|  |  |
| --- | --- |
|  | UFMS – Universidade Federal de Mato Grosso do Sul  CPTL – Campus de Três Lagoas  Curso de Bacharelado em Sistemas de Informação  Disciplina: Algoritmos e Programação II  Professores: IVONE PENQUE MATSUNO YUGOSHI  RONALDO FIORILO DOS SANTOS |

|  |  |  |
| --- | --- | --- |
| Nome do Aluno: Gabriel Portari de Moraes Oliveira | | RGA: 2019.0743.034-2 |
| Assunto: Relatório Trabalho | Data(s): 04/10/2021 | |

**Relatório do primeiro trabalho avaliativo de Algoritmos e Programação II**

Objetivos: Desenvolver um programa que leia uma matriz, que será utilizada como arena, um robô, e suas respectivas saídas, onde o algoritmo deve mostrar as saídas encontradas dado a quantidade de passos que o robô pode utilizar.

Para a realização dessa atividade, foi usado um algoritmo com um "contador" cuja base é 5, ou seja, representado apenas pelos algarismos 0, 1, 2, 3 e 4, onde cada algarismo, exceto o zero, representa uma direção, e, dado o orçamento que o robô pode se locomover, ele testa todas as possibilidades de 11111 até 44444, porém, é uma ideia um pouco falha, pois o robo pode ter um caminho repetido, como cbcb ou cdbe, onde sempre sai no mesmo lugar, mas como não há interesse em eficiência, esses erros não importam.

No inicio do programa, temos a abertura do arquivo na posição argv[1], para que o arquivo dado na linha de comando possa ser lido, seguido da leitura desse arquivo, até a parte das dimensões, logo é alocado um espaço na memória para a matriz, dados as dimensões lidas anteriormente, e após a alocação, a matriz é lida.

Como no arquivo os espaços em branco na matriz são representados como a tecla espaço, e não como \0, a conversão de " " para "\0" é feito já na leitura, tal como os "\n" no final da linha, que são excluidos pois atrapalham o funcionamento do programa.

Fecha-se o arquivo, e após a leitura da matriz, ela é varrida atrás da posição inicial ao robo, onde sua posição é salva.

Na parte principal do programa (onde os testes são calculados), após ser localizado a posição inicial do robô, temos um loop que só sera quebrado quando o nossa variável de contagem for igual a 4\*orçamento, que significa que todos os caminhos já foram testados.

Para cada numero de 1 a 4, é estabelecido uma direção(1-cima, 2-baixo, 3-direita, 4-esquerda), para que seja testado, e em cada caso de teste os passos são os mesmos, exceto a direção em que o robo anda, os passos são:

* Ver se o caminho está fechado, ou seja, testar se a proxima direção é igual a #, e caso seja, o algoritmo para de testar o resto do caminho, e vai para o proximo
* Ver se o caminho é uma saída, ou seja, testar se a proxima direção é diferente de # e também diferente de \0
* E por fim, ver se o caminho está livre, e caso esteja, move o robô de lugar, assim como as váriaveis de posição x e y, também incrementando o contador de passos que o robo deu

No caso do robô achar uma saida, é salvo em um registro a quantidade de passos que foram dados, a saída que foi encontrada, e o caminho que foi utilizado, e para que não fique salvo vários caminhos iguais, por ex cccdd, cccde, ccced, que foi um problema encontrado, salva-se o caminho até a quantidade de passos encontrado, por ex 3 passos, salva ccc, e após isso, o vetor de registros é varrido para procurar se existem registros iguais, como no exemplo acima, e caso encontre, ele é excluido e o vetor é puxado para tras.

Após isso o proximo caminho a ser testado é incrementado utilizado a base 5, que funciona da seguinte forma: dado um vetor com tamanho orçamento, é incrementado +1 na sua primeira posição, e após isso varre-se este vetor procurando numeros maiores que 4, e caso encontre incrementa +1 na proxima posição do vetor, iniciando de 0000 e indo até 4444.

No final da parte principal do programa, temos o reset de algumas variáveis, tais como a posição do robô na arena, e tambem suas váriaveis de posição inicial.

Após isso, abre-se o arquivo saida.txt, e é executada outra varredura no nosso vetor de registros, atrás da saída mais proxima, para ser mostrada no console, e depois é mostrado no arquivo as saídas encontradas.

Para que os caminhos sejam mostrados como direção, e não como um número, que é como são salvas, utiliza-se um switch, que seleciona o número salvo e mostra a direção que ele representa.

Para mostrar as saídas em ordem, como meu programa as salva na ordem em que são achadas, foi utilizada uma função, onde há um vetor de caracteres, em que cada posição deste vetor é o seu número respectivamente, por ex v[0] = '0', v[1] = '1'; deste modo eu posso testar a posição do vetor, e comparar seu conteudo com uma das saídas do registro para poder mostra-las em ordem.

E no final fecha-se o arquivo saida.txt, e libera-se o espaço alocado na memoria.

A minha principal dificuldade neste exercício, foi conseguir implementar o algoritmo que conta na base 5, assim que li o que devia ser feito, ja sabia como fazer, mas na hora de implementar levei um bom tempo para descobrir como fazer. Outro problema também foi o manuseio do vetor de registros, pois dentro desse vetor de registro há outro vetor que salva os caminhos, e para comparar o vetor de registro, com o vetor caminho, foram utilizados vários for, um dentro do outro e isso me deu um bom tempo para acertar tudo.

Em sumo, fiz o trabalho sozinho, e tentei fazer o melhor programa que pude, sem economizar código, colocando tudo que aprendi em prática, e desenvolvendo isso até que chegasse ao resultado final, acredito que minha nota seja um 9.