Practical 02

Aim: Write a program to Implement BFS for Missionaries and Cannibal Problem.

Theory:

Missionaries and Cannibal Problem:-

On one bank of a river are three missionaries and three cannibals. There is one boat available that can hold up to two people and that they would like to use to cross the river. If the cannibals ever outnumber the missionaries on either of the river's banks, the missionaries will get eaten. How can the boat be used to safely carry all the missionaries and cannibals across the river?

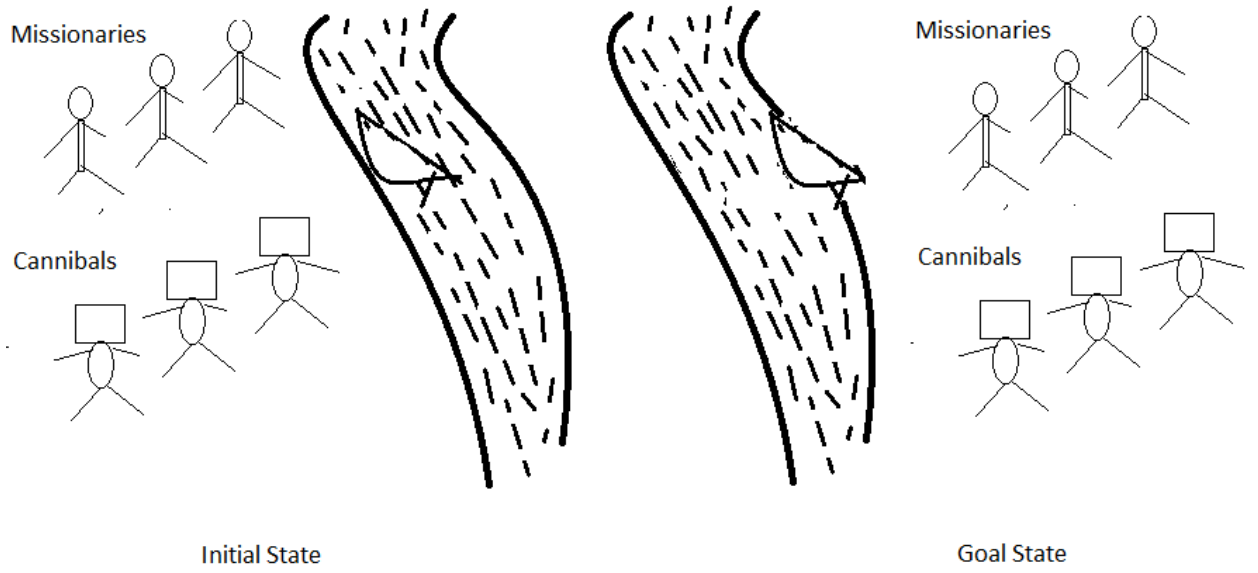


Fig.1: Missionaries and Cannibals: Initial and Goal State

Production Rules for Missionaries and Cannibals problem

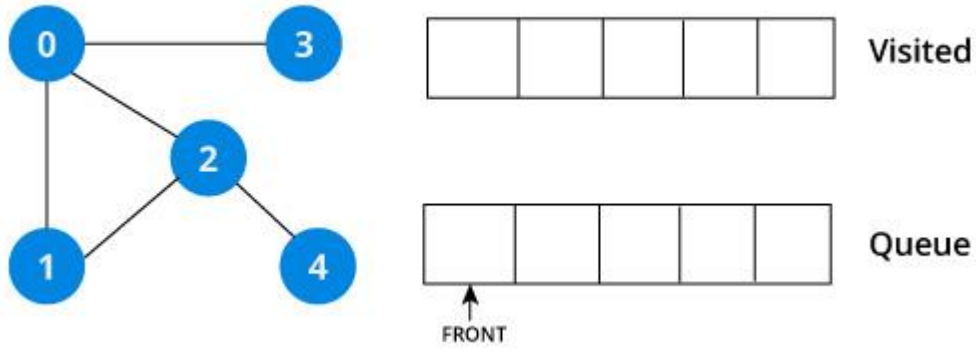
Rule No.	Rules and Action
1	(x,y): Two missionaries can go only when x-2 >= y or x-2 = 0 in one bank and x +2 >= y in the other bank
2	(x,y): Two cannibals can cross the river only when y-2 <= x or x = 0 in one bank and y +2 <= x or x=0 in the other bank.
3	(x,y): one missionary and one cannibals can go in the boat only when x-1 >= y-1 or x = 0 in one bank and x +1 >= y+1 or x=0 in the other bank.
4	(x,y): one missionary can cross the river only when x-1 >= y or x = 0 in one bank and x +1 >= y in the other bank.
5	(x,y): one cannibal can cross the river only when y-1 < x or x = 0 in one bank and y +1 <= x or y = 0 in the other bank.

