**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. **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. 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.

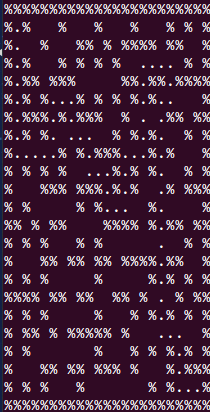
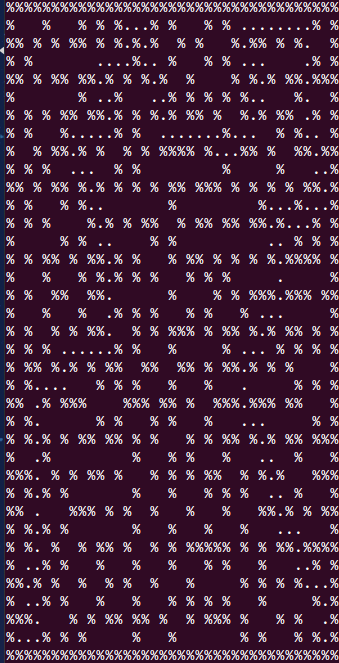
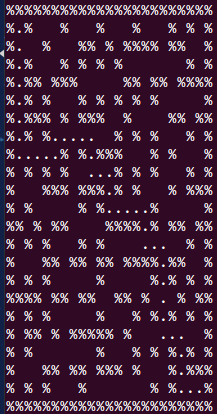
 

Figure 1 Solution paths for medium, big, and open maze using DFS

* + 1. 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.

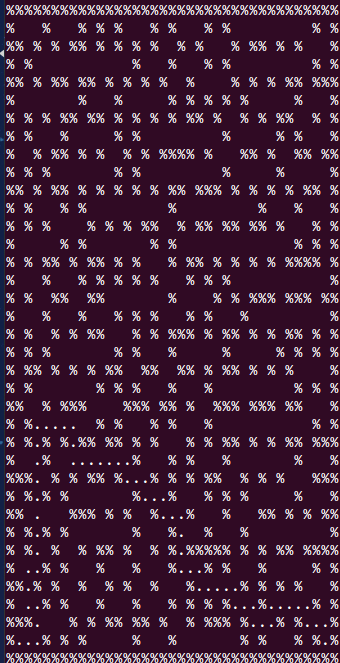
 

Figure 2 Solution paths for medium, big, and open maze using BFS

* + 1. 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.

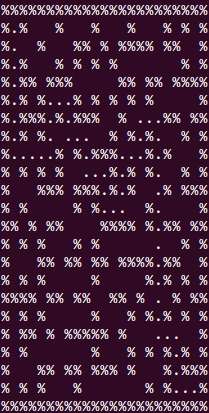
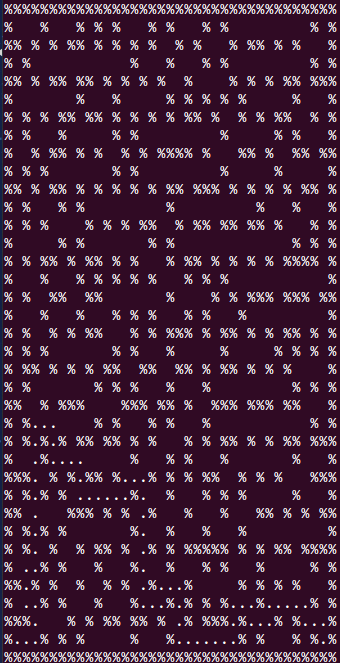
 

Figure 3 Solution paths for medium, big, and open maze using Greedy best-first search

* + 1. 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.

