LEETCODE | 841 |

DATE

Keys and Rooms

Input : rooms = [[1], [2], [3], []]

Output : true

Intuition We need to find if all rooms can be visited so we need to go branch by branch. So it is a Graph problem. We can solve it with BFS / DFS.

Approach BFS

Step1 0 → [1] (Looks like adjacency Matrix here)
1 → [2]
2 → [3]
3 → []

Queue <Integer> queue is the first step

visited = {0, 0, 0, 0} (length of rooms)

Take 1st element (0)th in queue

for loop → for (int i=0; i < rooms.get(0).size(); i++){
queue.offer(rooms.get(0).get(i));
visited[0] = 1
}

Step2 While loop → while (!queue.isEmpty){

→ poll queue → queue.poll()

Integer currentRoom

Step3 if (visited[currentRoom] == 0) {
visited[currentRoom] = 1;
for (Integer it : rooms.get(currentRoom)){
if (visited[it] == 0){
queue.offer(it); ← offer to queue the next room
}
}
}

Step3 Check for visited array if any element is still '0'

```
for (int i=0; i< visited.length; i++){
    if(visited[i]==0){
        return false;
    }
}
return true;
```

Some of any room is still unvisited

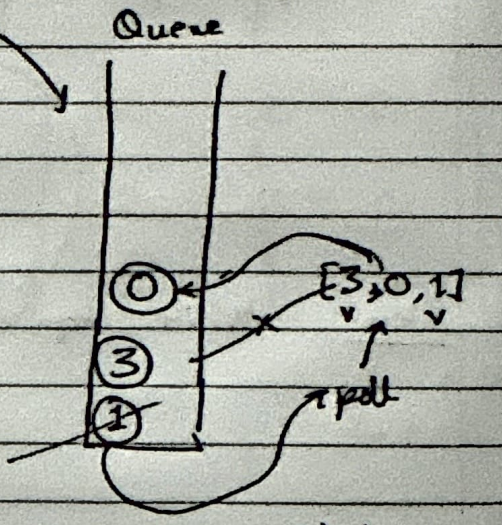
For

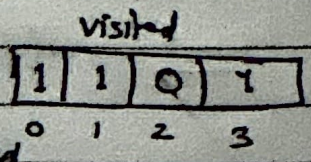
Input: 0->[1, 3]

1->[3, 0, 1]

2->[2]

3->[0]





Final Array Visited

Result : false (still 2nd room is unvisited)