Breath First Search

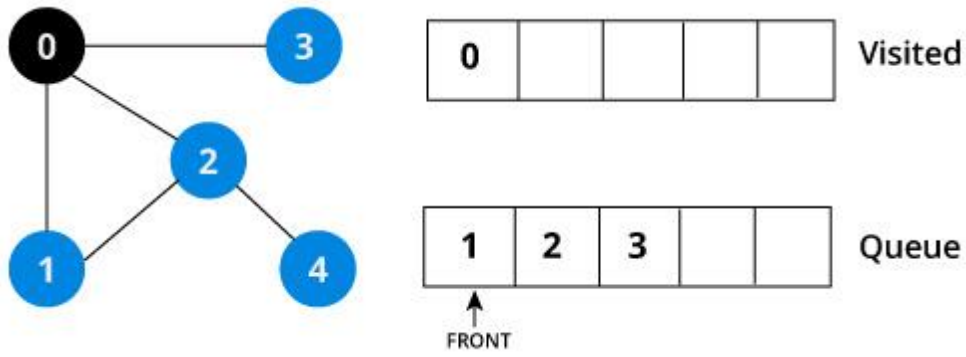
Algorithm

1. Create a variable called NODE-LIST and set it to the initial state.
2. Loop until the goal state is found or NODE-LIST is empty.
    a. Remove the first element, say E, from the NODE-LIST. If NODE-LIST was empty then quit.
    b. For each way that each rule can match the state described in E do:
        i) Apply the rule to generate a new state.
        ii) If the new state is the goal state, quit and return this state.
        iii) Otherwise add this state to the end of NODE-LIST

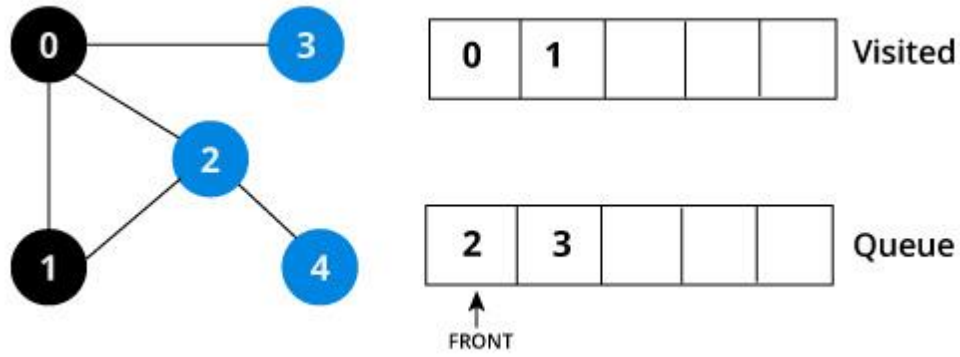
Let's see how the Breadth First Search algorithm works with an example. We use an undirected graph with 5 vertices. Here Visited is Closelist and Queue is Openlist.



