**Relatório**

# O objetivo deste trabalho é de que por meio dos algoritmos de Busca em Largura, Busca em Profundidade e Busca Baseada em Heurística um a gente(soldado) percorra o labirinto abaixo, do ponto inicial marcado com coordenadas [4, 11], até o objetivo marcado com coordenadas [0, 11].

# Código QR

# **-**Resultados **e Métodos:**

Os métodos utilizados nesse trabalho para a parte da interface gráfica do labirinto foi a biblioteca “PyGame” e para os algoritmos de busca nenhuma biblioteca foi utilizada.

Os resultados obtidos pelos algoritmos foram os seguintes:

-Busca em Largura: 56 passos do ponto inicial até o objetivo.

-Busca em profundidade: 38 passos do ponto inicial até o objetivo.

-Busca por Fluxograma(Heurística): 30 passos do ponto inicial até o objetivo.

**-Algoritmos:**

Para a resolução do problema o labirinto foi feito por meio de uma lista de listas em que cada uma das listas representa uma linha do labirinto e os elementos pertencentes ela representa cada coluna possuindo valores 0 para representar as paredes/obstáculos e 1 para representar os caminhos sem paredes/obstáculos.

* Na implementação da Busca em Largura, são recebidos como parâmetros o estado inicial do agente, o estado objetivo e o labirinto representado como uma matriz. Inicialmente, são criadas as variáveis necessárias: uma fila para gerenciar os estados a serem explorados, uma lista para registrar as coordenadas visitadas, uma variável para contar o número de passos executados e o estado atual, que é inicializado com o valor do estado inicial. O estado inicial também é adicionado à fila para iniciar o processo de busca. O algoritmo então executa um laço while enquanto o estado atual não for igual ao objetivo. Dentro desse laço, o número de passos é incrementado, e as coordenadas do próximo elemento na fila são adicionadas à lista de visitados. Em seguida, o algoritmo verifica os possíveis caminhos a partir da coordenada atual utilizando a função “direçõesPossives”. Essa função avalia se cada uma das direções (cima, esquerda, direita e baixo) está dentro dos limites do labirinto, garantindo que as movimentações não ultrapassem os limites da matriz. Para cada direção válida, as funções específicas de movimentação (cima, esquerda, direita, embaixo) são chamadas. Essas funções verificam se a coordenada da próxima célula possui valor 1 e se ainda não foi visitada. Caso ambas as condições sejam atendidas, a nova coordenada é adicionada à fila. Após avaliar todas as direções possíveis, o primeiro elemento da fila (que já foi processado) é removido, e o próximo estado na fila é definido como o estado atual. Esse processo continua até que o estado objetivo seja alcançado ou não haja mais estados na fila e por fim é retornado uma lista com os estados visitados pelo a gente e os passos dados.
* A implementação da Busca em Profundidade recebe como parâmetros o estado inicial do agente, o estado objetivo e o labirinto representado por uma matriz. Inicialmente, são definidas as variáveis essenciais: uma pilha para armazenar os estados a serem explorados, uma lista para registrar as coordenadas visitadas, um contador para monitorar o número de iterações e o estado atual, que é atualizado dinamicamente durante a execução. O estado inicial é adicionado à pilha para dar início à busca. O algoritmo executa um laço while enquanto o estado atual não for igual ao estado objetivo. A cada iteração, o contador de passos é incrementado, o estado atual é definido como o topo da pilha (último elemento), e essa coordenada é adicionada à lista de visitados. Em seguida, o estado atual é removido da pilha. Para determinar os próximos caminhos possíveis, a função direçõesPossiveis é utilizada, verificando quais direções (cima, esquerda, direita e baixo) estão dentro dos limites do labirinto. Em cada direção válida, as funções específicas de movimentação (direçãoCima, direçãoEsquerda, direçãoDireita, direçãoBaixo) são chamadas para verificar se a próxima célula é transitável (valor igual a 1) e ainda não foi visitada. Caso essas condições sejam atendidas, a nova coordenada é adicionada à pilha. O algoritmo continua explorando os caminhos de forma recursiva, priorizando sempre os estados mais recentes adicionados à pilha, o que caracteriza o comportamento de busca em profundidade. Esse processo se repete até que o estado objetivo seja alcançado ou todos os caminhos possíveis tenham sido explorados. No final, a função retorna a lista de coordenadas visitadas e o número total de iterações realizadas durante a busca.
* A implementação da busca baseada em fluxograma (heurística) utiliza como parâmetros o estado inicial, o objetivo e o labirinto. Inicialmente, é criada uma lista para armazenar as coordenadas visitadas e uma variável passos para contabilizar a quantidade de movimentos realizados pelo agente desde o ponto inicial até o objetivo. Além disso, são definidas as variáveis distancias e direções, que armazenarão, respectivamente, uma matriz representando as distâncias do objetivo e as direções que o agente deve seguir. Esses valores são calculados pela função fluxo, que recebe o labirinto e a posição do objetivo como parâmetros. A função fluxo constrói as listas de custos (distancias) e direções (direções). Usando uma fila, a função processa todas as coordenadas do labirinto com valor igual a 1, calculando os custos com base na quantidade de passos necessários para alcançar o objetivo. A cada movimento, o custo da coordenada atual é incrementado em 1. Se o custo calculado para uma coordenada for menor do que o custo registrado, a coordenada é atualizada na lista de distâncias, a direção correspondente é armazenada na lista de direções, e a próxima coordenada é adicionada à fila. Com as matrizes de distâncias e direções prontas, o agente executa a busca utilizando um laço while. Ele segue o caminho indicado pelas direções armazenadas, atualizando seu estado atual para a próxima coordenada a cada iteração. Durante o percurso, o agente registra as coordenadas visitadas e incrementa a variável passos. O processo continua até que o estado atual coincida com o objetivo ou até que não existam mais direções válidas para seguir. Ao final, a função retorna a lista de coordenadas visitadas e o número total de passos necessários para atingir o objetivo.

