Assignment 2

Fredrik Mårtensson

2019-12-03

Contents

Part 1 - Path planning

The project is structured to inherit an cost function depending on class parameter into the search algorithm. According to the table below it is possible to see that BFS roughly as good in cost against A. *However the expanded nodes is greatly higher in DFS followed by Random. Conclusion is that DFS and 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	cost	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 removed since an easier way to calculate each cost were found. This lowered complexity and allowed better readability. 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 one of the worst algorithm next to DFS with an extreme expanding followed by BFS. They all won even though it was suspected 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 that node again. Limitations on depth, hand draw and random step exists and once we find the goal we break and return the path. If no goal is found it will return -1 and skip storing data from that round.

Two different heuristics were implemented including the bonus part: One works towards how many bets that have been done, as it was described in the lab and the second how much money the agent have and the sum in the pot. The interesting about these tables is how the A* and greedy expands when searching for the biggest pot. However Greedy have a worse cost but lower search space. This test was done during 100 runs and the values are average of each parameter.

Using MANY as a heuristic the cost is lowered and performs overall better to find optimal path but the search space is higher. On the other hand BIG lowerest the search space but with increased cost.

Type	heuristic	Cost	Expanded nodes	Win raterounds
BFS	None	10.35	27145.29	100\100
DFS	None	19.7	130546.25	100\100
Random	None	16.36	938.25	100\100
Greedy	MANY bids	17.24	779.95	100\100
Greedy	BIG pot	16.8	43.53	100\100
A*	MANY bids	10.46	125376.066	100\100
A*	BIG pot	16.49	50.07	100\100

