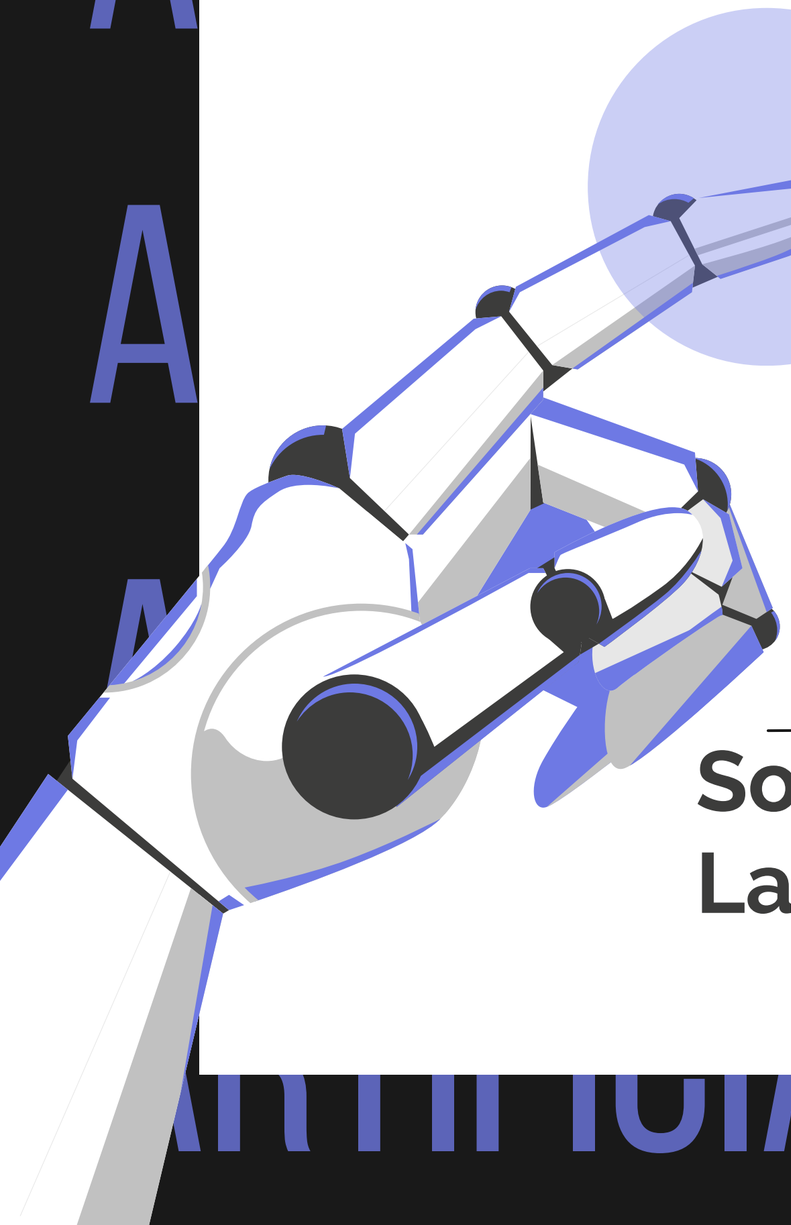


INTELIGÊNCIA ARTIFICIAL

Solução do Problema do Labirinto

Grupo: Mateus Nunes Lehmkuhl, Andrey Justen Júnior, Bárbara Prim de Souza, Leonardo Alves Silva, Kennedy da Silva Motta

