Placement
↓
Coding Round → TCS (3-4 LPA)

Interview

PW SKILLS

Dynamic Programming

DP

Lecture 1

Today’s checklist

1. What is DP
2. Nth Fibonacci number (Recursion)
3. Nth Fibonacci number (Recursion + Memoization)
4. Nth Fibonacci number (Tabulation)
5. Min Cost Climbing Stairs
6. MazePath (Right & Down)

What is Dynamic Programming?

PW SKILLS

1) optimised Recursion

↓

Solving a bigger problem using subproblems

DP

↓

overlapping subproblems

Ques:

PW SKILLS

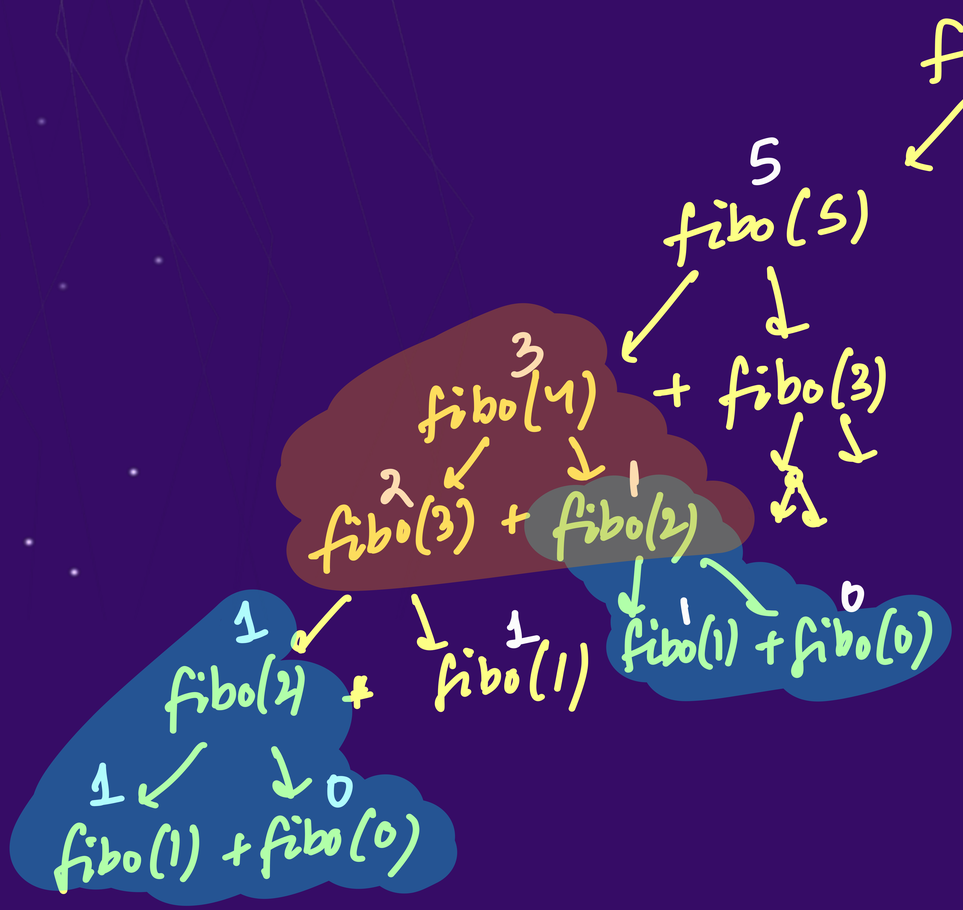
Q1 : Fibonacci Number (Recursion)

n =	0	1	2	3	4	5	6	7	8	9
	0	1	1	2	3	5	8	13	21	34

```
if (n==0 || n==1) return n;
fibo(n) = fibo(n-1) + fibo(n-2)
```

[Leetcode 509]

Problems in Recursive Solution: → Time Complexity → $O(2^n)$



Overlapping Subproblems

height of this tree = n

No. of nodes ~ 2^n

↓

no. of calls

S.C. = O(n)

