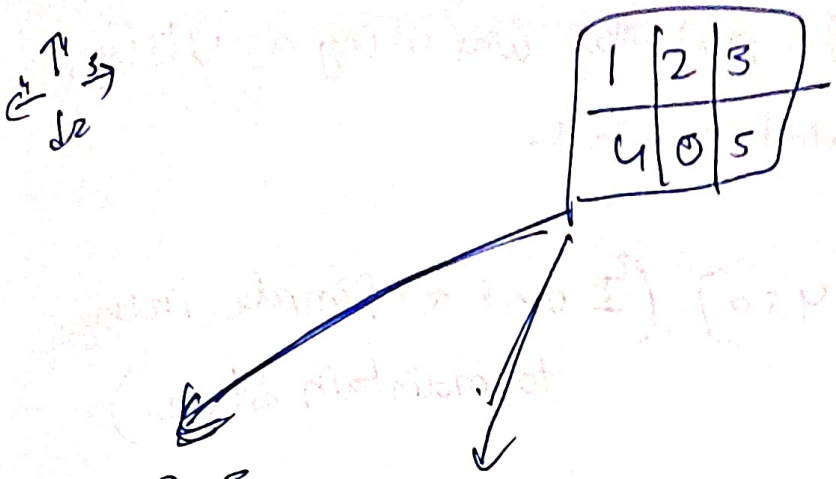
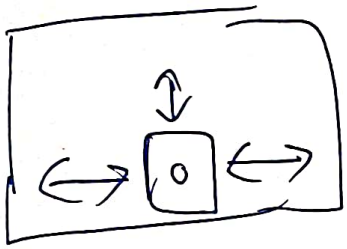


Assume that you have read the problem

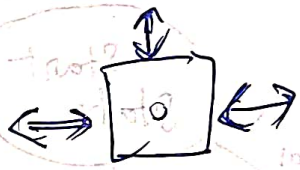
1 0 3

4 2 5

What happened when I first saw the problem?



Possible choices of back tracking came to my mind.



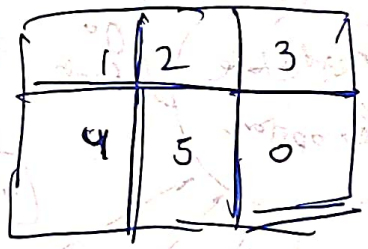
given a cell

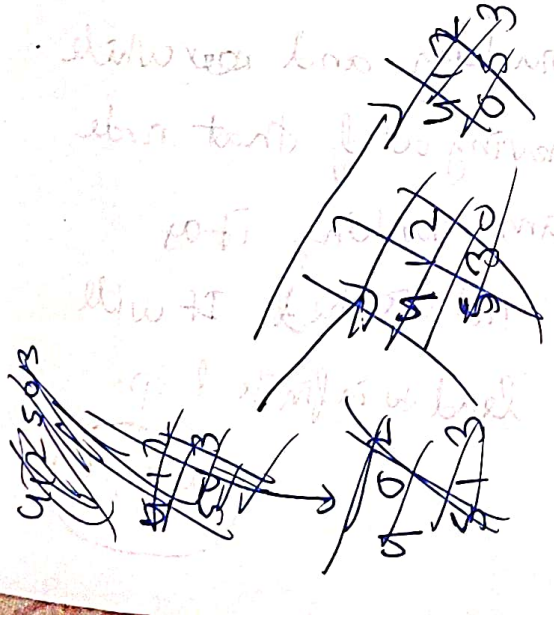
(i, j)

We will try to swap it with the all 4 directions of the cell.

We will try to see what match are we going to get on which state we are going to get next.