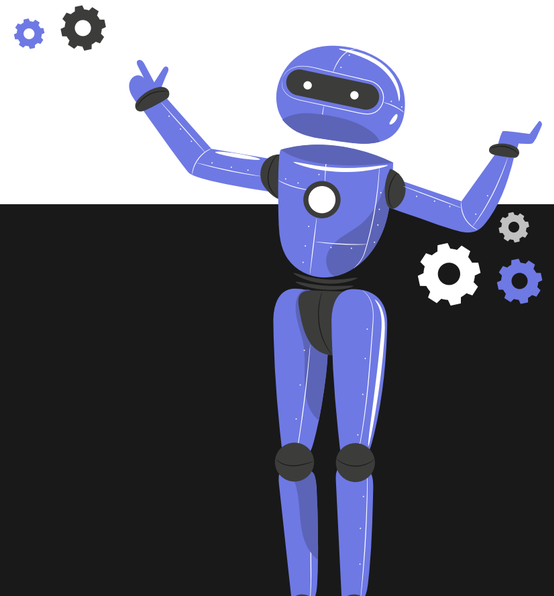
INTRODUÇÃO

Esta apresentação aborda a solução para o desafio de navegar um robô por um labirinto, partindo de um ponto inicial até um ponto final. Durante o percurso, o robô enfrenta obstáculos e precisa gerenciar sua energia. Para resolver o problema, utilizamos dois algoritmos de busca, a Busca em Largura e o A*, cada um com suas características e estratégias.

OBJETIVOS DO SISTEMA

/CAD/CAD/

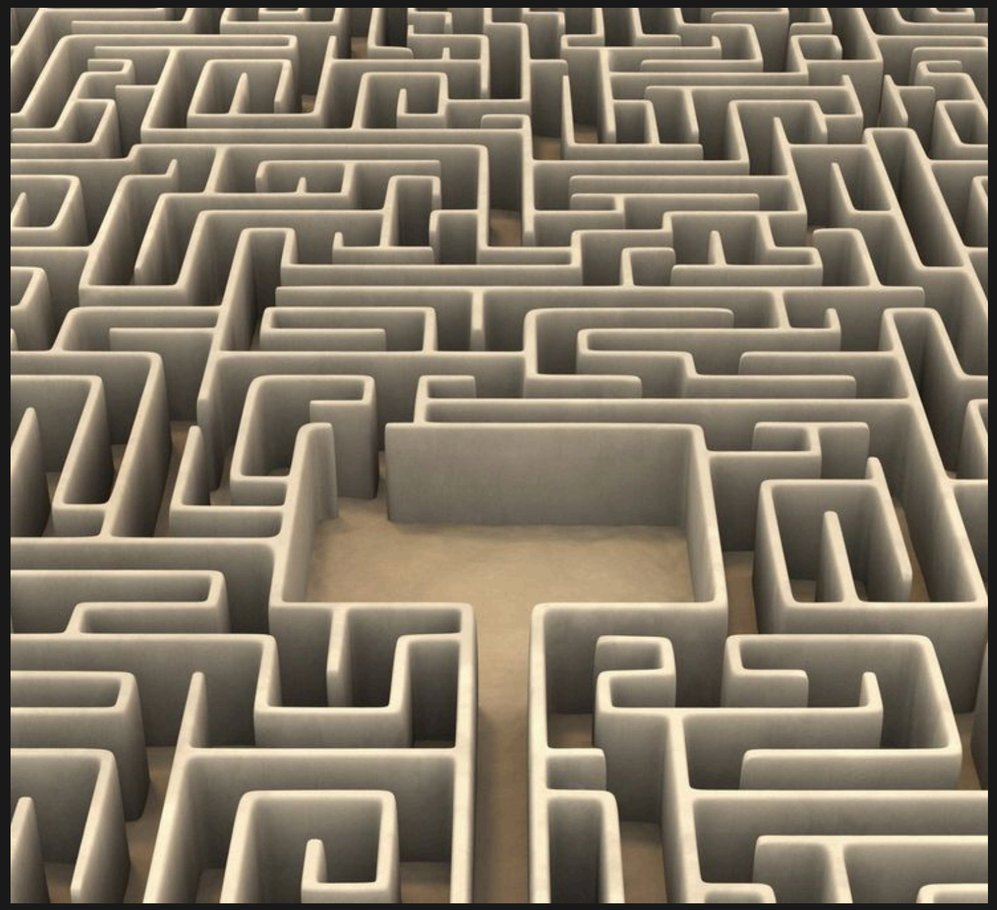
- Implementar os algoritmos de busca em largura e A* para encontrar caminhos através do labirinto.
- Gerar labirintos aleatórios com posições dos seus itens aleatórias (energias e obstáculos).
- Gerenciar o consumo e recuperação de energia do robô.
- Mostrar visualmente o labirinto e o caminho percorrido pelos algoritmos.
- Flexibilidade para testes.
- Produzir um relatório com tempos de execução.



