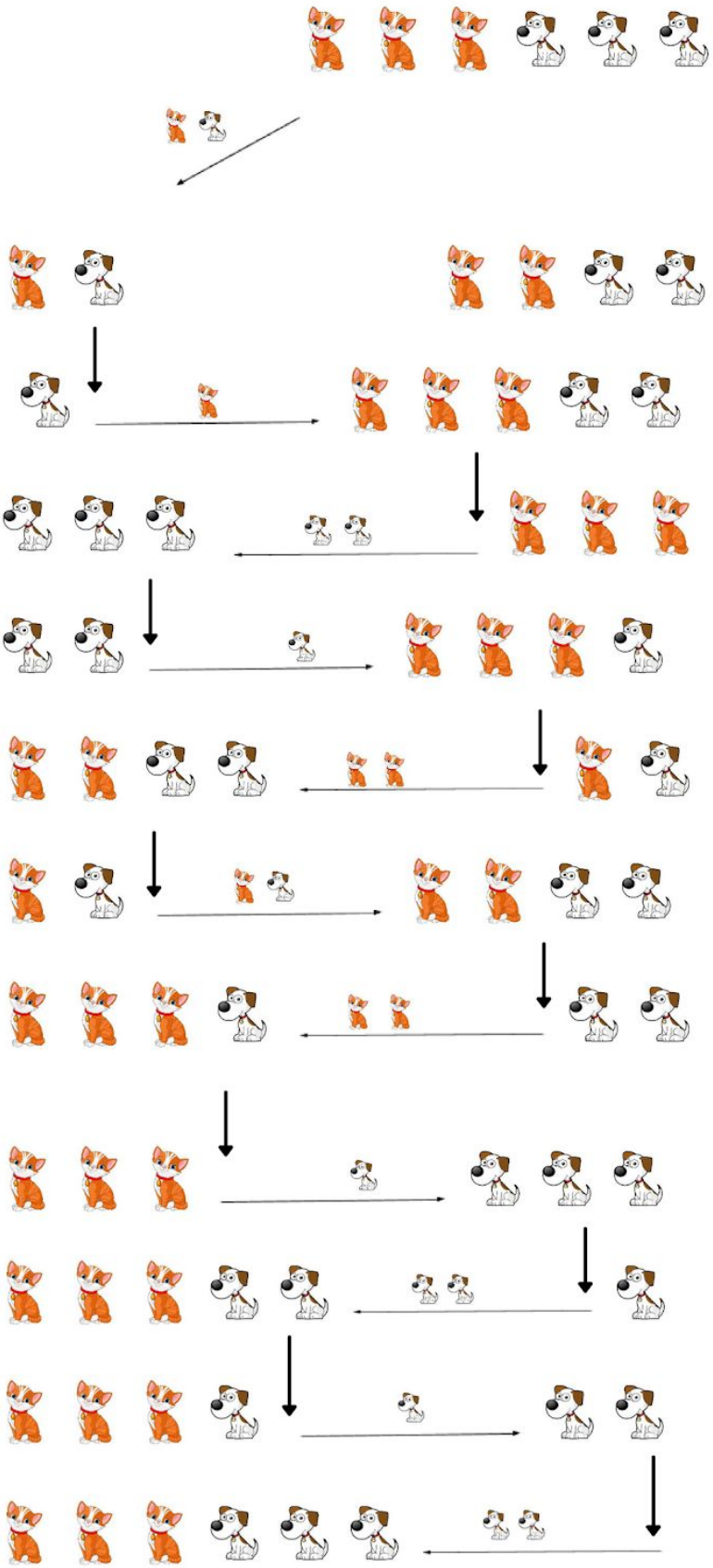


State 1: 3 dogs and 3 cats are on the right.
Action: 1 dog and 1 cat go to the left.
Transition: LEFT: 1 cat, 1 dog. RIGHT: 2 cats, 2 dogs

State 2:
Action: 1 cat from the left goes to the right
Transition: LEFT: 1 dog. RIGHT: 3 cats, 2 dogs

State 3:
Action: 2 dogs from the right go to the left
Transition: LEFT: 3 dogs. RIGHT: 3 cats

State 4:
Action: 1 dog from the left goes to the right.
Transition: LEFT: 2 dogs. RIGHT: 3 cats, 1 dog

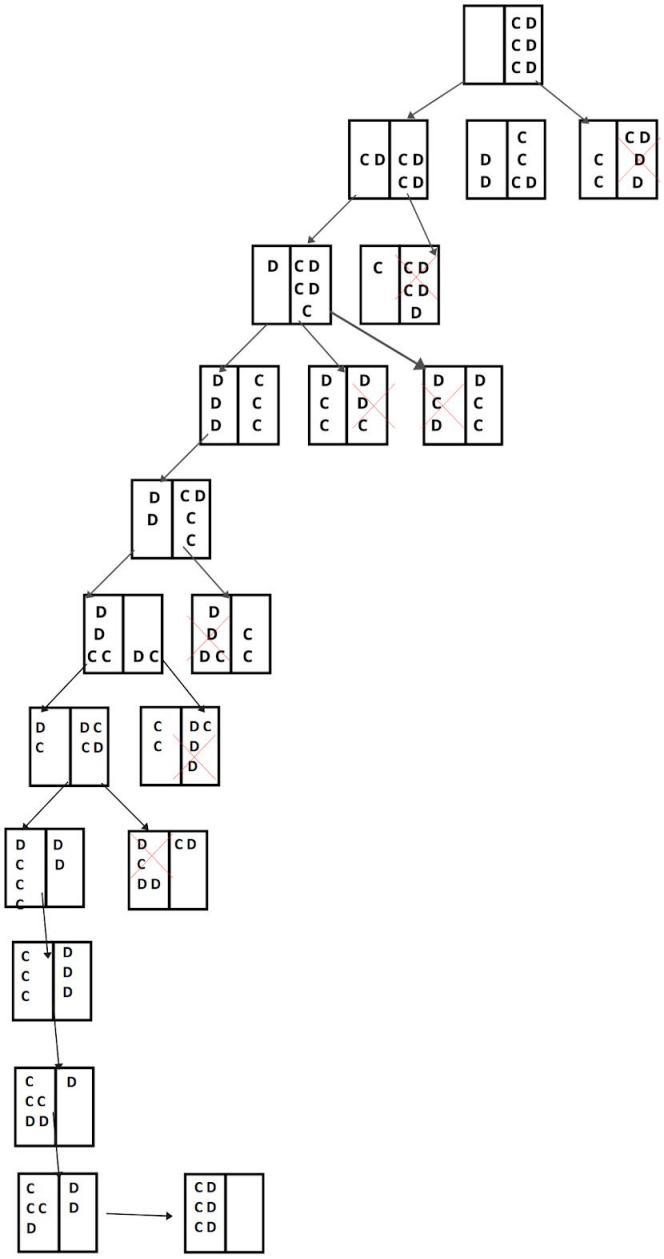
State 5:
Action: 2 cats from the right go the left
Transition: LEFT: 2 cats, 2 dogs. RIGHT: 1 cat, 1 dog

