My thought process,

First time when I read the problem, the thing that clicked in my brain →

f(startposition, prevHouse(colour)) {

(thats why we need to pass this parameter)

minCost = MAX_VALUE;

for(.....){

(Trigger this when K != prev color.)

minCost = min (minCost, cost of Colour K for this house + f(startPosition + 1, colorK))

}

}

total states = (n * K)

→ no of total colors.

no of total houses

And in each state we are iterating over K colors.

$(n \cdot K) * K = O(nK^2)$ state

Then I saw a follow up →

Can You do it in O(nK).

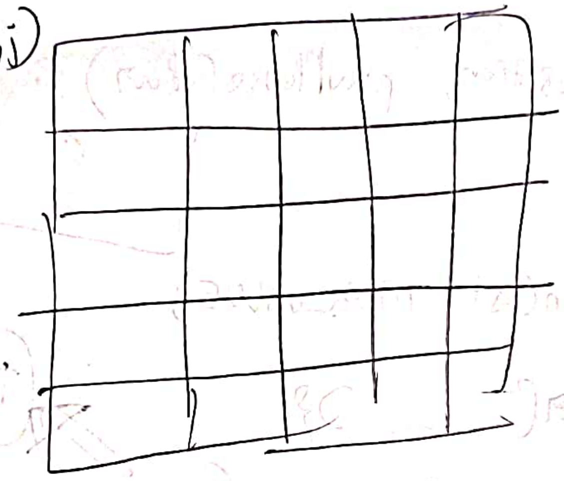
⇓

This got me thinking can I use iterative dp instead.

P.T.O

For, during recursion, we saw that the house which we colour first does not matter, just adjacent colour should not be the same.

So we did dp(i,j)



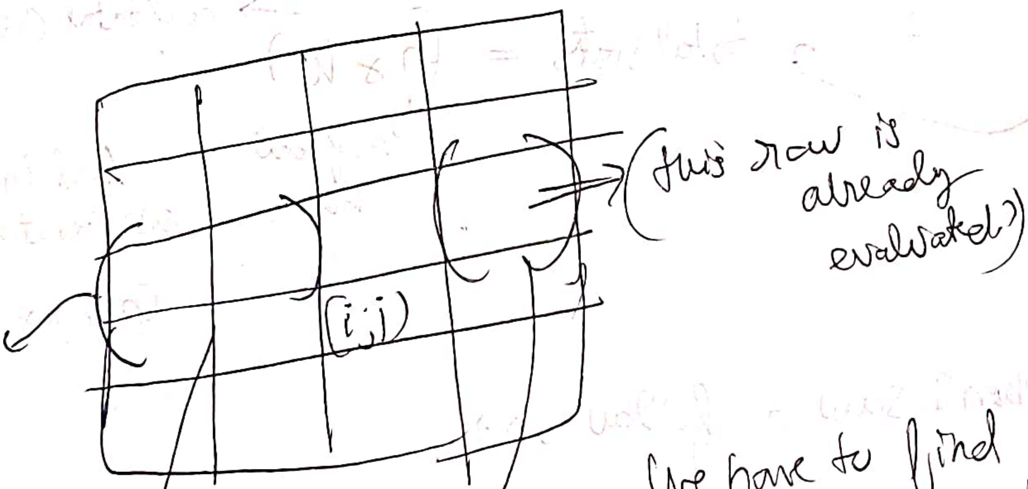
~~dp(...)~~

↓

dp(i,j).

⇓

tells one, what is the minimum cost to paint house i, with colour j, given all the (i-1) are painted, and we have tried all the colors for that house



For house i to have color j, house(i-1) cannot have color j. it can have any of [(0 - j-1) or (j+1 to k-1)] color

We have to find the minimum of both,

and

dp[i][j] = cost[i][j] + min(dp[0], ..., dp[j-1], dp[j+1] ... dp[k-1])

(consider dp[i-1][j])