UFMS – Universidade Federal de Mato Grosso do Sul

CPTL – Campus de Três Lagoas

Curso de Bacharelado em Sistemas de Informação

Disciplina: Algoritmos de Programação 2

Professores: Ivone Penque Matsuno Yugoshi e Ronaldo Fiorilo dos Santos

Nome: Vinicius Silva de Paula	**RGA: 2017.0743.043-8**
Assunto: Relatorio Trabalho 1 de Algoritmos 2	**Relatorio:07/10/2021**

Relatório Trabalho 1 de Algoritmos de Programação 2.

Sumário

Relatório Trabalho 1 de Algoritmos de Programação 2. 1

1. Introdução. 2
2. Método utilizado na resolução. 2
   - 2.1. Método de leitura e alocação. 2
   - 2.2. Método de busca. 2
   - 2.3. Método de Identificação de início e saída. 3
   - 2.4. Principais desafios. 3
   - 2.5. Resolução dos desafios. 3
3. Considerações Finais. 3
4. Avaliação dos participantes. 3

Três Lagoas - MS
Sistemas de Informação
UFMS - CPTL
2021

UFMS – Universidade Federal de Mato Grosso do Sul

CPTL – Campus de Três Lagoas

Curso de Bacharelado em Sistemas de Informação

Disciplina: Algoritmos de Programação 2

Professores: Ivone Penque Matsuno Yugoshi e Ronaldo Fiorilo dos Santos

1. Introdução.

O trabalho se baseia em um robô em um labirinto onde possui de 0 a 10 saídas, esse robô precisa encontrar a saída mais curta dentro da arena, para isso também é informado o tempo que o robô possui para sair do labirinto, ou seja o robô deve descartar as saídas que passarem do tempo estabelecido, além disso caso tenha obstáculos na arena o robô deve contornar, ele deve identificar também quando a saída estiver bloqueada por algum obstáculo assim não podendo ser alcançada, o tamanho da arena e as demais informações como a posição inicial do robô, posições das saídas entre outras serão informadas através de um arquivo.

Com isso pesquisamos métodos de resolução e encontramos dois métodos, primeiro método foi o A* e o segundo e o algoritmo de Djkistra, estudando os dois métodos, chegamos à conclusão de que o A* seria mais útil para o nosso caso.

2. Método utilizado na resolução.

2.1. Método de leitura e alocação.

A leitura do arquivo foi realizada utilizando uma variável lixo para descartar o que não seria útil para a implementação da solução, após a leitura dos parâmetros é feita a separação da dimensão para que seja possível a alocação da arena e do nó de mapeamento da arena.

A partir desse momento é feita a alocação da arena e do nó a partir dos parâmetros de dimensão, a leitura da arena e o nó sendo iniciado com as configurações padrão em toda interação com cada ponto de saída, a única informação que difere é se a posição do nó é andável ou não, para saber se é andável é verificado cada posição da matriz contendo o mapa da arena.

Além da alocação das matrizes também foi alocado um vetor de pontos para conseguir mapear o caminho de volta para o inicio do labirinto.

2.2. Método de busca.

Para resolução do problema foi estudado o método de busca A*, para que isso fosse possível mapeamos as entradas e saídas utilizando um registro de pontos, assim deixando mais fácil para reconhecer as coordenadas, logo após mapear as saídas e o inicio da arena foi utilizado um registro para mapear tudo o que pode ser andável e os obstáculos do mapa, a partir desse mapeamento implementamos o método de busca passando o início e as saídas uma por vez, o método de busca funciona da seguinte forma, é verificado todas as posições em volta da posição atual, com isso ela coloca o ponto atual na lista aberta identificando a lista que o ponto atual está por zero e um, sendo zero falso e um verdadeiro, após isso ele percorre toda a matriz procurando o menor custo total dentro das posições com o indicativo de lista aberta em um e seleciona ele como o próxima posição adicionando ele na lista fechada, assim ele calcula o custo utilizado para o movimento até aquela célula vizinha, para isso basta ir acumulando o valor da célula anterior somando um(no caso desse projeto pois o custo de movimento é constante 1), a distância daquela célula vizinha até o ponto de saída e por fim calcula o custo total do movimento em relação a saída, para calcular o custo total basta somar o custo do movimento e a distância da célula vizinha, após o termino do laço é feito o retorno do ponto final ao ponto inicial mapeando as direções andadas.

Para a impressão das direções foi salvo o caminho de volta em uma variável de pontos e essa variável de pontos e depois é impresso ao contrario para que seja mostrado o movimento do início para a saída.

2.3. Método de Identificação de início e saída.

Para encontrar os pontos de início e saídas foi feito uma busca na matriz procurando os símbolos com tal significado após encontra o símbolo ele salva a coordenado da posição, assim mapeando todos os pontos referentes a inicio e saída da arena.

2.4.Principais desafios.

Os desafios encontrados na implementação quase todos foram relacionados com a alocação de memorio, ao alocar a memória fora da função principal do código estava sendo alocada uma posição invalida assim levando a falha de segmentação ou retornando valores de outras variáveis, com isso os resultados não estavam saindo como o esperado, assim ocupando muito tempo para resolver e mesmo assim não foi totalmente solucionado, perdendo assim a modularização do código em certos pontos.

2.5.Resolução dos desafios.

Para que isso fosse solucionado a alocação precisou ser feita dentro da função principal do código quebrando a modularização, as funções de alocação estão implementadas no código porem não estão sendo utilizadas conforme descrito nas considerações finais.

3. Considerações Finais.

A arena e o nó estão sendo alocados na função principal do código seguindo as orientações do professor, existe uma função para alocar os dois, como conversado com o Ronaldo, a alocação estava dando erro e matando o código.

- OBS: O erro de alocação não foi identificado.

4. Avaliação dos participantes.

Avaliando o meu desempenho no trabalho e a interação que tive no desenvolvimento do código, tanto na parte de implementação quanto na análise de código levando em consideração o tempo empregado e disponível para esse projeto, será empregada uma nota levando em consideração todos os quesitos levantados acima, para a interação na parte de desenvolvimento a nota imposta será 8 (oito), já na análise dos códigos para a resolução de erros a nota imposta será 7 (sete), e para o tempo disponível utilizado a nota imposta será 9(nove), resultando assim uma média 8 (oito).

Para a avaliação do Julio irei utilizar os mesmos critérios citados acima, para interação com a implementação a nota imposta será 9 (nove), para a análise dos códigos para resolução de problemas e erros a nota imposta será 8 (oito) e para o tempo disponível e utilizado para o trabalho a nota imposta será 9 (nove), resultando em uma média 8,7 (oito virgula sete).