Melhor Rota com o A*

Apresentado por:

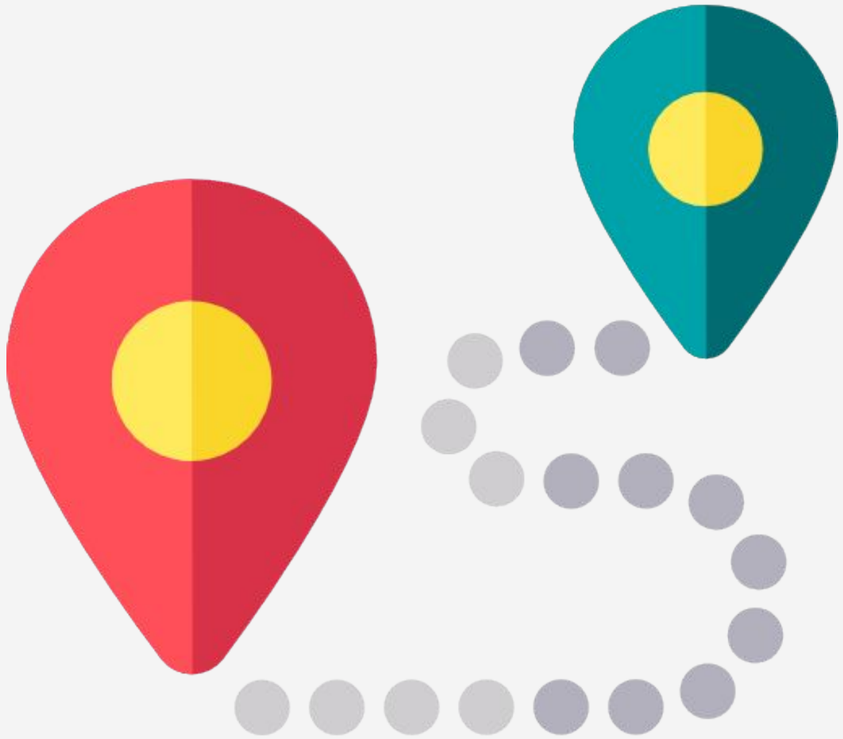
- Júlia de Souza Lima
- Magda Tainy
- Marcos Machado

Docente:

- Dr. Alcides Xavier Benicasa

Agenda

- Introdução
- Algoritmo A*
- O projeto
- Demonstração
- Conclusão



Introdução

O problema escolhido pela equipe foi encontrar a melhor rota para um percurso, seguindo o critério de menor caminho percorrido, utilizando do algoritmo A*.

O algoritmo A*

É um método que busca o caminho mais curto entre dois pontos. Esse método usa os valores de duas funções para estimar o melhor caminho em relação ao custo, cujos valores são o G e o H. A soma dos dois valores acima estimam o caminho mais eficiente de um ponto inicial até o ponto final.

Combina, de certa forma, as vantagens tanto do custo uniforme, como do algoritmo de busca gulosa, pois utiliza da fórmula

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

para seus cálculos.