↓

Call Stack space

> means faster

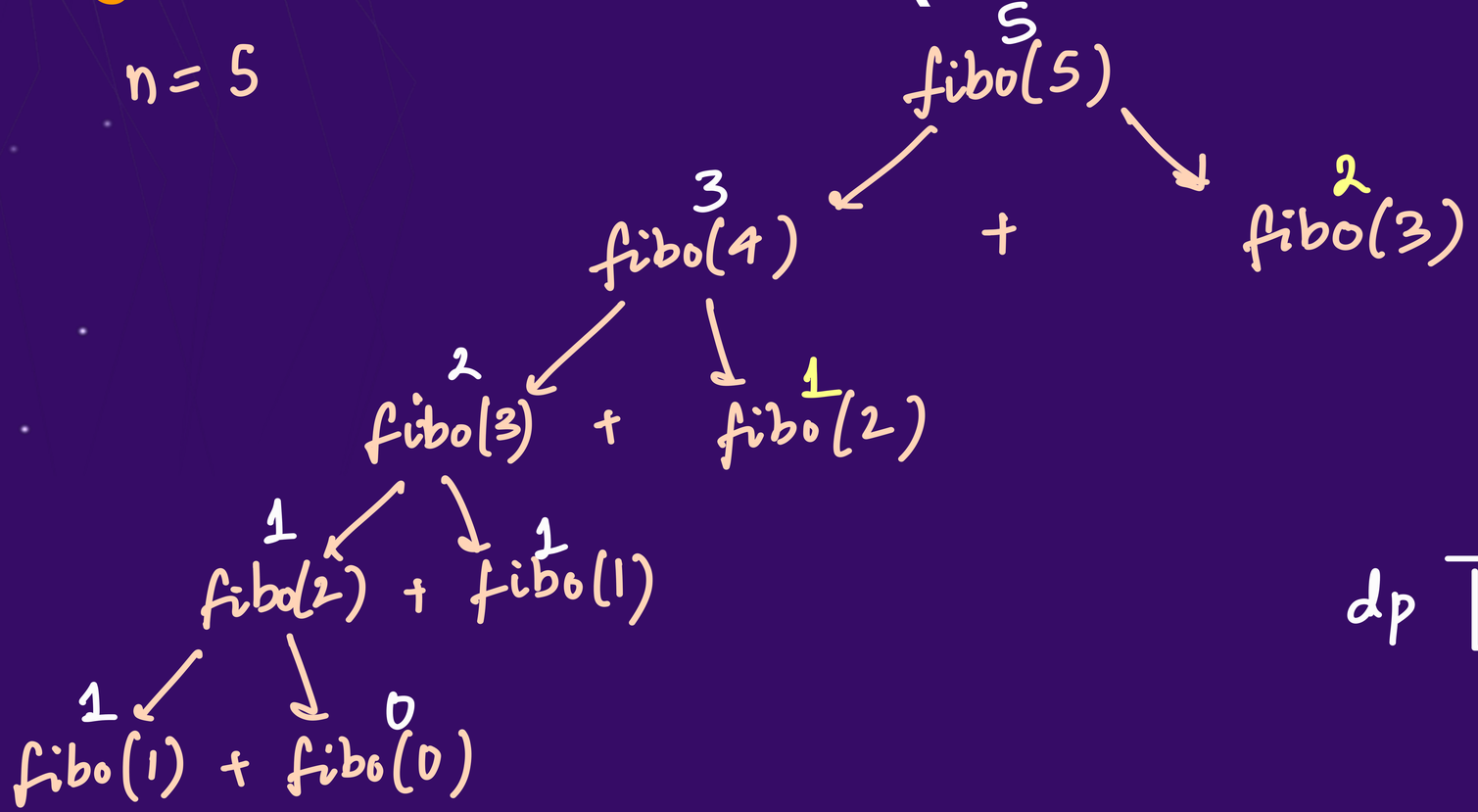
$O(\log n) > O(\sqrt{n}) > O(n) > O(n \log n) > O(n^2) > O(n^3) >>> O(2^n)$

DP → Recursion + Memoization (TOP DOWN DP)

↳ Tabulation (BOTTOM UP DP)

Ques:

