**Issue:** Cannibals and Missionaries

**Description:** There are 3 cannibals and 3 missionaries on a river bank. They need to cross the river to reach the other bank but they have only one boat to use. The boat can carry 2 persons at one time. They have to cross from current bank to the other with that one boat, considering the fact that the number of cannibals on either bank must not exceed the number of missionaries on the same bank. Because cannibals will eat the missionaries if they can outnumber the missionaries. The problem needs a solution by planning a sequence of crossings (combinations of cannibals and missionaries on the boat) that will take everyone on the other bank of river by maintaining the condition.

**Subject theory to solve the problem:** This kind of problem can be solved by implementing graph algorithm where each node can define the state of initial bank, with such information as the number of cannibals and the number of missionaries on the bank and the side the boat on. The edges can define the combination of missionaries and cannibals on the boat in each trip. And we can declare a starting node/state where all the cannibals and missionaries are at one side and a finish node/state where all the cannibals and missionaries are on the other side. Then we can traverse the graph nodes (bank state). Starting from the initial node, if we can reach our defined finish node by maintaining the condition, we will get a sequence of combinations from the path (connected by edges) for crossing the river.

**Task Analysis:** If we represent the problem on a graph, the nodes will represent the states of the bank after every move. The edges can define the combination of passengers on the boat (no. of missionaries and cannibals).

We can represent the states as,

* State (No. of missionaries on the starting bank, No. of Cannibals on the starting bank, The side the boat on)

And edges as,

* Edge (No. of missionaries on the starting boat, No. of Cannibals on the starting boat)

Since, the boat can only carry 2 persons at a time, there are limited number of combinations to carry missionaries and cannibals on the boat at the same time. The combinations can be as follows,

* Boat (2, 0)
* Boat (1, 0)
* Boat (1, 1)
* Boat (0, 2)
* Boat (0, 1)

Where Boat (m, c) means the combination, m is the number of missionaries and c is the number of cannibals.

So, from each states, the boat can only carry these combinations of passengers to other states. Thus, each node in the graph will have these five edges.

Boat (2,0)

Boat (0, 1)

Boat (1,1)

Boat (0, 2)

Boat (1,0)

At each state, we will look for possible states according to the possible boat passenger combinations (edges). Then we will check of the possible state if it maintains our conditions. If so, we will traverse to the next state until we find our goal state.

First, we need to define our initial state and goal state. According to the problem statement, the main motive was to cross the river. So, if we suppose, start from the right bank of the river, our initial state will be as follows,

* Initial State (3, 3, Right)

Where, Initial State (m, c, b) represents the first node of the graph, m is the number of missionaries, c is the number of cannibals on the initial bank and b is the side the boat currently on.

Our goal is to reach a node where initial side of the river has no missionaries and cannibals, so the other side has all of them. Our goal state will be,

* Goal State (0, 0, Left)

To reach the goal state from the initial state, we have to traverse through the possible edges, check the conditions and avoid the states which are already visited once. We can use various kind of graph searching algorithms such as Breadth First Search (BFS, traverse all possible immediate states connected by edges first), Depth First Search (DFS, traverse through only first possible state) for traversing.

* Conditions –
  + It is not feasible to move more missionaries or more cannibals than are present on one bank.
    - When the state is State (M1, C1, left) and we try Boat (M, C) then
      * M <= M1 and C <= C1 should be true.
    - When the state is State(M1, C1, right) and we try Boat (M, C) then
      * M + M1 <= 3 and C + C1 <= 3 should be true.
  + The number of cannibals on either bank must never exceed the number of missionaries on the same bank

**Code:**

% defining initial state

mandc(state(0, 0, right), \_, []).

% traversing through the graph

mandc(CurrentState, Visited, [Move | RestOfMoves]) :-

newstate(CurrentState, NextState),

not(member(NextState, Visited)),

make\_move(CurrentState, NextState, Move),

mandc(NextState, [NextState | Visited], RestOfMoves).

% make\_move(State1, State2, Move) builds the move(-,-,-) that gets you

% from State1 to State2.

make\_move(state(M1, C1, left), state(M2, C2, right), move(M, C, right)) :-

M is M1 - M2,

C is C1 - C2.

make\_move(state(M1, C1, right), state(M2, C2, left), move(M, C, left)) :-

M is M2 - M1,

C is C2 - C1.

% boat(X,Y) is true if the boat can carry X missionaries and Y cannibals.

boat (2, 0).

boat (1, 0).

boat (1, 1).

boat (0, 1).

boat (0, 2).

% legal(X, Y) is true if it is safe for the missionaries to have X

% missionaries and Y cannibals together on a river bank (left or right).

legal(X, X).

legal(3, X).

legal(0, X).

% newstate(State1, State2) is true if it is possible to get from State1

% to State2.

newstate(state(M1, C1, left), state(M2, C2, right)) :-

carry(M, C),

M <= M1,

C <= C1,

M2 is M1 - M,

C2 is C1 - C,

legal(M2, C2).

newstate(state(M1, C1, right), state(M2, C2, left)) :-

carry(M, C),

M2 is M1 + M,

C2 is C1 + C,

M2 <= 3,

C2 <= 3,

legal(M2, C2).

**Program Description:** mandc procedure is doing the traversing from a current state, which is initially set for the first node and dynamically set for the following states. It has three arguments, current state, a list of already visited states and a list of moves already made. It calls a procedure named newstate which essentially checks the conditions of the possible new state. If it violates any condition of the solution, program searches for another new state. If it is a legal state, then it checks if it is already visited or not. If the new state is not visited yet, it calls make\_move procedure to save the move into the path. After that it calls itself again but this time the new state will become the current state and the path is updated. That’s how the program runs recursively until it gets to the goal state.

**Expected Behavior:** Final results should reflect as follows,

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sequence of Traversed States | Sequence of moves (number of missionaries and cannibals per move) | Right bank status (number of missionaries and cannibals per move) | Left bank status (number of missionaries and cannibals per move) | The side the boat on |
| State (3, 3, R) | - | (3, 3) | (0, 0) | Right |
| State (3, 1, L) | (0, 2) | (3, 1) | (0, 2) | Left |
| State (3, 2, R) | (0, 1) | (3, 2) | (0, 1) | Right |
| State (3, 0, L) | (0, 2) | (3, 0) | (0, 3) | Left |
| State (3, 1, R) | (0, 1) | (3, 1) | (0, 2) | Right |
| State (1, 1, L) | (2, 0) | (1, 1) | (2, 2) | Left |
| State (1, 1, R) | (1, 1) | (2, 2) | (1, 1) | Right |
| State (0, 0, L) | (1, 1) | (0, 0) | (3, 3) | Left |