Ryan Phillips
April 18, 2014
CS 331 Project One

Methodology

All four algorithms were implemented in Python. They all used the same type of tree as a data structure, but only the necessary information was added to it for each algorithm (such as depth and h()).

DFS struggled to find a solution, unless that solution could be found on the first path it tried. I increased the max for expanded nodes (rather than depth, because I did not actually calculating depths while running DFS) all the way to 10,000 and it still didn't finish.

For A-star I just included an additional piece of info in the node object. For each new node/state, a measure of closeness to the final state was calculated by checking to see how many squares were currently in the correct spot. Then, after get successors was called, the one with the highest score (including all the others already in the fringe) was chosen next. There are other heuristics that could have been used, such as calculating how far away each item is from it's current location. But the one I choose felt like a pretty good balance of speed, simplicity, and utility.

Results

Test Case	Algorithm	Expanded Nodes	Length of Solution Path	Time (s)
1	bfs	7	3	5.87E-04
2	bfs	77	5	7.87E-03
3	bfs	506	7	7.60E-02
1	dfs	2	3	9.58E-05
2	dfs	error	error	error
3	dfs	error	error	error
1	iddfs	2	3	7.79E-04
2	iddfs	101	5	5.92E-02
3	iddfs	120	7	2.36E-01
1	astar	2	3	1.49E-04
2	astar	4	5	4.35E-04
3	astar	6	7	6.43E-04

Discussion

I don't know if if I made a mistake, but I was surprised that dfs wasn't working for this problem. I tried it with max iterations set at 10000, and it still didn't find a solution for the 2nd and 3rd test cases.

The most difficult part of this assignment for me was writing the data structure to be used by the algorithms. I quickly put together the framework, and found both iterative and

recursive means of arriving at a solution for bfs/dfs without using the psuedo code. But then I learned the hard way that I had no means of saving/reading/printing the solution.I needed a well connected data structure so that I could trace my way back from the final node to the start upon completion.

So after multiple rewrites, I was able to get it mostly working.

Conclusion

It makes sense that astar performs better than the rest for this problem, since we have a good heuristic (simple and informative). I would like to try these algorithms on another problem to see if the relative usefulness shifts.