**-Funcionalidade Extra:**

Como funcionalidade extra foi feito somente a interface gráfica em que é apresentado na tela o labirinto e as opções de algoritmos de busca ao usuário. Ao usuário escolher um dos algoritmos, o agente começa a se movimentar pelo labirinto conforma a lista das coordenadas visitadas retornada pela função que ele escolheu, e abaixo das opções de busca é mostrado a quantidade de passos necessários para que o agente chegue ao objetivo.

Código QR

Descrição gerada automaticamente

**-Desempenho:**

**Busca em Largura:**

No algoritmo de busca em largura, observa-se principalmente que, ao alcançar a coordenada (linha 9, coluna 7), surgem diversas possibilidades de caminhos a serem explorados. Como esse algoritmo possui um funcionamento por meio de fila, ele analisa todas as opções disponíveis antes de prosseguir na direção do objetivo. Esse comportamento resulta em um número significativo de passos adicionais, aumentando consideravelmente a quantidade de movimentos necessários para alcançar o destino.

**Busca em profundidade:**

Na busca em profundidade, o agente apresenta um desempenho mais eficiente devido ao seu funcionamento e à ordem predefinida para a escolha dos caminhos. Como o algoritmo explora um caminho até o final antes de retroceder e testar outras opções. Neste labirinto ele consegue encontrar o caminho correto necessitando de menos passos. Evitando a análise de múltiplos caminhos, como ocorre na busca em largura.

**Busca por fluxograma:**

Na busca utilizando a heurística do fluxograma, obtém-se o melhor desempenho devido à análise prévia do labirinto, em que é estabelecido as distâncias de cada coordenada até o objetivo. Com essas informações, o agente é capaz de escolher de forma estratégica o caminho mais curto até o destino, seguindo o fluxo previamente determinado.

**Comparação entre as buscas:**

Entre as buscas apresentadas, observa-se que a busca em largura apresenta o pior desempenho, considerando o objetivo e o labirinto estabelecidos. Em razão do agente explorar múltiplos caminhos possíveis antes de encontrar o correto, resultando em um grande número de passos extras. A busca em profundidade, por outro lado, mostra uma performance melhor neste labirinto, pois segue um caminho até o final antes de retroceder para explorar outras alternativas. No entanto, ainda percorre trajetos incorretos antes de alcançar o objetivo, o que compromete sua eficiência em razão da ordem de escolha dos caminhos. Por fim, a busca com heurística baseada no fluxograma demonstra o melhor desempenho. Isso se deve à análise prévia que define os custos de cada coordenada até o objetivo, permitindo que o agente siga, desde o início, o caminho mais eficiente e direto no labirinto. Entretanto, devido à análise prévia necessária para calcular os custos de cada coordenada até o objetivo, o algoritmo é mais caro computacionalmente. E em razão disso torna sua aplicação menos eficiente para labirintos maiores ou mais complexos, onde o esforço computacional seria significativo.