Sakshan Saini

Ryley Higa

James Jia

3 Credit Hours

MP1: Maze Search

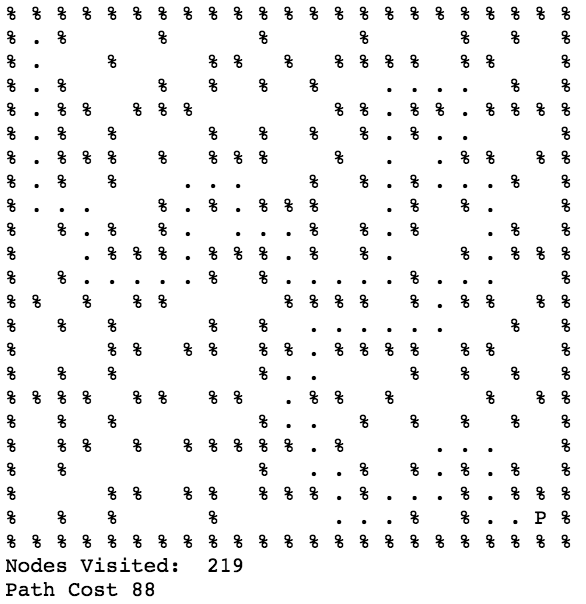
1.1: Maze Searching Algorithms

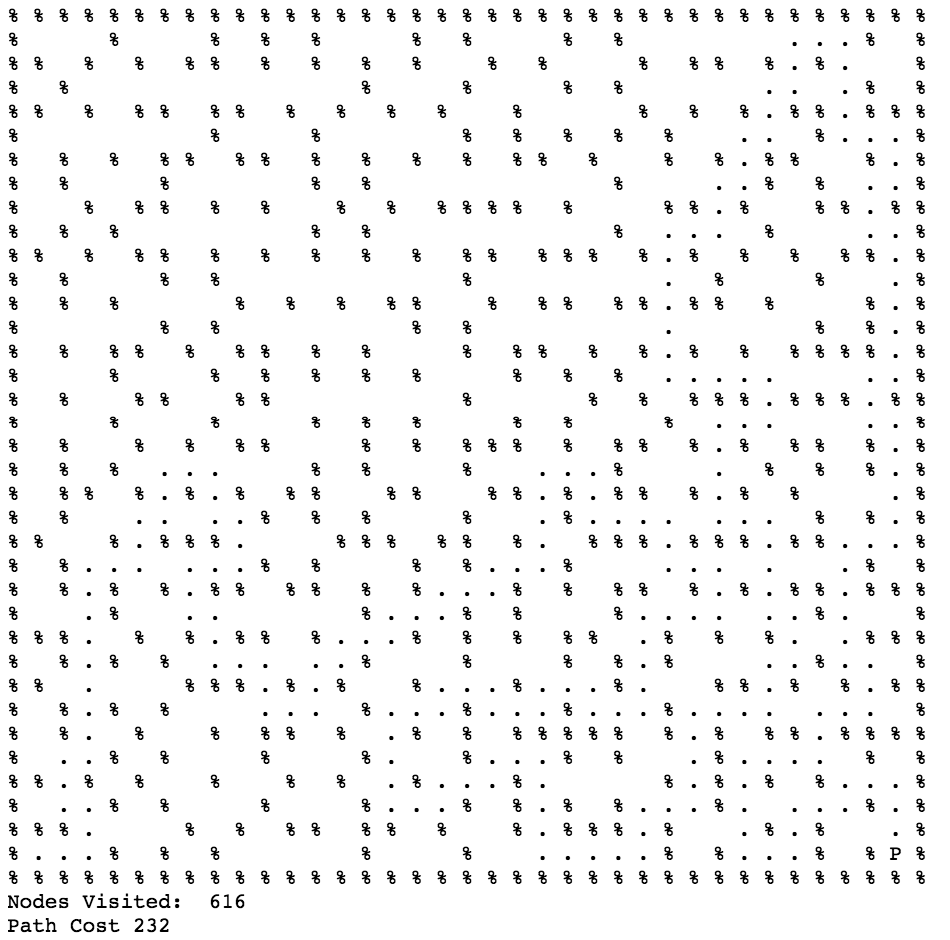
The maze class is represented by a two dimensional list of characters. The most significant feature of the maze class is that it implements a find children function that takes in a node and returns the successor states of the node. For part1.1, the successor states are adjacent positions.