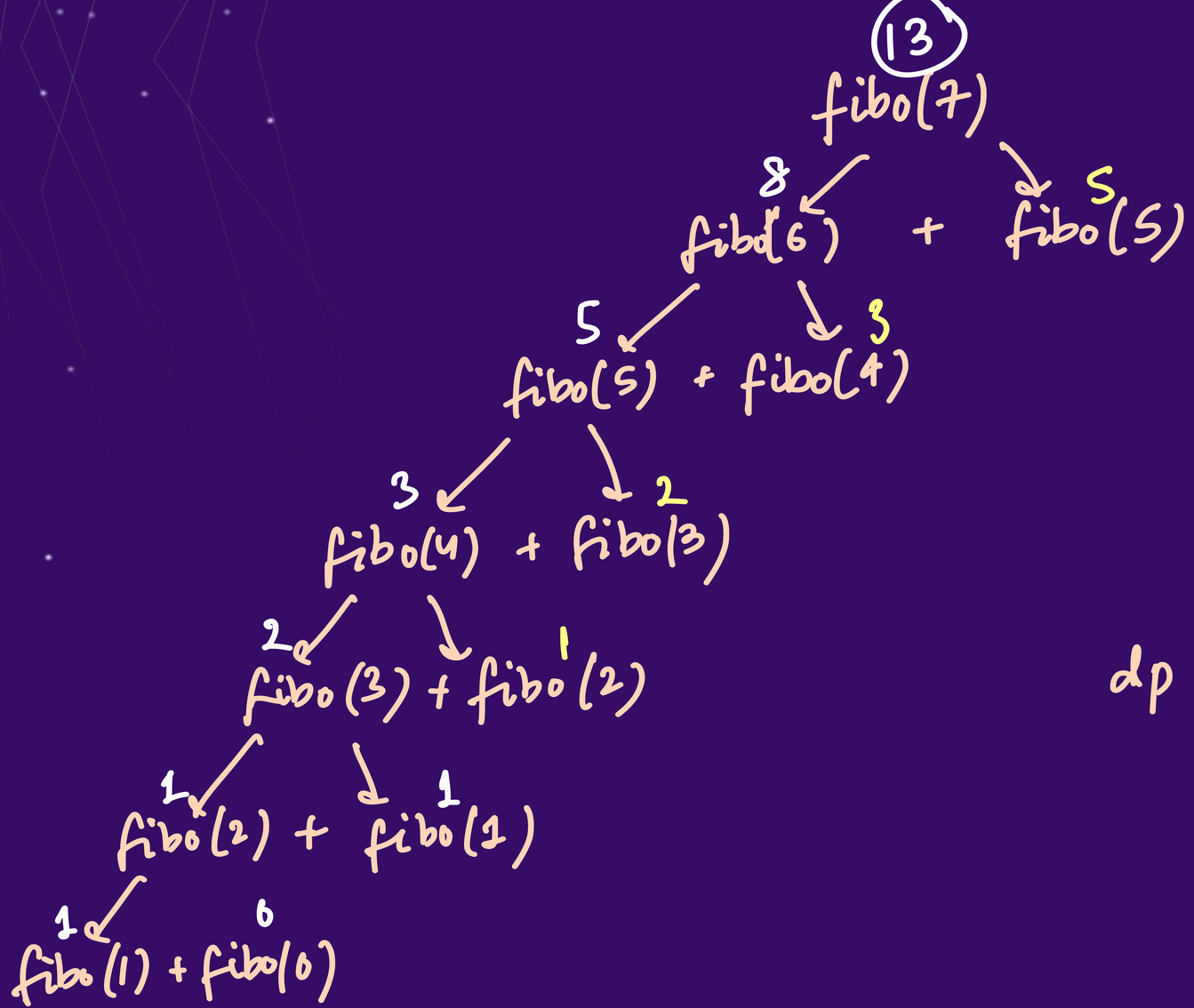
Q1 : Fibonacci Number (Recursion + Memoization)



dp

0	1	2	3	4	5
-1	-1	1	2	3	5

[Leetcode 509]



$T.C. = O(n)$

$S.C. = O(n) + O(n)$

$O(n)$ ↓ Extra array

$O(n)$ ↓ Call Stack

	0	1	2	3	4	5	6	7
dp	-1	-1	1	2	3	5	8	-1

```cpp
int fibo(int n, vector<int>& dp){
    if(n<=1) return n;
    if(dp[n]!=-1) return dp[n];
    return dp[n] = fibo(n-1,dp) + fibo(n-2,dp);
}
int fib(int n) {
    vector<int> dp(n+1,-1);
    return fibo(n,dp);
}
```

Ques:

