Search Algorithm	Missionaries and Cannibals	Towers of Hanoi	Farmer Fox Chicken and Grain
Iterative Depth First Search	Solution Path Length = 9 States Expanded = 10 Max Open Length = 2	Solution Path Length = 40 States Expanded = 40 Max Open Length = 7	Solution Path Length = 7 States Expanded = 7 Max Open Length = 3
Breadth First Search	Solution Path Length = 7 States Expanded = 10 Max Open Length = 2	Solution Path Length = 15 States Expanded = 70 Max Open Length = 16	Solution Path Length = 7 States Expanded = 9 Max Open Length = 2
Iterative Deepening DFS	Solution Path Length = 7 States Expanded = 19 Max Open Length = 2	Solution Path Length = 15 States Expanded = 129 Max Open Length = 9	Solution Path Length = 7 States Expanded = 17 Max Open Length = 2

Solution Paths

1. **Missionaries : [x, y] → x missionaries and y cannibals on the left bank (and the rest on the right bank)**
   a. IterDFS: [3,3] →[2,2] → [3,2] →[0,2] → [2,2] → [1,1] → [3,1] → [0,1] → [1,1] → [0,0] done!
   b. BFS: [3,3] → [2,2] → [3,2] → [0,2] → [2,2] → [0,1] → [1,1] → [0,0] done!
   c. IDDFS: [3,3] → [2,2] → [3,2] → [0,2] → [2,2] → [0,1] → [1,1] → [0,0] done!
2. **Farmer Fox: Fgcf → F: Farmer, g: Grain, c: Chicken, f: Fox. Given string lists which objects are on LEFT side of river**
   a. IterDFS: Fcfg → fg → Ffg → f → Fcf → c → Fc → done!
   b. BFS: Fcfg → fg → Ffg → f → Fcf → c → Fc → done!
   c. IDDFS: Fcfg → fg → Ffg → f → Fcf → c → Fc → done!
3. **Towers Of Hanoi (4) : [[list1][list2][list3]] → list i lists the plates on pole i ; 4 = largest plate, 1 = smallest plate**
   a. IterDFS

```
[[4, 3, 2, 1] ,[] ,[]]
[[4, 3, 2] ,[1] ,[]]
[[4, 3] ,[1] ,[2]]
[[4, 3, 1] ,[] ,[2]]
[[4, 3] ,[] ,[2, 1]]
[[4] ,[3] ,[2, 1]]
[[4, 1] ,[3] ,[2]]
[[4] ,[3, 1] ,[2]]
[[4, 2] ,[3, 1] ,[]]
[[4, 2, 1] ,[3] ,[]]
```

```
[[4, 2] ,[3] ,[1]]
[[4] ,[3, 2] ,[1]]
[[4, 1] ,[3, 2] ,[]]
[[4] ,[3, 2, 1] ,[]]
[[] ,[3, 2, 1] ,[4]]
[[1] ,[3, 2] ,[4]]
[[] ,[3, 2] ,[4, 1]]
[[2] ,[3] ,[4, 1]]
[[2, 1] ,[3] ,[4]]
[[2] ,[3, 1] ,[4]]
[[] ,[3, 1] ,[4, 2]]
[[1] ,[3] ,[4, 2]]
[[] ,[3] ,[4, 2, 1]]
[[3] ,[] ,[4, 2, 1]]
[[3, 1] ,[] ,[4, 2]]
[[3] ,[1] ,[4, 2]]
[[3, 2] ,[1] ,[4]]
[[3, 2, 1] ,[] ,[4]]
[[3, 2] ,[] ,[4, 1]]
[[3] ,[2] ,[4, 1]]
[[3, 1] ,[2] ,[4]]
[[3] ,[2, 1] ,[4]]
[[] ,[2, 1] ,[4, 3]]
[[1] ,[2] ,[4, 3]]
[[] ,[2] ,[4, 3, 1]]
[[2] ,[] ,[4, 3, 1]]
[[2, 1] ,[] ,[4, 3]]
[[2] ,[1] ,[4, 3]]
[[] ,[1] ,[4, 3, 2]]
[[1] ,[] ,[4, 3, 2]]
[[] ,[] ,[4, 3, 2, 1]]
```

b. BFS

```
[[4, 3, 2, 1] ,[] ,[]]
[[4, 3, 2] ,[1] ,[]]
[[4, 3] ,[1] ,[2]]
[[4, 3] ,[] ,[2, 1]]
[[4] ,[3] ,[2, 1]]
[[4, 1] ,[3] ,[2]]
```

```
[[4, 1] ,[3, 2] ,[]]
[[4] ,[3, 2, 1] ,[]]
[[] ,[3, 2, 1] ,[4]]
[[] ,[3, 2] ,[4, 1]]
[[2] ,[3] ,[4, 1]]
[[2, 1] ,[3] ,[4]]
[[2, 1] ,[] ,[4, 3]]
[[2] ,[1] ,[4, 3]]
[[] ,[1] ,[4, 3, 2]]
[[] ,[] ,[4, 3, 2, 1]]
```

c. IDDFS:

```
[[4, 3, 2, 1] ,[] ,[]]
[[4, 3, 2] ,[1] ,[]]
[[4, 3] ,[1] ,[2]]
[[4, 3] ,[] ,[2, 1]]
[[4] ,[3] ,[2, 1]]
[[4, 1] ,[3] ,[2]]
[[4, 1] ,[3, 2] ,[]]
[[4] ,[3, 2, 1] ,[]]
[[] ,[3, 2, 1] ,[4]]
[[] ,[3, 2] ,[4, 1]]
[[2] ,[3] ,[4, 1]]
[[2, 1] ,[3] ,[4]]
[[2, 1] ,[] ,[4, 3]]
[[2] ,[1] ,[4, 3]]
[[] ,[1] ,[4, 3, 2]]
[[] ,[] ,[4, 3, 2, 1]]
```