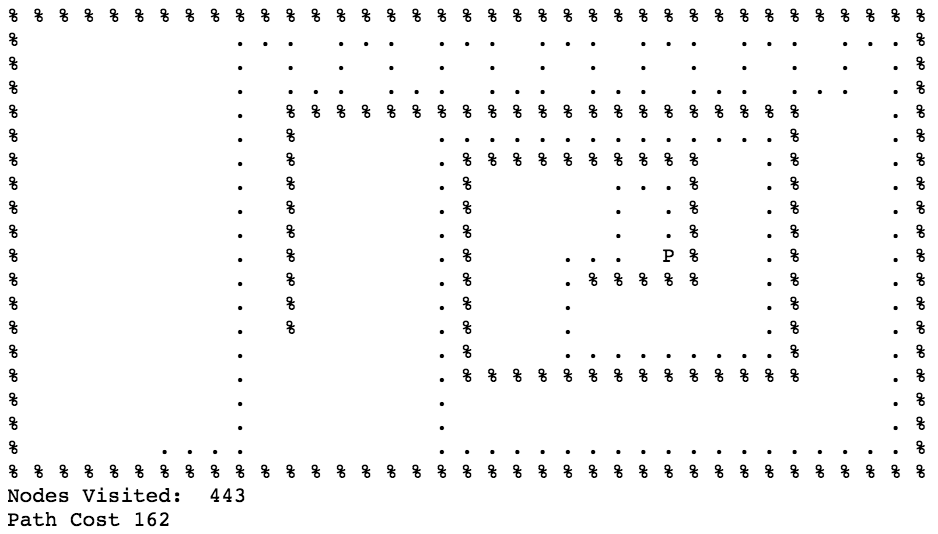
Each of the searching algorithms was implemented in separate classes that inherit from a base maze search class. The base maze search class is abstract—meaning it should never have instances—and outlines certain functionalities that every maze search algorithm should have such as adding nodes to the frontier, choosing nodes from the frontier, detecting duplicates nodes, finding a path, tracing back a path from a goal state to the initial state, and detecting whether you reach the goal state. Since the path finding routine and the backtracking routine are similar for all maze searching algorithms implemented in this section, these routines are implemented in the base class and directly inherited by its derived classes. The derived functions should implement an initializer function (\_\_init\_\_) to initialize the frontier and values associated with the frontier, a function to add a node, a function to choose a node, a function to detect duplicates, and a function to check the end state.

Depth First Search

The depth-first-search class implements a queue as the frontier. To add a node to the frontier, the function takes in a node and appends it to the right of a list. To choose a node, the function takes in a node and pops the node from the left of the list.

