Programming Assignment #1
Kwanghyuk Kim & Hojun Shin
CS 331
April 19, 2021

1. **Methodology (5 points)**

We analyzed the wolves and chickens puzzle solution by using three test files and the modes of BFS, DFS, IDDFS, A* as the uninformed and informed search algorithm. We first created a node class that takes a node, a method, and a state as parameters. Within the class, each of the five actions is executed depending on whether the boat has moved left or right.
For the implementation of both BFS and DFS, we used the *deepcopy()* function in the copy package imported to apply the state of the current node as a child node that is appended to a que variable. During iteration, *Queue()* of the Queue package imported is used to remove the first stored frontier for BFS as a FIFO method, and *LifoQueue()* of the Queue package imported is used to remove the last appended frontier for DFS as a LIFO method.
For the implementation of ID-DFS, we set the depth limit to 10000000 with the use of the depth limit search function. By having the recursive function run inside the depth limit search function, we set the search to stop and exit the function when a solution is found.
For the implementation of A*, we used a heuristic search function, which keeps subtracting the current number to the goal number by comparing how many chickens and wolves in the current state remain to the left side until reaching the goal. Then, we used a priority queue to queue the successors with order.

2. **Results (10 points)**

	Test Case 1		Test Case 2		Test Case 3	
	Solution path	Nodes Expanded	Solution path	Nodes Expanded	Solution path	Nodes Expanded
BFS	12	46	∞	∞	∞	∞
DFS	12	25	80	305	564	2071
ID-DFS	12	18	∞	∞	∞	∞
A*	12	12	44	44	378	378

3. Discussion (10 points)

For the breadth-first search (BFS), it was only the first test of small numbers that could expect to show the result based on the time and space complexity of $O(b^d)$, and it was not even the optimal one. From the second test, the results were not shown, which may be due to the size of nodes expanded.
For the depth-first search (DFS), it showed results in a relatively short period of time for all three tests compared to the BFS. However, due to the characteristic of DFS, this is not a complete and optimal solution for a problem, as the search ends when the solution is reached.
For the iterative deepening depth-first search (ID-DFS), we expected this to be complete and optimal with the time complexity of $O(b^d)$ and space complexity of $O(bd)$, but it would be meaningless and slower than BFS because the graph search-based approach requires a lot of memory. However, the first test showed fewer expanded nodes than BFS, and the second and third took forever to run like BFS.
A* with a heuristic search function was the most efficient of the four algorithms and provided the optimal solution as we expected. One thing that was interesting to us was that the solution path and the number of expanded nodes were the same while running three test cases.

4. Conclusion (5 points)

We can conclude that the A* search algorithm using heuristic functions is the most optimal of the four algorithms. BFS is easy to implement, but inefficient in terms of time and space complexity along with IDDFS, and IDDFS was even difficult to implement. DFS is easy to implement like BFS and faster than BFS, but it did not provide an optimal solution. As a result, we have chosen the most efficient A* search algorithm as the wolves and chickens puzzle solution.