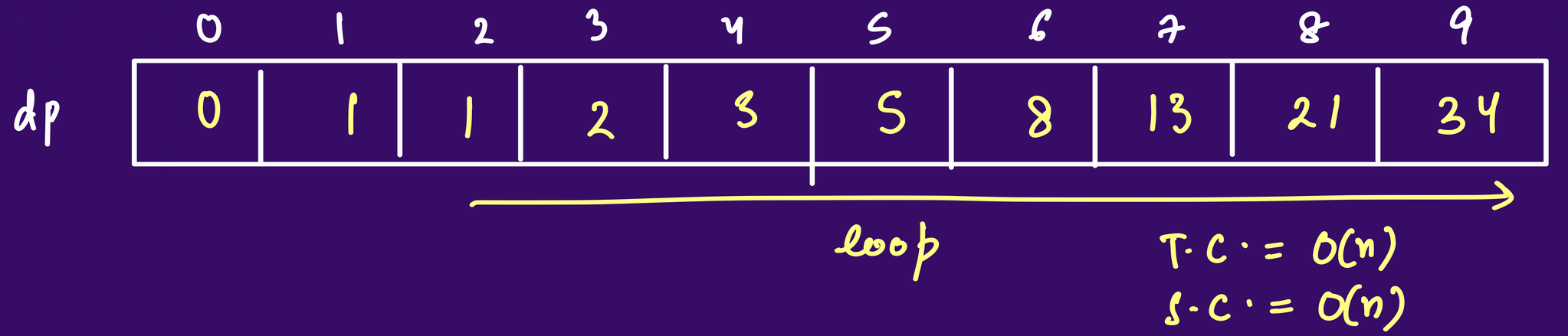
PW SKILLS

Q1 : Fibonacci Number (Tabulation)

Iterative DP

Bottom UP DP

$n = 9$ $fibo(9) = ?$




```
dp[i] = dp[i-1] + dp[i-2];
```

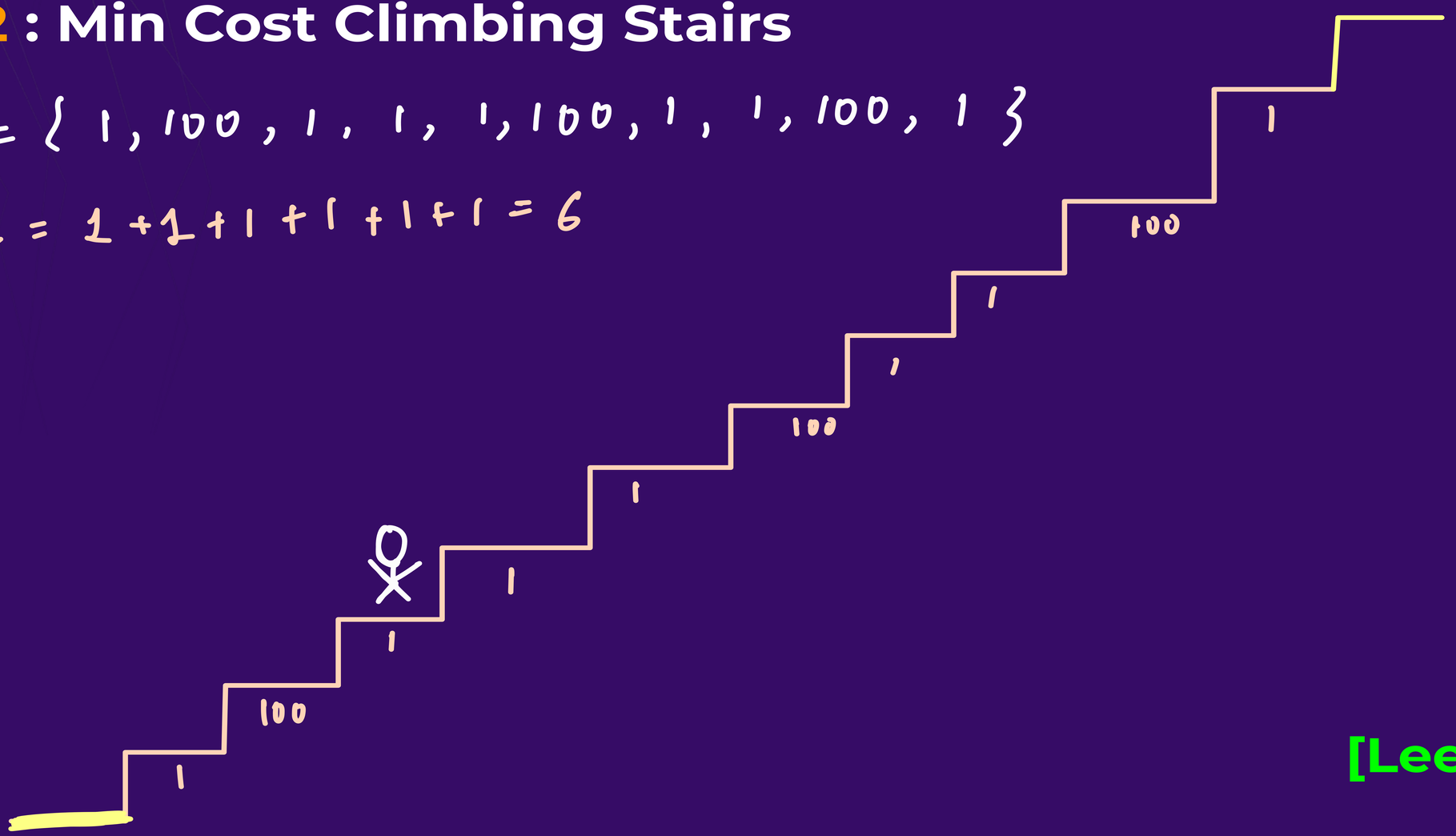
[Leetcode 509]

