Dylan Wulf
CSC380: Artificial Intelligence
Project 1: Searching
February 12, 2017

1.

	BFS			DFS			A* Straight Line			A* Taxi Distance		
Map	Nodes	Time	Cost	Nodes	Time	Cost	Nodes	Time	Cost	Nodes	Time	Cost
1	8	0.3 ms	7	8	0.4 ms	7	8	0.4 ms	7	8	0.4 ms	7
2	15	0.4 ms	14	15	0.5 ms	14	15	0.5 ms	14	15	0.5	14
3	33	0.6 ms	14	21	0.6 ms	14	27	1.0 ms	14	29	0.8 ms	14
4	45	1.1 ms	22	44	1.1 ms	26	45	1.4 ms	22	45	1.1 ms	22
5	289	2.6 ms	15	64	1.2 ms	21	81	2.1 ms	15	47	1.1 ms	15
6	*	*	*	*	*	*	*	*	*	*	*	*
7	45	0.8	–	39	0.9 ms	–	45	1.2 ms	–	39	1.0 ms	–
8	30	0.6 ms	21	18	0.5 ms	23	21	0.6 ms	15	20	0.6 ms	15

*I killed the map6 tests after they ran for 30 minutes, so I did not get any data.
–Map 7 had no solution, which my program correctly computed
Time was computed with the Java method System.nanoTime().

2. The first heuristic I used was the the straight line distance between a node and the goal. It uses the distance formula, sqrt($(x2-x1)^2 + (y2-y1)^2$) This is admissible because a straight line is the absolute shortest distance possible between two points, so there is no way the path could cost any less than that distance.
The second heuristic I used was the "taxi distance," also known as "snake distance" or "Manhattan distance." It is usually bigger than the straight line distance, but it is still admissible because it describes the minimum number of steps needed to get from one point to another on a grid, if each square in the grid is open and has a path cost of 1.

3. Any situation where the search tree could go down infinitely could cause DFS to fail, but not BFS. This is because BFS expands the shallowest unexpanded node first, which means every single node of every single layer of the search tree is explored before it goes on to the next layer. This guarantees that a solution will be found eventually, if one exists. But DFS can get stuck in one branch that goes down infinitely, and it may never find the solution since the solution could be in a different branch. One example of a problem that could cause this would be an infinite maze. At each point there are only 4 options: up, down, left, right; but these options could be expanded infinitely. BFS would eventually get to the goal, but DFS could get stuck going in one direction forever.

(4 and 5 on next page)

4. Any situation where the search tree has an infinite branching factor but a finite tree depth could cause BFS to fail where DFS would succeed. BFS would be stuck infinitely on the first level, but DFS could traverse every node in every level and will eventually reach the solution if there is one. One example of a problem like this could be a jigsaw puzzle that has infinite possible pieces, but the puzzle board size is finite. BFS would be stuck looking at every single possible piece and would never get past the first piece. But DFS tries to fill in the entire board before looking for another solution, and eventually it will find the correct solution.

5. A* will not be beneficial over BFS in any problem where there is only one path to the solution.