O algoritmo A*

É uma busca ótima

Pode ter um tempo curto de execução

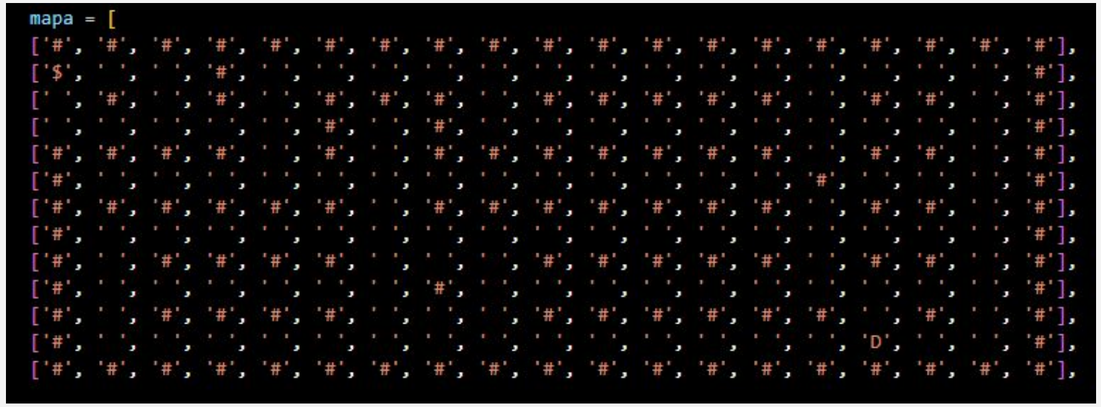
Escolhe o menor caminho

O projeto

Problema	Critério	Solução utilizada
Melhor rota de um mapa	Menor percurso	A*

O projeto

Foi definido um trajeto, com o ponto de partida e o objetivo desejado, por meio de uma matriz.



```
mapa = [
['#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#'],
['$', ' ', ' ', ' ', '#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#'],
[' ', '#', ' ', '#', ' ', '#', '#', '#', ' ', '#', '#', '#', '#', '#', ' ', '#', '#', ' ', '#'],
[' ', ' ', ' ', ' ', ' ', ' ', '#', ' ', '#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#'],
['#', '#', '#', '#', ' ', '#', ' ', '#', '#', '#', '#', '#', '#', '#', ' ', '#', '#', ' ', '#'],
['#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#'],
['#', '#', '#', '#', '#', '#', ' ', '#', '#', '#', '#', '#', '#', '#', ' ', '#', '#', ' ', '#'],
['#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#'],
['#', ' ', '#', '#', '#', '#', ' ', ' ', ' ', '#', '#', '#', '#', '#', ' ', '#', '#', ' ', '#'],
['#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '#'],
['#', ' ', '#', '#', '#', '#', ' ', ' ', ' ', '#', '#', '#', '#', '#', '#', ' ', '#', ' ', '#'],
['#', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', 'D', ' ', ' ', ' ', '#'],
['#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#'],
```

A função calcula_heuristica realiza o cálculo do menor caminho, comparando a distância da posição atual com a distância do vizinho, em relação ao objetivo.

```
def calcular_heuristica(pos_atual, pos_objetivo):
    return abs(pos_atual[0] - pos_objetivo[0]) + abs(pos_atual[1] - pos_objetivo[1])
```

O projeto

A função dentro_limites(pos) verifica se as coordenadas da posição estão dentro dos limites do mapa.

```
def dentro_limites(pos):
    return 0 <= pos[0] < altura_mapa and 0 <= pos[1] < largura_mapa
```

```
def obter_vizinhos(pos):
    vizinhos = [(pos[0]-1, pos[1]), (pos[0]+1, pos[1]),
                (pos[0], pos[1]-1), (pos[0], pos[1]+1)]
    return [vizinho for vizinho in vizinhos if dentro_limites

custos = [[float('inf')] *
          largura_mapa for _ in range(altura_mapa)]
```

A função obter_vizinhos retorna os vizinhos válidos de uma determinada posição no mapa.

