CS331Assignment #1
Uninformed and Informed Search

Bryant Hayes

Methodology

For this assignment three different experiments were run. The first was a simple test case that required moving three of each missionary and cannibals from one side to the other using a boat with a capacity of two. The second was similar but with 10 missionaries and 8 cannibals and the third used 100 missionaries and 90 cannibals. Below is a list of parameters set for the three test cases.

- DFS & IDDFS: Depth limit of 500 because completing a search of that size would already take an unrealistic amount of time.
- BFS: n/a
- A*: I am using a heuristic found by removing the constraint that says you cannot have more cannibals on one side than missionaries. By removing this constraint the total number of moves needed would be:

  (# of missionaries + cannibals on opposite shore) – 1

- This is because one person must remain on boat the whole time so each trip results in one more missionary/cannibal getting to the new shore. The minus one is because on the final trip two of them can get off. This heuristic is admissible because it is guaranteed to take equal to or more moves than this with the constraint (missionaries>cannibals) active.

In regards to my implementation of these algorithms, I used Python2.7 and made a "Node" class that was used to spawn instances of nodes, which represented various states, that problem could be in. These states have different properties and internal variables used to store data about a given state. For my closed list, used in each test, I used a python dictionary, which is implemented like a hash table and has close to constant look up time.

Results

	BFS	DFS	IDDFS	A*
3x3	Nodes Expanded: 14 Nodes in path: 11	Nodes Expanded: 11 Nodes in path: 11	Nodes Expanded: 106 Nodes in path: 11	Nodes Expanded: 13 Nodes in path: 11
10x8	Nodes Expanded: 72 Nodes in path: 33	Nodes Expanded: 62 Nodes in path: 47	Nodes Expanded: 2888 Nodes in path: 33	Nodes Expanded: 68 Nodes in path: 33
100x90	Nodes Expanded: 2184 Nodes in path: 377 Time: 0.1s	Nodes Expanded: 124120 Nodes in path: 499 Time:	Nodes Expanded: 18467204 Nodes in path:377 Time: 12m11s	Nodes Expanded: 2180 Nodes in path: 377 Time: 0.124s

Discussion

My results are as I expected for the most part. I knew that IDDFS would expand many more nodes that the other algorithms and I also new BFS would find the optimal solution where DFS might not. I was surprised that DFS didn't expand more nodes, but after drawing out the tree for the 3x3 test I saw that DFS had a pretty straight path down to a solution, but it was definitely not the optimal one. I was also surprised to see that most nodes only had one or two possible successors so there were less nodes expanded than I anticipated because of that.

An interesting thing I noticed in regards to the A* search was that when I squared the (# of people on opposite shore) before subtracting one in the heuristic, I got a much better result. In all the cases I got the same solutions but in nearly half the nodes expanded. The problem is that squaring that term makes the heuristic non-admissible. I'm sure it must just be luck that it works so well.

Conclusion

In conclusion I found that A* definitely performed the best, but it also required an informed heuristic from external knowledge about the problem so it makes since that it is the best. IDDFS and BFS both got optimal solutions as well, but I would say BFS was better than IDDFS for this problem because IDDFS took much longer to run and expanded a massive amount of nodes. DFS was pretty fast, but that is because it found a non-optimal solution. These results were pretty much as I expected with the exception of IDDFS taking as long as it did, I thought I had optimized well enough to speed it up to under ten minutes, but it takes about twelve.

Optimal Paths

3x3:
[1, 1], [1, 0], [0, 2], [0, 1], [2, 0], [1, 1], [2, 0], [0, 1], [0, 2],
[1, 0], [1, 1]

10x8:
[2, 0], [1, 0], [1, 1], [1, 0], [2, 0], [1, 0], [1, 1], [1, 0], [2, 0],
[1, 0], [1, 1], [1, 0], [2, 0], [1, 0], [1, 1], [1, 0], [2, 0], [1, 0],
[1, 1], [1, 0], [2, 0], [1, 0], [1, 1], [1, 0], [2, 0], [1, 0], [1, 1],
[1, 0], [2, 0], [1, 0], [2, 0], [1, 0], [1, 1]