REPRESENTAÇÃO DO LABIRINTO

O labirinto é representado como uma matriz de 10x10, onde:

0 célula (passável).
-1 (obstáculo).
5 e 10 (células de recarga de energia.)
1 e 2 (posição inicial e final.)



ENERGIA E MOVIMENTOS DO ROBÔ

- O robô inicia com 50 pontos de energia.
- Cada movimento reduz a energia em 1 ponto.
- Ao passar por células de recarga, a energia é aumentada conforme o valor da célula (limitada ao máximo de 50).

Movimentos Válidos

O robô pode se mover em quatro direções:

- Para cima: (i-1, j)
- Para baixo: (i+1, j)
- Para a direita: (i, j+1)
- Para a esquerda: (i, j-1)

BUSCA EM LARGURA (BFS)

O que é BFS?

- Um algoritmo de busca não informada.
- Explora todas as possibilidades de movimento de forma sistemática, a partir do ponto inicial.

Como funciona o BFS?

- Utiliza uma fila (queue) para armazenar os estados a serem explorados.
- Expande os estados em ordem de chegada (primeiro a entrar,primeiro a sair - FIFO).
- Continua até:
- Encontrar a solução (nó objetivo).
- Ou esgotar as possibilidades.

Pontos Fortes

- Completude: Garante encontrar uma solução se houver um caminho até o objetivo.
- Simples: É intuitivo e fácil de implementar.

Limitações

- Não é ótimo: Pode não encontrar o caminho mais curto (ótimo em termos de custo de energia ou número de passos).
- Consumo de memória: Em labirintos grandes, a fila pode crescer rapidamente.

A*

O que é A*

- Um algoritmo de busca que utiliza uma função heurística para priorizar estados promissores.
- A heurística utilizada é a distância de Manhattan.
- Combina a heurística com o custo do caminho percorrido até o momento.

Como funciona o A*

- O algoritmo A* combina dois elementos: o custo do caminho já percorrido e uma estimativa da distância restante até o objetivo.

Limitações

- Não é ótimo: O A* pode não encontrar o caminho mais curto se a heurística não for bem projetada ou superestimar os custos, o que afeta a precisão.
- Consumo de memória: Em áreas grandes ou com muitos caminhos possíveis, o A* pode ocupar muita memória, já que armazena todos os caminhos em consideração.

Pontos Fortes

- Completude: O A* sempre vai achar o caminho para o objetivo, se for possível chegar lá. Ele não "se perde" ou "desiste" enquanto ainda existe uma chance de encontrar o destino.
- Simples: O A* funciona de forma lógica: ele calcula o custo de cada passo já feito e faz um bom palpite de qual caminho seguir em seguida, com base no objetivo. Isso o torna fácil de entender e usar, desde que você tenha uma boa ideia de como estimar o caminho mais curto.

DIFERENÇAS

BUSCA E LARGURA

- A Busca em Largura é garantida em encontrar uma solução, mas pode não ser a mais eficiente.
- a Busca em Largura é mais simples de implementar e não depende de estimativas.

A*

o A* utiliza informações adicionais para priorizar caminhos melhores, geralmente encontrando a solução mais curta.

