CS440 Assignment 1: Maze Search

INTRODUCTION

In this assignment, search algorithms are applied to different scenarios of maze searching problems, the problem in Part 1 is letting the "Pacman" search the paths through three different sizes of mazes to reach the goal, then on top of it, the "Pacman" needs to avoid ghosts to reach the goal. In Part 2, the "Pacman" is going to tackle a more complex scenario that multiple goals need to be reached. Java is used to solve the problems.

PART 1

1.1 Basic Pathfinding

This problem is to find a path through three different kinds of maze from a given start to a goal state using four different kinds of searching algorithms: Depth-first search, Breadth-first search, Greedy best-first search, and A* search.

1.1.1 Results of DFS

The node which DFS expands is the deepest node in the frontier, this results in that the search goes to the deepest level of the search tree where the nodes have no children. The solution paths obtained from the code implementing the DFS are shown in the figures below.

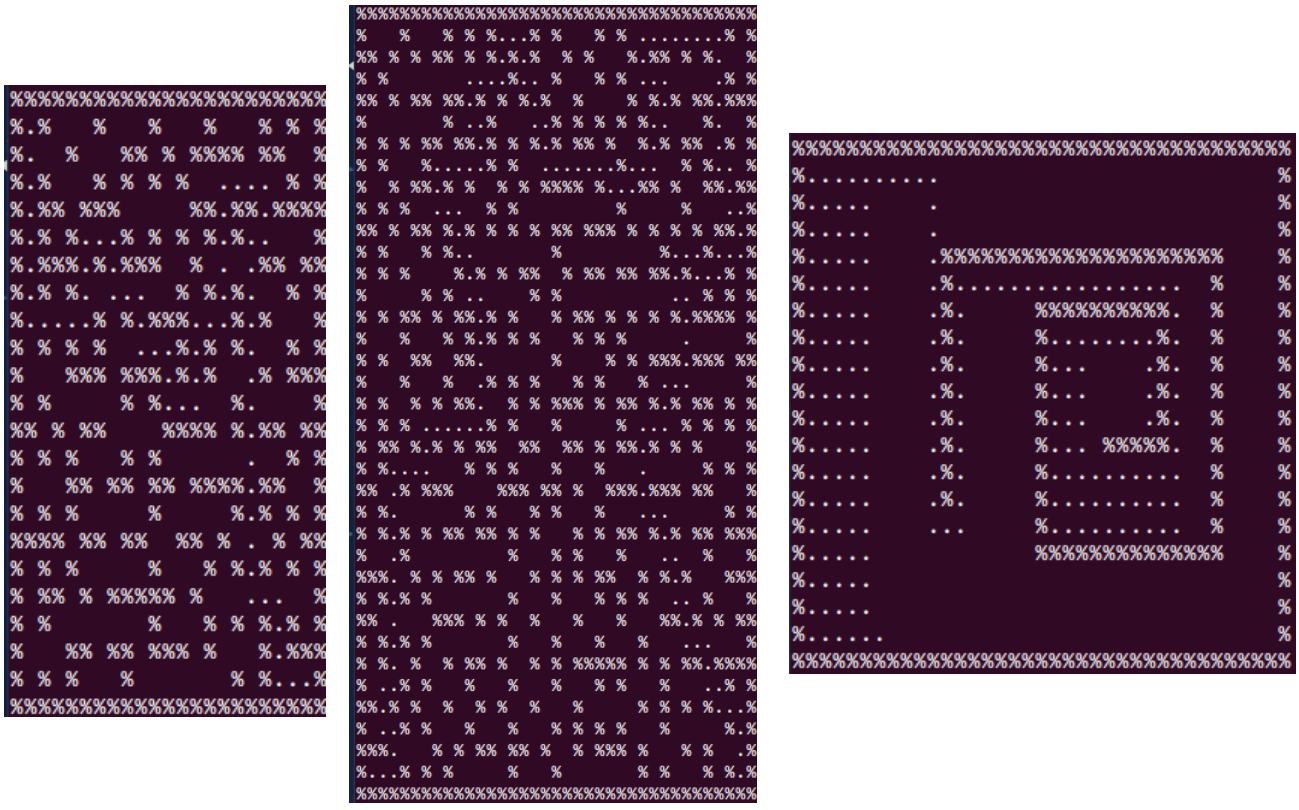

Figure 1 Solution paths for medium, big, and open maze using DFS

1.1.2 Results of BFS

In the Breadth-first search, all the successors of the root node are expanded right after the root node is expanded, then the successors of the root's successors are expanded, and the process goes on in this way. The solution paths obtained from the code implementing the BFS are shown in the figures below.

