**CS 440 Assignment 1**

R section 3 credits

Group members:

Chendi Lin, Yulun Yao, Zhuoyue Wang

1. **Overview**

In this assignment, we design and use different searching algorithms to solve "food pellets" puzzles.

In the first part, we need to find the shortest path to the puzzle has only one goal state by a given start state. To approach this problem, we use BFS, DFS, Greedy and A\* search algorithms to find solution paths and their path cost and number of expanded nodes.

In the second part, we are asked to solve the maze with multiple goals. As a starting point, we apply BFS on “tinySearch” to double check that our state representation. Then, we use graph as the maze representation to compress the number of states, and use the total length of the Minimum Spanning Tree of the graph as the heuristic to improve the searching efficiency. The result turns out to be pretty good.

For the extra credit part, we utilize the similar strategy, but with a different way to record the dots collection and use an overestimate heuristic to decrease the computational expense but as a trade off, only suboptimal solution can be found.

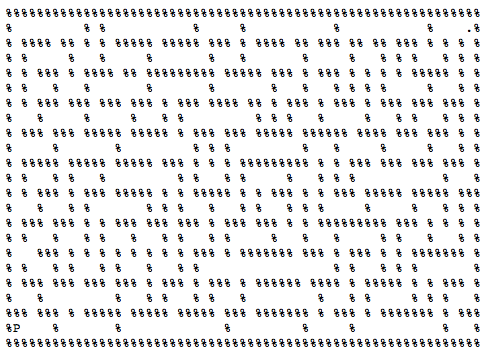
1. **Work distribution**

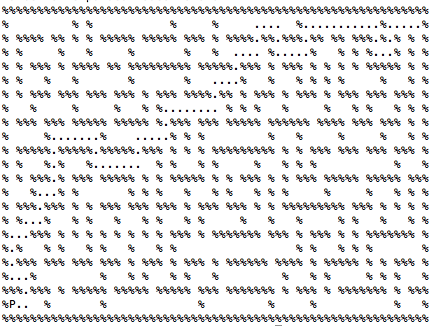
Chendi Lin: Greedy best-first search algorithm & A\* search algorithm, 1.2 multiple dot search, extra credit, report

Yulun Yao: A\* search algorithm, 1.2 multiple dot search, report

Zhuoyue Wang: Breadth-first search & Depth-first search, read\_maze, report

1. **Section 1.1 Basic pathfinding**

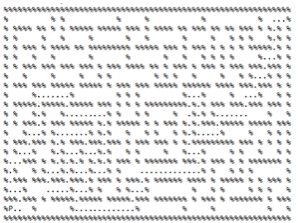
Input: mediumMaze

Breadth-first search:

Solution:

Path Cost: 94

Number of nodes expanded: 610

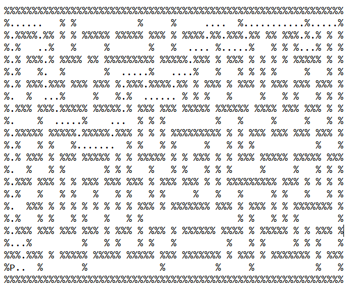


Depth-first search:

Solution:

Path Cost: 162

Number of nodes expanded: 212

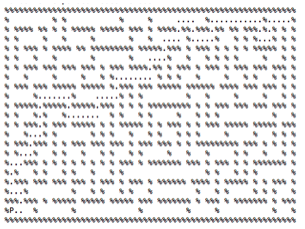


Greedy best-first search:

Solution:

Path Cost:

Number of nodes expanded:

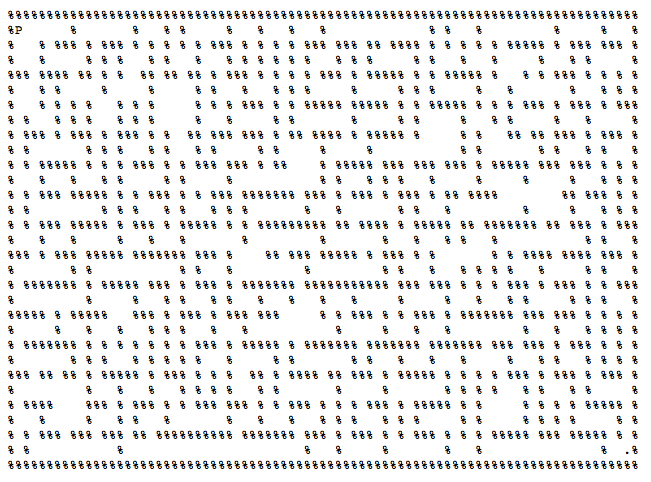


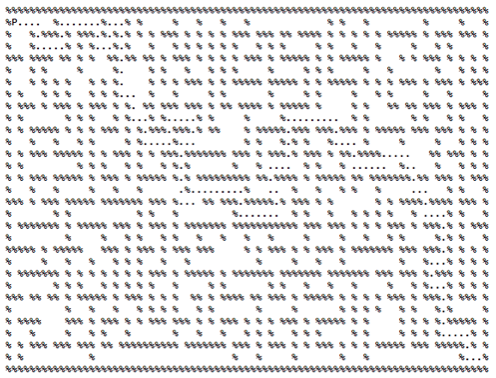
A\* search:

Solution:

Path Cost:

Number of nodes expanded:

Input: bigMaze

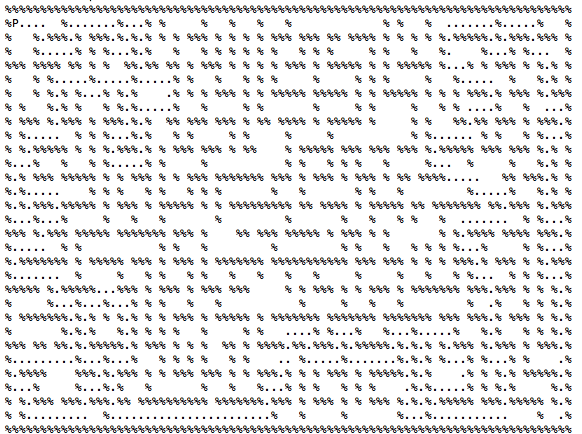


Breadth-first search:

Solution:

Path Cost: 148

Number of nodes expanded: 1256



Depth-first search:

Solution:

Path Cost: 386

number of nodes expanded: 540

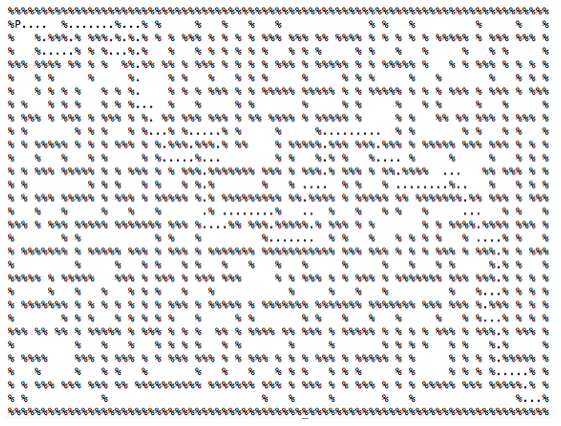


Greedy best-first search:

Solution:

Path Cost: 234

Number of nodes expanded: 297

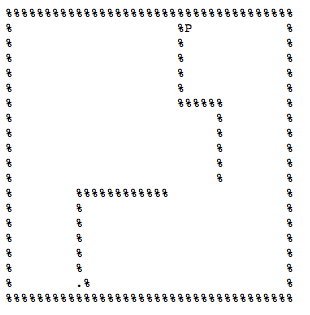


A\* search:

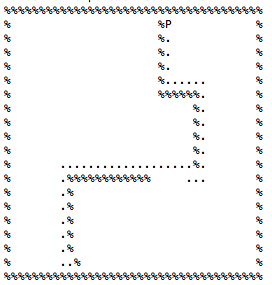
Solution:

Path Cost:148

Number of nodes expanded: 1148



Input: openMaze

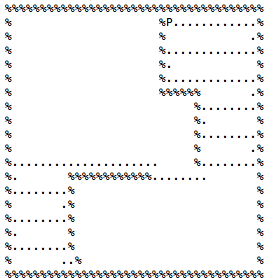


Breadth-first search:

Solution:

Path Cost: 45

Number of nodes expanded: 524

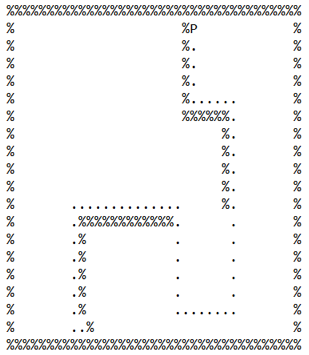


Depth-first search:

Solution:

Path Cost: 125

Number of nodes expanded: 335

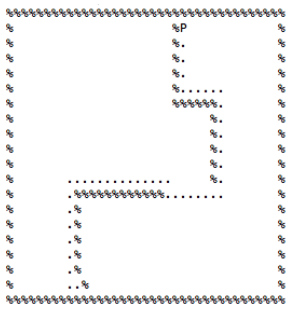


Greedy best-first search:

Solution:

Path Cost:

Number of nodes expanded:



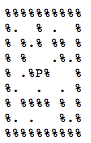
A\* search:

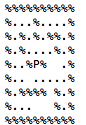
Solution:

Path Cost:

Number of nodes expanded:

1. **Section 1.2 Search with multiple dots**

Breadth-first search for tinySearch



Breadth-first search:

Solution:

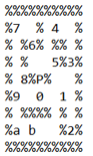
Path Cost: 36

Number of nodes expanded: 51356

A\* Search:

Heuristic Algorithm Description:

For this part, we made some change to improve the efficiency. We firstly set up a graph, with starting point and all goals as the nodes in the graph, and the shortest path length between them as the edge of the graph. The time complexity here is O(n^2) and it is super fast even with “bigDots”. Then instead of having the whole maze as our searching objective, we only search the graph, so the number of all the states becomes way smaller than before.

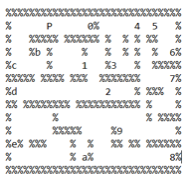
The heuristic here is the total length of the Minimum Spanning Tree (MST) of the graph. It is an admissible heuristic because essentially, after setting up the problem as a graph, we are solving a Travelling Salesman Problem (TSP), where the solution is the shortest path with the salesman only passing each node once. So it is clear that the solutions of TSP is the subset of the solutions of MSt, because in MSP we only need to connect all the nodes. Thus the actual heuristic, which is the result of TSP, is always larger than the estimated heuristic, which is the result of MST. This heuristic is also consistent, because obviously, whenever we expand a new node, a goal is collected, and the total length of MST will be shorter.

Input: tinySearch

Solution:

Path Cost:36

Number of nodes expanded:54

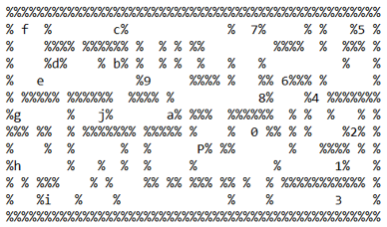


Input: smallSearch

Solution:

Path Cost: 143

Number of nodes expanded: 503

Input: mediumSearch

Solution:

Path Cost:207

Number of nodes expanded:3981

1. **Extra credit: Suboptimal search**

Here, to reduce the computational cost by a huge amount, and because we only care about suboptimal solution, we made some change to both the A\* ‘s structure and the heuristic. Instead of storing which dots have been collected, we store how many dots have been collected. This results in the omission of a lot of states but can lead to a solution. Also, the heuristic is multiplied by a large constant, which made this algorithm kind of similar to greedy first search now because the heuristic takes much more weight that it should in A\*.