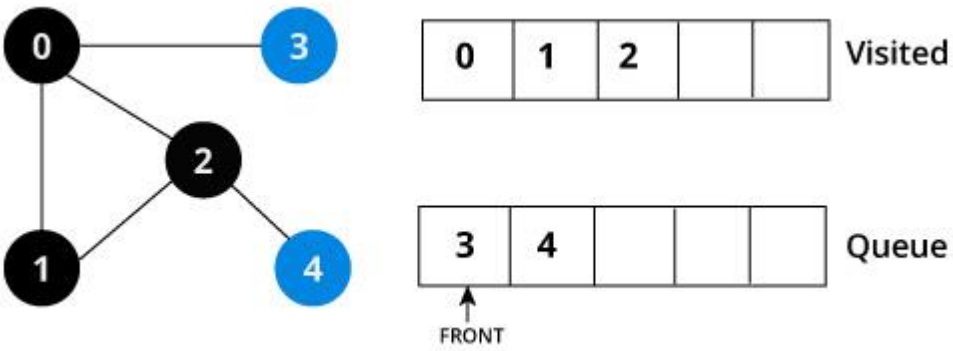
We start from vertex 0, the BFS algorithm starts by putting it in the Visited list and putting all its adjacent vertices in the stack.

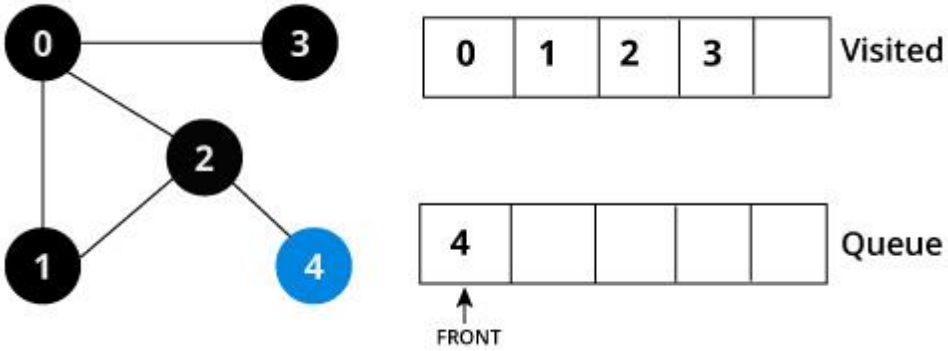


Next, we visit the element at the front of queue i.e. 1 and go to its adjacent nodes. Since 0 has already been visited, we visit 2 instead.

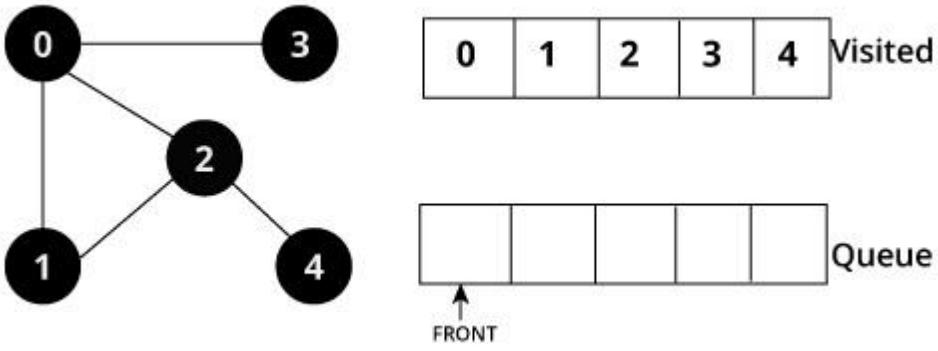


Vertex 2 has an unvisited adjacent vertex in 4, so we add that to the back of the queue and visit 3, which is at the front of the queue.





Only 4 remains in the queue since the only adjacent node of 3 i.e. 0 is already visited. We visit it.



