**Problem :** [**https://youtu.be/cDHdlG51Ums?t=199**](https://youtu.be/cDHdlG51Ums?t=199)

**Approach : BFS.**

-> Imagine it as a graph .

-> Every lvl represents the number of seconds.

-> Each node has following info :

1. The cell number reached currently
2. The position of the grid

position 0 : grid is in original position.

position 1 : grid directions are rotated once clockwise.

same way 2 and 3 and back to 0.

-> So from starting cell , you can go to adjacent possible cell according to given direction,and enter node representing **[adjacent cell number,grid in same position]** or enter the node representing **[adjacent cell number,grid in position rotated clockwise once].**

**But it’s certain that the nodes added at this time will represent 1 second .**

-> Now next time both these will be popped and this will generate 4 other possibilities.

-> **(imp)**But what if we get to the same cell again. We maintain a visited array and it has 2 states

**visited[cell location][how many times rotated] which represents a unique subproblem,** when we have already visited a cell, we are sure that we have visited it in optimal time possible, but the configuration of the grid might be such that we may reach destination late, so the other state is also required.

-> So **Time = O((n\*m)\*4) = O(n\*m)**

**Space = O(n\*m\*4) = O(n\*m\*4)**