Documentação Trabalho Prático 0 de AED's 2

Aluno: Marcus Vinícius de Oliveira **Matricula: 2014144804**

1. Introdução

1.1 Contexto

A história de Teseu e o Minotauro é uma das mais conhecidas da Mitologia Grega. Conta o mito que, em certa época, Atenas era obrigada a enviar anualmente jovens para que fossem colocados num labirinto e devorados por um Minotauro, como forma de tributo à Creta.

Teseu, um corajoso jovem, filho de Egeu (ou Poseidon, dependendo da versão) rei de Atenas, decidiu voluntariar-se para que pudesse vencer o monstro e libertar Atenas.

Ao chegar em Creta, conhece Ariadne que apaixonada o presenteia com um novelo de fio para facilitar seu retorno. Dessa forma, ele marcaria o caminho de volta do labirinto e quando o combate com o Minotauro terminasse, ele poderia retornar para fora do labirinto sem se perder.

1.2 Nosso problema

Nosso problema é similar, porém com singelas diferenças. Em nosso caso o jovem Teseu não está em busca do Minotauro, mas sim de uma poderosa espada capaz de mata-lo! Essa espada também se encontra em um labirinto.

Apesar da coragem, nosso jovem Teseu não é tão galã quanto o original da mitologia, logo não dispõe de nenhum novelo de fio. Desta forma ele nos pede ajuda para encontrar a espada.

Teseu tem as coordenadas de entrada do labirinto, bem como seu mapa, e as coordenadas de onde se encontra a espada. Assim, devemos escrever um programa que dado esses dados retorne o caminho até a espada, se possível.

2. Implementação

O arquivo de entrada é dado no seguinte modelo:

```
N x y sx sy
L(0,0) L(0,2) L(0,3) ... L(0,N-1)
L(1,0) L(1,2) L(0,3) ... L(1,N-1)
...
L(N-1,0) L(N-1,2) L(N-1,3) ... L(N-1, N-1)
```

Onde N é a dimensão do labirinto, x e y são as coordenadas da entrada do labirinto e sx e sy são as coordenadas da espada.

Conforme especificação, o arquivo de entrada é um .txt passado por parâmetro através da função argv[1]. Dentro da função main esse parâmetro é passado para a função LeLabirinto, onde ocorre toda a leitura e transferência desses dados para uma struct Labirinto.

Juntamente com a matriz solução NxN alocada na main, a struct Labirinto é passada como parâmetro para função CaminhaLabirintoRecursivo ou para função CaminhaLabirintoIterativo. A decisão de qual função será chamada é dada pelo valor de entrada do argumento argv[3]: se argv[3] igual à 0 então, CaminhaLabirintoRecursivo. Se argv[3] igual à 1, CaminhaLabirintoIterativo.

Tanto no CaminhaLabirintoRecursivo quanto CaminhaLabirintoIterativo, através dos dados obtidos da leitura e organização do arquivo de entrada, ocorre a computação do caminho entre a entrada do labirinto e o local da espada. A diferença entre um e outro é a forma como isso é feito. Detalhes de cada algoritmo podem ser obtidas nas próximas subseções deste documento.

Caso exista um caminho para o labirinto, essas funções retornam true, caso não exista false. Este valor é armazenado na função main pela variável resultado.

A matriz solução, passada como parâmetro para essas duas funções, guarda o caminho que Terceu deve percorrer para chegar até a espada (caso exista) e pode ser acessado através da função main.

De volta a main a sulução para o labirinto, caso ela exista, é escrita em um arquivo em disco, conforme indicado por argv[2]. Caso não exista, é gravado neste arquivo o numero 0.

Posteriormente é liberada a memoria utilizada e fechado os arquivos abertos. É retornado o valor do resultado e finalizado o programa.

2.1 Funções

labirinto.h || labirinto.c

Labirinto *LeLabirinto(char *entrada): Está função tem como objetivo ler os dados da imagem e armazena-los em uma estrutura de dados Labirinto

Função: Le os dados da imagem e armazena em uma estrutura de dados Labirinto conforme asseguir:

```
typedef struct {
        int N; // Dimensão do labirinto, lmebre-se que o mesmo é N x N
        int x; // Coordenada x da entrada
        int y; // Coordenada y da entrada
        int sx; // Coordenada x da espada
        int sy; // Coordenada y da espada
        unsigned char **mapa; // variável para armazenar o mapa (matriz)
}Labirinto;
```

Sua entrada é composta de um ponteiro para char. Durante a execução da main, é passado como parâmetro o argumento argv[1], contendo o endereço/nome do arquivo de entrada.

