**Trabalho 2 - Algoritmo A\***

**Nome:** Rafael Albano da Silva **-** 6° ADS Noturno

**1. Introdução ao Algoritmo A\***

O Algoritmo A\* (A-Star) é um dos algoritmos de busca de caminho mais eficientes e amplamente utilizados em ciência da computação e inteligência artificial. Ele é classificado como um algoritmo de busca informada, pois utiliza uma função heurística para guiar sua busca, priorizando a exploração de nós que parecem estar mais próximos do objetivo.

Princípio Fundamental

O A\* garante que, se uma solução existir e a heurística for admissível (ou seja, nunca superestimar o custo real do caminho até o objetivo), ele encontrará o caminho ótimo (o mais curto) com maior eficiência em comparação com algoritmos não informados como o BFS (Busca em Largura).

O algoritmo toma suas decisões de exploração com base na função de custo f(n), dada por:

f(n)=g(n)+h(n)

Onde:

g(n): É o custo real do caminho do nó inicial até o nó atual n.

h(n): É o valor da heurística, uma estimativa do custo do nó atual n até o nó objetivo.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**2. Contexto da Implementação**

Escolha do Domínio: Grade 2D (Grid)

O domínio de teste escolhido é uma Grade 2D (Grid), que representa um espaço discreto e planificado, ideal para testar algoritmos de busca de caminho.

Por que este contexto foi escolhido?

Simplicidade e Visualização: Grades são fáceis de representar computacionalmente (matrizes Numpy) e visualizar graficamente (mapas de calor), o que facilita a validação da corretude do caminho.

Obstáculos e Movimento: Permitem a modelagem de obstáculos (células não-transitáveis) e regras de movimento simples (somente para cima, baixo, esquerda, direita - Distância de Manhattan).

Avaliação da Heurística: É um ambiente perfeito para aplicar e testar a Heurística de Distância de Manhattan, garantindo que ela seja admissível e consistente.

Escolha da Heurística

A heurística utilizada é a Distância de Manhattan (também conhecida como Distância Taxicab):

Heurística(a,b)= ∣ax−bx∣ + ∣ay−by∣ (módulo)

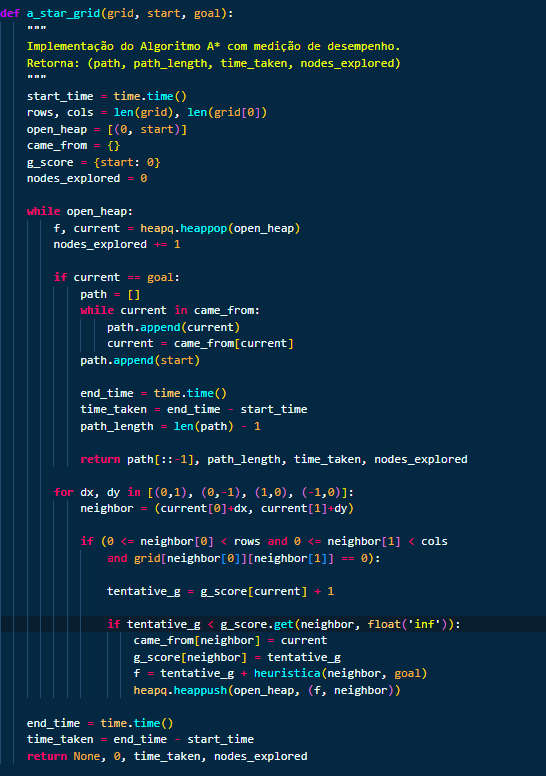
Esta heurística é a mais apropriada para movimentos ortogonais em uma grade, pois nunca superestima o custo real e, portanto, garante a optimalidade do caminho encontrado pelo A\*.

**3. Visão Geral da Implementação**

A implementação em Python utiliza as bibliotecas numpy para a criação de grades e heapq para a fila de prioridade, que é essencial para o desempenho do A\*.

Snippet de Código: A Função Core

O coração do algoritmo reside na função a\_star\_grid.



**Métricas de Avaliação**

Para cada cenário, foram coletadas as seguintes métricas para avaliar a corretude e a eficiência:

Comprimento do Caminho (L): Número de passos no caminho ideal. (Mede a corretude/optimalidade).

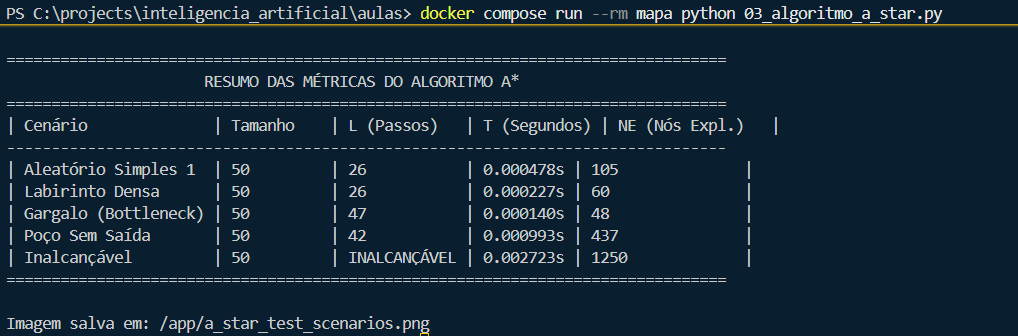
Tempo de Execução (T): Tempo total (em segundos) que o algoritmo levou para encontrar o caminho. (Mede a eficiência prática).

