Lance Umagat
CS331 Intro to AI
4/20/17

Cannibals and Missionaries

Methodology

For this assignment I ran each of the three test cases against BFS, DFS, IDDFS, and A* to test the traits of each of the search algorithms. This was done under the various intensity levels of the Cannibals and Missionaries puzzle. Also because BFS, DFS, and IDDFS are uninformed and A* is an informed algorithms then in accordance with GraphSearch and Expand functions from class we know that these all share numerous commonalities.

Method Overview

For the uninformed searches I implemented the actual parameters in each search function. Within each of search algorithms I included a fringe, initial state, and the final goal state. As for the fringe, it was used in the uninformed searches using the deque object from python collections library. This library contains deque.popleft() and deque.pop() which made for easy access to the data structure from either side. As for the informed A* search, I ended up using a priority queue that implemented heapq to push and pop the state and state cost. For this the initial and goal states are implemented using Nodes that contain attributes. Since we iterate through the search and expand functions like state to the left/right, parent state, action, state depth, and cost. Lastly a closed list is used in conjunction with a hash table to store states that have been checked already.

Algorithm Discussion

Due to there being a time consumption problem with DFS and IDDFS I decided to set an arbitrary depth limit of around 400. This is because much more after that would be problematic.

As for A*, the heuristic is based on the fact that the game will end when there are more cannibals then missionaries on one of the banks. Thus because of this and the fact that the boat would need at least one person to move it back then the heuristic becomes:

$$\mathrm{h(n)} = \frac{Individuals\ on\ Starting\ bank}{Boat\ Size}$$

and pertaining to the coding problem for us would be:

$$\mathrm{h(n)} = \frac{n}{2}$$

Results

Algorithm	**Test 1**		**Test 2**		**Test 3**	
	Solution length	*Total Expanded*	*Solution length*	*Total Expanded*	*Solution length*	*Total Expanded*
Breath First Search	9	22	33	120	377	2724
Depth First Search	15	15	61	114	401	174995
Iterative Deepening Depth First Search	9	175	33	6108	377	22595920
*A**	9	22	33	116	377	2720

Discussion

Before creating the algorithms to tests these searches I thought that the order of efficiency would have been A*, BFS, DFS, IDDFS. These were just guesses from the knowledge that I got in class. The reason I believe that A* would be the most efficient was because with a strong heuristic it would have existing knowledge of non-goal states.

First with A*, it had all the advantages of being informed and incorporating knowledge of the non-goal states. Additionally given the appropriate heuristic the efficiency become optimal and complete.

Moving on to BFS it can be seen from the table above that BFS performed at a satisfactory rate. Also it was definitely not the worst performing algorithm. However, it does have exponential time and space complexity.

For DFS, it did not perform as well as I thought it would have. From the results above we can see that it is incomplete and lacked optimality. I tested this a few times but as far as I can tell the reason for this outcome is due to the lack of optimality and completeness of DFS.

Finally IDDFS, the solution for this algorithm were in line with the other searches except for DFS. This performed better than I initially thought but not by much. It had exponential time complexity and was out shined by BFS in terms of runtime completion.

Conclusion

At the end of all of this it was evident that A* did the best overall and most efficient out of all of the search algorithms here. This was due to it being given an admissible and consistent heuristic. From the uninformed searches, which performed below A*, both BFS and IDDFS perform satisfactorily in terms of finding the same optimal path. Although both were slower than A* but DFS was overall the worse with its incompleteness and optimality. Overall the results obtained in this exercise was about what I expected except for DFS which I thought would perform better.