Esse arquivo é aberto e através de operações simples é lido e seus dados armazenados.

O arquivo é fechado e em sua saída é retornada a struct Labirinto devidamente preenchida.

void LiberaLabirinto(Labirinto *lab): Está função é auxiliar à LeLabirinto. Durante a execução da função LeLabirinto, é alocado dinamicamente uma estrutura Labirinto.

LiberaLabirinto tem como função desalocar por completo toda uma estrutura Labirinto desde sua matriz mapa até a própria estrutura.

CaminhaLabirintoIterativo(Labirinto* lab, int x, int y, int ** sol): Está função recebe uma estrutura Labirinto, as coordenadas x e y da entrada do labirinto, as coordenadas sx e sy da localização da espada e a matriz solução. Tem como objetivo computar interativamente o caminho entre a entrada do labirinto até o local em que a espada se encontra.

Durante a modelagem do problema percebi que todo o caminho possível para se percorrer dentro do labirinto poderia ser representado por uma arvore.

Após perceber este fato, pesquisei formas de se percorrer uma arvore e descobri que o algoritmo que necessitava para resolver este problema seria o um algoritmo de busca em profundidade.

Com apoio do material didático do professor Haroldo G. Santos, foi possivel montar o seguinte algoritmo:

. Cria-se uma pilha para armazenar as posições corretas que Terseu deve percorrer.
. Considero a entrada como meu ponto inicial e armazeno sua coordenada na pilha.
. Enquanto a coordenada atual for diferente da coordenada da espada
    . Encontro as coordenadas vizinhas acessíveis da coordenada atual ainda não visitada.
    . Se uma ou mais forem encontradas
        . Escolho ela como coordenada atual.
    .Empilho a coordenada atual na pilha.

    Senão
        . Desempilho posição da coordenada atual.
    . Considero a coordenada anterior (atual topo da pilha) como a atual.

Se em algum momento a coordenada atual voltar a ser a coordenada de entrada do Labirinto, então este não tem solução.

Se não existe solução, é retornado false para a main.

Caso exista, no momento em que ela é encontrada, a matriz solução é zerada. Posteriormente é usado o caminho armazenado na pilha para marcar na matriz solução o caminho a ser percorrido. Por fim, é retornado true para main.

CaminhaLabirintoRecursivo(Labirinto* lab, int x, int y, int ** sol): Está função recebe uma estrutura Labirinto, as coordenadas x e y da entrada do labirinto, as coordenadas sx e sy da localização da espada e a matriz solução. Tem como objetivo computar recursivamente o caminho entre a entrada do labirinto até o local em que a espada se encontra.

Seu algoritmo foi dado pelos monitores e consiste basicamente em algo semelhante ao anterior. Porém com o uso da recursão, a utilização de pilhas é desnecessária.

A medida que o algoritmo percorre os caminhos possíveis, a matriz solução vai sendo preenchida e ao encontrar a solução a função retorna imediatamente um true para main.

Caso não encontre solução é retornado false para a main.

3. Resultado

O programa rodou sem problemas para os dois casos testes propostos. O numero de testes foi pouco para uma analise mais detalhada. E graças a minha capacidade de gerenciamento de tempo, não será possível fazer os estudos dos resultados! =/

4. Conclusão

Durante a implementação uma dificuldade inicial com parâmetros da argv[] bem como uma dificuldade com ponteiros foi sentida, entretanto com auxilio da comunidade stackoverflow, as dificuldades foram vencidas e o trabalho continuou sem muita dificuldade. E graças a minha capacidade de gerenciamento de tempo, não será possível fazer uma conclusão decente! =/

5. Referências

https://pt.wikipedia.org/wiki/Busca_em_profundidade

http://www.decom.ufop.br/haroldo/disciplinas/aedsIII/files/aedsiii_05.pdf

http://www.ii.uni.wroc.pl/~wzychla/maze_en.html

http://stackoverflow.com/questions/9777117/char-isnt-converting-to-int