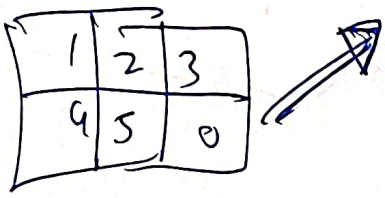
Goal is to reach





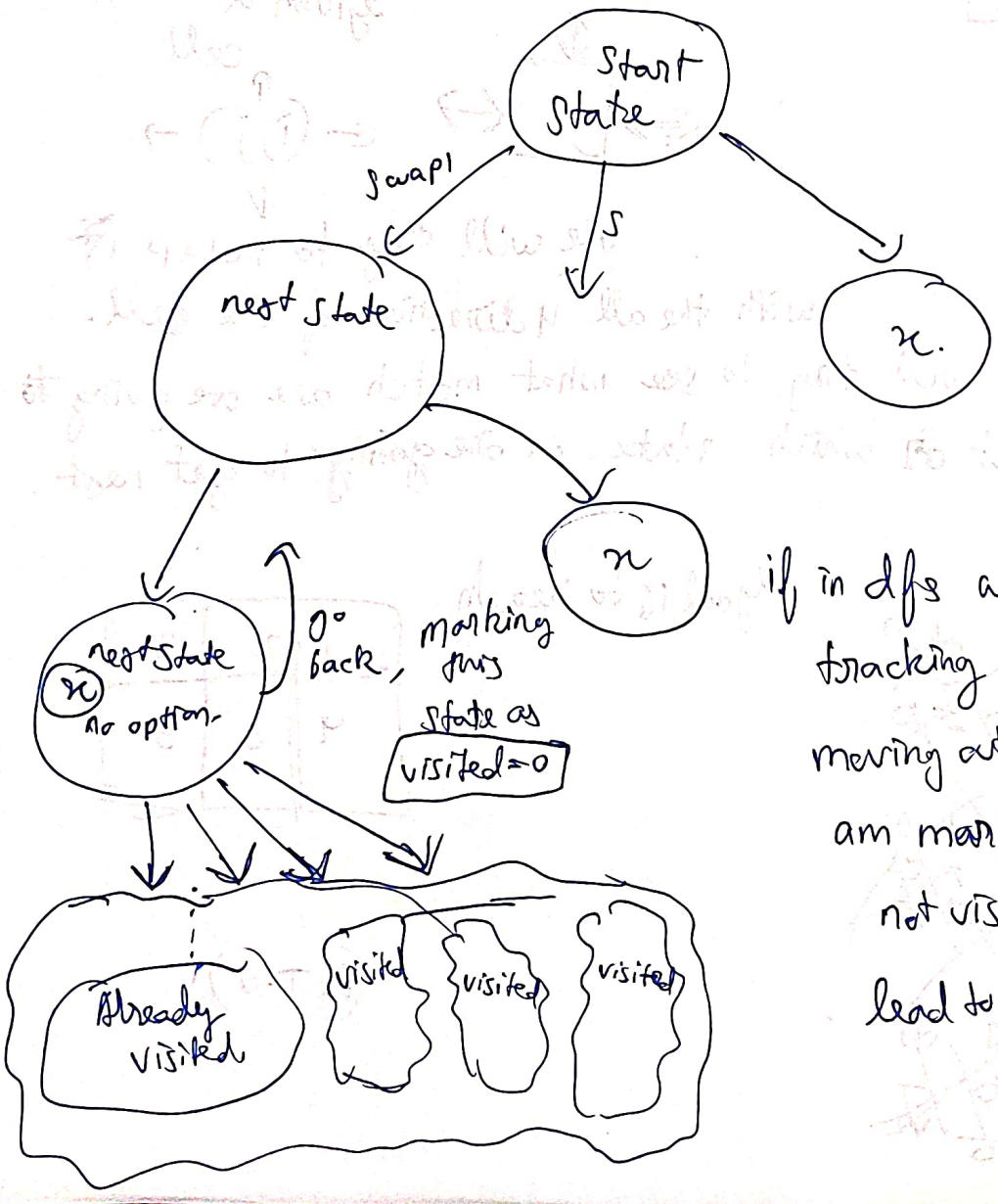
(P.T.O)

I started with dfs and naively way using a visited datastructure to maintain state



[1 2 3 4 5 0] (I used a simple integer to maintain state)

At the end of the dfs call
I was again marking the visited of the node as false
which was resulting in TLE.

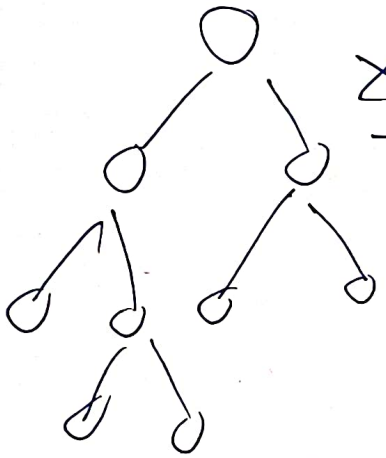


if in dfs when an back tracking and while moving out of that node am marking it as not visited. It will lead to infinite loop.

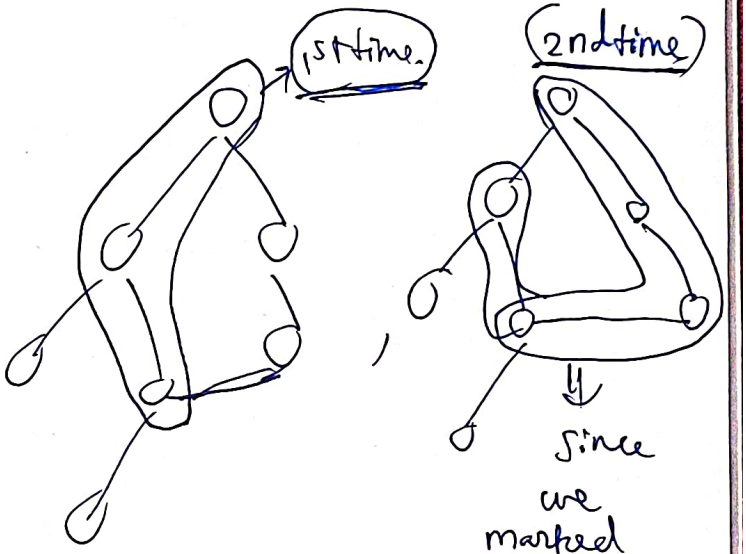
Why?

Because I made the mistake of thinking the transfers to State will represent a undirected tree.

But in reality, its like a undirected possible cycle containing graph.



✗ Not → case

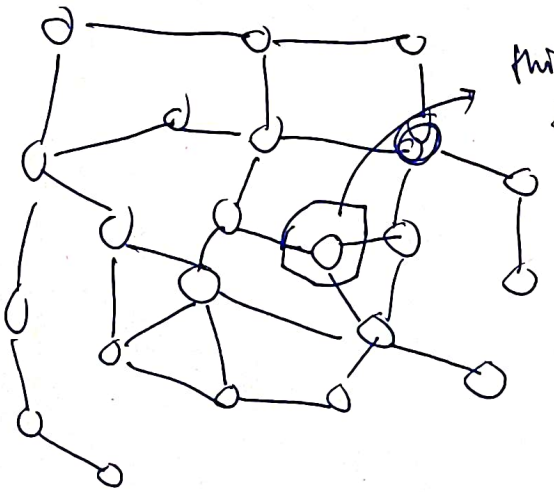


Since we marked visited=false while backtracking which is a problem.

That why TLE was happening

(John) I saw editorial.

Hassle Free way, do a BFS.



this is the state, simple visited and BFS solves

Another way,
Like in dijkstra,
we keep (node, distance) map,
and we go inside that node only if the distance calculated presently is less than the earlier one

So have a (State, Score) map.

More code complexity.