Nicholas Skinner
Samuel Bonner

Methodology:

Compile:
Javac ThreeChickensSolver.java

Usage:
Java ThreeChickensSolver testStart1.txt testGoal1.txt <Search Method> solved.txt

Breadth-First:
Java ThreeChickensSolver testStart1.txt testGoal1.txt **bfs** solved.txt

The Breadth-First search function will explore every node in the tree level by level. This algorithm will attempt all of the neighbor nodes to the end of the data structure before continuing to the next level of the tree.

Depth-First:
Java ThreeChickensSolver testStart1.txt testGoal1.txt **dfs** solved.txt

The Depth-First search function is implemented similarly to Breadth-First, but instead of adding neighbor nodes to the overall data structure, neighbors are added to the front. With each node's neighbors added to the to the front, the algorithm will continue to explore a path until it reaches a terminal.

Iterative Deepening Depth First Search:
Java ThreeChickensSolver testStart1.txt testGoal1.txt **iddfs** solved.txt

The Iterative Deepening Depth First search will begin at the root node, and then based on the depth limit, will perform a Depth-First search. If no solution is found within the limit, we will increase it, and the process will start over.

A-Star:
Java ThreeChickensSolver testStart1.txt testGoal1.txt **astar** solved.txt

The A* search function will expand nodes based on a cost so far (g) and a calculated heuristic value (h). The node that has the lowest h + g value will be explored and expanded until a solution state is found. The chosen heuristic function was the sum of all animals on the starting (right) side of the river, minus one. This heuristic function is an estimate of the total moves that may be necessary to move all animals to the other side, and it is based on a relaxation (no accounting for wolves outnumbering chickens) of the original problem.

Nicholas Skinner
Samuel Bonner

Results:

	BFS	DFS	IDDFS	ASTAR
3x3	Total: 14 Solution: 11	Total:14 Solution:11	Total:123 Solution:11	Total: 13 Solution: 11
9x8	Total: 64 Solution: 31	Total:54 Solution:31	Total:1575 Solution:31	Total: 51 Solution: 31
100x95	Total: 1344 Solution: 387	Total: 1342 Solution: 387	Total:2471073 Solution:387	Total: 1339 Solution: 387

Discussion:

Our results were generally as we expected. BFS and DFS performed very similarly in terms of states expanded, with DFS proving to expand slightly fewer than BFS. A* had the lowest total expansions for all test cases, due to its use of a heuristic function and priority queue.

Something that was not expected was that DFS managed to find the optimal solution for every test state. Although DFS can find the optimal solution, it is not guaranteed that it will expand the optimal solution first.

Conclusion:

This was an interesting assignment to assess the performance of these searching algorithms. What we can conclude from the results of our tests is that one search algorithm isn't able to do it all. For memory conservation, the Depth First algorithm should be used, for quickness we use Breadth First, If you want to optimize the path you are using, you will want to use either A Star or IDDFS. Determining what one of the algorithms is best is difficult due to the specific conditions of each that was listed above, but if we're to judge based on the minimized solution, then IDDFS and BFS should on average be equal to, if not better than the others.