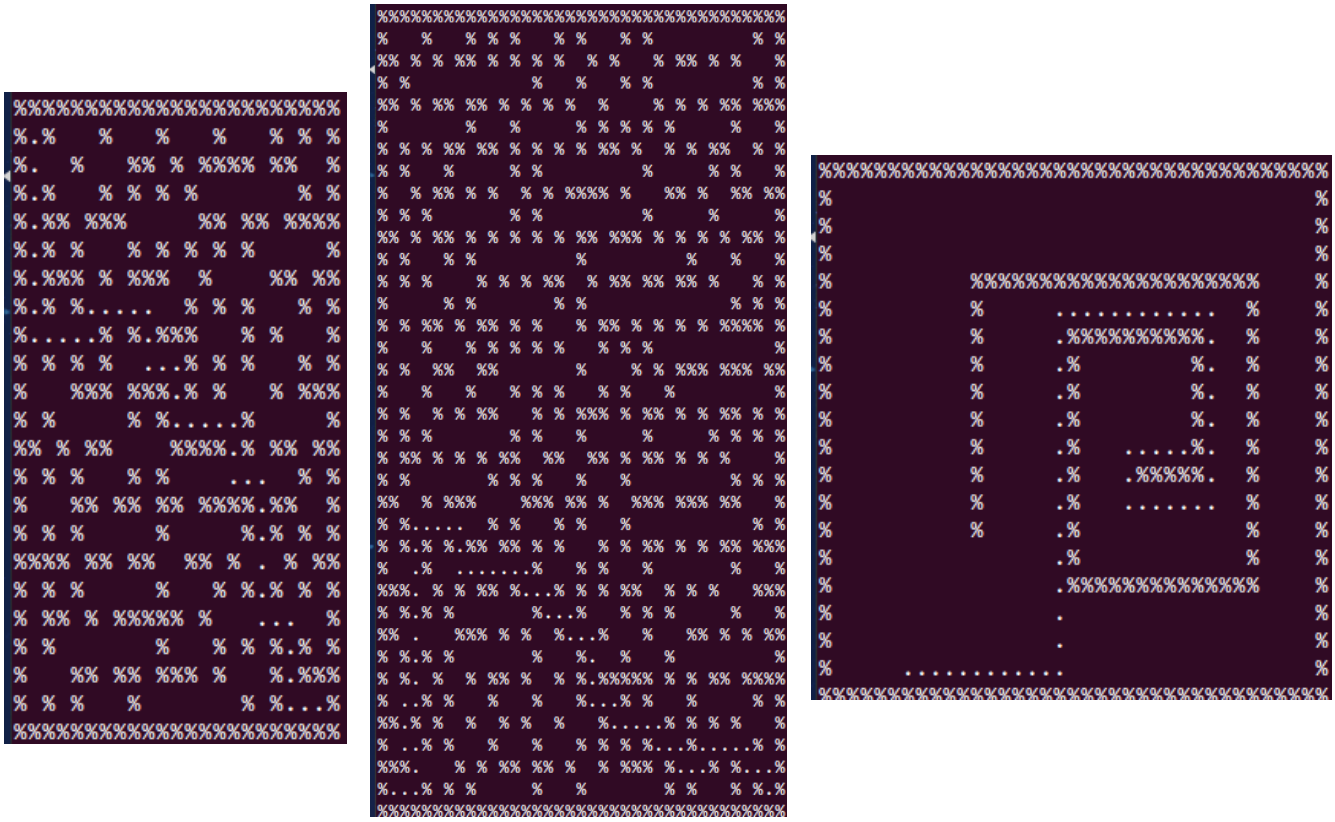

Figure 2 Solution paths for medium, big, and open maze using BFS

1.1.3 Results of Greedy best-first search

The node whose has the smallest heuristic function value is expanded first in Greedy beat-first search. The solution paths obtained from the code implementing the Greedy best-first search are shown in the figures below.