Appendix

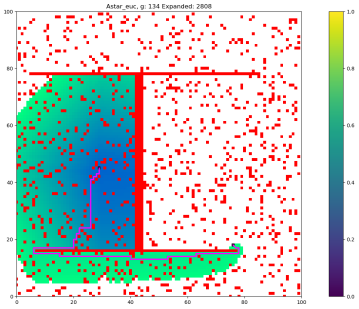

Figure 1: A* with euclides heuristics

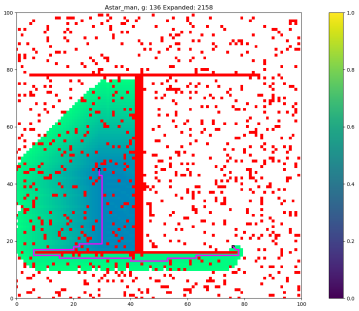

Figure 2: A* with manhattan heuristics

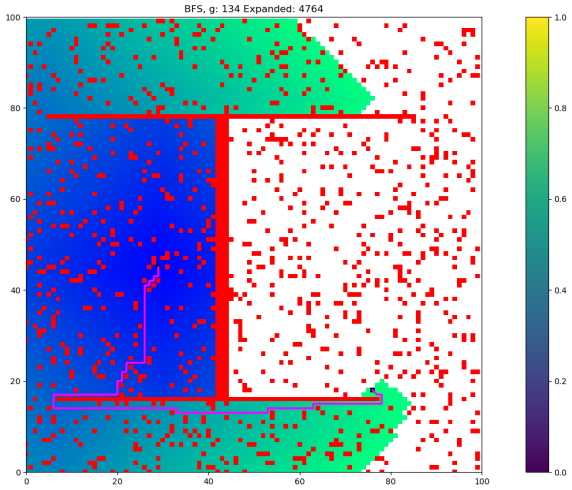

Figure 3: BFS

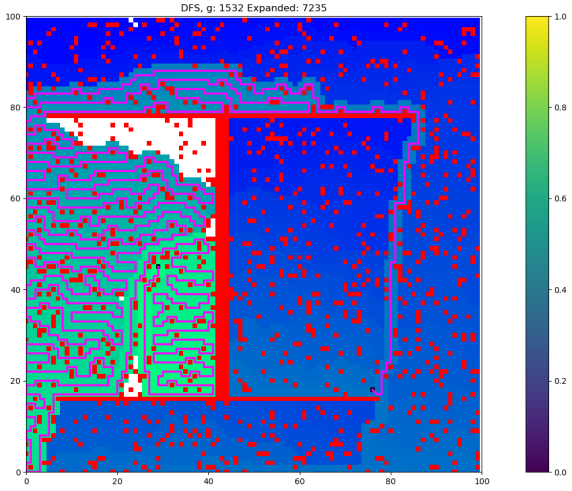

Figure 4: DFS

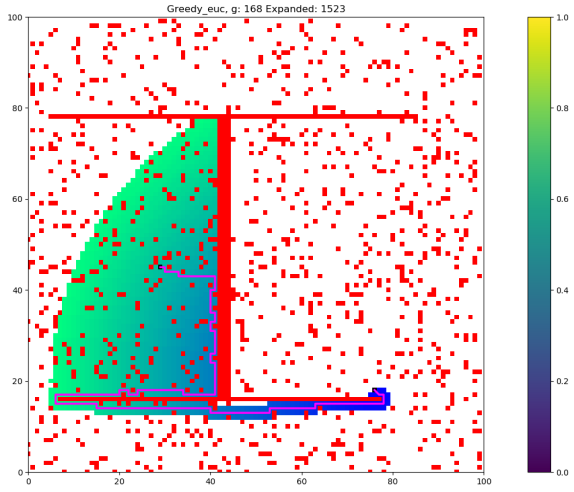

Figure 5: Greedy with euclides heuristics

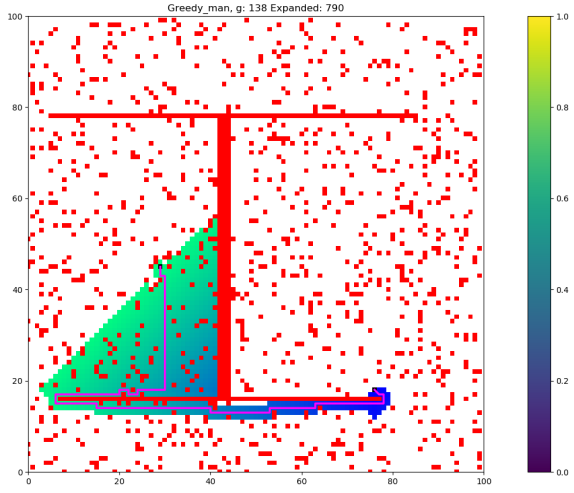

Figure 6: Greedy with euclides heuristics

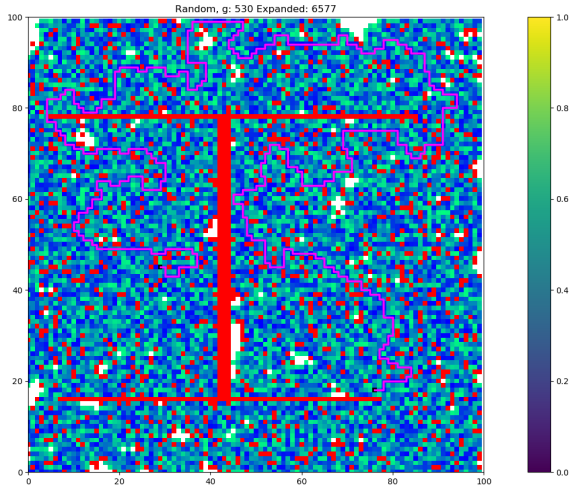

Figure 7: Random with euclides heuristics

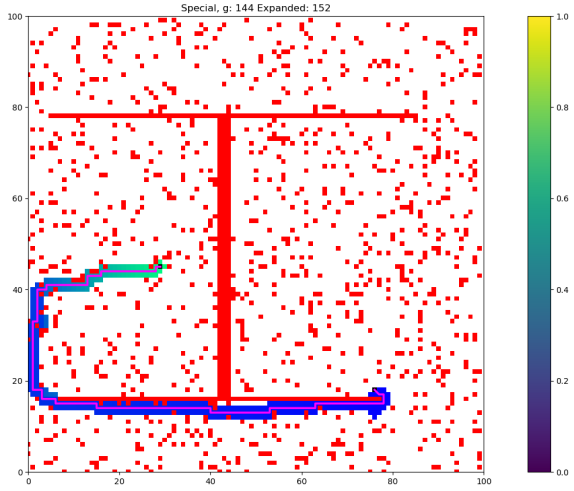

Figure 8: Special with euclides heuristics