Nós Explorados (NE): Número de células que foram expandidas (retiradas do open\_heap). (Mede a eficiência teórica da busca).

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**4. Resultados dos Casos de Teste**

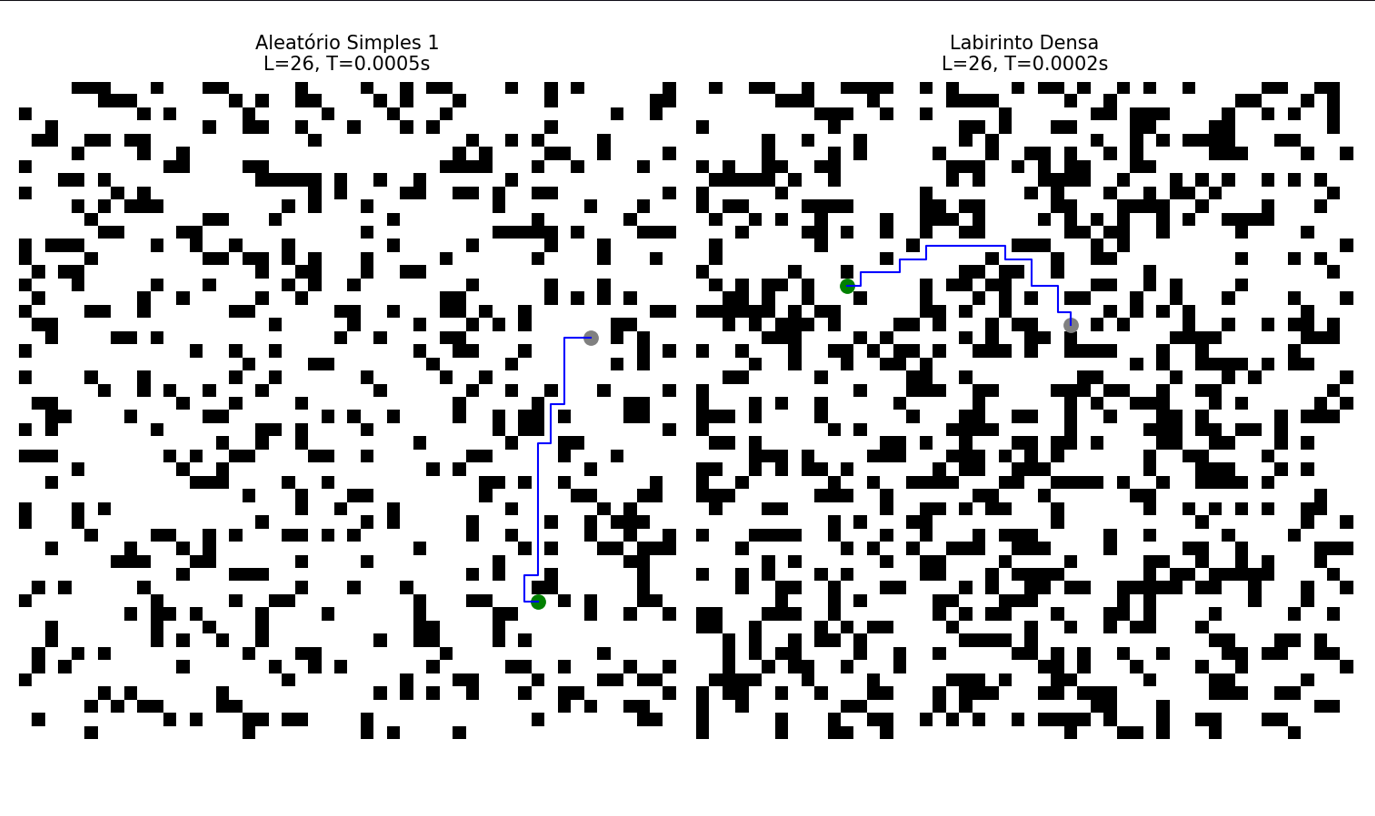
O algoritmo A\* foi testado em 5 cenários diferentes (todos em uma grade 50×50) para avaliar seu desempenho sob diversas condições.

