BFS Small
Path length : 19
Fringe max size : 8
Visited : 92

DFS Small
Path length: 37
Fringe max size : 6
Visited : 38

A* Small
Path length: 29
Fringe max size : 4
Visited : 40

BFS Medium
Path length : 68
Fringe max size: 8
Visited : 270

DFS Medium
Path length : 130
Fringe max size : 8
Visited: 147

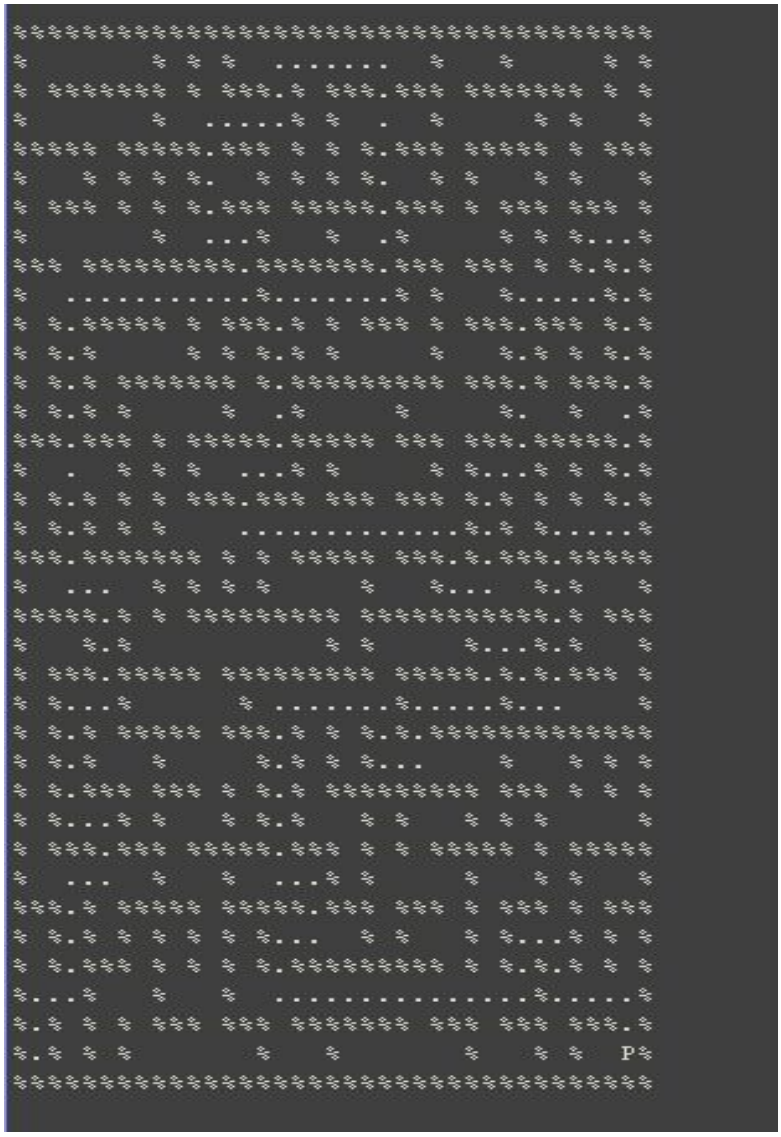
A* Medium
Path length : 152
Fringe max size: 7
Visited : 156