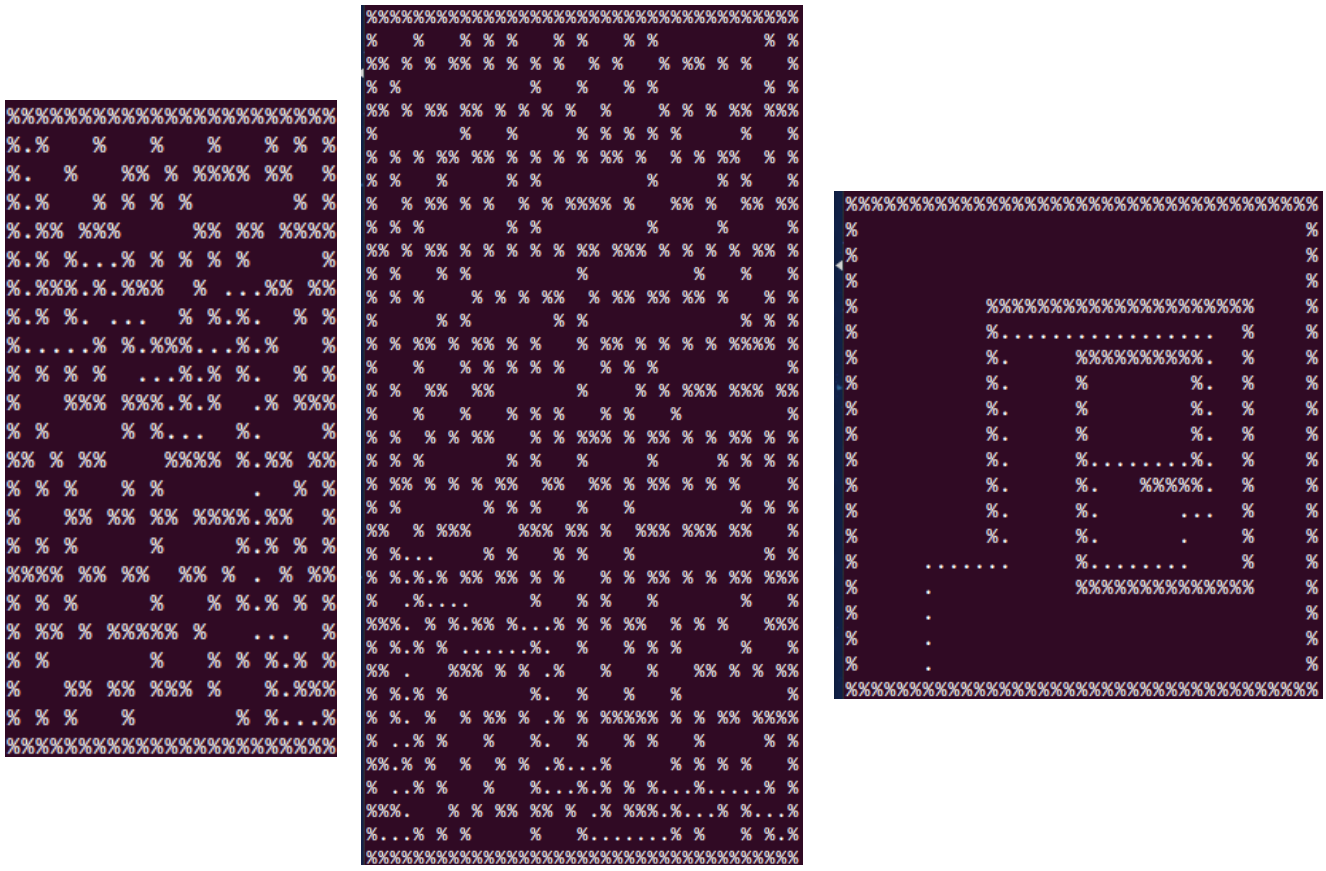

Figure 3 Solution paths for medium, big, and open maze using Greedy best-first search

1.1.4 Results of A* search

A* search is a well-known best-first search algorithm which evaluates the summation (represented by f(n)) of the cost from start state to reach the node g(n) and the cost to get the goal from the node h(n), then the node who has the smallest f(n) is expanded first. The solution paths obtained from the code implementing the A* search are shown in the figures below.