Ques:

Q2 : Min Cost Climbing Stairs

cost = { 1, 100, 1, 1, 1, 100, 1, 1, 100, 1 }

money = 1 + 1 + 1 + 1 + 1 + 1 = 6



[Leetcode 746]

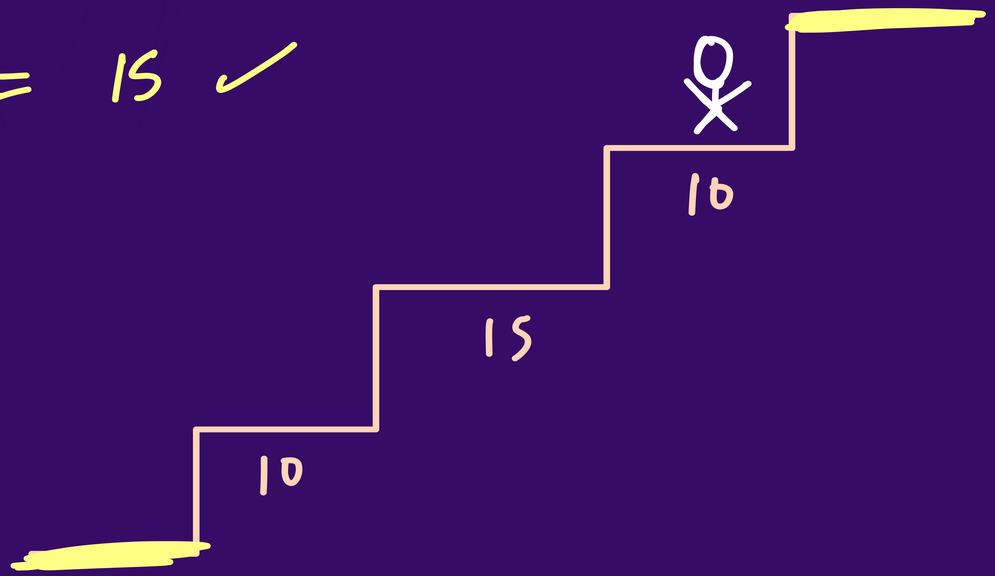
Ques:

Q2 : Min Cost Climbing Stairs

arr = {10, 15, 10}

money = 10 + 10 = 20 ✗

= 15 ✓



#Note :

It is VIMP to identify the problem that if it is Greedy or DP

dp[i] = the min cost to reach ith stair

[Leetcode 746]

Ques:

Q2 : Min Cost Climbing Stairs

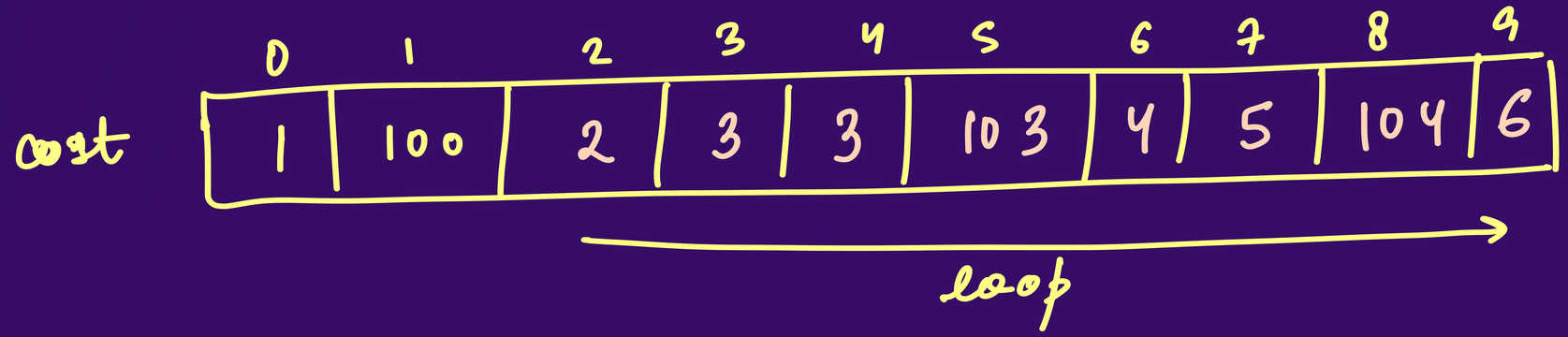
```
int helper(vector<int>& cost, int i, vector<int>& dp){
    if(i==0 || i==1) return cost[i];
    if(dp[i]!=-1) return dp[i];
    return dp[i] = cost[i] + min(helper(cost,i-1,dp),helper(cost,i-2,dp));
}
int minCostClimbingStairs(vector<int>& cost) {
    int n = cost.size();
    vector<int> dp(n,-1);
    return min(helper(cost,n-1,dp),helper(cost,n-2,dp));
}
```

[Leetcode 746]

Ques:

Q2 : Min Cost Climbing Stairs (Tabulation)

Ex-1 : cost = { 1, 100, 1, 1, 1, 100, 1, 1, 100, 1 }




```
cost[i] += min(cost[i-1], cost[i-2]);
```

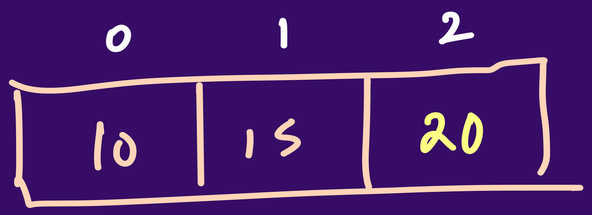
[Leetcode 746]

Ques:

Q2 : Min Cost Climbing Stairs

0	1	2
10	15	10

arr = { 10, 15, 10 }



minCost → min(arr[n-1], arr[n-2])

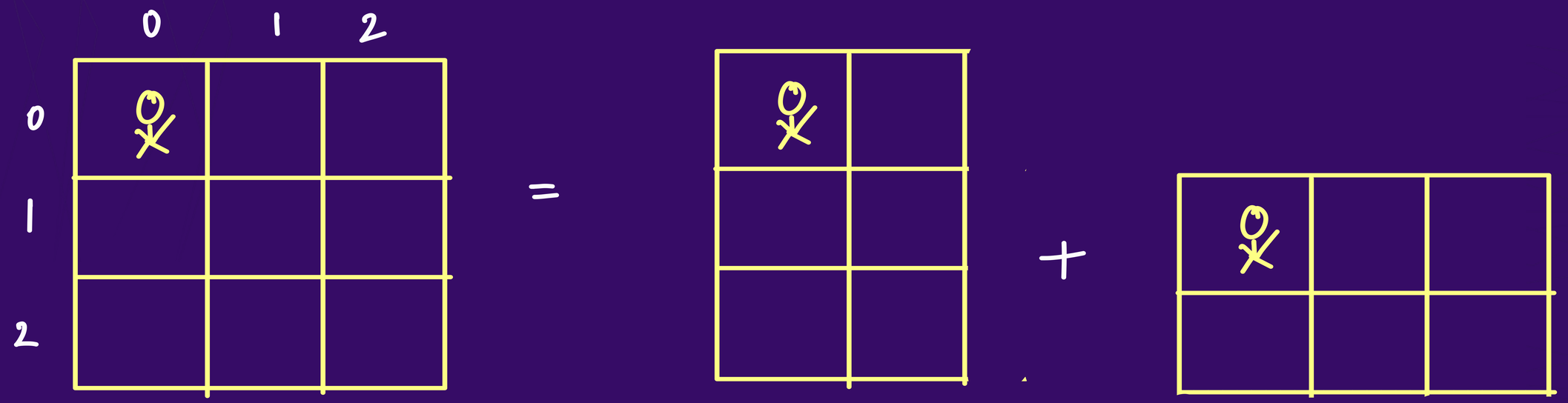
[Leetcode 746]

