A PROJECT REPORT ON

THE SEARCHING PAC-MAN

Project 01: Artificial Intelligence (CSE 537)

Instructor: Professor I.V. Ramakrishnan

BY

Nisha Gandhi (111496495)

Department of Computer Science

Stony Brook University

2017-2018

1. Depth-First Search (Question 1)

Data structures used: Stack & List

Expansion of nodes: Each successor node of the current node being expanded is stored in the fringe (stack push) along with the tuple of directions till that node. The successors are expanded in the same order provided by *getSuccessors* function. For every node being expanded from the fringe (stack pop), a check is made if it has already been expanded, and if so, the next node is chosen. This continues till the goal state has been reached.

Return values: Returns a list of directions to the goal.

Maze Type	**Search nodes expanded**	**Total Cost & Total Time**	**Average score**
Tiny Maze	15	10 & 0.0 seconds	500.0
Medium Maze	146	130 & 0.0 seconds	380.0
Big Maze	390	210 & 0.0 seconds	300.0

2. Breadth-First Search (Question 2)

Data structures used: Queue & List

Expansion of nodes: Each successor node of the current node being expanded is stored in the fringe (queue push) along with the tuple of directions till that node. The successors are expanded in the same order provided by *getSuccessors* function. For every node being expanded from the fringe (queue pop), a check is made if it has already been expanded, and if so, the next node is chosen. This continues till the goal state has been reached.

Return values: Returns a list of directions to the goal.

Maze Type	**Search nodes expanded**	**Total Cost & Total Time**	**Average score**
Tiny Maze	15	8 & 0.0 seconds	502.0
Medium Maze	269	68 & 0.0 seconds	442.0
Big Maze	620	210 & 0.1 seconds	300.0

3. Uniform-Cost Search (Question 3)

Data structures used: Priority Queue & List

Expansion of nodes: Each successor node of the current node being expanded is stored in the fringe (priority queue push) along with the tuple of directions till that node, as well as the cost required to reach that node. The successors are expanded in the same order provided by *getSuccessors* function. For every node being expanded from the fringe (priority queue), a check is made if it has already been expanded, and if so, the next node is chosen. This continues till the goal state has been reached.

Return values: Returns a list of directions to the goal.

Maze Type	**Search nodes expanded**	**Total Cost & Total Time**	**Average score**
Medium Maze	269	68 & 0.1 seconds	442.0
Medium Dotted Maze	186	1 & 0.0 seconds	646.0
Medium Scary Maze	108	68719479864 & 0.0 seconds	418.0

4. A* Search (Question 4)

Data structures used: Priority Queue & List

Parameters in function: An object of Problem, Heuristic

Expansion of nodes: Each successor node of the current node being expanded is stored in the fringe (priority queue push) along with the tuple of directions till that node, as well as the summation of the cost required to reach that node and the heuristic to the goal state. The priority of the node is decided by its heuristic value. The successors are expanded in the same order provided by *getSuccessors* function. For every node being expanded from the fringe (priority queue), a check is made if it has already been expanded, and if so, the next node is chosen. This continues till the goal state has been reached.

Return values: Returns a list of directions to the goal.

Maze Type	**Search nodes expanded**	**Total Cost & Total Time**	**Average score**
Tiny Maze	14	8 & 0.0 seconds	502.0
Medium Maze	221	68 & 0.0 seconds	442.0
Big Maze	549	210 & 0.2 seconds	300.0

5. Corners Problem (Question 5)

Start state: Start state is taken as a combination of the starting Pac-man position and the tuple (list) of goals (corners) visited. This list is initialized with *False* for every corner.

Goal state: The goal state returns *True* only if all corners have been visited i.e. the list of goals contains all *True.*

Successors Function: The *getSuccessors* function generates a list of successors; each element in the list containing the node, goals list, action required to reach that node and the cost. This list is then returned.

Logic behind the chosen state encodings: Since every state needs to maintain its current goal state (corners visited till that state) along with its coordinates, the above state encodings have been chosen.

Maze Corners Type	Search nodes expanded	Total Cost & Total Time	Average score
Tiny Corners Maze	252	28 & 0.0 seconds	512.0
Medium Corners Maze	1966	106 & 0.5 seconds	434.0
Big Corners Maze	7949	162 & 7.0 seconds	378.0

6. Corners Heuristic (Question 6)

Idea behind Corners Heuristic: The entire distance from the current successor's position to its nearest unvisited corner, from this corner to the next nearest unvisited corner and so on is computed by using Manhattan distance till all the corners are visited. This complete distance is the admissible Corners Heuristic.

Why is it admissible and consistent?

This heuristic is admissible because, since the shortest Manhattan distances from a particular node position are being computed, it is less than or equal to the direct distances computed from that particular node. Moreover, the function ***f(x) = g(x) + h(x)*** increases monotonically, hence the consistency.

Maze Corners Type	Search nodes expanded	Total Cost & Total Time	Average score
Tiny Corners Maze	154	28 & 0.0 seconds	512.0
Medium Corners Maze	692	106 & 0.1 seconds	434.0
Big Corners Maze	1725	162 & 0.5 seconds	378.0

7. Food Heuristic (Question 7)

Idea behind Food Heuristic: The entire distance from the current successor's position to its nearest food position, from this food position to the next nearest uneaten food position and so on is computed by using Manhattan distance till all the food is consumed. This complete distance is the admissible Food Heuristic.

Why is it admissible and consistent?

This heuristic is admissible because, since the shortest Manhattan distances from a particular node position are being computed, it is less than or equal to the direct distances computed from that particular node. Moreover, the function ***f(x) = g(x) + h(x)*** increases monotonically, hence the consistency.

Maze Corners Type	**Search nodes expanded**	**Total Cost & Total Time**	**Average score**
Tiny Maze	470	27 & 0.3 seconds	573.0
Tricky Search Maze	6761	60 & 15.6 seconds	570.0