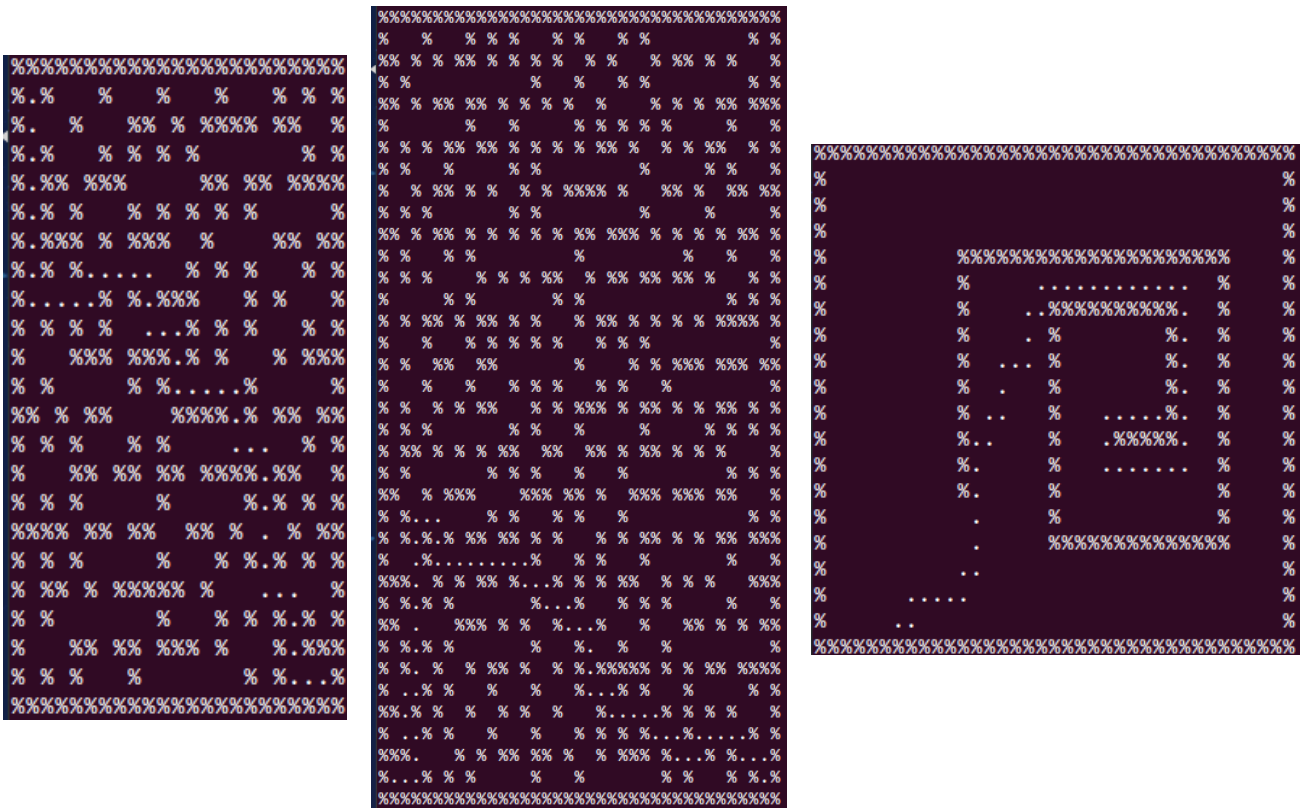

Figure 4 Solution paths for medium, big, and open maze using A* search

1.1.5 Summary

Table 1 Summary of four different kinds of search algorithm (medium maze)

Search algorithms	DFS	BFS	Greedy search	A* search
Path cost	65	42	56	42
# of nodes expanded	74	224	80	126

Table 2 Summary of four different kinds of search algorithm (big maze)

Search algorithms	DFS	BFS	Greedy search	A* search
Path cost	155	62	70	62
# of nodes expanded	300	719	121	323

Table 3 Summary of four different kinds of search algorithm (open maze)

Search algorithms	DFS	BFS	Greedy search	A* search
Path cost	195	54	64	54

# of nodes expanded	213	269	123	418

1.2 Penalizing Turns

In this case, the actions of "Pacman" are categorized as: move forward, turn left, and turn right, and different actions can have different costs. First, A* using Manhattan distance heuristic is used to find the optimal path for the two following cases:

Case1: forward movement has cost 2 and any turn has cost 1.

Case2: forward movement has cost 1 and any turn has cost 2.

