Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamente de Ciência da Computação
Algorítmos e Estruturas de Dados II

Trabalho Prático 1
Documentação

Julia Tiemi Alkmim Morishita
2015004240
Abril de 2016

Introdução

O mito de Teseu e o Minotauro faz parte da mitologia grega. A história se inicia quando o filho de Minos, Androgeu, foi enviado por Egeu para combater o touro de Creta. Quando Androgeu morre tentando matar o touro, Minos declara guerra contra Atenas. Ao ganhar a guerra, ele define que a cada nove anos, sete rapazes e sete moças atenienses deveriam ser enviados para Creta a fim de serem presos num labirinto com seu filho meio humano meio touro, o Minotauro. Na terceira remessa de jovens, Teseu decidiu ir para derrotar o poderoso monstro.

Na versão adaptada para este trabalho, a única forma de derrotar o Minotaura é com a ajuda de uma espada mágica que está localizada dentro desse labirinto. Tendo em conhecimento a localização da entrada, da espada e do mapa do labirinto, cabe a nós, programadores, ajudar Teseu a encontrar a arma.

Implementação

1) Estrutura Labirinto:

Foi criado uma estrutura para armazenar todos os dados de um labirinto. A estrutura Labirinto consiste em cinco variáveis do tipo int – N, x, y, sx e sy – que representam a dimensão do labirinto, os par de coordenadas da entrada e o par de coordenadas da espada, respectivamente. Além disso ela possui um ponteiro de ponteiro para um unsigned char mapa, que virá a armazenar a matriz que representa o mapa do labirinto.

2) Labirinto* LeLabirinto(const char *entrada):

Para registrar os dados de um labirinto na estrutura citada acima, foi criada a função LeLabirinto. Essa função recebe como parâmetro um dos argumentos de entrada. Isso significa que na linha de comando da execução do programa:

./exec labirinto.txt caminho_espada.txt alg

O primeiro argumento argv[0] é exec, argv[1] é labirinto.txt, argv[2] é caminho_espada.txt e argv[3] é alg. Para esse função, estamos interessados apenas em argv[1]. A saída dessa função é um ponteiro para uma estrutura Labirinto com os dados de um certo mapa registrados. A função abre o arquivo do labirinto e aloca dinâmicamente uma estrutura Labirinto para o armazenamento dos dados. Ao abrir o arquivo, foi escolhido o mode r para leitura. Após ter aberto o arquivo, ela lê as informações do labirinto, tais como suas dimensões e as coordenadas da entrada e da espada. A seguir, ela usa suas dimensões pra alocar dinamicamente a matriz lab->mapa, que posteriormente é preenchida com os valores do arquivo original. Terminado esse processo, o arquivo é fechado e a estrutura Labirinto é retornada para a main.

3) Função int CaminhaLabirintoRecursivo(Labirinto *lab, int x, int y, int **sol):

Ao ser indentificado na main que o usuário deseja percorrer o caminho recursivamente, entramos na função CaminhaLabirintoRecursivo. Ela recebe de parâmetro a estrutura Labirinto, duas coordenadas, x e y e a matriz onde fica salvo o caminho até a espada. Primeiramente, a função checa se as coordenadas atuais não são as mesmas coordenadas da espada. Neste caso, teríamos a solução final, chegaríamos ao nosso destino. Se esse não for o caso, ele checa se a entrada e a posição da espada são coordenadas válidas de valor '0' no mapa. Se forem, ele chama recursivamente a função para as coordenadas aos lados, acima e abaixo da posição

atual, gravando sempre o rastro deixado na matriz sol.
A seguir as funções do TAD de pilha.
1) Estrutura Item: essa estrutura armazena todos os componentes de informação de uma célula da pilha. Nesse caso, ela possui duas variáveis do tipo int, x e y, que determinariam as coordenadas do caminho de Teseu.
2) Estrutura Celula: uma célula de uma pilha precisa armazenar duas variáveis – uma estrutura Item e um apontador para uma estrutura Celula que determina a ordem da pilha.
3) Estrutura Pilha: essa estrutura determina a própria pilha, com dois apontadores para uma estrutura Celula, que determinaria a primeira e a última células da pilha, além de uma variável do tipo int que armazena a quantidade de células na pilha.
4) Função FazPilha: cria uma pilha, inicializando-a com os atributos de uma pilha vazia.
5) Função Vazia: verifica se uma pilha está vazia (true) ou não (false), retornando 0 ou 1.
6) Função Empilha: cria uma nova célula e a adiciona no topo da pilha.
7) Função Desempilha: libera e remove a célula do topo da lista, salvando suas informações num tipo Item.
8) Função Tamanho: retorna a quantidade de células na pilha.

Resultados

Foram realizados dois testes:

```
6 1 0 4 4
1 1 1 1 1 1
0 0 1 1 1 1
1 0 0 1 0 1
1 1 0 1 1 1
1 0 0 0 0 1
1 1 1 1 1 1
```

O primeiro teste consistiu num arquivo de entrada onde era possível que Teseu encontrasse a espada. Escolhendo o algorítmo recursivo, a matriz resultado foi impressa no arquivo de saída perfeitamente, sem encontrar nenhum problema.

```
6 1 0 4 4
1 1 1 1 1 1
0 0 1 1 1 1
1 0 0 1 0 1
1 1 0 1 1 1
1 0 1 0 0 1
1 1 1 1 1 1
```

Porém, no segundo teste, o caminho de Teseu estava bloqueado por uma parede. O problema não foi resolvido e resultou numa falha de segmentação.

Sobre a análise de complexidade das funções temos que:
A função LeLabirinto possui ordem de complexidade $O(n^2)$, devido à leitura da matriz que necessidade de dois loops, um dentro do outro.
A função CaminhaLabirintoRecursivo possui ordem de complexidade $O(2^n)$ no pior caso.
A função CaminhaLabirintoIterativo que não foi realizada, poderia ter complexidade $O(n^2)$, de forma que fosse percorrendo o caminho e alterando o labirinto ao mesmo tempo, criando uma parede por onde já tivesse passado. Esse é o custo mais baixo possível para resolver esse problema.
A função main tem complexidade $O(n^2)$, pois a matriz resultado é impressa na main, criando dois loops, um dentro do outro.
As funções do TAD pilha são todas O(1).

Conclusão

O Trabalho Prático 1 foi realizado parcialmente, devido à diversos fatores. Primeiramente, não fui capaz de encontrar uma forma de implementar a função CaminhaLabirintoIterativo. Isso fez com que eu administrasse mal meu tempo e não me dedicasse ao restante do trabalho, tendo em mãos apenas uma fração do que deveria ser feito. O TAD de pilha foi feito, porém não utilizado. Ele está anexado ao trabalho apenas para fim de avaliação.

Referências

https://pt.wikipedia.org/wiki/Teseu#Teseu_e_o_Minotauro_de_Creta