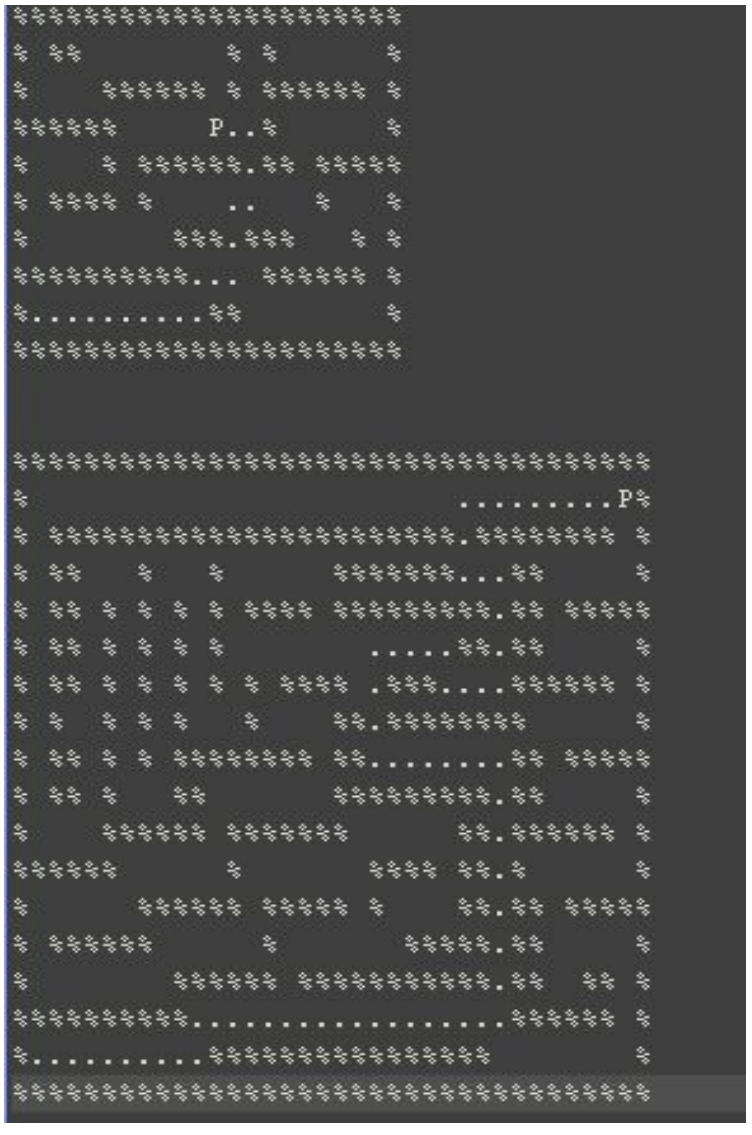
BFS Large
Path length : 210
Fringe max size: 7
Visited : 621

DFS Large
Path length : 210
Fringe max size: 38
Visited : 388

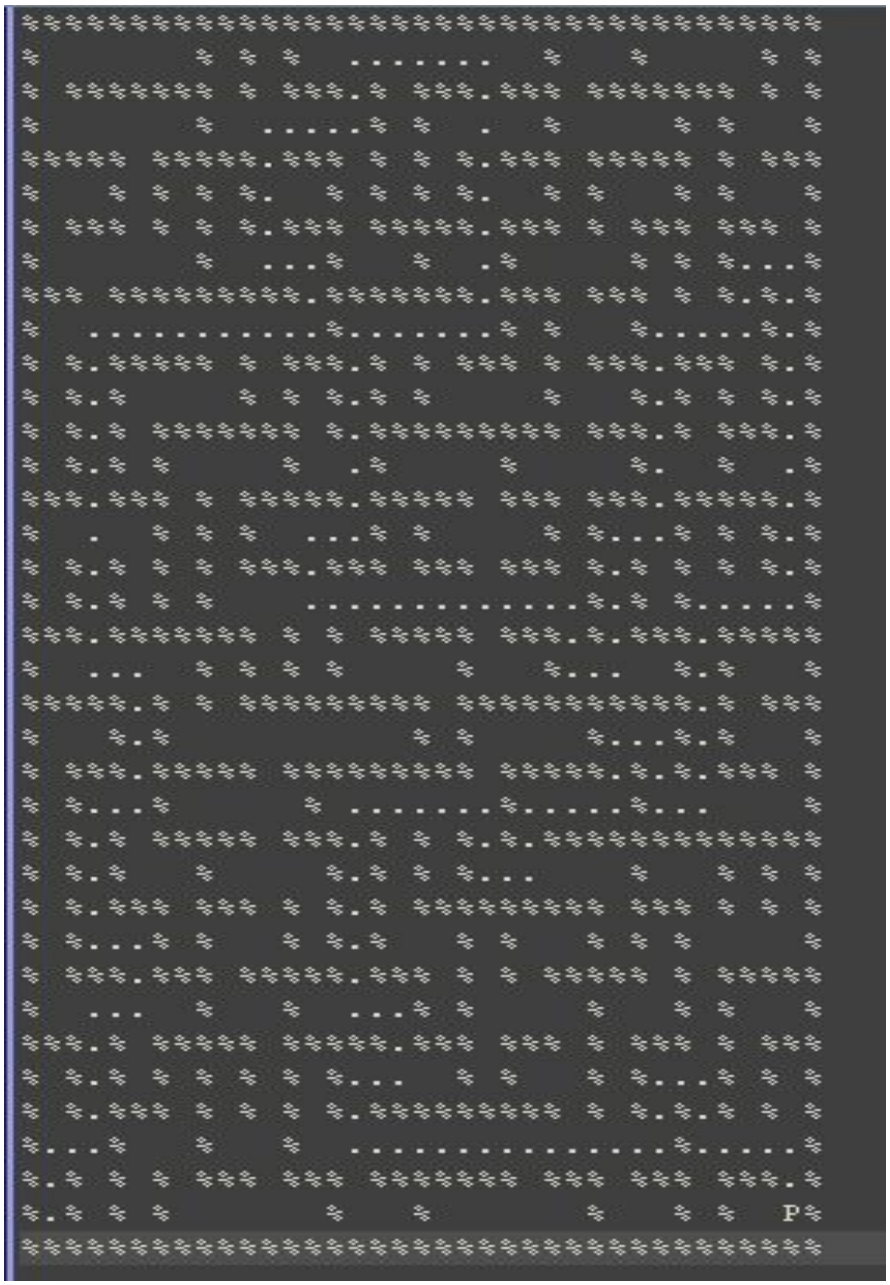
A* Large
Path length : 210
Fringe max size : 24
Visited : 457

