

**INTRODUCTION TO AI (AI101B)**

**AI based pathfinding with A\* algorithm**

**Name of Student: Dhruv kesarwani**

**Branch: CSEAI**

**Lib. ID: 2428cseai1245**

**Section: B**

**Session: 2024-2028**

**Roll no: 202401100300101**

**Submitted to: Mr. Shivansh Prasad**

Introduction:

**The A\* (A-star) algorithm is a widely used AI-based pathfinding method that efficiently finds the shortest path between two points. It combines Dijkstra’s algorithm and Greedy Best-First Search, using a cost function that balances actual distance traveled and an estimated distance to the goal. Due to its accuracy and efficiency, A\* is commonly used in robotics, gaming, and navigation systems. This report explores its working principles, implementation, and applications.**

Methodology:

1. **Problem Definition – Represent the environment as a graph/grid with nodes, edges, and movement costs.**
2. **Algorithm Implementation – Use an open list (nodes to explore) and a closed list (explored nodes). Compute f(n) = g(n) + h(n), expand the lowest-cost node, and update neighbors.**
3. **Heuristic Function Selection – Choose a heuristic like Manhattan, Euclidean, or Diagonal Distance based on the environment.**
4. **Path Reconstruction – Trace back from the goal node to the start using parent pointers.**
5. **Performance Evaluation – Test in different environments, comparing efficiency with other algorithms.**

**def a\_star(grid, start, end):**

**rows, cols = len(grid), len(grid[0])**

**open\_list = [start]**

**came\_from = {}**

**g\_score = {start: 0}**

**f\_score = {start: abs(start[0] - end[0]) + abs(start[1] - end[1])}**

**while open\_list:**

**open\_list.sort(key=lambda pos: f\_score.get(pos, float('inf')))**

**current = open\_list.pop(0)**

**if current == end:**

**path = []**

**while current in came\_from:**

**path.append(current)**

**current = came\_from[current]**

**path.append(start)**

**return path[::-1]**

**neighbors = [(current[0] + d[0], current[1] + d[1]) for d in [(0, 1), (0, -1), (1, 0), (-1, 0)]]**

**neighbors = [pos for pos in neighbors if 0 <= pos[0] < rows and 0 <= pos[1] < cols and grid[pos[0]][pos[1]] == 0]**

**for neighbor in neighbors:**

**temp\_g\_score = g\_score[current] + 1**

**if temp\_g\_score < g\_score.get(neighbor, float('inf')):**

**came\_from[neighbor] = current**

**g\_score[neighbor] = temp\_g\_score**

**f\_score[neighbor] = temp\_g\_score + abs(neighbor[0] - end[0]) + abs(neighbor[1] - end[1])**

**if neighbor not in open\_list:**

**open\_list.append(neighbor)**

**return None  # No path found**

**# Example usage**

**grid = [**

**[0, 1, 0, 0, 0],**

**[0, 1, 0, 1, 0],**

**[0, 0, 0, 1, 0],**

**[1, 1, 0, 1, 0],**

**[0, 0, 0, 0, 0]**

**]**

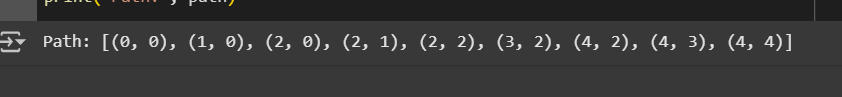
**start = (0, 0)**

**end = (4, 4)**

**path = a\_star(grid, start, end)**

**print("Path:", path)**

**#output snippet:**

****