100x90:
[2, 0], [1, 0], [2, 0], [1, 0], [2, 0], [1, 0], [2, 0], [1, 0], [2, 0],
[1, 0], [2, 0], [1, 0], [2, 0], [1, 0], [2, 0], [1, 0], [2, 0], [1, 0],
[1, 1], [1, 0], [2, 0], [1, 0], [1, 1], [1, 0], [2, 0], [1, 0], [1, 1],
[1, 0], [2, 0], [1, 0], [1, 1], [1, 0], [2, 0], [1, 0], [1, 1], [1, 0],
[2, 0], [1, 0], [1, 1], [1, 0], [2, 0], [1, 0], [1, 1], [1, 0], [2, 0],
[1, 0], [1, 1], [1, 0], [2, 0], [1, 0], [1, 1], [1, 0], [2, 0], [1, 0],
[1, 1], [1, 0], [2, 0], [1, 0], [1, 1], [1, 0], [2, 0], [1, 0], [1, 1],
[1, 0], [2, 0], [1, 0], [1, 1], [1, 0], [2, 0], [1, 0], [1, 1], [1, 0],
[2, 0], [1, 0], [1, 1], [1, 0], [2, 0], [1, 0], [1, 1], [1, 0], [2, 0],
[1, 0], [1, 1], [1, 0], [2, 0], [1, 0], [1, 1], [1, 0], [2, 0], [1, 0],
[1, 1], [1, 0], [2, 0], [1, 0], [1, 1], [1, 0], [2, 0], [1, 0], [1, 1],
[1, 0], [2, 0], [1, 0], [1, 1], [1, 0], [2, 0], [1, 0], [1, 1], [1, 0],
[2, 0], [1, 0], [1, 1], [1, 0], [2, 0], [1, 0], [1, 1], [1, 0], [2, 0],
[1, 0], [1, 1], [1, 0], [2, 0], [1, 0], [1, 1], [1, 0], [2, 0], [1, 0],
[1, 1], [1, 0], [2, 0], [1, 0], [1, 1], [1, 0], [2, 0], [1, 0], [1, 1],
[1, 0], [2, 0], [1, 0], [1, 1], [1, 0], [2, 0], [1, 0], [1, 1], [1, 0],
[2, 0], [1, 0], [1, 1], [1, 0], [2, 0], [1, 0], [1, 1], [1, 0], [2, 0],
[1, 0], [1, 1], [1, 0], [2, 0], [1, 0], [1, 1], [1, 0], [2, 0], [1, 0],
[1, 1], [1, 0], [2, 0], [1, 0], [1, 1], [1, 0], [2, 0], [1, 0], [1, 1],
[1, 0], [2, 0], [1, 0], [1, 1], [1, 0], [2, 0], [1, 0], [1, 1], [1, 0],
[2, 0], [1, 0], [1, 1], [1, 0], [2, 0], [1, 0], [1, 1], [1, 0], [2, 0],
[1, 0], [1, 1], [1, 0], [2, 0], [1, 0], [1, 1], [1, 0], [2, 0], [1, 0],
[1, 1], [1, 0], [2, 0], [1, 0], [1, 1], [1, 0], [2, 0], [1, 0], [1, 1],
[1, 0], [2, 0], [1, 0], [1, 1], [1, 0], [2, 0], [1, 0], [1, 1], [1, 0],
[2, 0], [1, 0], [1, 1], [1, 0], [2, 0], [1, 0], [1, 1], [1, 0], [2, 0],
[1, 0], [1, 1], [1, 0], [2, 0], [1, 0], [1, 1], [1, 0], [2, 0], [1, 0],
[1, 1], [1, 0], [2, 0], [1, 0], [1, 1], [1, 0], [2, 0], [1, 0], [1, 1],
[1, 0], [2, 0], [1, 0], [1, 1], [1, 0], [2, 0], [1, 0], [1, 1], [1, 0],
[2, 0], [1, 0], [1, 1], [1, 0], [2, 0], [1, 0], [1, 1], [1, 0], [2, 0],
[1, 0], [1, 1], [1, 0], [2, 0], [1, 0], [1, 1], [1, 0], [2, 0], [1, 0],
[1, 1], [1, 0], [2, 0], [1, 0], [1, 1], [1, 0], [2, 0], [1, 0], [1, 1],
[1, 0], [2, 0], [1, 0], [1, 1], [1, 0], [2, 0], [1, 0], [1, 1], [1, 0],
[2, 0], [1, 0], [1, 1], [1, 0], [2, 0], [1, 0], [1, 1], [1, 0], [2, 0],
[1, 0], [1, 1], [1, 0], [2, 0], [1, 0], [1, 1], [1, 0], [2, 0], [1, 0],
[1, 1], [1, 0], [2, 0], [1, 0], [1, 1], [1, 0], [2, 0], [1, 0], [1, 1],
[1, 0], [2, 0], [1, 0], [1, 1], [1, 0], [2, 0], [1, 0], [1, 1], [1, 0],
[2, 0], [1, 0], [1, 1], [1, 0], [2, 0], [1, 0], [1, 1], [1, 0], [2, 0],
[1, 0], [1, 1], [1, 0], [2, 0], [1, 0], [1, 1], [1, 0], [2, 0], [1, 0],
[1, 1], [1, 0], [2, 0], [1, 0], [1, 1], [1, 0], [2, 0], [1, 0], [1, 1],
[1, 0], [2, 0], [1, 0], [1, 1], [1, 0], [2, 0], [1, 0], [1, 1], [1, 0],

[2, 0], [1, 0], [1, 1], [1, 0], [2, 0], [1, 0], [1, 1], [1, 0], [2, 0], [1, 0], [1, 1], [1, 0], [2, 0], [1, 0], [2, 0], [1, 0], [1, 1]