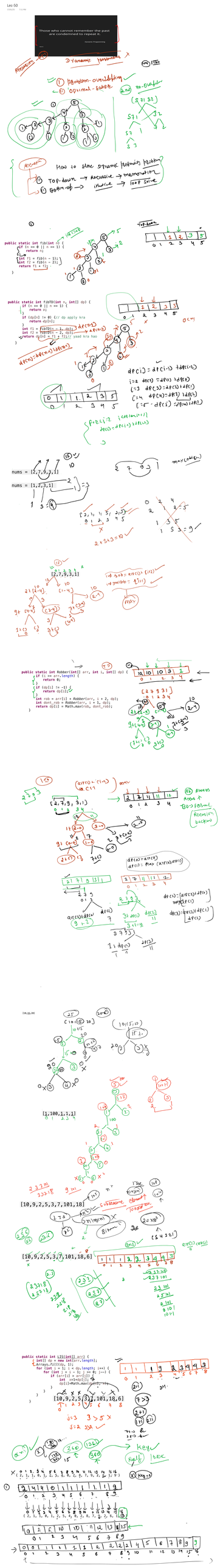
Lec-50


Those who cannot remember the past are condemned to repeat it.
— George Santayana
Recursion
Dynamic programming
① Recursion - overlapping
② optimal - substructure
Top-down
Bottom-up
How to store dynamic programming problem
① Top-down → Recursive → memoization
② Bottom-up → iterative → tool table
public static int fib(int n) {
if (n == 0 || n == 1) {
return n;
}
int f1 = fib(n - 1);
int f2 = fib(n - 2);
return f1 + f2;
}
public static int fibTD(int n, int[] dp) {
if (n == 0 || n == 1) {
return n;
}
if (dp[n] != 0) {// dp apply kra
return dp[n];
}
int f1 = fibTD(n - 1, dp);
int f2 = fibTD(n - 2, dp);
dp[n] = f1 + f2;// yaad kra hao
}
dp[i] = dp[i-1] + dp[i-2]
i=2 dp[2] = dp[0] + dp[1]
i=3 dp[3] = dp[1] + dp[2]
i=4 dp[4] = dp[2] + dp[3]
i=5 dp[5] = dp[3] + dp[4]
nums = [2,7,9,3,1]
nums = [1,2,3,1]
[2,7,9,3,1]
max(dp[n])
2+9+1 = 12
max
public static int Robber(int[] arr, int i, int[] dp) {
if (i >= arr.length) {
return 0;
}
if (dp[i] != -1) {
return dp[i];
}
int rob = arr[i] + Robber(arr, i + 2, dp);
int dont_rob = Robber(arr, i + 1, dp);
return dp[i] = Math.max(rob, dont_rob);
}
[2,7,9,3,1]
dp[i] = max(arr[i]+dp[i+2], dp[i+1])
[1,100,1,1,1]
[10,9,2,5,3,7,101,18]
[10,9,2,5,3,7,101,18,6]
[10,9,2,5,3,7,101,18,6]
i=3 3 > 5 X
i=2 2 > 5 X
Key
Key[0]
Key[1]
Key[2]
[0,1,1,2,3,5]
[0,1,1,2,3,5,4]
[0,0,1,1,1,1,1,2,2,2,2,4,5,6,7,8,9,3]