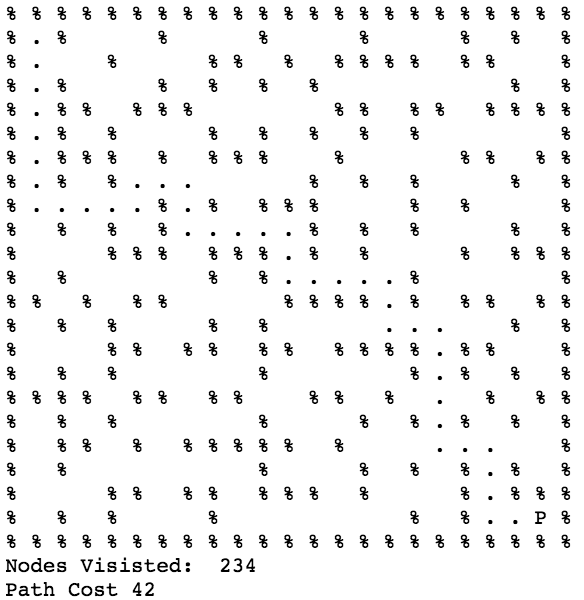


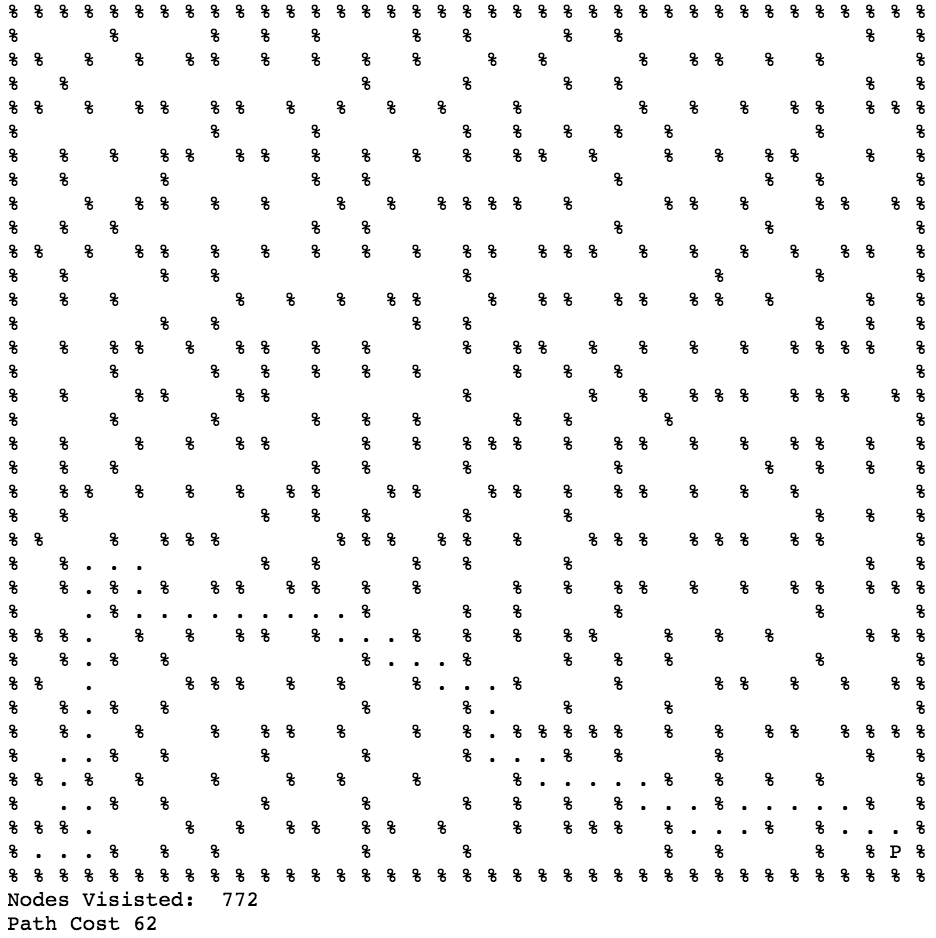


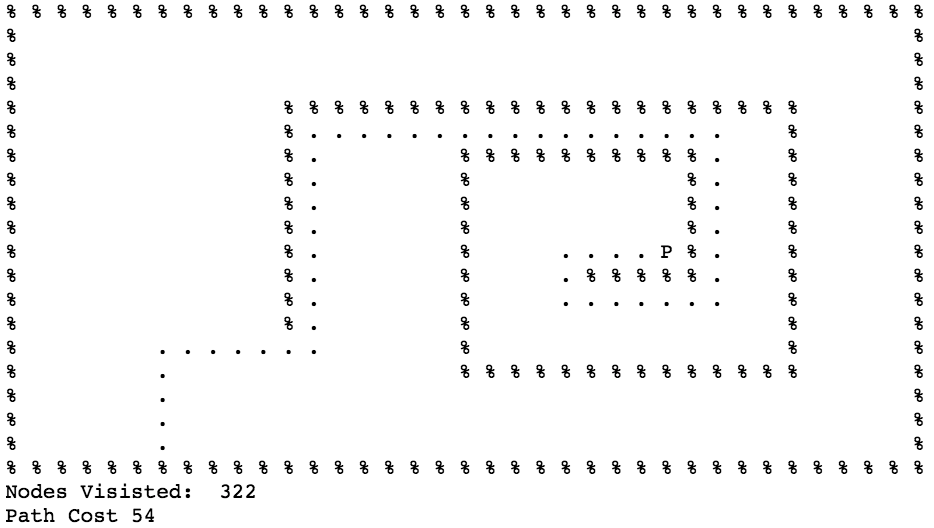


Breadth First Search

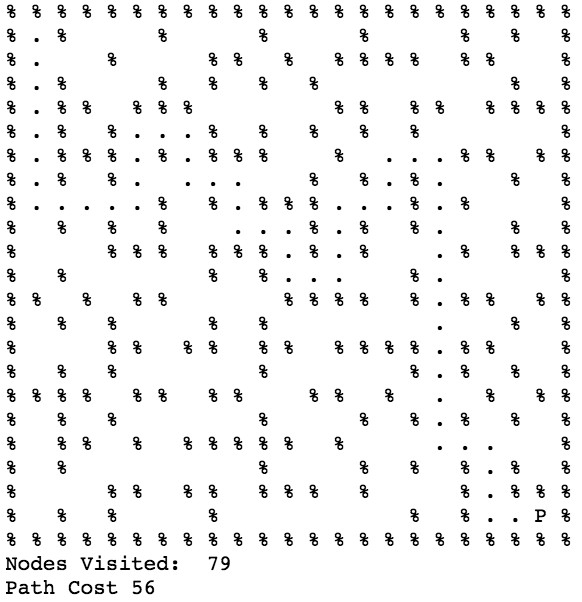
The breadth-first-search class implements a stack as a frontier a similar data structure is used to keep track of the path costs associated with each node so that if we popped a node off the frontier, we would also pop the path cost associated with the node. Add a node involves appending the node and path cost to a list, and choosing a node involves poping the node and path cost from the right of the list.

