Pacman Search Algorithms Assignment Report

Student Name: Cecil Kioko – SCT212-0047/2023
Course: Artificial Intelligence
Assignment: Search Algorithms in Pacman

---

1. Introduction

The goal of this assignment is to implement and analyze several classical **search algorithms** in the context of the Pacman game. The algorithms studied include **Depth-First Search (DFS)**, **Breadth-First Search (BFS)**, **Uniform Cost Search (UCS)**, and **A*** search. Additionally, advanced search problems such as **Corners Problem** and **Food Search Problem** were solved using heuristic-based A* agents.

This project focuses on:

- Exploring different search strategies in various maze configurations.
- Measuring algorithm efficiency in terms of **nodes expanded**, **path cost**, and **game score**.
- Understanding the role of **heuristics** in improving search performance.

---

2. Methodology

2.1 Search Algorithms

- **Depth-First Search (DFS)**: Explores the deepest unvisited nodes first. May not find the shortest path but is memory-efficient.
- **Breadth-First Search (BFS)**: Explores all neighbors level by level. Guarantees the shortest path.
- **Uniform Cost Search (UCS)**: Expands the node with the lowest cumulative cost, ensuring optimality.
- **A***: Uses a heuristic function to estimate remaining cost and guides search toward the goal efficiently.

2.2 State Representation

- **PositionSearchProblem**: Tracks Pacman's position in the maze.
- **CornersProblem**: Pacman must visit all four corners.

- **FoodSearchProblem**: Pacman must collect all food dots in the maze.

2.3 Heuristics

- **Manhattan Distance Heuristic**: Estimates the cost from the current position to the goal using the sum of horizontal and vertical distances.
- **Corners Heuristic**: Estimates the shortest path covering all unvisited corners.
- **Food Heuristic**: Uses maximum distance to the remaining food dots to guide A* search.

2.4 Optimizations

- Implemented caching for repeated state evaluations.
- Optimized the order of node expansion to reduce computation time.
- Avoided unnecessary recomputation in heuristic functions.

---

3. Results

Maze / Problem	Agent / Algorithm	Path Cost	Score	Nodes Expanded	Result
tinyMaze	DFS	10	500	15	Win
mediumMaze	BFS	68	442	269	Win
mediumMaze	UCS	68	442	269	Win
bigMaze	A* (Manhattan)	210	300	549	Win
mediumCorners	A*Corners	106	434	1136	Win
trickySearch	A*Food	60	570	4137	Win

---

4. Discussion

1. **DFS** performed efficiently on small mazes but may fail to find the shortest path in larger mazes.
2. **BFS** guaranteed optimal paths in medium mazes but expanded more nodes than DFS.
3. **UCS** matched BFS in performance for uniform-cost paths.
4. **A*** with heuristics significantly reduced node expansions in complex problems like bigMaze, Corners, and Food Search.

5. Heuristics improved efficiency by guiding Pacman toward goals intelligently, reducing computation compared to blind searches.

---

5. Observations

- **Path Cost vs Nodes Expanded**: Algorithms with heuristics reduced node expansions without compromising path optimality.
- **Score Trends**: Higher scores correspond to faster solutions with fewer unnecessary moves.
- **Complexity Trade-offs**: Heuristic design is critical in complex mazes; poorly designed heuristics may increase computation.

---

6. Conclusion

This assignment demonstrates a solid understanding of search algorithms and heuristic design:

- DFS and BFS provided baseline performance and understanding of search strategies.
- UCS ensured optimality where path costs mattered.
- A* search combined heuristic guidance with optimality to efficiently solve complex problems.

Overall, the implementation and results confirm that **heuristics significantly improve search efficiency** while maintaining correctness, especially in larger mazes and multi-goal problems.