Assignment 2

Fredrik Mårtensson

2019-12-01

Contents

Part 1 - Path planning **1**

Part 2 - Poker **3**

Appendix **5**

Part 1 - Path planning

The project is structured to inherit an cost function depending on input into the search algorithm. Every function uses the Queue. According to the table below it is possible to see that BFS were equal mach in g value against A* and random. However the expanded nodes is greatly higher in Random followed by BFS. Conclusion is that Random is bad, BFS is good but compared to A* it is not good enough. However, Greedy search have the lowest expanded nodes next to Special but the path cost increases. Finally we can see that Special performs best in the matter of expanded nodes but due to its static path of avoidance it is not recommended for changing environments. Comparing the heuristics (Observe the images) we can see that manhattan performs better for both A* and Greedy with small margins in A*. However the gap between the expanded nodes is relative high. The special heuristics main target is to go where posX,posY that is the position current position and x, y1,y2 is the given positions of the obstacle. So the following is implemented: If $posX < x$ and $y1 > posY > y2$ then move to the left and multiply with a huge number so it wont go back into the obstacle. Then apply manhattan distance to start searching for the node.

Type	g	Expanded nodes
BFS	134	4764
DFS	1532	7235
Random	530	6577
Greedy euclides	168	1523
Greedy manhattan	138	790
A* euclides	134	2808
A* manhattan	136	2158
Special	144	152

Part 2 - Poker

There is a few differences from the first part and part 2. The inheritances were reduced since an easier way to calculate each cost were found. However this cost a little more performance but not very noticable. In the table below we can see how the different algorithms performs. Interesting enough A* was the worst one with an extreme expanding followed by BFS while focusing on doing as many bets as possible. They all won even though a theory that Random would fail at least once. This could have been prevented since none of the algorithms are allowed to visit a node more than once. So if it is visited they cant visit it again. Limitations on depth, hand draw limitations and random step exists and once we find the goal we break and return the path.

Two different heuristics were implemented as complement (bonus part): One works towards how many bets that have been done and the second how much money the agent have and the sum in the pot. The interesting about these tables is little the A* and greedy expands when searching for the biggest pot. However A* have a worse g value but greedy performs overall better. This test was done during 30 runs and the values are average of each parameter. By using the biggest pot we reduced the total expanded nodes from 173256.7 to 33676.1

Using MANY as a heuristic the g value is lowered and performs overall better but the search space is higher. On the other hand BIG lowerest the search space but increases the g value

Using MANY as heurisic (default):

Type	g	Expanded nodes	Win raterounds
BFS	10.3	28124.3	30\30
DFS	15.2	1590.6	30\30
Random	14.9	680.6	30\30
Greedy	17.3	421.2	30\30
A*	10.8	142440	30\30

Using BIG pot as heurisitc:

Type	g	Expanded nodes	Win raterounds
BFS	10.4	30669.4	30\30
DFS	15.0	1866.9	30\30
Random	16.4	860.7	30\30
Greedy	15.2	89.8	30\30
A*	15.9	79.3	30\30

