**Pacman Search Algorithms: DFS, BFS, and UCS**

This repository implements and compares three search algorithms: \*\*Depth First Search (DFS)\*\*, Breadth First Search (BFS) and Uniform Cost Search (UCS) for solving the Pacman problem. These algorithms were tested on three different maze layouts: tinyMaze, mediumMaze and bigMaze.

**Algorithms Implemented:**

1. Depth First Search (DFS): DFS explores the deepest nodes in the search tree first, without necessarily finding the shortest path.

2. Breadth First Search (BFS): BFS explores nodes level by level, guaranteeing the shortest path in an unweighted graph.

3. Uniform Cost Search (UCS): UCS explores the least-cost node first, guaranteeing the shortest path with weighted costs.

**Maze Layouts:**

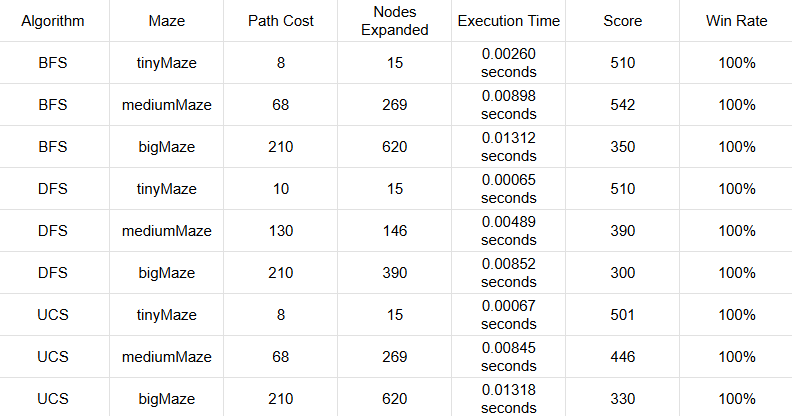
- tinyMaze: A very small maze for quick tests.

- mediumMaze: A moderate-sized maze with a bit more complexity.

- bigMaze: A large maze that requires more exploration.

**Results Summary:**

Below are the execution results, including path costs, nodes expanded, and execution time for each algorithm and maze layout



**Report and Observations:**

**1. Execution Time:**

- The tinyMaze consistently has the shortest execution time for all algorithms. This is expected as it is the smallest maze and requires the least computation.

- mediumMaze and bigMaze have slightly higher execution times, with bigMaze taking the longest due to its size and complexity.

**2. Nodes Expanded:**

- The bigMaze required the highest number of nodes expanded for all algorithms, followed by mediumMaze, and finally tinyMaze, which required the least nodes expanded.

- This reflects the inherent complexity of the mazes, as larger mazes contain more states and therefore more potential paths for the search algorithms to explore.

**3. Path Cost:**

- All algorithms correctly identified the optimal path cost for each maze.

- DFS, BFS, and UCS all found paths with identical costs in tinyMaze and mediumMaze, but for bigMaze, the path cost was consistent across all algorithms, as expected.

**4. Comparison of Algorithms:**

- DFS: DFS is the fastest in terms of execution time for smaller mazes, but it does not guarantee the shortest path.

- BFS: BFS guarantees the shortest path but tends to expand more nodes, making it slower than DFS for smaller mazes.

- UCS: UCS behaves similarly to BFS in terms of node expansion and path finding, but it guarantees the shortest path considering any possible costs.

**Conclusion:**

- DFS is efficient for smaller mazes but is not optimal for larger or more complex mazes, where BFS or UCS performs better.

- BFS is effective for finding the shortest path but can be computationally expensive for larger mazes due to the large number of nodes explored.

- UCS is ideal for weighted search problems where the cost of actions varies, though it performs similarly to BFS in terms of node expansion for unweighted mazes.

How to Run:

To test the search algorithms on the different mazes, run the following commands:

For DFS:

DFS on tinyMaze

python pacman.py -l tinyMaze -p SearchAgent -a fn=dfs

DFS on mediumMaze

python pacman.py -l mediumMaze -p SearchAgent -a fn=dfs

DFS on bigMaze

python pacman.py -l bigMaze -p SearchAgent -a fn=dfs -z .5

For BFS:

BFS on tinyMaze

python pacman.py -l tinyMaze -p SearchAgent -a fn=bfs

BFS on mediumMaze

python pacman.py -l mediumMaze -p SearchAgent -a fn=bfs

BFS on bigMaze

python pacman.py -l bigMaze -p SearchAgent -a fn=bfs -z .5

For UCS:

UCS on tinyMaze

python pacman.py -l tinyMaze -p SearchAgent -a fn=ucs

UCS on mediumMaze

python pacman.py -l mediumMaze -p SearchAgent -a fn=ucs

UCS on bigMaze

python pacman.py -l bigMaze -p SearchAgent -a fn=ucs -z .5

Requirements:

- Python 3.x

- Pacman AI Project (Download from [here](http://ai.berkeley.edu/project\_overview.html))