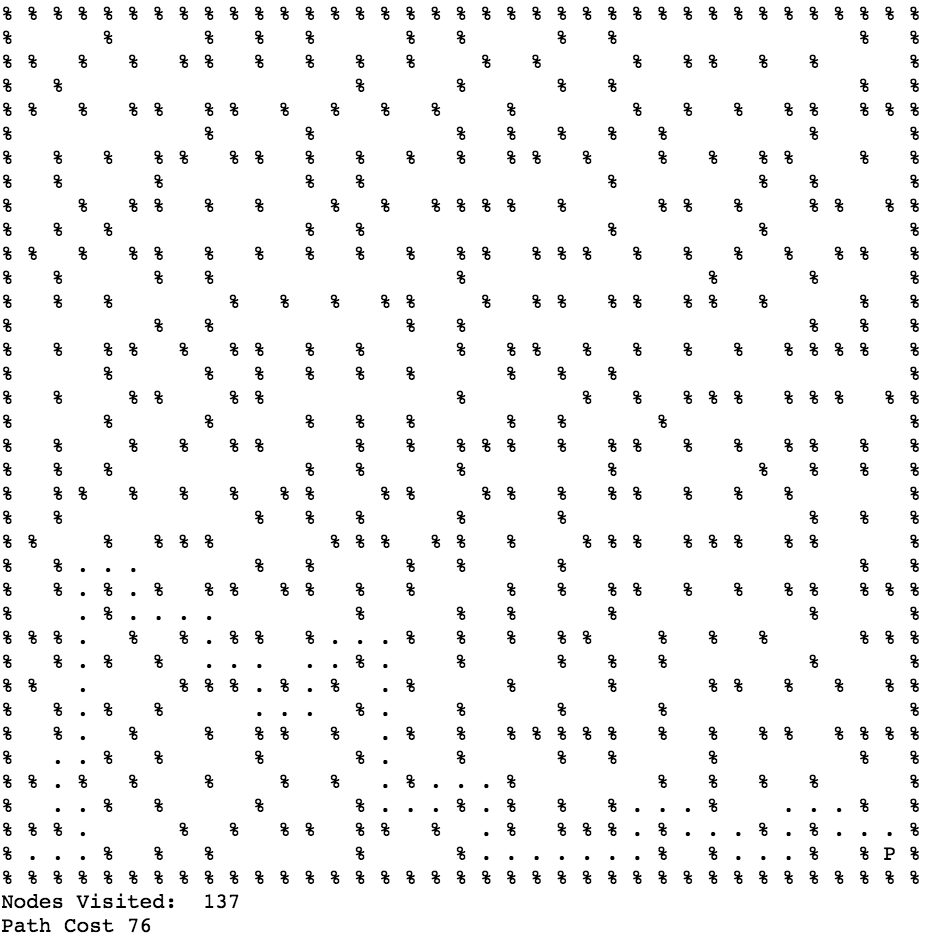


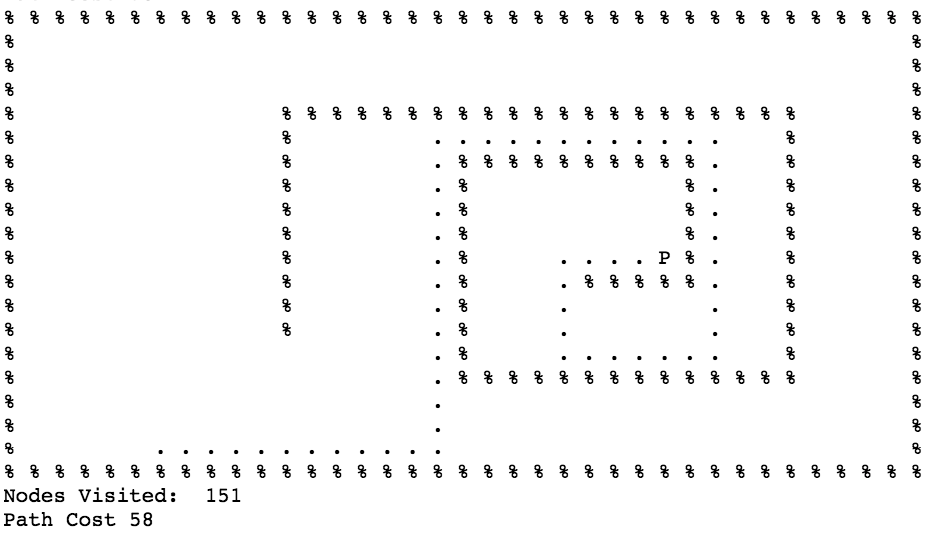




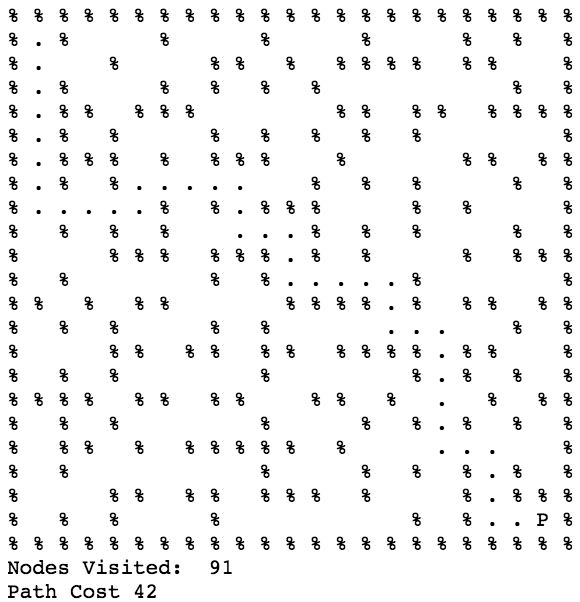
Greedy Best-First Search

