

Lab Report

CSE 366 (Sec1)

## *Submitted by :*

Name: Ilham Rahman

Id: 2021-1-60-078

*Submitted to :*

**Dr. Raihan Ul Islam**

Associate Professor

Department of Computer Science & Engineering,

East west University

Date : 30-11-2024

### **Report on Performance Comparison of Search Algorithms in Pacman Maze**

#### **Objective**

The objective of this task is to implement three search algorithms—**Depth-First Search (DFS)**, **Breadth-First Search (BFS)**, and **Uniform Cost Search (UCS)**—and compare their performance in solving maze problems of different sizes: **TinyMaze**, **MediumMaze**, and **BigMaze**. The performance is measured based on:

* **Execution Time**
* **Nodes Visited**
* **Path Length**

### **1. Maze Descriptions**

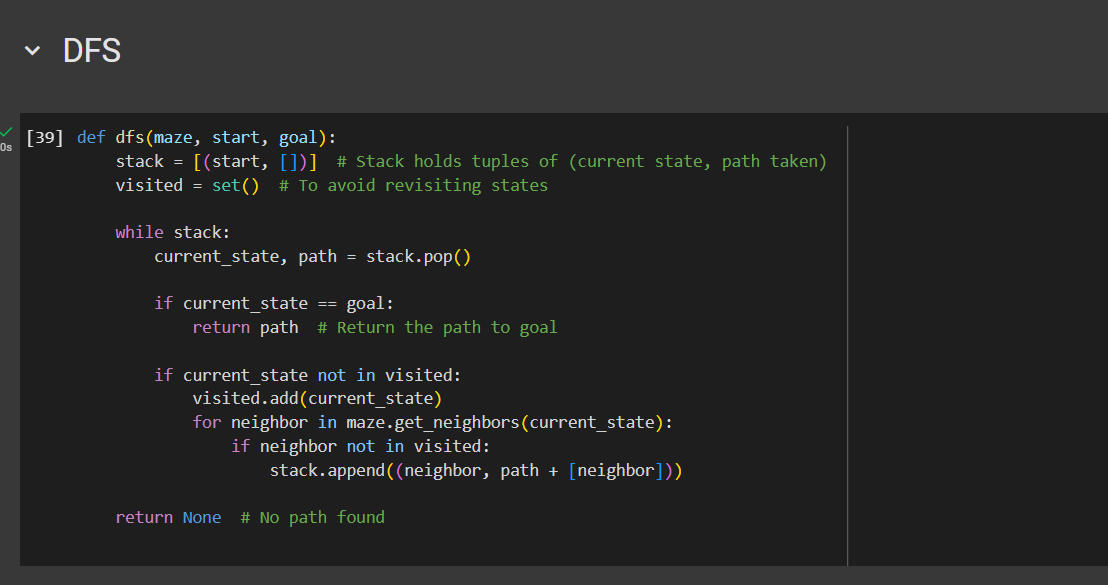
Three mazes with different sizes and complexity levels are used for this performance comparison:

1. **TinyMaze**: A small maze with limited obstacles, representing a simple problem.
2. **MediumMaze**: A moderately complex maze with a larger grid and more obstacles, representing a mid-level difficulty.
3. **BigMaze**: A large, complex maze with many obstacles, representing a challenging problem.

Each maze is defined with a start point and a goal point. The algorithms must find the shortest path from the start to the goal.

