

Algoritmo A Star

Planejamento de caminhos

Julio Cezar Lossavaro

2018.0743.029-4

Três Lagoas

2021



Sumário

1.	Estruturas		3
	1.1.	Node	3
1	.1.1.	Variáveis	3
	1.2.	Pontos	4
1	.1.2.	Variáveis	4
2.	Funçõ	es e métodos	4
	2.1.	leitura_parametro:	4
	2.2.	calcula_saida:	4
	2.3.	aloca_arena:	4
	2.4.	libera_arena:	4
	2.5.	leitura_arena:	4
	2.6.	Identifica_saidas:	5
	2.7.	iniciaNode:	5
	2.8.	aloca_no:	5
	2.9.	liberaNo:	5
	2.10.	isWalkable:	5
	2.11	encontraMenorCaminho:	5

1. Estruturas

1.1. Node

A estrutura nó foi criada com o objetivo de armazenar valores importantes para realizarmos o calculo da distância até o destino, cada posição da arena terá um Node com as suas informações de cálculo.

1.1.1. Variáveis

- Int andavel: o atributo andável vai ser utilizado para mapear a arena, sendo 1, caso a posição não esteja bloqueada e 0, caso não.
- Int listaAberta, listaFechada: essas duas variáveis serão incrementadas, caso a determinada posição estiver como aberta ou fechada, para a verificação.
- **Int g:** A variável g é uma constante 1, que será incrementada a cada vez que o nó avançar em direção ao ponto final.
- Int h: Essa variável será responsável por armazenar o modulo da diferença entre o caminho inicial e o caminho final, ignorando obstáculos.

Exemplo: Entrada: [2,5], Saída: [5,5], h = |((2-5)-(5-5))| = 3.

• Int f: o f será o responsável por armazenar o "peso" de cada posição.

Exemplo: F = G + H

- Int parenteLinha, parenteColuna: o parente será responsável por apontar para a posição pai da posição analisada.
- Char direção: essa variável demonstra a posição da posição atual para o seu pai. Exemplo: caso o seu pai esteja a direita, ele recebera a letra 'd'.



1.2. Pontos

Esta classe é responsável por armazenar dois inteiros, sendo eles x e y, ela é utilizada com o intuito de armazenar valores de posições de um determinado caminho.

1.1.2. Variáveis

Int x: posição de linha, que será utilizada em conjunto com y para apontar uma determinada posição(ponto).

Int y: posição de colunas, que será utilizada em conjunto com y para apontar uma determinada posição(ponto).

2. Funções e métodos

2.1. leitura_parametro:

Este método é responsável por ler os parâmetros passados por um arquivo e em seguida os atribuir a variáveis.

2.2. calcula saida:

Este método recebe por parâmetro uma matriz de caracteres, e a partir dela mapeia todas as saídas e as armazena em um vetor de Pontos.

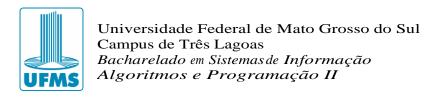
2.3. aloca_arena:

Recebe uma matriz com as suas colunas não iniciadas, em seguida aloca as suas colunas com base no parâmetro passado, ou seja, ele é responsável por alocar dinamicamente a matriz de caracteres.

2.4. libera arena:

Recebe uma matriz, **seu tamanho e em seguida** libera toda a memória alocada para a mesma.

2.5. leitura_arena:



Recebe o arquivo de entrada, uma matriz de caráteres e em seguida mapeia a arena dentro desta matriz.

2.6. Identifica_saidas:

Recebe uma matriz de char, junto com o seu tamanho e mapeia todos os caracteres que representam uma saída, em seguida os armazena em um vetor de Pontos o qual é recebido por parâmetro.

2.7. iniciaNode:

Essa função recebe uma matriz de nós já previamente alocados e em seguida os mapeia com base em uma matriz de caracteres, iniciando os seus valores e verificando se o caractere é equivalente a uma posição bloqueada ou não, essa função utiliza o método isWalkable para fazer essa verificação de bloqueio.

2.8. aloca no:

Essa função recebe um**a** matriz de nodes com as suas linhas já iniciadas, em seguida aloca as suas colunas.

2.9. liberaNo:

Recebe uma matriz de nos, junto ao seu tamanha, em seguida libera a memoria alocada.

2.10. isWalkable:

recebe uma matriz de caracteres, junto a uma posição em seguida verifica se este caractere corresponde a uma posição bloqueada, retornando 0 caso bloqueado, se não, 1.

2.11. encontraMenorCaminho:



Essa função é baseada no algoritmo de busca "A estrela", o qual na teoria anda com base no formato de uma estrela, porém foi adaptado para caminhar somente em linhas horizontais e verticais, respeitando a premissa que o robô anda sempre em linha reta.

Resumidamente o algoritmo caminha baseado em uma função de "peso"(F), atribuído a cada nó analisado, tal calculo leva em consideração a distancia do seu destino(H) e a quantidade de nós percorridos acumulados(G), além de possuir uma lista a qual marcará se o nó em questão já foi percorrido ou não e como último passo escolhe o nó com menor peso F como o seu novo nó atual para ser analisado, respeitando a seguinte formula(F = G + H).

OBS: Cada nó possui um atributo chamado parente, o qual é responsável por armazenar o nó "pai" de cada nó percorrido, possibilitando que possamos retornar o caminho percorrido com facilidade.