# CSE 537 - Artificial Intelligence

# Project-01: The Searchin' Pac Man

Submitted by: Nilesh Rustagi - 113259870, Varun Sabnis - 113272172

## 1. Depth First Search

| **Stats** | Tiny Maze | Medium Maze | Big Maze |
|---|---|---|---|
| Expanded Nodes | 15 | 146 | 390 |
| Run Time | 0.0s | 0.0s | 0.0s |
| Cost | 10 | 130 | 210 |

- DFS uses the stack data structure which holds the list of nodes to be visited next i.e fringe list.
- If the state space graph consists of cycles, then DFS is not complete. But if the tree is finite, then we can say that DFS is complete
- DFS is not optimal because it disregards the cost or the depth of the node as it finds the solution in the leftmost part of the search tree

## 2. Breadth First Search

| **Stats** | Tiny Maze | Medium Maze | Big Maze |
|---|---|---|---|
| Expanded Nodes | 15 | 269 | 620 |
| Run Time (in sec) | 0.0s | 0.0s | 0.0s |
| Cost | 8 | 68 | 210 |

- BFS uses the queue data structure which holds the list of nodes to be visited next i.e fringe list.
- In comparison to DFS, we can see that in BFS, the number of nodes expanded is more as on each level, all the nodes on one level are expanded before moving onto the next.
- Moreover, BFS gives a less cost solution wrt DFS because the goal is reached only after exploring all paths.

## 3. Uniform Cost Search

| **Stats** | Tiny Maze | Medium Dotted Maze | Medium Scary Maze | Medium Maze | Big Maze |
|---|---|---|---|---|---|
| Expanded Nodes | 15 | 186 | 108 | 269 | 620 |
| Run Time (in sec) | 0.0s | 0.0s | 0.0s | 0.0s | 0.0s |
| Cost | 8 | 1 | 68719479864 | 68 | 210 |

- To implement uniform cost search we use priority queue as the fringe list. We always pick the lowest cost state from the fringe list.
- Once we pick a goal state out from the fringe list we return the path. We find that the UCS expansion of nodes is the same as BFS because the edge cost is 1.

## 4. A* Search

| **Stats** | Tiny Maze | Medium Maze | Big Maze | Open Maze |
|---|---|---|---|---|
| Expanded Nodes | 14 | 221 | 549 | 535 |
| Run Time (in sec) | 0.0 | 0.0 | 0.0s | 0.0s |
| Cost | 8 | 68 | 210 | 54 |

- In A* search algorithm we use priority queue as well, but the cost function includes heuristic cost from current state to goal state.
- We have used the Manhattan heuristic. The heuristic is admissible and consistent, hence will find the goal state in a number of steps.
- The A* search algorithm expands the minimum number of nodes when compared to other algorithms.

## 5. Corners Problem

| Stats | Tiny Corners | Medium Corners |
|---|---|---|
| Expanded Nodes | 435 | 2448 |
| Run Time (in sec) | 0.0 | 0.1 |
| Cost | 28 | 106 |

- The corners problem can be solved by additionally keeping track of how many corners have been visited along with the current position of the pacman.
- The breadth first search approach expands a large number of nodes to cover all the corners. A* search would do better as it expands contours that are targeted to the corners (one after the other) and hence reaches the goal faster.

## 6. Corners Heuristic

| Stats | Medium Corners |
|---|---|
| Expanded Nodes | 978 |
| Run Time (in sec) | 0.0 |
| Cost | 106 |

- We use an admissible heuristic for the corners problem. The heuristic is computed as the maximum maze distance (which uses BFS) between current position to any of the 4 corners.
- This is admissible, at least this distance must be covered before a goal state is reached. This is a lower bound on the actual optimal path cost to reach the goal state.

## 7. Food Search Problem

| Stats | Tricky Search |
|---|---|
| Expanded Nodes | 4137 |
| Run Time (in sec) | 6.4 |
| Cost | 60 |

- We have used mazeDistance to compute distance from current state to the food points. We will take the max of all distances to the food points as done in question 6.