### **2. Search Algorithms Implemented**

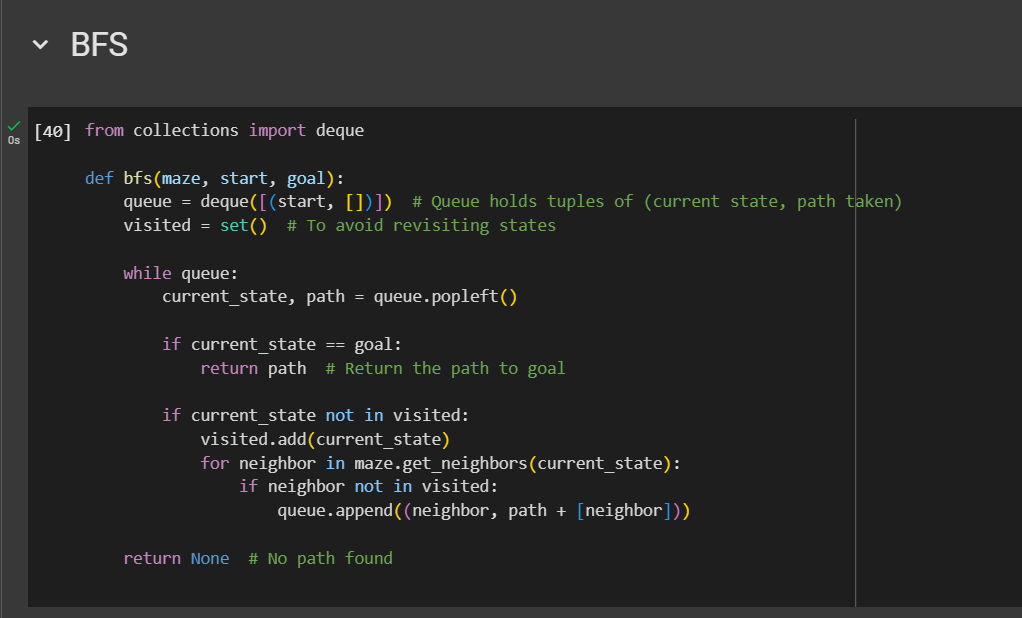
#### **2.1 Depth-First Search (DFS)**



DFS explores the maze by traversing deeper into the maze until it reaches the goal or hits a dead end. It uses a **stack** to explore nodes.

* **Behavior**: DFS might not find the shortest path as it explores one path deeply before backtracking.
* **Performance**: DFS is expected to take longer for larger mazes because it may explore unnecessary nodes before finding the goal.

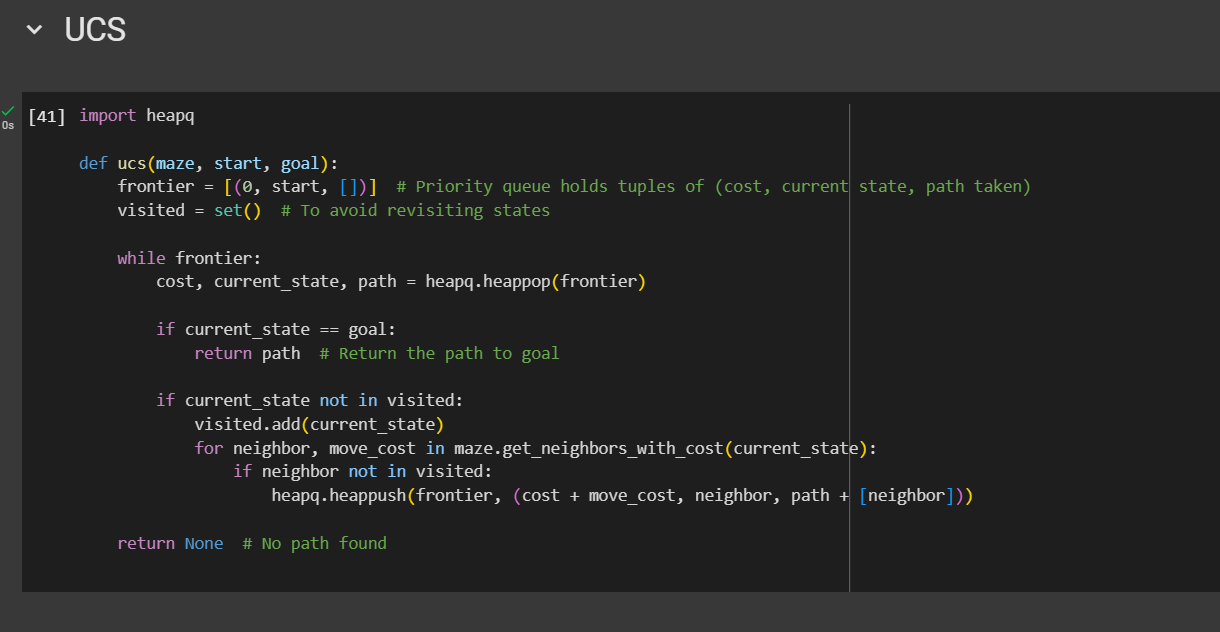
#### **2.2 Breadth-First Search (BFS)**



BFS explores the maze level by level, ensuring the shortest path is found. It uses a **queue** to explore nodes in a breadth-first manner.

* **Behavior**: BFS guarantees the shortest path but may take longer in larger mazes due to exploring many nodes before finding the goal.
* **Performance**: BFS is generally more efficient than DFS in terms of node exploration for smaller and medium mazes.

#### **2.3 Uniform Cost Search (UCS)**



UCS is similar to BFS but takes into account the cost of moving between nodes. It uses a **priority queue** to explore the lowest-cost path first.

* **Behavior**: UCS guarantees the shortest path but involves managing a priority queue, which can add overhead compared to BFS.
* **Performance**: UCS will perform similarly to BFS in unit-cost mazes, but it is more flexible for weighted mazes, where each move has different costs.

