Mark Bereza
April 4, 2018

CS331 Programming Assignment #1 Report

I. Methodology

The purpose of this assignment was to implement and test various search algorithms learned in class. Three test cases for the wolves and chickens puzzle were provided with the first having only three of each, the second having nine chickens and eight wolves and the third having 100 chickens and 95 wolves. The experiments performed on these test cases were three forms of uninformed search, namely BFS (Breadth-First Search), DFS (Depth-First Search), and IDDFS (Iterative-Deepening Depth-First Search), and one type of informed search: A*. To make comparisons between the results for each of these searches more meaningful and to reduce computation time, each was implemented with a Graph Search algorithm. A general Graph Search function was created that changed its behavior based on which of the four search modes it was set to, like restarting the search once the frontier was empty with an increased depth for the IDDFS, utilizing a priority queue for the frontier instead of a simple list for the A* case, etc.

The parameters of interest were the maximum depth for IDDFS and the heuristic function used for A*. The maximum depth was initially set to 50,000 just to be safe, but after running it several times and seeing that the algorithm found the solution at a depth of 388, the upper bound was lowered to a more appropriate 500. As for the heuristic function, it assigns an expected cost of 2 per animal on the right bank because in almost every case it takes a minimum of two moves to get one animal off the right bank (two animals to the left bank at most, one animal must come back). The only time this isn't true is during the final trip in which up to two animals can be moved over in one turn. This exception is offset in two ways. First, in the case where the boat is at the right bank, a cost reduction of 3 is applied, which outweighs the slight deviation from the average animals per turn for the last turn. In the case where the boat is at the left bank, the cost reduction need not be applied because that is when the expected cost is already overly optimistic. This is because neither the animal that still must be sent back (+2 cost) nor the turn it takes to go back to the right bank (+1 cost) is factored into the heuristic cost in the left bank case. Thus, the function is an optimistic approximation for all states and is therefore admissible.

II. Results

TEST #1	BFS	DFS	IDDFS	A*
Nodes Expanded	13	11	84	13
Nodes in Solution	12	12	12	12

TEST #2	BFS	DFS	IDDFS	A*
Nodes Expanded	52	42	939	48
Nodes in Solution	32	32	32	32

TEST #3	BFS	DFS	IDDFS	A*
Nodes Expanded	1341	900	3191844	1337
Nodes in Solution	388	486	388	388

Figure 1: results of performing BFS, DFS, IDDFS, and A searches on the test cases.*

III. Discussion

The results of the aforementioned experiments were judged using two metrics: the total number of nodes expanded and the total number of nodes in the solution for each search algorithm. The first is a rough measure of the time complexity of each algorithm and the second is a rough measure of the optimality of each since optimal means fewest steps for this puzzle. With that in mind, the biggest surprise here is how much longer IDDFS took than the other three searches, seemingly growing exponential faster in nodes expanded compared to the other three. This was shocking because the worst-case time complexity is supposed to be exponential with respect to the depth limit, making it comparable to the exponential time complexities of DFS and BFS. This was such an issue that the initial implementation which used Tree Search had to be scrapped since it simply would not return in any reasonable amount of time for the third test case. Another surprise was that DFS had fewer nodes expanded than A* despite not being informed by a heuristic function. Besides that, the algorithms behaved as expected, with all but DFS appearing to be “optimal” and A* beating our BFS in nodes expanded, albeit marginally.

IV. Conclusion

What we can conclude from these results is that IDDFS is completely unfeasible for this puzzle, with relatively small ($n = 100$) input sizes taking minutes to compute, which was surprising given how downplayed its repetition of states was in the textbook. Additionally, we can conclude that A* performed the best overall because despite taking second place in fewest nodes expanded, it continued to be (seemingly) optimal for larger input sizes, whereas DFS began to show a significant decrease in optimality relative to the other three algorithms at $n = 100$.