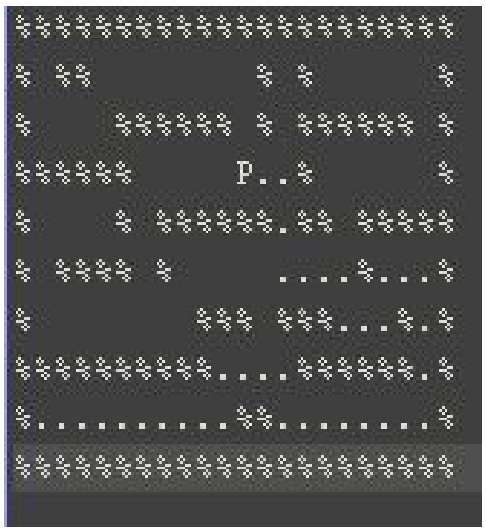
BFS Solutions:



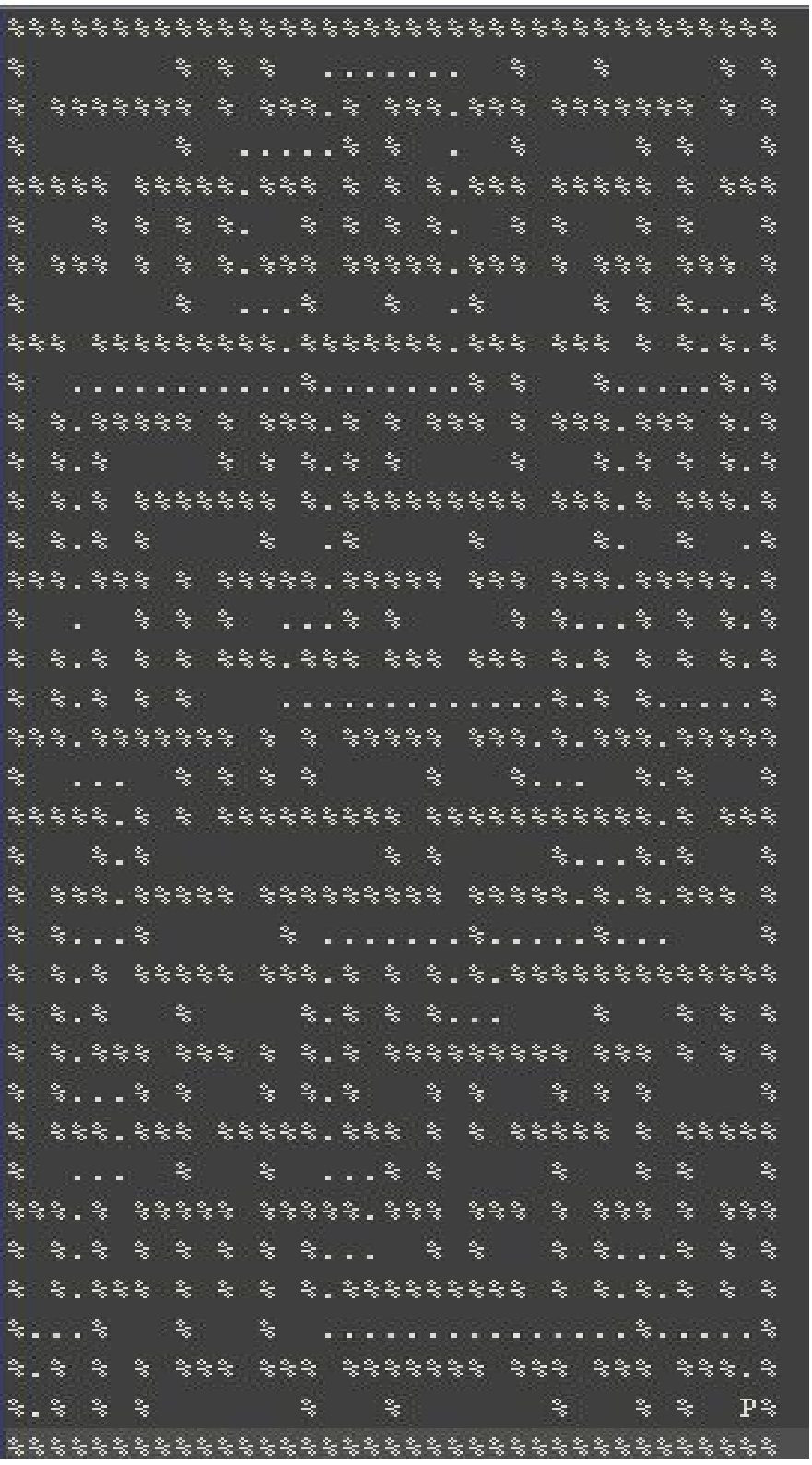


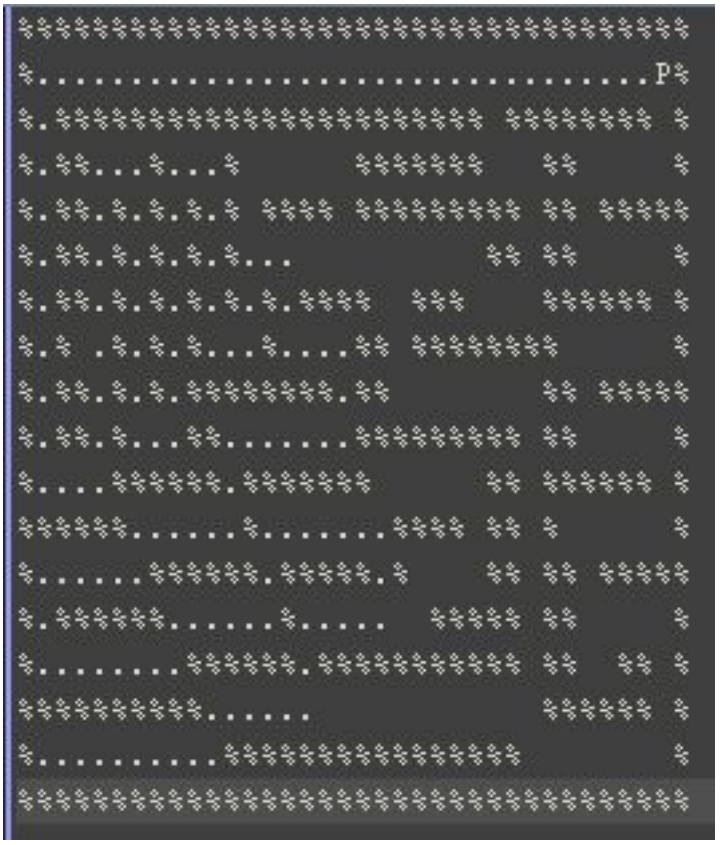
DFS Solutions:





A* Solutions :





For my implementation I worked from the first assignment to keep an adjacency list in the gameboard class. I then used the fringe/frontier for BFS/DFS. For BFS the priority was set to FIFO, and for DFS it was set to FIFO. For A* I implemented a function to find the Manhattan distance between two nodes. I kept a dictionary that converted the node number to its position in the 2d array. The element that was visited next was the elements whose Manhattan distance to the goal was minimal.

For the path, I kept a dictionary of parents and iterated backwards from the goal until I hit the start node. The start node is reached when curr == next, which means that the start node's parent is itself.

The code saves into a file called "sol.txt", and you have to type in the maze file name, and function to call (game_board.DFS(), game_board.BFS(),game_board.Astar()).