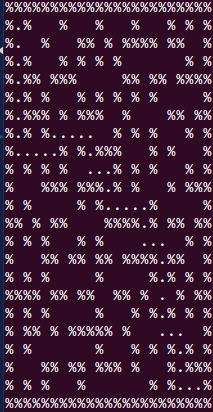
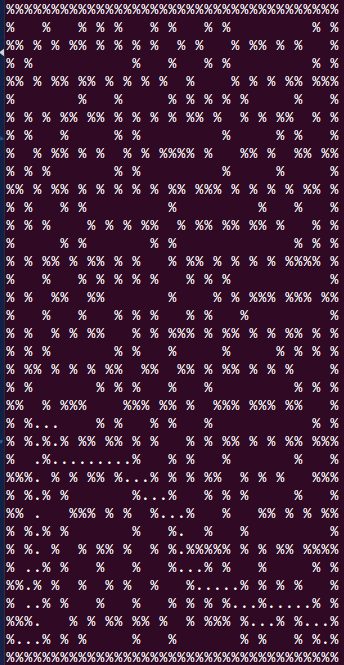
 

Figure 4 Solution paths for medium, big, and open maze using A\* search

* + 1. 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. **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:

Where = the original heuristic function value based on Manhattan distance

= 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, is the minimum “moving forward cost” and 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. Results of A\* using Manhattan distance

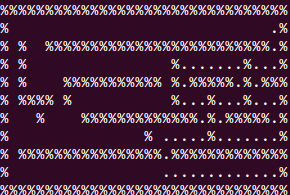
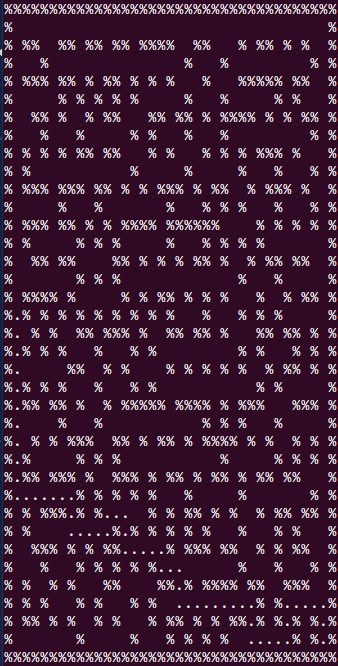
 

Figure 5 Solution paths for small and big mazes using A\* search with Manhattan distance heuristic

(case 1)

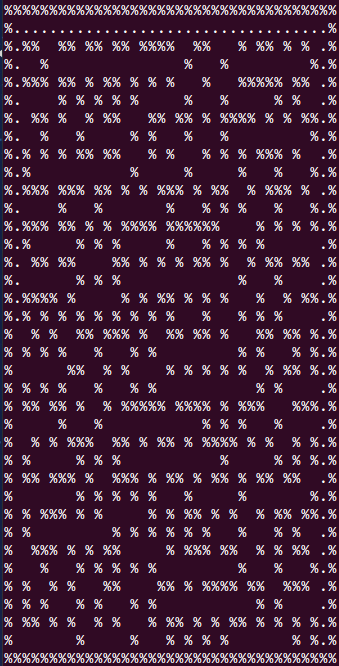
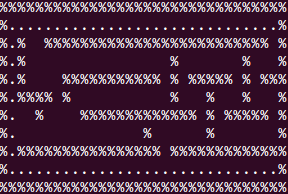


Figure 6 Solution paths for small and big mazes using A\* search with Manhattan distance heuristic

(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 |

* + 1. Results of A\* using more informed heuristic function

Table 6 Summary of results using more informed heuristic (case 1)

|  |  |  |
| --- | --- | --- |
| Maze size | Small Turns | Big Turns |
| path cost |  |  |
| # of nodes expanded |  |  |

Table 7 Summary of results using more informed heuristic (case 2)

|  |  |  |
| --- | --- | --- |
| Maze size | Small Turns | Big Turns |
| path cost |  |  |
| # of nodes expanded |  |  |

* 1. **Pacman with A Ghost**

In this problem, the definition of actions of “Pacman” is the same as section 1.1, but there’s a ghost walking back and forth in horizontal direction in a certain region between two walls, and the ghost moves the same time when “Pacman” moves, if “Pacman” tries to occupy the same square with the ghost or try to swap position with the ghost, then “Pacman” dies and the search fails.

