Daniel Ruegamer
Thursday April 6, 2017

Exercise 3.3

Directions: Perform various searches on the following grid

		G					
			S				

a) Depth First Search

18	17	16	15	14	13	12	11
19	20	21	22	23	24	25	10
		G30	29	28	27	26	9
							8
		3	2	4	5	6	7
			S1				

b) Best First Search

Manhattan Distance Grid

4	3	2	3	4	5	6	7
3	2	1	2	3	4	5	6
2	1	**G**0	1	2	3	4	5
3	2						6
4		2	3	4	5	6	7
5	4		**S**4	5	6		8
6	5	4	5	6	7	8	9
7	6	5	6	7	8	9	10

Order of Nodes Expanded Grid

		G14	13	12	11	10	9
							8
		3	2	4	5	6	7
			S1				

c) Heuristic Depth First Search

		G14	13	12	11	10	9
							8
		3	2	4	5	6	7
			S1				

d) A*
Tiebreaker Policy: FIFO then Up Right Down Left
Path highlighted in red

f12	f12						
8	9	Gf10					
7	8						
6		X	1	2	3	4	f12
5	4		S0	1	2		
4	3	2	1	2	3	f12	
f12	4	3	2	3	f12		

Heurstic

4	3	2	3	4	5	6	7
3	2	1	2	3	4	5	6
2	1	G0	1	2	3	4	5
3	2						6
4		2	3	4	5	6	7
5	4		S4	5	6		8
6	5	4	5	6	7	8	9
7	6	5	6	7	8	9	10

Assume cells are indexed with the origin at bottom left, and indices start at 1
Format is (horizontal, vertical)

FINAL PATH:
start(4,3) -> (4,2) -> (3,2) -> (2,2) -> (2,3) -> (1,3) -> (1,4) -> (1,5) -> (1,6) -> (2,6) -> goal(3,6)

e) Dynamic Programming

Cost-to-Goal Grid

4	3	2	3	4	5	6	7
3	2	1	2	3	4	5	6
2	1	G0	1	2	3	4	5
3	2						6
4		12	11	10	9	8	7
5	6		S10	11	10		8
6	7	8	9	10	11	10	9
7	8	9	10	11	12	11	10

Path highlighted in red

Assume cells are indexed with the origin at bottom left, and indices start at 1
Format is (horizontal, vertical)

Tiebreak Policy: Up, Right, Down, Left

FINAL PATH:
start(4,3) -> (4,2) -> (3,2) -> (2,2) -> (2,3) -> (1,3) -> (1,4) -> (1,5) -> (1,6) -> (2,6) -> goal(3,6)

f) I would say that, due to the small nature of this problem, that finding the least cost path is of higher priority than saving computational space. Dynamic Programming would likely be the best in this respect. The problem is small enough to hold the cost-to-goal values in memory, and we also get the least cost path as a result.

g) Both Depth-First Search and Dynamic Programming would have issues with an infinitely extended graph. Depth-First has a problem when it chooses a path that is not in the same direction as the goal. It will move infinitely away from the goal in this case. It is clear to see how this happens with in problem (a), in which the search only finds the goal because it is confined into a space. Dynamic Programming has an issue because it must hold the cost-to-goal for EVERY node in the space, and we obviously do not have infinite memory. Again, assuming we are prioritizing the finding of the least cost path, A* would be the best. It does not require storing large amounts of data, and operates on a frontier basis. In this respect, A* creates it's own boundaries, and an infinite space becomes artificially finite.