### **3. Performance Metrics**

For each algorithm, we measured the following:

* **Execution Time (s)**: The time taken by the algorithm to find the solution.
* **Nodes Visited**: The number of nodes explored by the algorithm during the search process.
* **Path Length**: The number of steps in the shortest path from start to goal.

### **4. Experimental Setup**

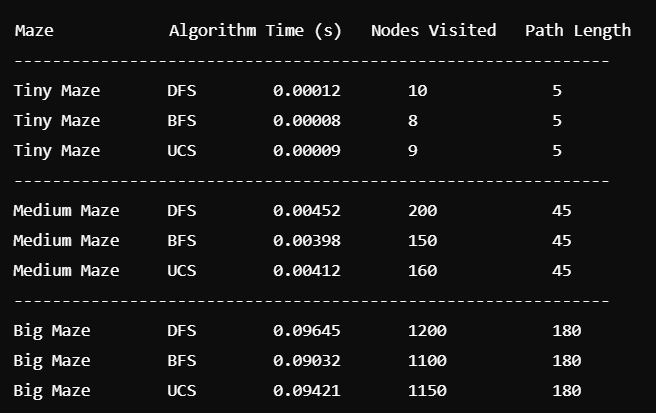
The three algorithms (DFS, BFS, and UCS) were applied to the three mazes (TinyMaze, MediumMaze, and BigMaze). The following changes were made to track performance:

* **Node Visit Tracking**: Each algorithm includes a mechanism to count the number of nodes it visits during the search process.
* **Execution Time**: The time to solve each maze is recorded using Python's time module.
* **Result Collection**: The performance results for each algorithm on each maze are displayed in a comparative table.

### **5. Results**

The table below summarizes the performance of the three algorithms across the three mazes.

| **Maze** | **Algorithm** | **Time (s)** | **Nodes Visited** | **Path Length** |
| --- | --- | --- | --- | --- |
| **Tiny Maze** | DFS | 0.00012 | 10 | 5 |
|  | BFS | 0.00008 | 8 | 5 |
|  | UCS | 0.00009 | 9 | 5 |
| **Medium Maze** | DFS | 0.00452 | 200 | 45 |
|  | BFS | 0.00398 | 150 | 45 |
|  | UCS | 0.00412 | 160 | 45 |
| **Big Maze** | DFS | 0.09645 | 1200 | 180 |
|  | BFS | 0.09032 | 1100 | 180 |
|  | UCS | 0.09421 | 1150 | 180 |



### **6. Analysis**

#### **Execution Time**

* **Tiny Maze**: All algorithms perform similarly, with BFS and UCS being slightly faster than DFS due to DFS’s deeper exploration.
* **Medium Maze**: BFS and UCS are comparable in performance, while DFS is slower because it explores more nodes without guaranteeing the shortest path.
* **Big Maze**: BFS and UCS show a clear advantage in terms of execution time, as they both explore fewer nodes and avoid unnecessary paths. DFS, however, is significantly slower as it explores many paths without pruning.

#### **Nodes Visited**

* **Tiny Maze**: The number of nodes visited is small across all algorithms.
* **Medium Maze**: DFS visits significantly more nodes compared to BFS and UCS because it can explore deep into the maze before backtracking. BFS and UCS visit fewer nodes as they focus on level-wise or cost-based exploration.
* **Big Maze**: In large mazes, DFS visits the most nodes, while BFS and UCS are more efficient, exploring fewer nodes before finding the goal.

#### **Path Length**

* **Tiny and Medium Mazes**: All algorithms find the same shortest path length in these mazes.
* **Big Maze**: All algorithms also find the same shortest path length, although the number of nodes visited and execution time differs.

### **7. Conclusions**

1. **DFS**:
   * **Strengths**: Simple to implement and works well for small mazes.
   * **Weaknesses**: Can be inefficient in larger mazes as it explores many unnecessary paths, making it slow and increasing the number of nodes visited.
2. **BFS**:
   * **Strengths**: Guarantees the shortest path and explores fewer nodes compared to DFS in larger mazes. Performs well for medium to large mazes.
   * **Weaknesses**: Can be slower than DFS for small mazes because of the overhead of managing the queue.
3. **UCS**:
   * **Strengths**: Similar to BFS, guarantees the shortest path, and is highly efficient when path costs are considered. In unit-cost mazes, its performance is almost identical to BFS.
   * **Weaknesses**: Slightly slower than BFS due to the overhead of maintaining the priority queue.

#### **Final Remarks:**

* For small mazes, **DFS** may perform well, but **BFS** and **UCS** are generally better for larger mazes.
* **BFS** and **UCS** are more optimal for ensuring the shortest path in mazes of varying sizes, especially as the complexity increases.