Artificial Intelligence (DT8012) Laboration 2 Report

Path Finding

Andreas Häggström

December 3, 2019

Contents

1 Introduction

This lab-report is a part of my studies in the course *Artificial Intelligence* , **DT8012** at Halmstad University. In section 2 the first part of the lab is covered, here the assignment is to implement different path finding algorithms for a map of nodes. Section 3 covers the second part where similar path finding algorithms are to be implemented for a game of poker. Both tasks are written using **Python-3.7.2 64 bit**.

2 Task-1

This task is about implementing different path finding algorithms for a map of nodes. The agents receive following input stated below.

- The map to operate within, in the form of a matrix.
- The coordinates for the start node.
- The coordinates for the goal node.
- For 1b, agents can also receive additional input discussed in section 2.2.

The goal of the task is to find a path between the start and the goal node. The algorithms to be implemented are the following.

- A* using Manhattan distance
- A* using Euclidean distance
- Greedy using Manhattan distance
- Greedy using Euclidean distance
- Breadth First
- Depth First
- Randomized

Here the **A*** algorithms can be explained as the following:

$$f(x, y) = h(x, y) + g(x, y) \tag{1}$$

Here h is the distance to the goal node and g is the cost of getting to the current node. The Manhattan and the Euclidean implementations vary only on how they calculate h [1]. The **Greedy** algorithms are similar to **A*** with the difference being that they don't use the heuristic g. The implementations of Breadth First, **BFS** and Depth First, **DFS** are also similar, even though their outcomes are not. **BFS** is simply a Last-In-First-Out implementation while **DFS** is implemented with First-In-First-Out.

2.1 Task-1a

In this task, various obstacle-nodes are placed in the map. The agent is to navigate between these and find its way to the goal node. The average result of 100 rounds can be seen in Table 2.1.

AgentType	PathLenght	Expanded
A* Manhattan	60.78	790.91
A* Euclidean	60.78	790.91
Greedy-Manhattan	63.92	161.20
Greedy-Euclidean	63.92	161.20
BreadthFirst	60.76	4059.26
DepthFirst	1986.64	5303.61
Random	97.16	4461.56

Where **PathLenght** is the length of the path between the start and the goal node and **Expanded** is the amount of nodes expanded in order to find this path. It is to be noted that **A*** seems to find somewhat longer paths than **BFS** even though they should have the same path lengths.

2.2 Task-1b

This task is similar to the one described in Section 2.1 with the difference being that a larger obstacle is placed in the middle of the map. The result of 100 rounds can be seen in Table 2.2.

AgentType	**PathLenght**	**Expanded**
A* Custom	159.08	2483.30
A* Manhattan	155.92	3686.59
A* Euclidean	155.92	3686.59
Greedy-Manhattan	184.14	2256.53
Greedy-Euclidean	184.14	2256.53
BreadthFirst	154.04	6564.70
DepthFirst	2081.04	5328.54
Random	278.86	6457.71

The columns in this table represent the same as in Table 2.1. It should be noted that this map requires much more nodes expanded for the agents to find a solution. Also no significant difference between the Euclidean and Manhattan implementation of **A*** can be seen neither here or in Table 2.1.

3 Task-2

In this task, different search algorithms were to be implemented for poker. The goal of the task is to win 100$ from your opponent within 4 hands. The result of 100 rounds can be seen in Table 3.

AgentType	**AgentStack**	**OpponentStack**	**PathLength**	**Expanded**	**Wins**	**Hands**
Greedy	501.375	298.625	10.50	5865.48	100	2.15
Greedy Improved	506.775	293.225	11.73	715.91	100	2.18
Depth First	25.000	775.000	24962.39	100002.60	1	24962.39
Breadth First	500.000	299.800	10.33	117775.70	100	2.29
Randomized	273.425	421.225	15.97	100002.82	3	9.23

Something that was noted during testing was that the **Greedy** agents would a small amount of nodes for the most times, but sporadically get a large number of explored nodes. For example, **Greedy Improved** would get a number of expanded nodes of about 100 for the most of the time.

4 Bonus Task

The bonus task this week was 2c. This is implemented in Section 3 as the **Greedy Improved** agent. This agent uses the heuristic of:

$$h = opponentStack - agentStack \tag{2}$$

This heuristic is used together with the one of the regular **Greedy** agents heuristic of *number of hands played*. Together these heuristics makes sure that the agent increases its stack against the opponent while also keeping the amount of hands played low.

References

[1] Archana Singh, Avantika Yadav, and Ajay Rana. K-means with three different distance metrics. *International Journal of Computer Applications*, 67(10), 2013.