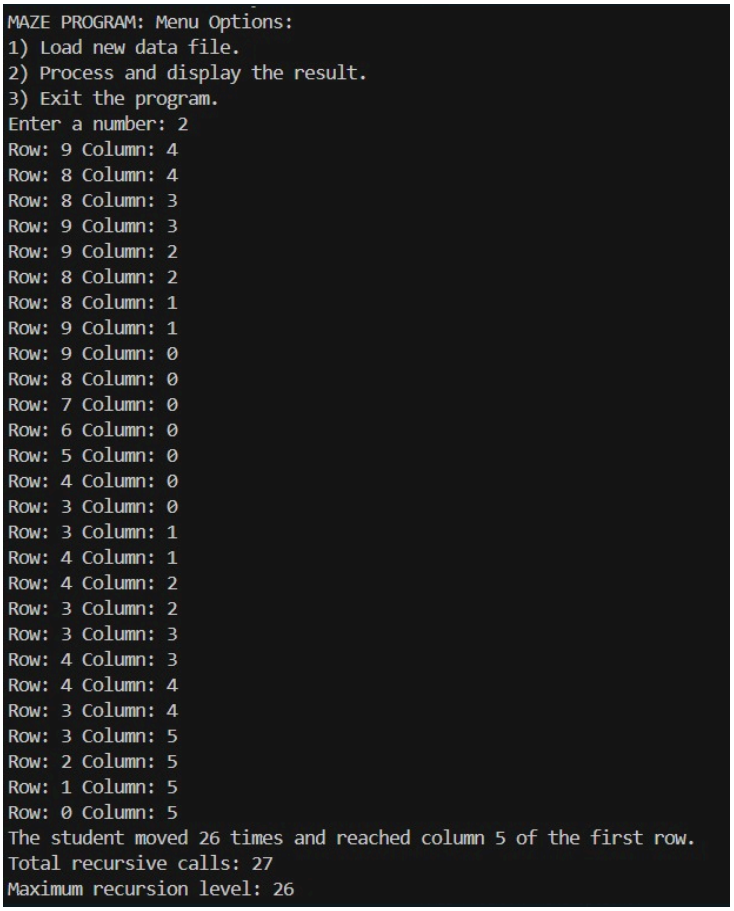

Figura 2 : Saída do programa.

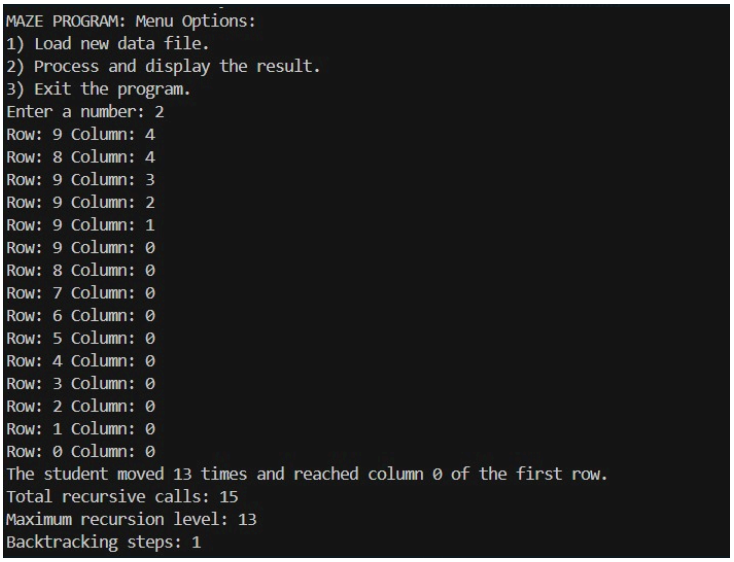

Figura 3: Exemplo da saída que mostra o backtracking, usando outro arquivo teste.

Limpeza do executável

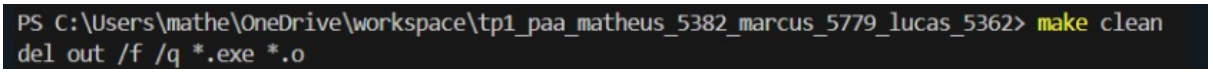

Figura 4: Limpeza do executável.

4. Decisões de Projeto

1. O programa utiliza um `#define` para habilitar ou desabilitar o modo de análise.
2. Foi implementada a alocação dinâmica para permitir a leitura de labirintos de tamanho variável.
3. As movimentações consideram portas e uso de chaves.
4. Limitações: Não foi implementado suporte para labirintos com chaves no chão (células amarelas).

5. Detalhes

Dado o uso de cmake no projeto aqui vai um guia para instalação.

Windows:

1. Baixe o instalador do CMake no site oficial: https://cmake.org
2. Durante a instalação, selecione a opção "Add CMake to the system PATH for all users".

Linux:
Instale via gerenciador de pacotes:

sudo apt-get install cmake

MacOS:

Instale usando o Homebrew:

brew install cmake

6. Conclusão

O trabalho permitiu aprofundar o entendimento do paradigma de **backtracking** e de conceitos como recursão e alocação dinâmica de memória. Apesar das limitações, o programa cumpre os requisitos propostos e pode ser expandido com funcionalidades adicionais.

7. Referências

1. Sites e fóruns como Stack Overflow e GeeksforGeeks.
2. Material da disciplina: Projeto e Análise de Algoritmos (UFV-Florestal).
3. Uso da ferramenta cmake https://cmake.org.