As different moves have different costs, so a better heuristic function is design to improve the performance of the A* search, instead of using the Manhattan distance as the heuristic, when calculating the heuristic function value of the successors, the following expression is used as the heuristic function:

$$h(\mathrm{n}) = h_o(\mathrm{n}) + t(n)$$

Where $h_o(\mathrm{n})$= the original heuristic function value based on Manhattan distance

$t(n)$= the minimum necessary cost of turning from the node to the reach the goal, which depends on the facing direction of "Pacman" at that node and the position of the goal.

As we can see, $h_o(\mathrm{n})$ is the minimum "moving forward cost" and $t(n)$ is the minimum "turning cost" for "Pacman" to reach the goal from that node, the actual cost won't be smaller than this predicted cost, so this heuristic is admissible.

The results of this section are shown below:

1.2.1 Results of A* using Manhattan distance

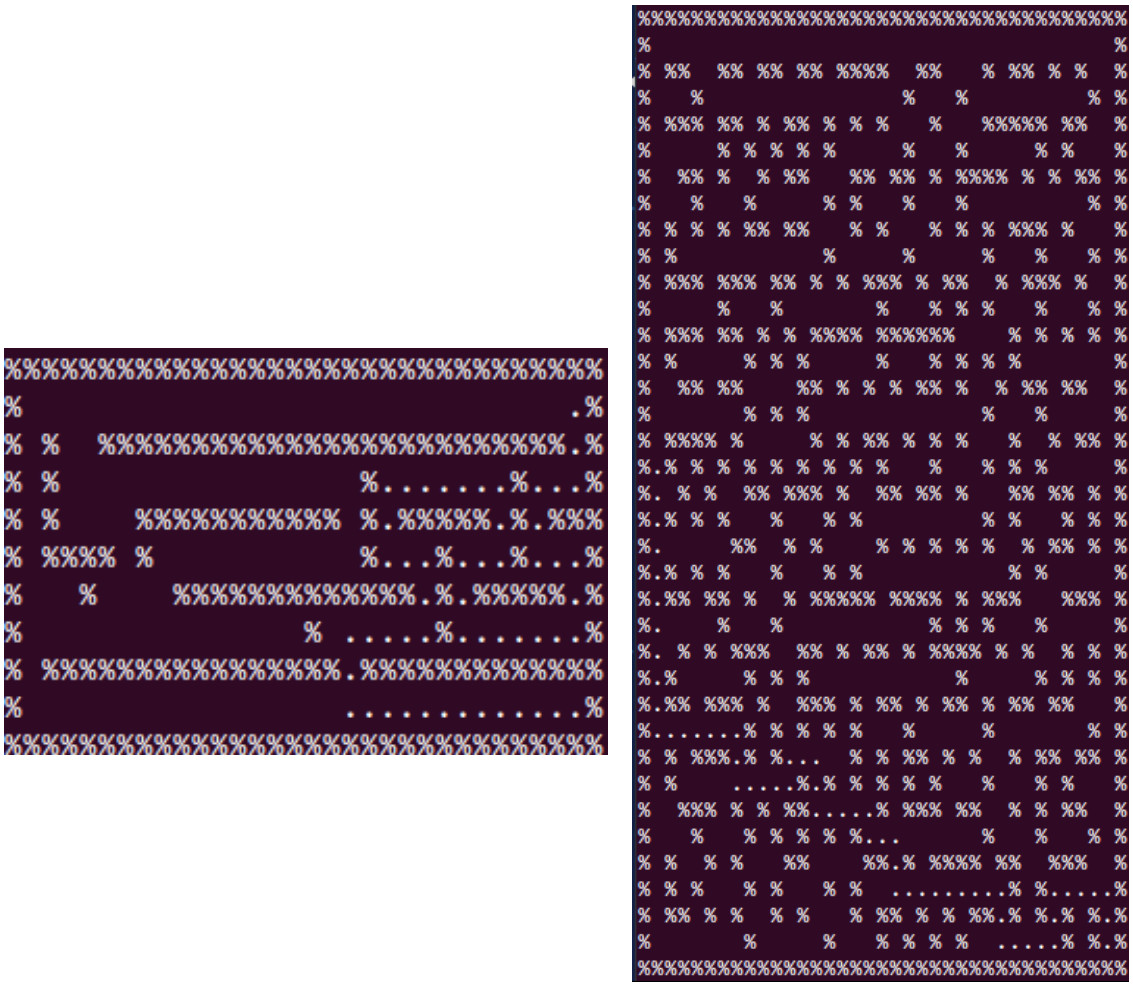

Figure 5 Solution paths for small and big mazes using A* search with Manhattan distance heuristic (case 1)