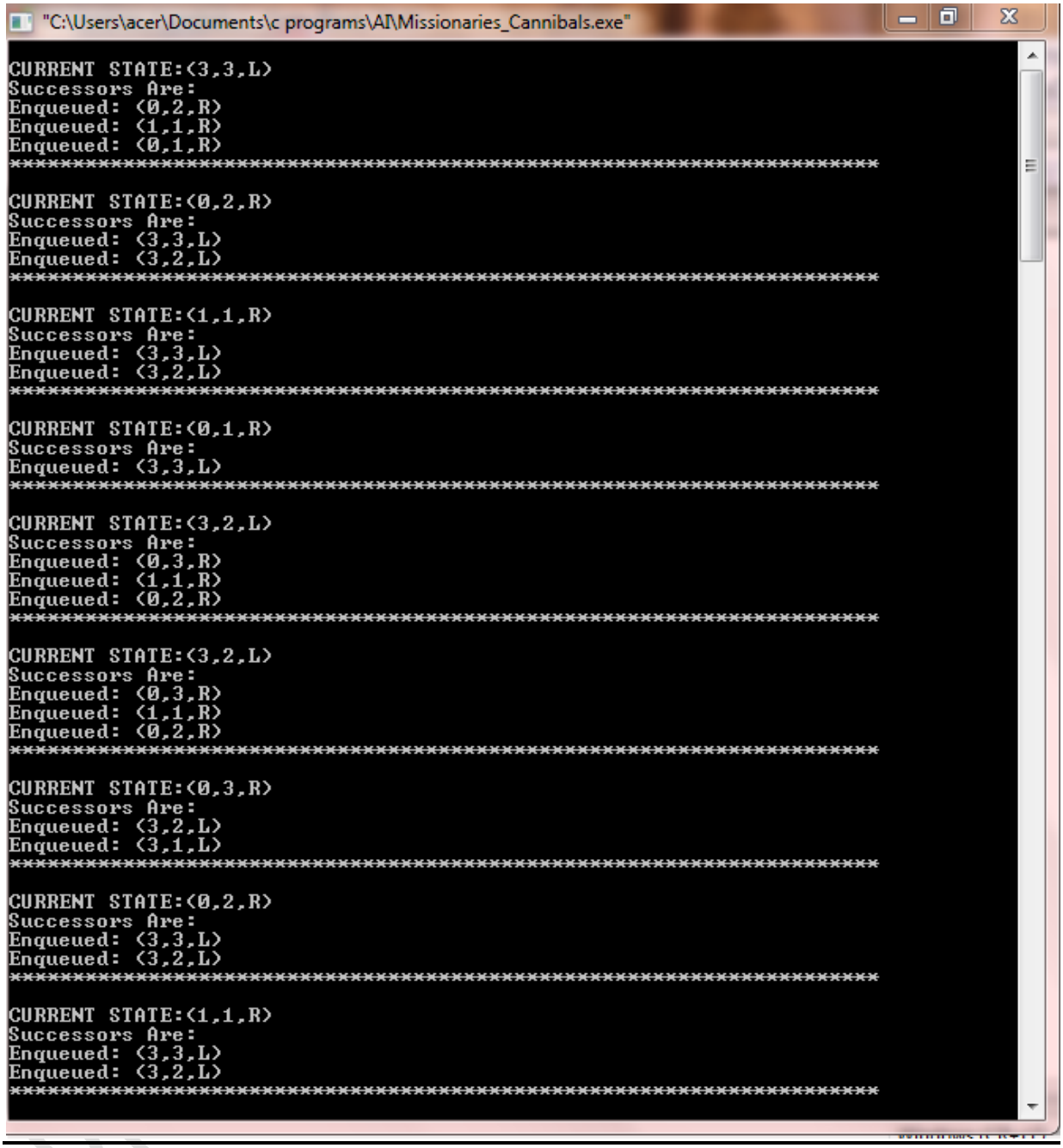
Since the queue is empty, we have completed searching.

Procedure/Algorithm:

Here openlist is Queue and Closelist is visited

1. Create a structure for storing current state
2. Function to generate successors.
3. Create a structure for storing all successors of current state.
4. Create a structure for storing non-repeating/Not visited(Open) successors
5. Create a structure for storing repeating/Visited (Close) successors
6. Function for moving from current to next state
7. If generated successor has been already traversed, putting it into closed structure
8. If generated successor is a new state, putting it into open structure
9. Print if goal state achieved else Backtracking if goal state not achieved

Sample Output:



Conclusion: Successfully implemented BFS for missionaries and cannibal problem.

Review Questions:

1)What is BFS? How it Works?

2)What is Missionaries and Cannibal Problem?

3)How to implement BFS for missionaries and cannibal?