- No entanto, o A* depende da qualidade da heurística para ser efetivo

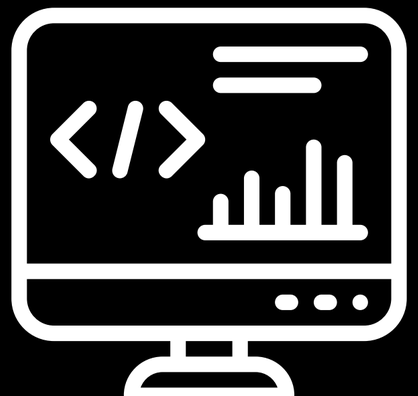
IMPLEMENTAÇÃO

BACKEND PYTHON COM FLASK

- Gerar labirintos aleatórios com configurações definidas pelo usuário.
- Resolver o labirinto utilizando BFS e A*.
- Retornar os resultados (labirinto e caminhos) para o frontend.

FRONTEND (HTML/CSS/JS)

- Gera todo o visual, tanto do labirinto, quanto botões e etc...



CONCLUSÃO

O SISTEMA ATENDE AOS REQUISITOS E PERMITE:

>>>>

01.

TESTAR DIFERENTES CONFIGURAÇÕES DE LABIRINTOS.

02.

VISUALIZAR E COMPARAR SOLUÇÕES ENCONTRADAS POR BFS E A*.

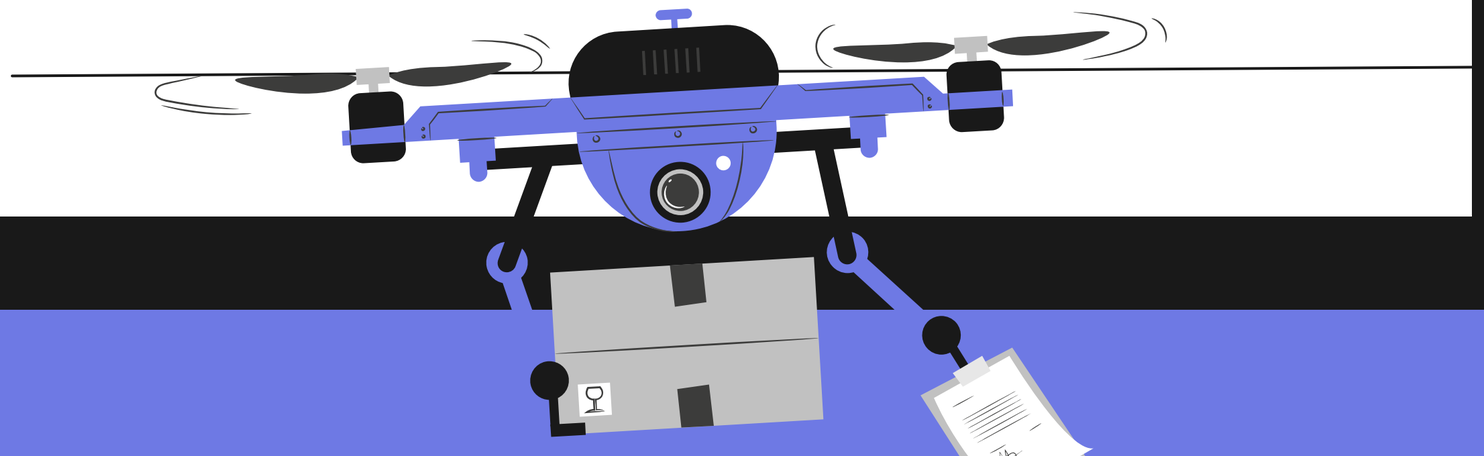
03.

DEMONSTRAR O IMPACTO DE OBSTÁCULOS E POSIÇÕES DE RECARGA NO DESEMPENHO DOS ALGORITMOS.

ARTIFICIAL INTELLIGENCE (AI)

CÓDIGO FONTE

https://github.com/AsLeonardo/AI-Maze-Solver-and-Generator



WEB SERVICE

Utilizando do repositório do Github e do Host do “Render”, o código python se encontra disponível em Web Service no seguinte link;

https://maze-solver-and-generator.onrender.com/

ARTIFICIAL INTEL (AI)

FIM.

(AI)