Ques:

(sr, sc, er, ec)

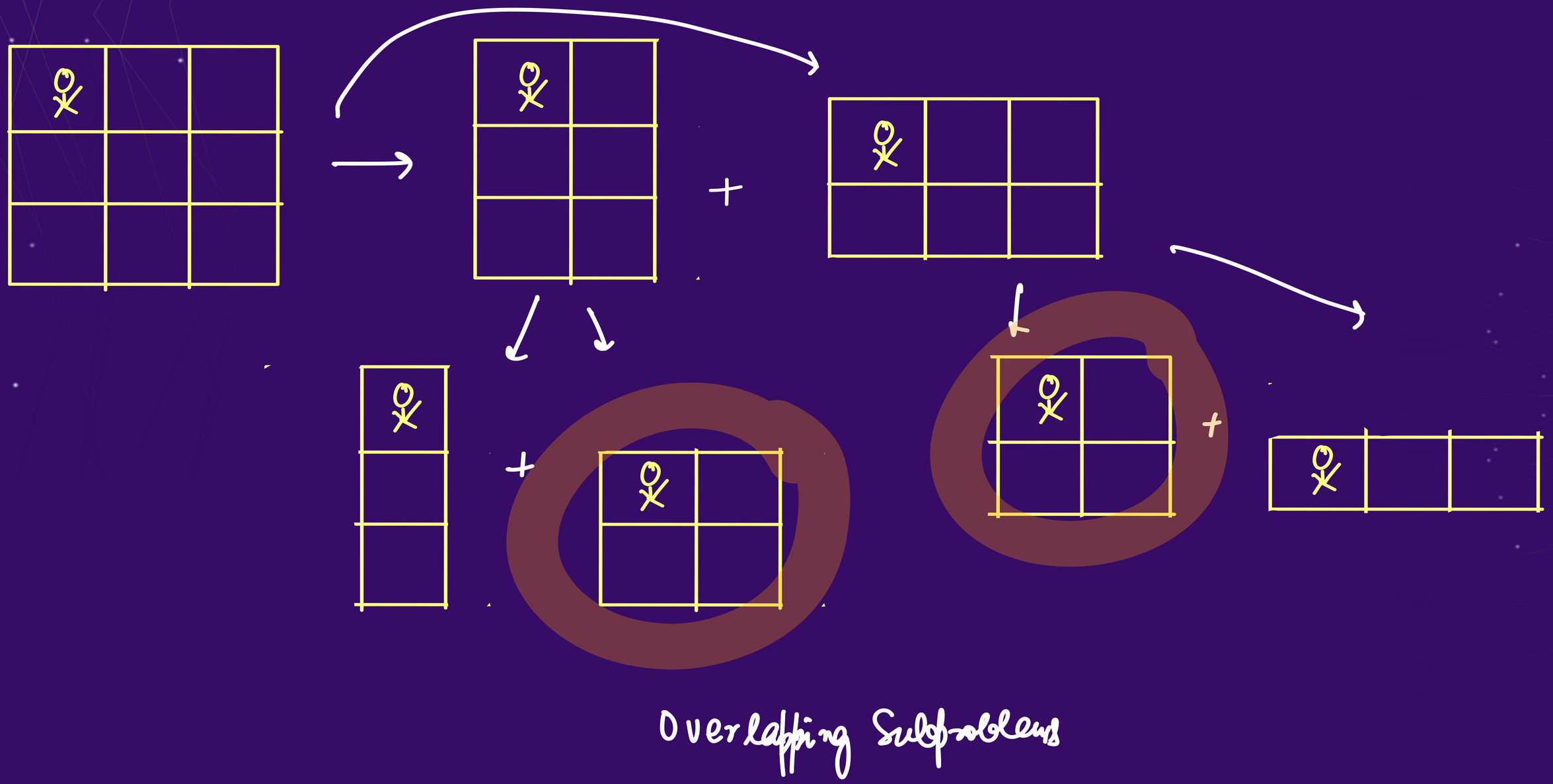
Q3 : Unique Paths

m = 3, n = 3



total ways = rightways + downways

[Leetcode 62]