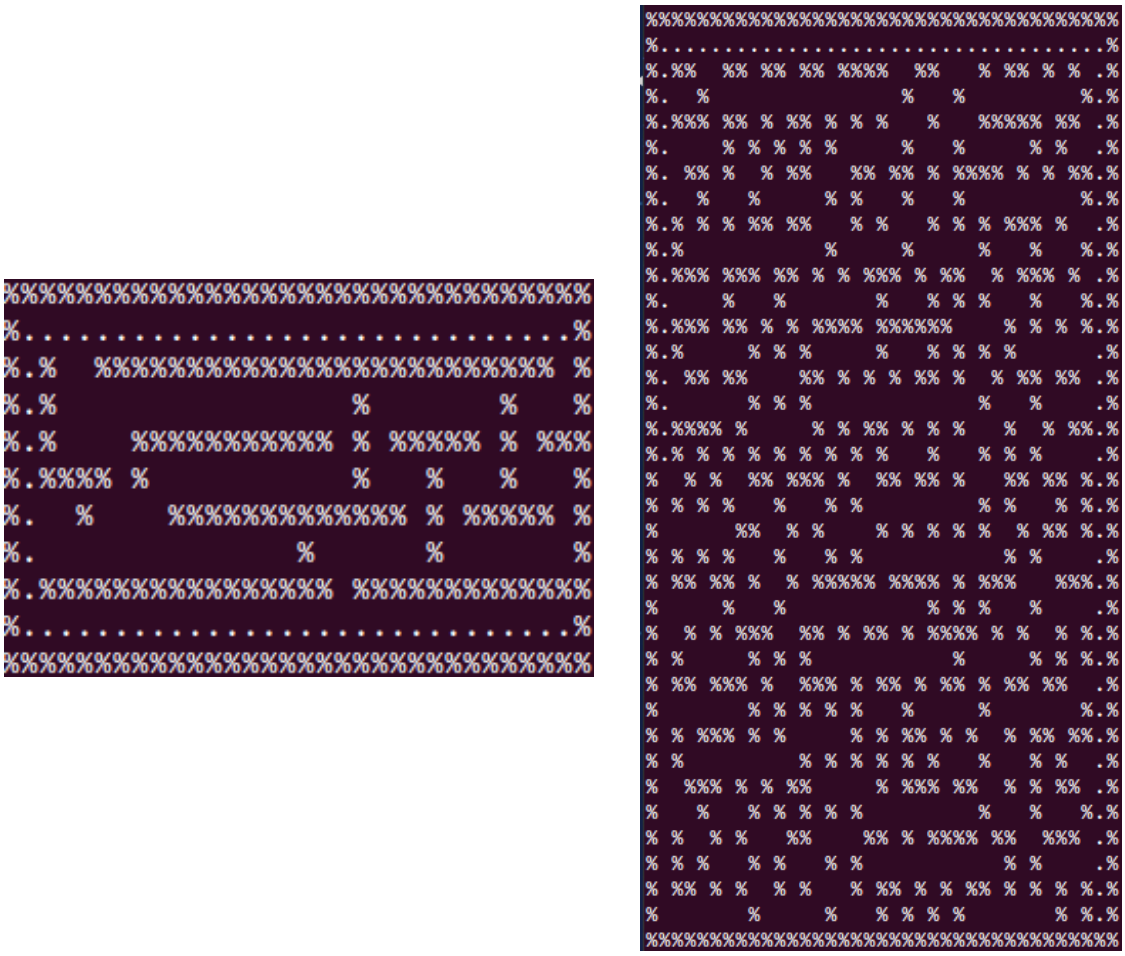

Figure 6 Solution paths for small and big mazes using A* search with Manhattan distance heuristic (case 2)

Table 4 Summary of results using Manhattan distance heuristic (case 2)

Maze size	Small Turns	Big Turns
path cost	120	133
# of nodes expanded	208	664

Table 5 Summary of results using Manhattan distance heuristic (case 2)

Maze size	Small Turns	Big Turns
path cost	74	90
# of nodes expanded	155	583