Missionary And Cannibal

Assignment No. 1

Submitted By:
Ashutosh Chapagain (08)
Krishu Kr. Thapa (58)

Problem Statement

Three missionaries and three cannibals come to the bank of a river they wish to cross. There is a boat that will hold only two and any of the group is able to row. If there are ever more missionaries than cannibals on any side of the river the cannibals will get converted.

Strategies Used To Accomplish the Goal

- Depth-First Search
- Breadth-First Search
- Best-First Search

Assumption:

All 3 Missionaries and Cannibals are on the left side of the river which indicates boat position 1. Other side represents boat position 0.

Initial State:

3,3,1

Final State:

0,0,0

Constraints:

The no. of cannibals should never exceed the no. of missionaries at any side.

Condition To Satisfy The Constraint:

```
if node.missionary >= 0 and (3-node.missionary) >= 0 \
            and node.cannibal >= 0 and (3-node.cannibal) >= 0 \
            and (node.missionary == 0 or node.missionary >= node.cannibal) \
            and ((3-node.missionary) == 0 or (3-node.missionary) >= (3-node.cannibal)):
                return True
        else:
                return False
```

Depth-First Tree:

Algorithm:
1) Place the node at top of the stack.
2) Pop out the element and find its descendants.
3) Check if the descendant are already explored.
4) Add it to the active list as a stack.
5) If empty, the conclusion is we did not reach goal state.
6) Otherwise goto 2 unless goal state is reached.

Output
In order of
MissionaryLeft,CannibalLeft,BoatPos

3,3,1
3,1,0
3,2,1
3,0,0
3,1,1
1,1,0
2,2,1
0,2,0
0,3,1
0,1,0
1,1,1
0,0,0

Breadth-First Tree:

Similar to DFS, but here we maintain a queue.

Algorithm:
1) Place the node at top of the queue.
2) Pop out the element and find its descendants.
3) Check if the descendant are already explored.
4) Add it to the active list as a queue.
5) If empty, the conclusion is we did not reach goal state.
6) Otherwise goto 2 unless goal state is reached.

Output
In order of
MissionaryLeft,CannibalLeft,BoatPos

3,3,1
2,2,0
3,2,0
3,1,0
3,2,1
3,0,0
3,1,1
1,1,0

2,2,1
0,2,0
0,3,1
0,1,0
1,1,1
0,0,0

Best-First Search

A greedy approach is taken to find the most promising descendant of a given node. It can also be said as a greedy approach.
Heuristics Needed : **h(x)= (MissionaryLeft+CannibalLeft)/2**

1) Place the node at top of the list.
2) Pop out the element and find its descendants.
3) Check if the descendant are already explored.
4) Among the unexplored descendants find the most promising one using the heuristic function.
4) Add it to the active list as a list.
5) If empty, the conclusion is we did not reach goal state.
6) Otherwise goto 2 unless goal state is reached.

MissionaryLeft,CannibalLeft,BoatPos
3,3,1
3,1,0
3,2,1
3,0,0
3,2,1
3,1,1
1,1,0
2,2,1
0,2,0
0,3,1
0,1,0
1,1,1
0,0,0

Analysis

Strategies Used	No. Of Steps Required
Depth-First Search	11
Breadth-First Search	14
Best- First Search	11

Note: Best-First Tree is still better than Depth-First tree because the steps in DFS depends on the node which appears first. In Best-First Search, it does not depend upon the arrangement of nodes.