CS331 Programming Assignment 1 Report

Jaelyn Litzinger and Lindsey Kvarfordt
4/19/20

Methodology

In order to implement the three uninformed search algorithms (breadth first search, depth first search, and iterative deepening depth first search), we utilized python 3's queue class to standardize the method names for adding and retrieving elements from the data structures used as the frontier in each respective algorithm. BFS uses queue.Queue, DFS/IDDFS uses queue.LifoQueue, and ASTAR uses queue.PriorityQueue. For IDDFS, the depth limit was set to 750 to make sure that the largest test case completed.

The heuristic that we used for A-star search was the number of animals left to move to the goal bank. This heuristic will always underestimate the number of steps left because the original problem has stricter rules like only moving two animals at a time and watching the comparative number of wolves vs chickens. This heuristic also works because it is largest at the initial state and 0 at the goal state.

Some trial and error had to be done while implementing A-star because of the backtrace function. The backtrace dictionary had to be altered to also store the heuristic value so that duplicate states with different costs had unique back pointers.

Results

Algorithm	# Nodes Expanded			#Nodes on Solution Path		
	Test 1	**Test 2**	**Test 3**	**Test 1**	**Test 2**	**Test 3**
BFS	14	60	970	11	35	387
DFS	12	36	615	11	35	387
IDDFS	12	36	615	11	35	387
ASTAR	12	51	765	11	35	387

Discussion

The results were as expected. All algorithms were able to find the optimal solution. DFS and IDDFS did the best job with respect to space complexity, expanding minimal extra nodes. A-star did better than BFS but not as well as IDDFS/DFS. It makes sense that BFS expanded the most nodes because it has exponential complexity (time and space).

Conclusion

IDDFS/DFS did the best job of the algorithms when it came to space complexity, expanding the fewest nodes. This makes sense because IDDFS combines the benefits of DFS and BFS to increase the search's efficiency. DFS was particularly effective for this set of test cases because none of the search tree's paths had infinite depth; whereas BFS was less effective because there were at most five possible successor states for each state. A-star had a reasonably good heuristic which helped minimize the complexity. However, the heuristic was not consistent, so A-star search was not optimally efficient (ie could not be guaranteed to expand the fewest nodes of all the algorithms)