Methodology:

Overall experiment: Tested with 3 of the test cases provided in the assignment. Additional test cases are done by swapping the start and goal file, also some arbitrary test case where the number of chickens is larger or equal to the number of wolves.

BFS – We run the search problem on all 3 of the test cases. They all come out to reach the goal state with no problem. Other than the 3 test cases, we also tried to flip around the initial file and goal file of the first test case to see if the algorithm works the other way around. There is no particular parameter used in this search except passing in the initial state and check it with goal state.

DFS – This search algorithm is also run the same experiments as BFS. It is performed by an auxiliary death limited search(dls) function. Which we pass in an INT_MAX as the limit parameter for the dls function. That way, we mimic an infinite depth which happens while performing DFS.

IDDFS – As above, same experiment is done for this search algorithm. It is done by iterating through current limit and pass it to the same dls function used in DFS until current limit reach INT_MAX (again mimicking infinite depth).

A* – The same overall experiment was performed. The heuristic function we used in the A* algorithm is by calculating the least amount of move required to transfer all the animals from one side to the other ignoring the rules. By running test for test cases that contain only chickens, we found that there is a relationship between the minimum move and the total number of chickens. That is, if the number of chickens is smaller than 3, the shortest path to take all chickens to the other bank will be 1, otherwise, the shortest path will be 3 + (number of chickens – 3) * 2. Using this pattern, we then apply it to our heuristic, replacing the number of chickens with the number of wolves. Since the heuristic calculate the shortest path to move all animals from one bank to the other, it is always optimistic and never overestimate. Therefore, our heuristic used in this A* function is admissible.

Result:

# nodes on the solution path

	Test1	Test2	Test3
Bfs	11	35	387
Dfs	11	35	391
Iddfs	11	35	389
A*	11	35	387

nodes expanded

	Test1	Test2	Test3
Bfs	14	60	970
Dfs	11	48	856
Iddfs	97	1185	206630
A*	14	44	780

According to the results, we can see that the path lengths of the solutions of different algorithms are approximately the same. Strictly speaking, the results are exactly the same for the first two tests. Dfs and iddfs use more path to reach the goal. Bfs and A* solution are the same. For number of nodes expanded, Dfs has the least number of nodes expanded. For the second and third tests, A* has a lowest number and iddfs has the largest number. The number of nodes expanded of iddfs exceeds that of other algorithms a lot. Especially in the third test, when the number of other algorithms is close to one thousand, the number of iddfs reaches 20,000.

Discussion:

Overall, we think most of the figures are in line with our expectations. Iddfs results did not meet expectations. First of all. A* and bfs found the same optimal solution as an optimal search method. Iddfs did not find the same optimal solution as A * and bfs. Second, the number of nodes expanded by iddfs far exceeds that of other algorithms. This is inconsistent with our slides mentioned that when the cardinality is large enough, the number of nodes expanded by iddfs should be one power less than bfs. A* has the least number of nodes expanded except for the first test case which dfs found the solution on one branch because of luck. We can feel that teaching an algorithm to execute each step with human thinking can effectively improve the efficiency of the algorithm, filtering out most attempts with a low success rate. In the A * algorithm, we teach the algorithm to identify the step closest to the target as the next step, resulting in a significant reduction in the number of expanded nodes. This is the difference between uninformed search and informed search. Informed search combines known logic and decision rules to maximize traversal or looping. The most difficult part is to find the basis for judging the nodes that are close to the target which is h(n). In the path example of slides, the criterion for the optimal step is the direct distance from the target. We need equation calculation to find the standard to measure progress. An interesting thing that we found is that even though both A * and bfs yields the optimal solution, the path that they are using are the same. That probably means that the problem has several optimal paths, causing the algorithm to yield different optimal solution due to different search method.

Conclusion:

A* is the most efficient search method among these all. That being said, we will say informed search are better than uninformed search for this problem. We say that DFS is not an optimal search but found the very close result. The nodes expanded of iddfs is more than other search methods due to small search capacity. We think that it is because the test cases are still not large enough, so that the duplicate part of iddfs cannot be omitted. $O(b^{d+1})>>O(b^{d})$+duplicate is true when the b and d are both large enough. In our case, the b is 5 which is probably not large enough to cover duplicate parts.