Project Title: Comparative Analysis of Search Strategies for Maze Pathfinding.

Course: Artificial Intelligence (CS,IS Level 3)

1. Problem Definition

Selected Problem: Maze Pathfinding Navigation.

Description:

The project aims to develop an intelligent agent simulation capable of navigating through a grid-based environment (N \times N) containing static obstacles (walls). The primary objective is to find a valid path from a starting coordinate (Start) to a target coordinate (Goal).

Motivation:

This problem serves as a fundamental model for understanding state-space search in Artificial Intelligence. It simulates real-world challenges found in:

Robotics path planning (e.g., warehouse robots).

Autonomous vehicle navigation.

Game development (NPC movement).

2. Methodology & Algorithms

We will implement and analyze 5 distinct search algorithms, categorizing them into Uninformed (Blind) and Informed

(Heuristic) strategies to compare their efficiency in solving the maze:

2.1 Uninformed Search Strategies (Blind Search):

2.1.1 Breadth-First Search (BFS):

Explores the search space level by level.

Key Feature: Guarantees finding the shortest path (optimal solution) in an unweighted grid.

2.1.2 Depth-First Search (DFS):

Explores as deep as possible along each branch before backtracking.

Key Feature: Memory efficient but does not guarantee the shortest path and may get stuck in deep loops.

2.1.3 Uniform-Cost Search (UCS):

Expands the node with the lowest cumulative path cost g(n).

Key Feature: Ensures optimality based on path cost, behaving similarly to BFS in uniform-cost environments.

2.1.4 Iterative Deepening Search (IDS):

A hybrid approach that combines the space efficiency of DFS with the completeness of BFS by repeatedly running DFS with increasing depth limits.

Key Feature: Finds the optimal solution with linear memory complexity (O(bd)).

2.2 Informed Search Strategies (Heuristic Search):

2.2.1 A Search (A-Star):

Uses a combined function f(n) = g(n) + h(n) to direct the search towards the goal.

Heuristic Used: Manhattan Distance (|x_1 - x_2| + |y_1 - y_2|).

Key Feature: The most efficient algorithm among the selected set, balancing optimality and speed.

3. Implementation Tools

Programming Language: Python.

GUI Framework: tkinter (Standard Python Library) is used to visualize the maze generation, obstacles, and the agent's movement in real-time.

Data Structures: The implementation utilizes Deque (for BFS), Stack (for DFS), and Heap Priority Queue (for UCS and A*) to manage the frontier.

4. Evaluation Metrics

We will conduct a comparative analysis of the algorithms based on the following performance indicators:

Time Complexity: The execution time required to find a solution (measured in milliseconds).

Space Complexity: The total number of nodes expanded (visited) during the search process.

Optimality: Whether the algorithm successfully finds the shortest possible path.

Completeness: The ability of the algorithm to find a solution whenever one exists.