Artificial Intelligence (Lab)

Lab 02: Introduction to search in AI: Problem spaces, states, and goals

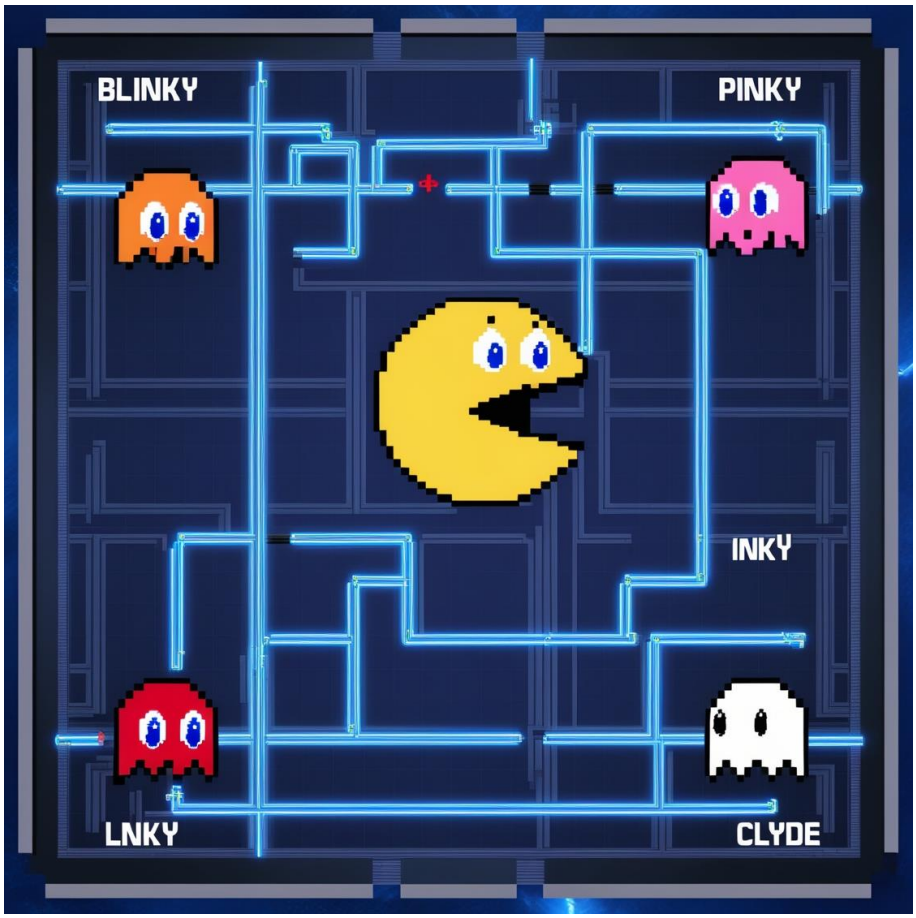
SAAD ALI

2022512

CYBER SECURITY

This directory contains an implementation of the A* search algorithm as part of a Pac-Man simulation game. The game involves Pac-Man navigating through a grid with obstacles, trying to reach a designated finish point, while ghosts chase him using the same A* pathfinding technique. The A* algorithm ensures optimal pathfinding by combining the cost from the start to a node (g-score) and the heuristic estimate to the goal (f-score), making it efficient for finding the shortest path even in the presence of obstacles and enemies.

A* Algorithm in Pac-Man Game:



How A* Algorithm Works:

A* is an informed search algorithm that finds the shortest path from a start node to a goal node, combining aspects of both greedy search and Dijkstra's algorithm.

The key formula used by A* is:

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

g(n): The cost from the start node to the current node n.

h(n): The heuristic estimate from node n to the goal node, often using Manhattan distance in a grid (since Pac-Man moves up, down, left, or right).

f(n): The total cost of the path, which combines g(n) and h(n) to decide which node to explore next.

Why A* Is Called "Informed" Search:

A* is called an informed search algorithm because it uses heuristic information to make better choices. Unlike uninformed search algorithms (like BFS or DFS), A* doesn't just explore all possible paths blindly. Instead, it uses a guess (h(n)) to focus on the most promising paths that are likely to lead to the goal faster.

A* Implementation in the Code:

1.Priority Queue (open_set): A priority queue (min-heap) is used to explore nodes with the smallest f(n) values first.

Initially, the starting node (ghost's position) is added with a cost of 0.

2. G-Score and F-Score: g_score keeps track of the actual cost of reaching each node from the start. f_score is the total cost, which includes the heuristic value (h(n)).

3. Direction Exploration: The algorithm checks possible movements (up, down, left, right) from the current position.

It ensures that the next position is within bounds and not an obstacle.

4. Path Reconstruction: Once the goal (Pac-Man's position) is reached, the algorithm traces back from the goal to the start, reconstructing the path.

The path is then returned and visualized in the grid.

Key Points of the Code:

1. Grid Creation (create_pacman_grid):

The function initializes a grid of a specified size with empty spaces (' '). Pac-Man is placed at the top-left corner ((0, 0)).

Ghosts are placed at random positions on the grid, ensuring they don't overlap with Pac-Man.

2. Obstacle Placement (add_obstacles):

Obstacles ('X') are randomly placed in the grid, ensuring they don't overwrite Pac-Man or ghosts.

This adds complexity to the paths that both Pac-Man and ghosts have to navigate.

3. *A Algorithm (a_star)**:

This function implements the A* pathfinding algorithm to allow the ghosts to chase Pac-Man.

It calculates the shortest path from the ghosts to Pac-Man, avoiding obstacles.

4. Path Visualization (print_grid_with_path):

This function prints the grid with the path marked using '*', showing how the ghosts will approach Pac-Man based on the calculated path.

Pac-Man ('P') and ghosts ('G') remain fixed in their positions.

5. Game Simulation (pacman_game):

This is the main function that runs the simulation.

The grid is created, obstacles are added, and A* is used to move ghosts toward Pac-Man.

The grid is printed to show both the initial state and how the ghosts attempt to chase Pac-Man.

Conclusion:

The A* algorithm is a powerful, informed search method that combines the best features of Dijkstra's algorithm (minimizing cost so far) and greedy best-first search (using heuristics). It ensures that the path it finds is both efficient and optimal, making it a great choice for solving problems where you need to find the best path while considering both the cost so far and the estimated cost to the goal. In summary: g(n): Actual cost so far from the start node to the current node. h(n): Heuristic estimate of the remaining cost to the goal. f(n) = g(n) + h(n): The total estimated cost for the node. A* picks the node with the smallest

f(n) to explore next. This makes A* a highly effective algorithm for finding the shortest path in a wide range of scenarios.

rs