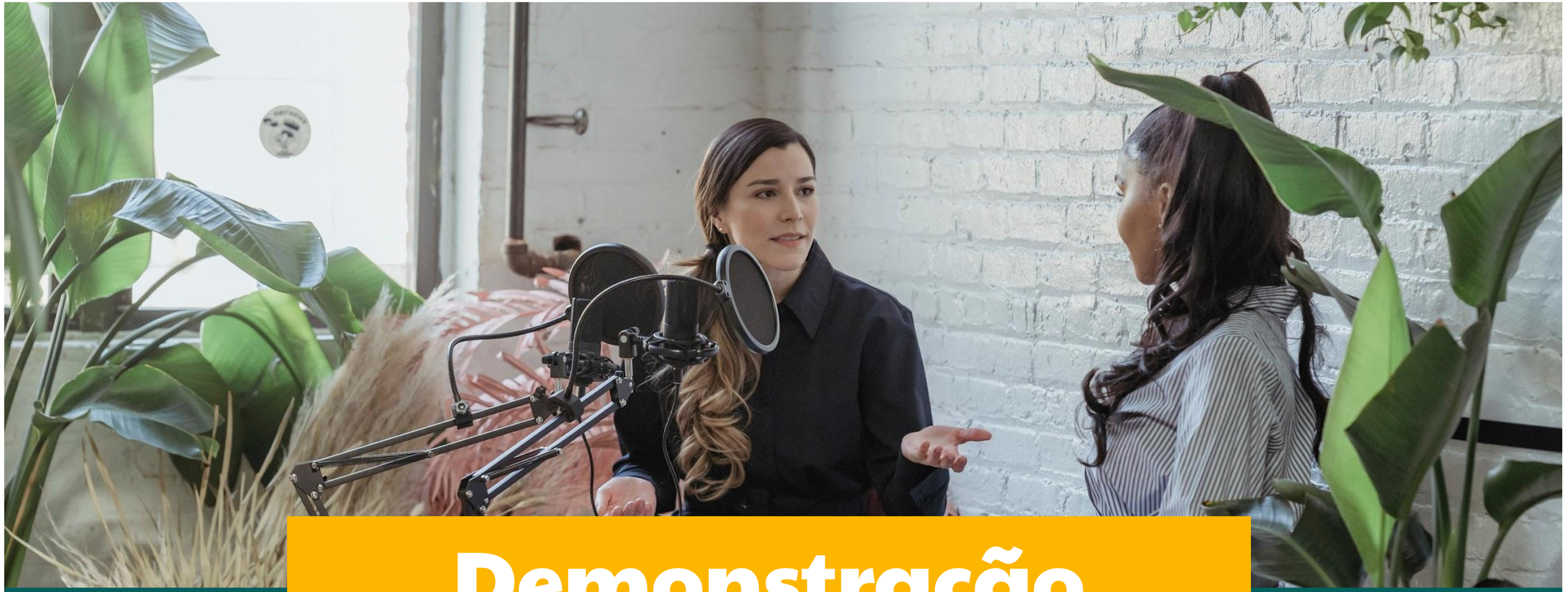
O projeto

```
custos = [[float('inf')] *
    |   |   | largura_mapa for _ in range(altura_mapa)]
custos[inicio[0]][inicio[1]] = 0
pais = [[None] * largura_mapa for _ in range(altura_mapa)]
pais[inicio[0]][inicio[1]] = inicio

fila_prioridade = queue.PriorityQueue()
fila_prioridade.put((0, inicio))
```

A matriz custos armazena os custos acumulados para alcançar cada posição no mapa.

```
fila_prioridade = queue.PriorityQueue()
fila_prioridade.put((0, inicio))

while not fila_prioridade.empty():
    _, pos_atual = fila_prioridade.get()

    if pos_atual == objetivo:
        caminho = []
        pos = objetivo
        while pos != inicio:
            caminho.append(pos)
            pos = pais[pos[0]][pos[1]]
        caminho.append(inicio)
        caminho.reverse()
        return caminho

    for vizinho in obter_vizinhos(pos_atual):
        novo_custo = custos[pos_atual[0]][pos_atual[1]] + 1

        if novo_custo < custos[vizinho[0]][vizinho[1]]:
            custos[vizinho[0]][vizinho[1]] = novo_custo
            prioridade = novo_custo + \        You, há 6 dias
                calcular_heuristica(vizinho, objetivo)
            fila_prioridade.put((prioridade, vizinho))
            pais[vizinho[0]][vizinho[1]] = pos_atual

return None
```

- Loop principal
- Exploração dos vizinhos
- Verificação do objetivo



Demonstração

Conclusão

Obrigado(a)!