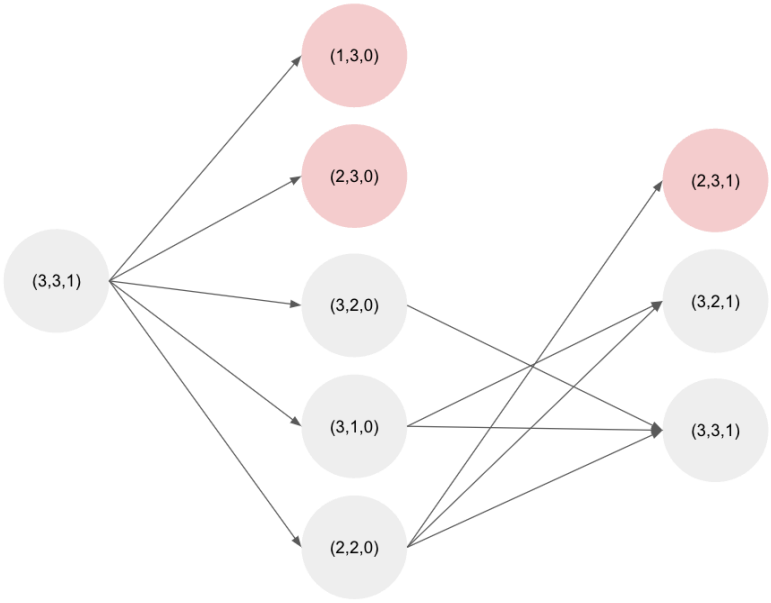
COSC276 Fall 2022, PA1, Gibran Erlangga

Problem Setup

The Chickens and Foxes problem is specifically represented in *FoxProblem.py* under the *FoxProblem* object. The state is represented by the (i, j, k) with i represents the total number of chickens on the left side, j represents the total number of foxes on the left side, and k represents the boat position (1 if the boat is on the left side, and 0 if the boat is on the right side). With this setup, we can deduce the number of chickens and foxes on the right side by subtracting the initial number of chickens and foxes in the current state. The Upper bound on the number of states without taking states legality into account are: 3 x 3 x 2 (chicken x fox x boat position) = 18 possible states.

some rules I applied to filter out the illegal actions are:
- checks whether the total of current state and action exceeds the total chickens or foxes available
- ensure that the number of chickens are equal or more than the number of foxes on both sides
- the boat maximum capacity is two animals and at least one animal should be there to move it to the other side
- The action for the boat movement direction is divided into two: right to left and vice versa. For right to left, I also checked whether the boat position is correct (if the direction is right to left then the boat should initially be on the right). Furthermore, if the direction is right to left, then any animals cannot move to the right.

Below is the sample of state graph along with the actions up until the first-reached states:



The red circles shown above are the invalid states because it was produced by illegal actions, or actions that do not pass the filter above. For instance, the top red circle (1,3,0) is classified as invalid because the number of chickens is less than the number of foxes. The second red circle (2,3,0) is classified as invalid for the same reason, and so does the last red circle (2,3,1).

Search algorithms

I implemented the Breadth-first search, Depth-first search with path-checking and iterative deepening search algorithms. Here are the results from the test cases provided:

Case 1: (3,3,1) - 3 chickens and 3 foxes

Algorithm	# nodes visited	Solution length
BFS	15	11
DFS	11	11
IDS	66	11

Case 2: (3,3,1) - 5 chickens and 5 foxes

Algorithm	# nodes visited	Solution length
BFS	13	N/A
DFS	13	N/A
IDS	1212	N/A

Case 3: (3,3,1) - 5 chickens and 4 foxes

Algorithm	# nodes visited	Solution length
BFS	30	15
DFS	25	19
IDS	244	19