Overlapping Subproblems

2D Array to memoize

1st option → int dp[m+1][n+1]

2nd option → int dp[m][n]

```
int helper(int sr, int sc, int er, int ec, vector<vector<int>>& dp){
    if(sr==er && sc==ec) return 1;
    if(sr>er || sc>ec) return 0;
    if(dp[sr][sc]!=-1) return dp[sr][sc];
    return dp[sr][sc] = helper(sr,sc+1,er,ec,dp) + helper(sr+1,sc,er,ec,dp);
}
int uniquePaths(int m, int n) {
    vector<vector<int>> dp(m,vector<int>(n,-1));
    return helper(0,0,m-1,n-1,dp);
}
```

Tabulation

int dp[m][n];

Only Right or Down

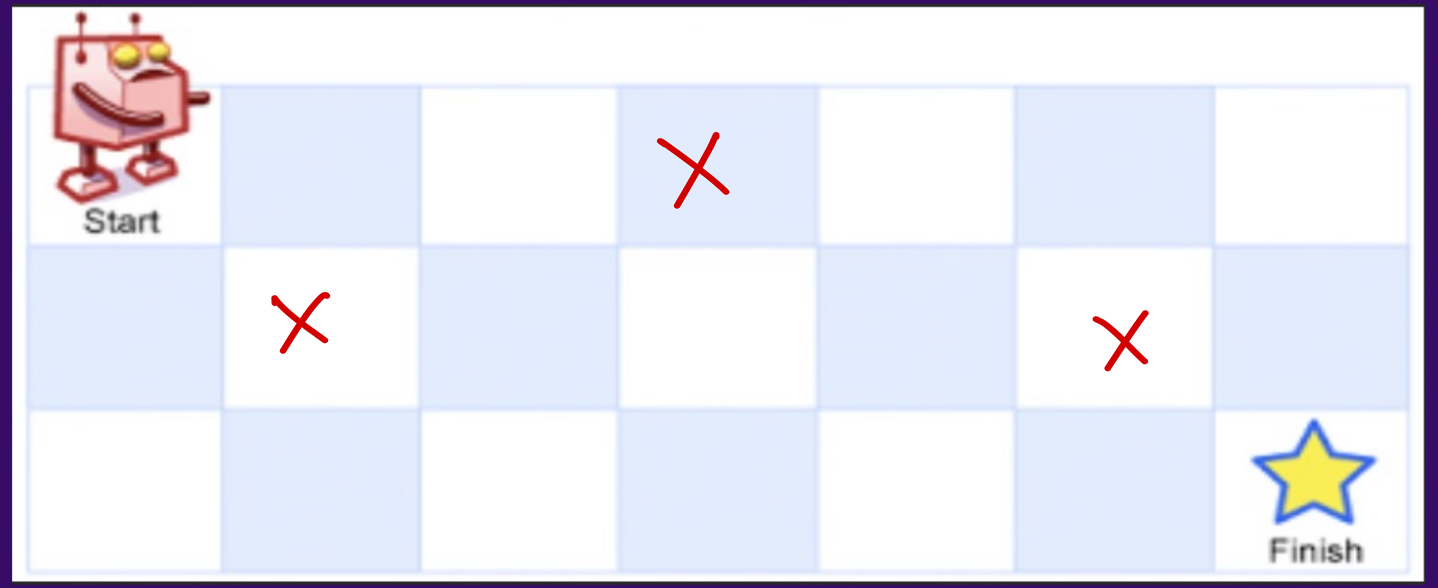
	0	1	2	3	4	5	6
0	 Start	1	1	1	1	1	1
1	1	2	3	4	5	6	7
2	1	3	6	10	15	21	 28 Finish

$$dp[i][j] = dp[i-1][j] + dp[i][j-1]$$

m = 3, n = 7

- Homework

Unique Paths II (Leetcode 63)



Homework:

Q : Nth Tribonacci Number

Recursion + Memoization

&

Tabulation

[Leetcode 1137]

THANK YOU