State 6:
Action: 1 cat, 1 dogs from the left go to the right
Transition: LEFT: 1 cat, 1 dog. RIGHT: 2 cats, 2 dogs

State 7:
Action: 2 cats from the right go the left
Transition: LEFT: 3 cats, 1 dog. RIGHT: 2 dogs

State 8:
Action: 1 dog from the left goes to the right
Transition: LEFT: 3 cats. RIGHT: 3 dogs
State 9:
Action: 2 dogs from the right go to the left
Transition: LEFT: 3 cats, 2 dogs. RIGHT: 1 dog
State 10:
Action: 1 dog from the left goes to the right
Transition: LEFT: 3 cats, 1 dog. RIGHT: 2 dogs
State 11:
Action: 2 dogs from the right go to the left
Transition: LEFT: 3 cats, 3 dogs. RIGHT: 0

```
function treeSearch(problem)
        initialize frontier with initial state of problem
        Loop
                if frontier is empty,
                        return failure
                choose a leaf node and remove it from frontier
                if the node contains a goal state,
                        Return solution
                expand the chosen node,
                add resulting nodes to frontier
```

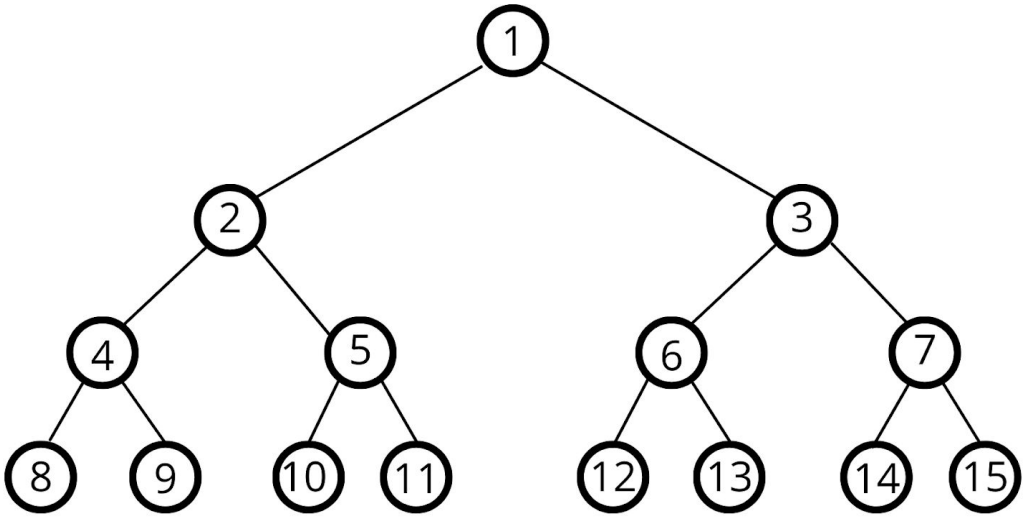


In my opinion, people have a hard time solving this puzzle because there are many conditions to follow such as there must be one or two animals in the boat to move so every time we bring one or two animals to the other side, we need to bring one back. That takes more states to finish the puzzle. Moreover, the hardest condition is we need to keep the number of cats always equal or less than the number of dogs in any place so we need to keep an eye on both sides of the river. We need to bring all the cats to the left first so the number of cats on the left is always larger or equal to the number of dogs then we bring the rest of the dogs later.

2.

- Draw the portion of the state space for states 1 to 15

Start state = 1 => return two states 2(1) = 2 and 2(1) +1 =3



- Suppose the goal state is 11. List the order in which nodes will be visited for breadth first and depth-limited search with limit 3
  - Breadth first search: 1 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 9 → 10 → 11
  - Depth-limited search with limit 3: 1 → 2 → 4 → 8 → 9 → 5 → 10 → 11

- Call the action going from k to 2k Left and the action going to 2k+1 Right. Can you find an algorithm that outputs the solution to this problem without any search at all?

  Pick a node and divide it by 2, then take the result and continue divide it by 2 until nothing is left.
  If the remainder is 1 -> Left
  If the remainder is 0 -> Right
  Then reverse the path.

  For example:
  13/2 = 6 remain 1 -> Left
  6/2 = 3 remain 0 -> Right
  3/2 = 1 remain 1 -> Left

  Therefore, the path from 0 to 13 is Left-Right-Left