Appendix

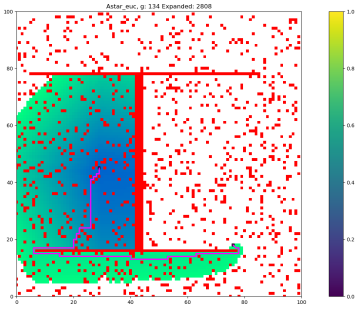

Figure 1: A* with euclides heuristics

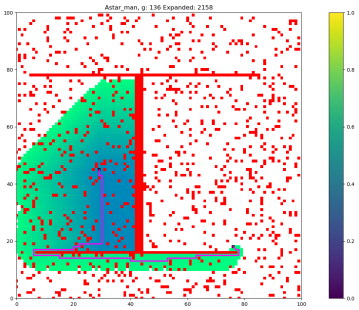

Figure 2: A* with manhattan heuristics

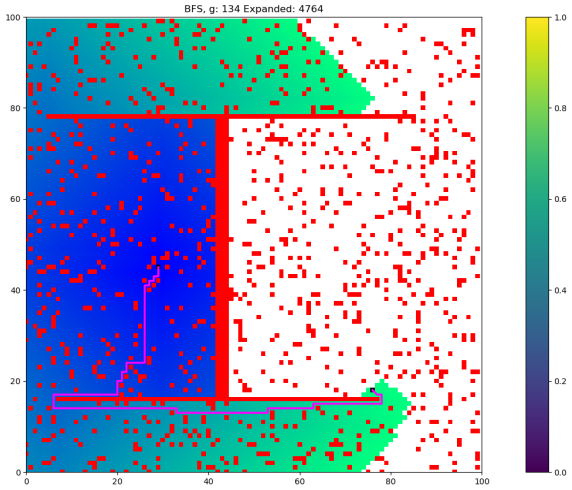

Figure 3: BFS

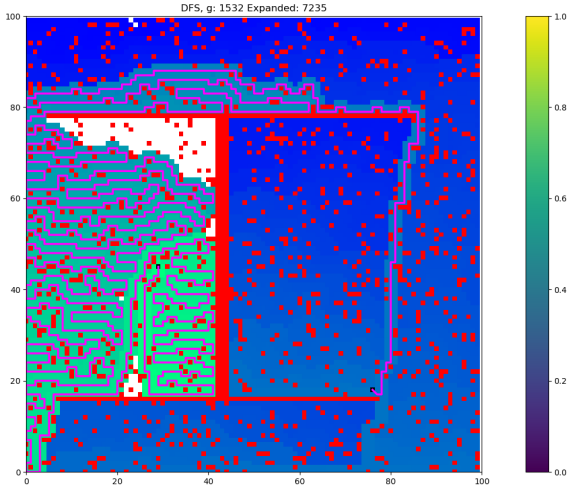

Figure 4: DFS

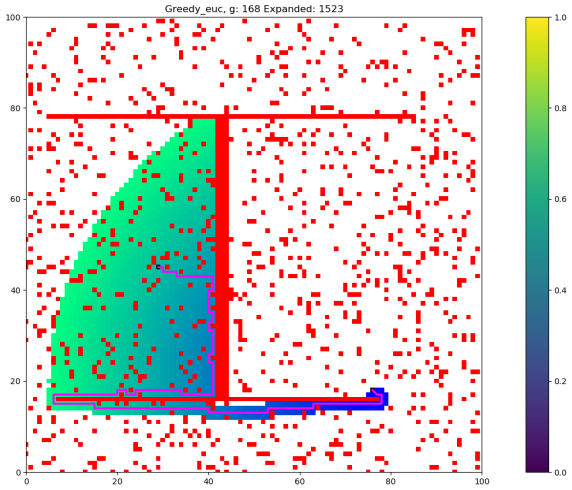

Figure 5: Greedy with euclides heuristics

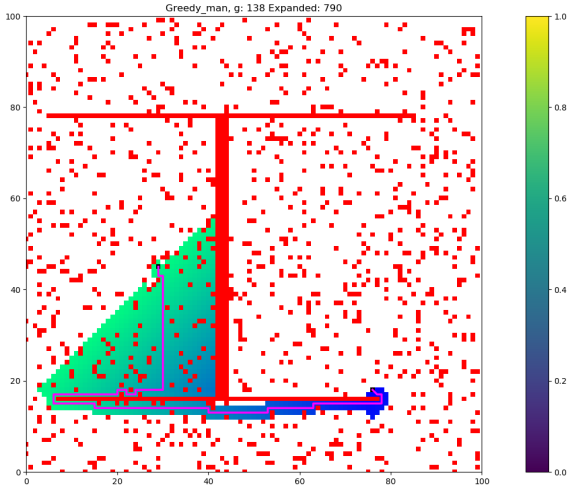

Figure 6: Greedy with euclides heuristics

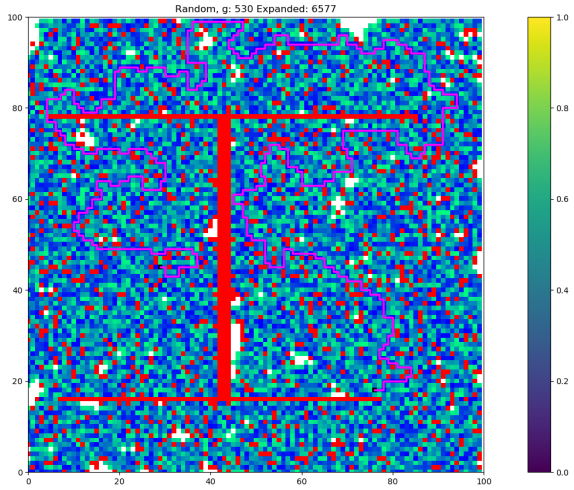

Figure 7: Random with euclides heuristics

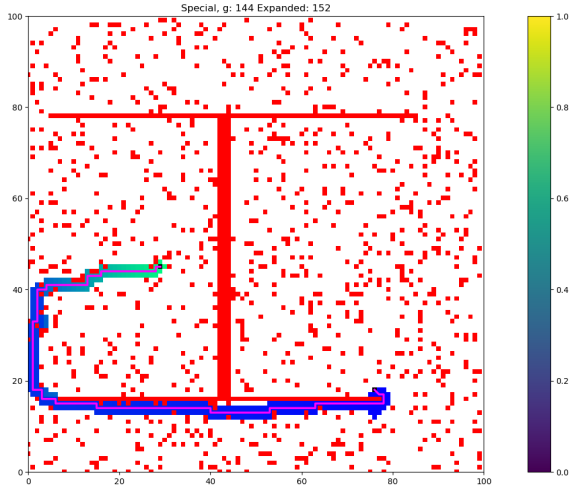

Figure 8: Special with euclides heuristics