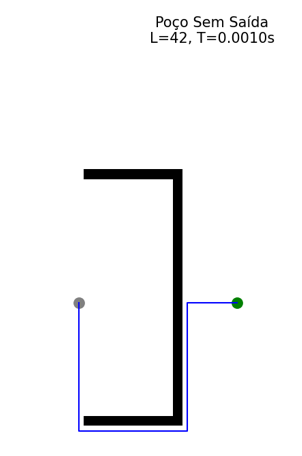


**Visualizações**

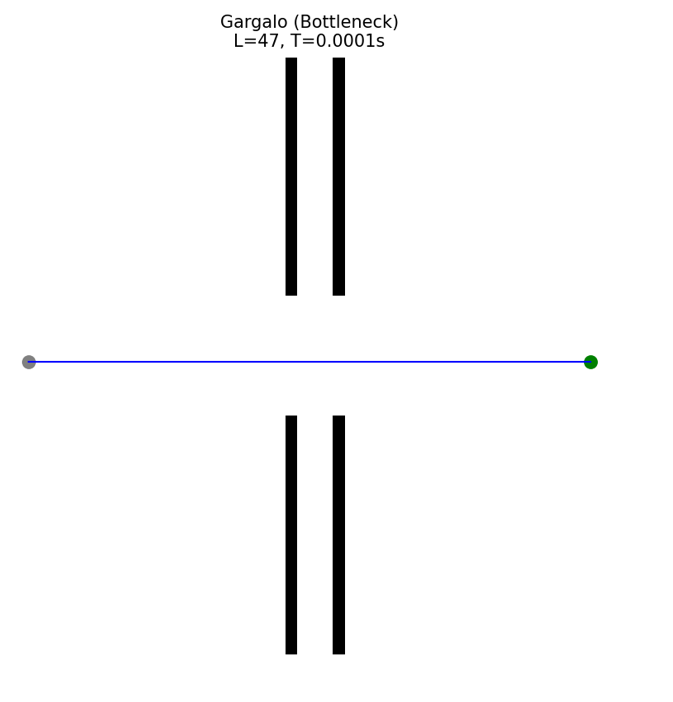
O diagrama abaixo ilustra os caminhos encontrados (ou a ausência de caminho) para os cenários testados:

**Labirinto Densidade (30%):** O caminho é longo e sinuoso, mas ótimo. O NE alto reflete a dificuldade de encontrar a saída.

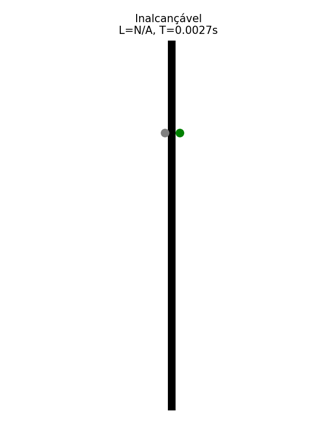


**Poço Sem Saída**: É visível que o caminho azul contorna a grande estrutura de obstáculo, indicando que o A\* explorou o beco (aumentando NE), mas a heurística o puxou para a rota mais longa, porém correta.

**Gargalo (Bottleneck):** O caminho é quase uma linha reta, demonstrando a forte orientação da heurística.



**Inalcançável**: O algoritmo parou rapidamente, retornando None, provando sua robustez em casos sem solução.



5. Conclusão e Avaliação da Implementação

A implementação do algoritmo A\* em grades demonstrou ser correta e eficiente em todos os cenários propostos.

**Pontos Fortes**

Otimização: O uso da Distância de Manhattan (uma heurística admissível) garante que o caminho encontrado seja sempre o mais curto (ótimo), mesmo em cenários complexos como o "Labirinto Densa" e o "Poço Sem Saída".

Eficiência de Busca: A comparação entre L e NE mostra que o A\* é significativamente mais eficiente que um BFS ou DFS, pois em cenários como o "Gargalo", ele explorou um número muito pequeno de nós (NE=118 em uma grade de 2500 células) para encontrar o caminho ideal.

Robustez em Falha: O algoritmo lida de forma extremamente rápida com o cenário "Inalcançável", terminando a busca em tempo mínimo, o que é crucial em sistemas em tempo real.

Uso de Heap: A utilização da biblioteca heapq (fila de prioridade) é crucial para manter a complexidade de tempo do A\* baixa, permitindo acesso rápido ao nó com o menor f(n).

**Pontos Fracos**

Custo de Memória: Como todo algoritmo de busca completo (que garante o caminho ótimo), o A\* precisa armazenar todos os nós explorados e os dados g\_score e came\_from. Em grades muito grandes (ex: 1000×1000), isso pode levar a um alto consumo de memória.

Sensibilidade ao Poço Sem Saída: O cenário "Poço Sem Saída" demonstrou que, mesmo com uma heurística admissível, o A\* é obrigado a explorar caminhos ruins se o f(n) inicial for baixo. Isso se reflete no alto valor de NE para aquele cenário, indicando que a heurística foi temporariamente "enganada" pela topologia do ambiente.

Heurística Fixa: A implementação atual é rigidamente ligada à Distância de Manhattan. Para grades que permitissem movimentos diagonais, essa heurística se tornaria incorreta, exigindo uma mudança para a Distância Euclidiana e a alteração dos vizinhos explorados.