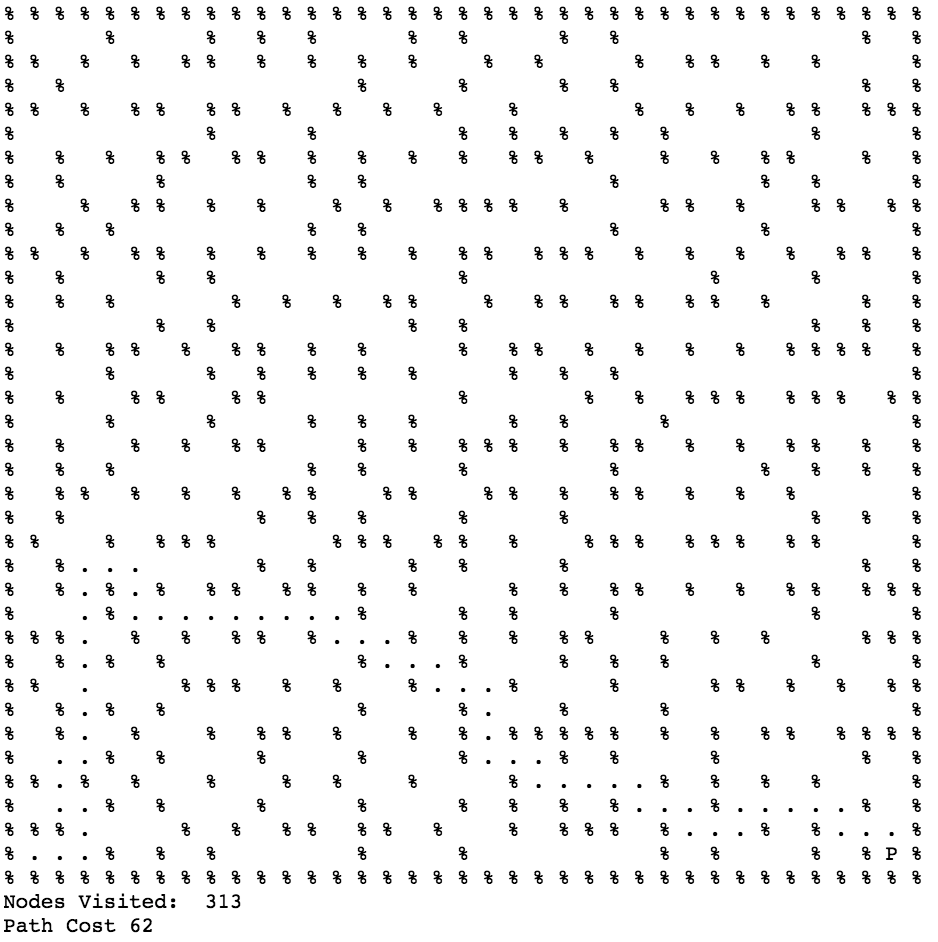


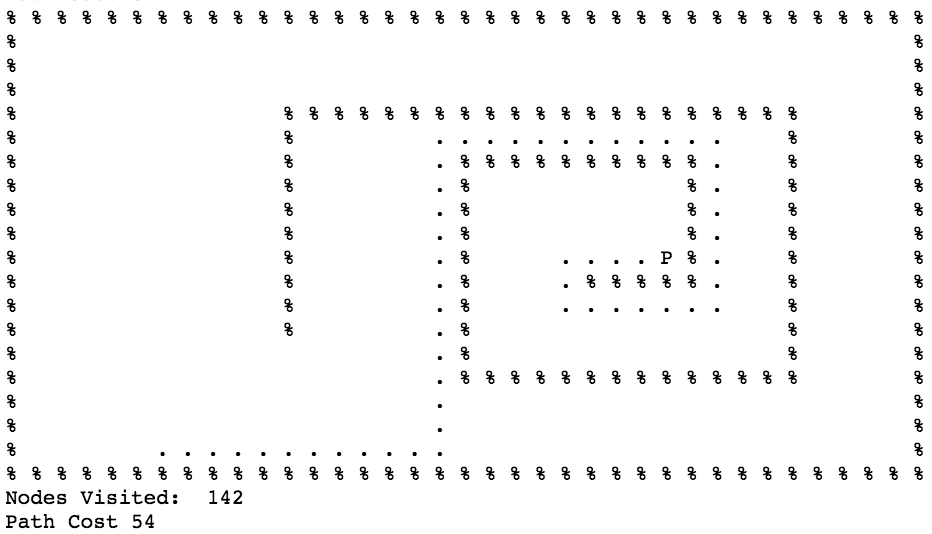




A\* Search







1.2:

1.3: Maze with Ghost