To avoid the death of “Pacman”, we give “Pacman” the ability to avoid the ghost, that is when “Pacman” is going to die if it moves forward, it will move backwards one step, and then it continues to move forward, each of this kind of action costs “Pacman” two more steps, the additional costs will be added to the g(n) of that corresponding node in the A\* search.

Three mazes with ghost are given, and the solution paths found by A\* search using Manhattan distance heuristic are given in the following figures:

Comparisons of the solution cost and the number of nodes expanded of the two cases with and without ghost are listed in the table below:

Table 2 Comparison of the number of costs of the two different cases

|  |  |  |  |
| --- | --- | --- | --- |
| Maze size | Small | Medium | Big |
| With ghost |  |  |  |
| Without ghost |  |  |  |

Table 8 Comparison of the number of expanded nodes of the two different cases

|  |  |  |  |
| --- | --- | --- | --- |
| Maze size | Small | Medium | Big |
| With ghost |  |  |  |
| Without ghost |  |  |  |

* 1. **For Full Credit: A More Informed Heuristic**

In mazes where a lot of walls exist, it is not possible for “Pacman” to move along Manhattan distance to reach the goal, chances are the number of necessary steps is larger than the Manhattan distance from the node to the goal, which means “Pacman” has to detour a little bit to reach the goal, and how long the distance “Pacman” needs to detour depends on the complexity of the maze, assuming that we don’t know the complexity of maze in advance, we can try to multiply the Manhattan distance with a coefficient to calculate the h(n), that is:

Where c=coefficient that is larger than 1.

**1.5 For Extra Credit: A New Maze with More Complicated Ghost Movement**

**PART 2**

**2.1 Search with Multiple Dots**

In this harder scenario, “Pacman” is going to find path to collect multiple goals, the final goal is to reach all the goals.

Because now there are multiple goals, if the heuristic is simply based on the Manhattan distance to a certain goal, the heuristic is likely to underestimate the real cost too much. So a new heuristic is designed for the multiple goals search.

For the new heuristic, g(n) is still the cost from start state to reach the node, when calculating h(n), we find out two “boundary goals” out of the set of the goals, if we set a coordinate to the entire maze, among the goals with the smallest x value (or maybe there’s just one), we take the one with the smallest y value and we called it the “lower left goal”, then we take the “upper right goal” in the same manner. Then we define:

=Min (MDLL, MDUR) + MDLL-UR

MDLL= Manhattan distance from the node to “lower left goal”;

MDUR= Manhattan distance from the node to “upper right goal”;

MDLL-UR= Manhattan distance from “lower left goal” to “upper right goal”.

To achieve the final goal that all the goals are reached, “Pacman” has to reach the “lower left goal” and the “upper right goal”, and the above is the minimum distance the “Pacman” can move to reach these two goals, so in real search, the total distance “Pacman” moved should be larger than the above, is admissible in this case.

The solution paths are shown below:

The summary of the costs and number of expanded nodes are listed in the table below:

|  |  |  |  |
| --- | --- | --- | --- |
| Maze size | Tiny | Small | Medium |
| path cost |  |  |  |
| # of nodes expanded |  |  |  |

**2.2 Suboptimal search**

Even using the A\* searchwith the heuristic described in 2.1, the performance of the search is not efficient, that is because the heuristic still underestimates the real cost when “Pacman” achieves the final goal, then a suboptimal search is preferred, to do that, the following is used:

Where c=coefficient that is larger than 1.

The summary of the costs and number of expanded nodes are listed in the table below:

|  |  |  |  |
| --- | --- | --- | --- |
| Maze size | Tiny | Small | Medium |
| path cost |  |  |  |
| # of nodes expanded |  |  |  |