ॐ

[1, 0, -1],
[1, 1, 1]]

How to identify if it is a DP Problem.

1. Count total no:of ways
2. There are multiple ways of doing it & you are asked to find Minimum / Maximum

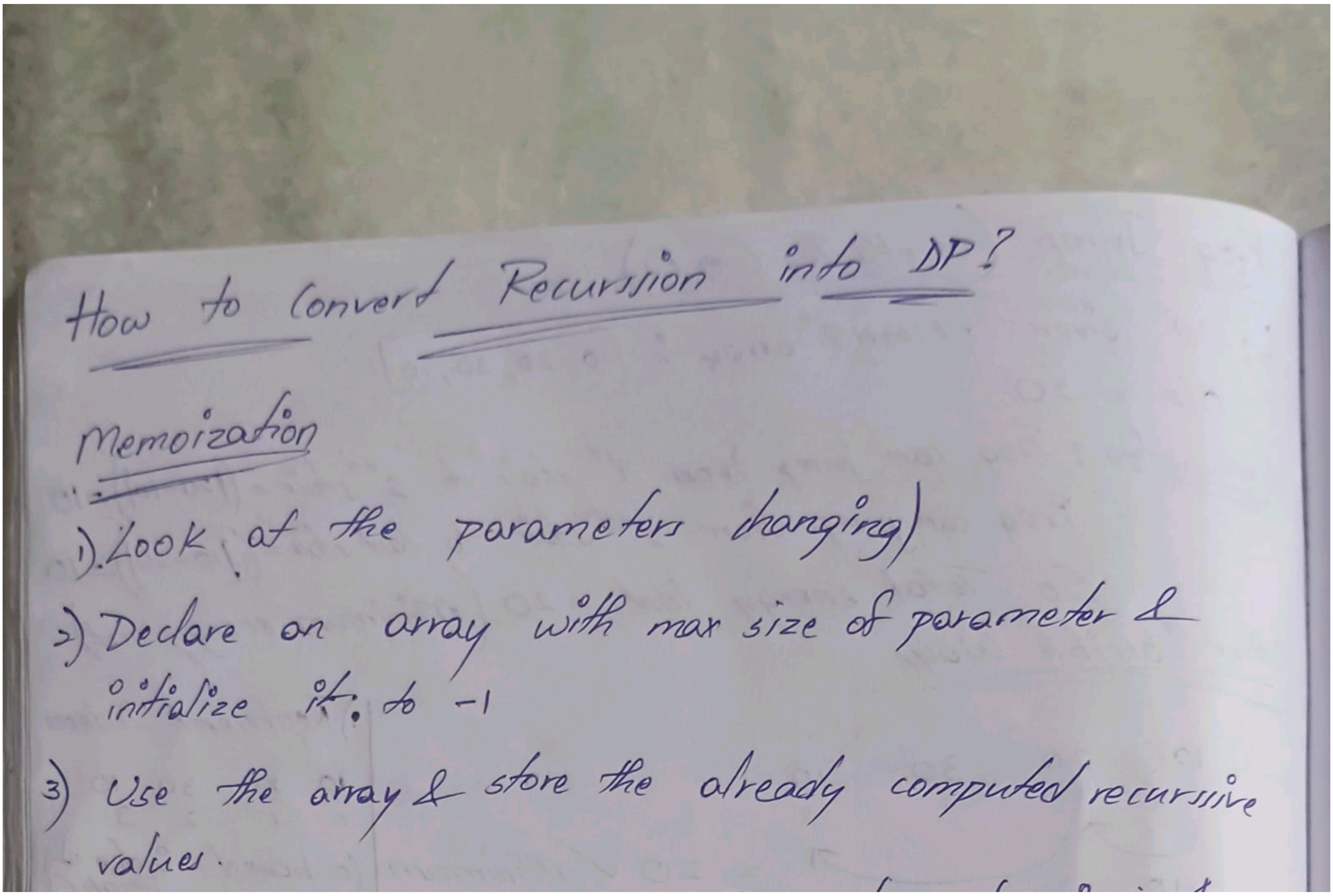
} Tend to do it Using Recursion

Try all possible ways
- Count
- Best Way

} This is when you try to apply recurssion

(Shortcut to Recursion)

1) Try to represent the problem in terms of index
2) Do all possible stuffs on that index according to the problem statement.
3) If Count All Ways ⟹ Sum up all stuffs
If Find Min ⟹ Minimum of all stuffs
If Find Max ⟹ Maximum of all stuffs



How to Convert Recursion into DP?

Memoization

1). Look at the parameters changing)

2) Declare an array with max size of parameter & initialize it to -1

3) Use the array & store the already computed recursive values.

Tabulation

In recurssion you start from the top f(5) and go to the bottom hence it is a Top-Down Approach, Tabulation is the exact opposite i.e.., we calculate & Bottom-Up

Steps

1) Initialize the dp with dp[n+1] = -1

2) Check for Base Cases & write in the form of dp

In Frog Jump if (ind = = 0) return 0 $\Longrightarrow$ dp[0] = 0;

In memoization/Recurssion jump1, jump2 operations are being performed from f(5) ---> f(0) (since it is Top Down Approach).

In Tabulation

Space Optimization

wherever you see dp[ind-1], dp[ind-2], you can perform space optimization. In Space Optimization instead of wing array you store the values in variables.

FROG JUMP

	dp[i-2]	dp[i-1]	dp[i]			
0	1	2	3	4	------	n-1

↓ next step

		dp[i-2]	dp[i-1]	dp[i]		
0	1	2	3	4	-----	n-1

At any position you only need dp[i-1] & dp[i-2] & you don't need anyone else. Therefore you store dp[i-1] & dp[i-2] in variables as prev1, prev2 instead of wing array

As next steps you update prev1 = curr

prev2 = prev1

Code