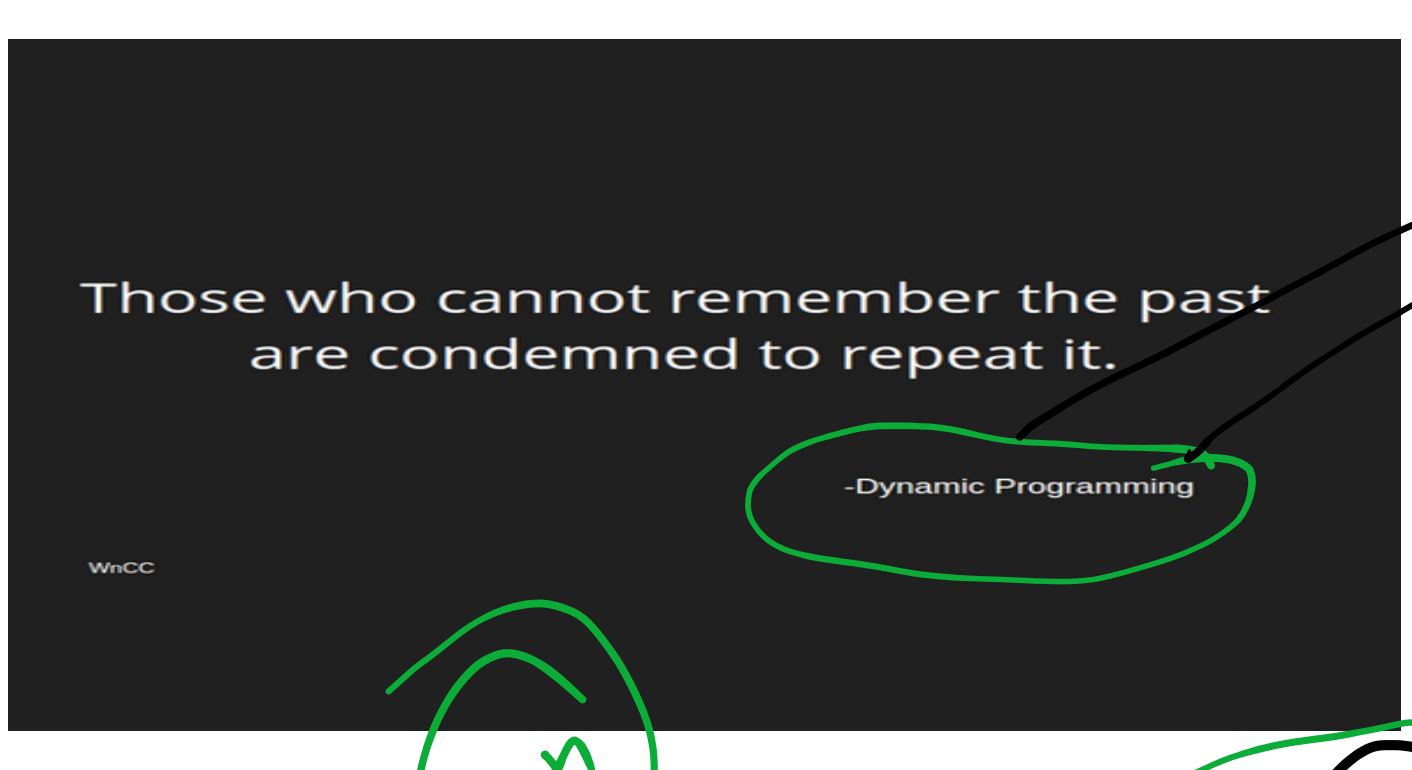
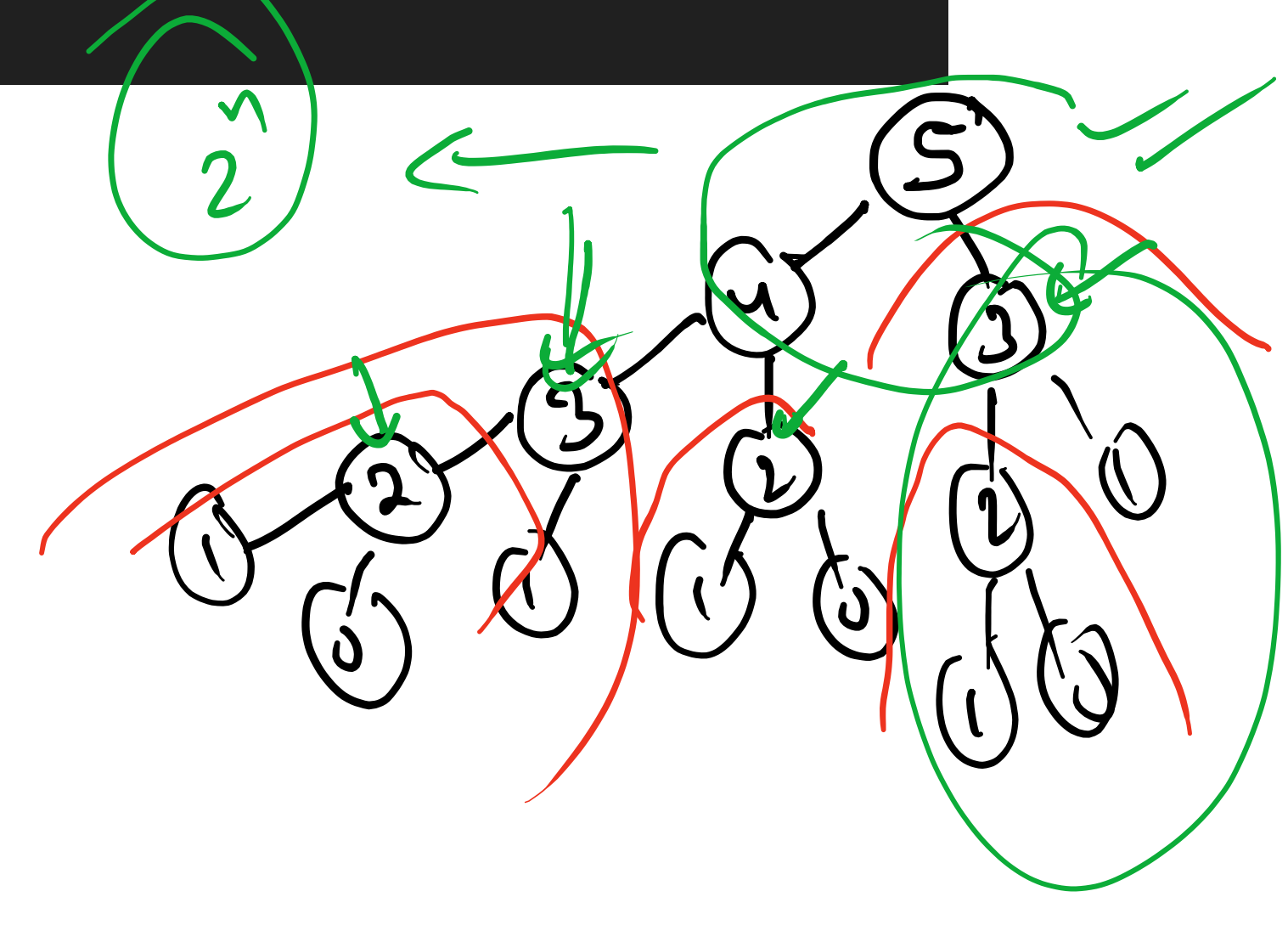
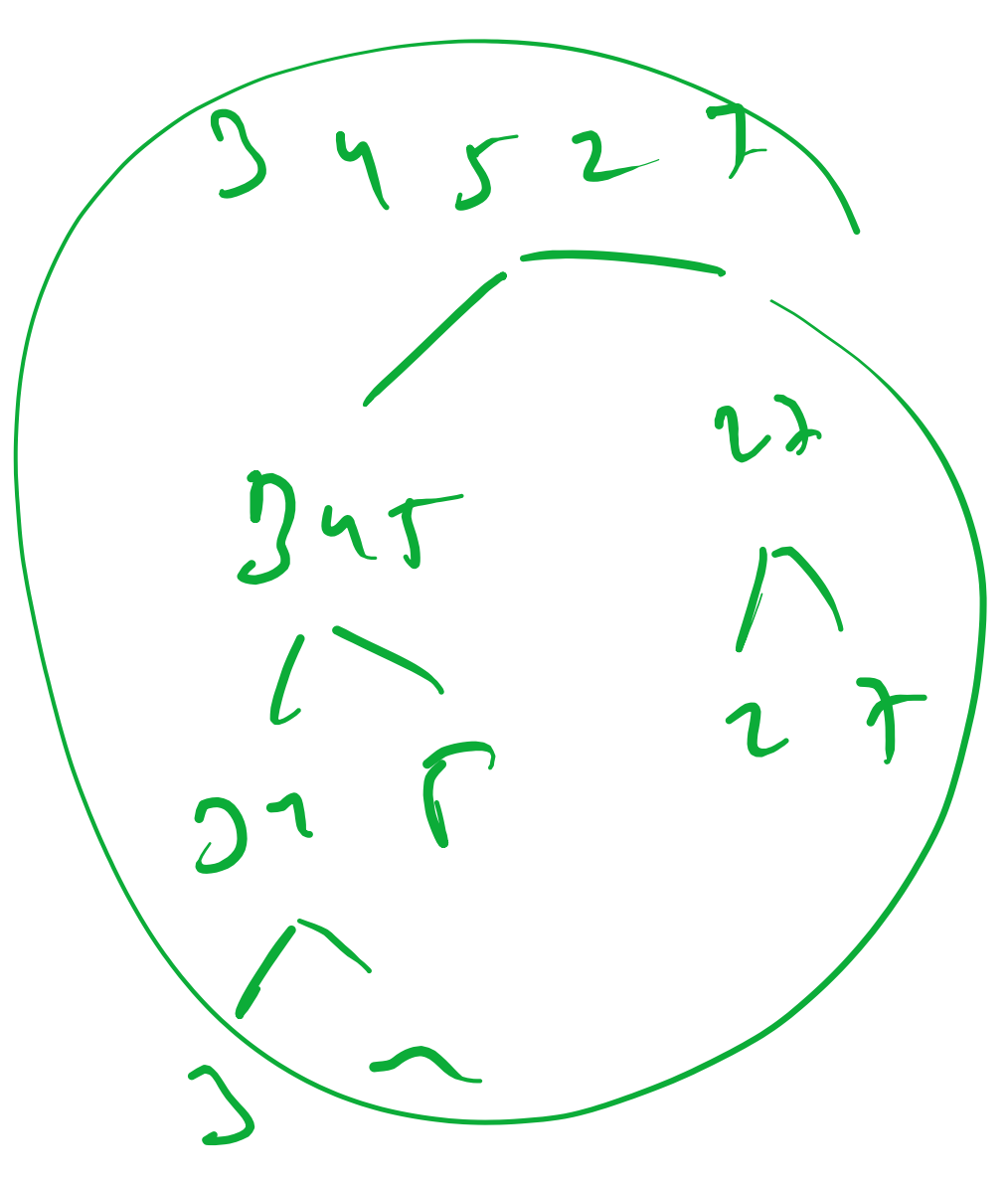
Lec-42

01/06/25 4:48 PM



1. D.P
- problem - overlapping
- optimal substructure



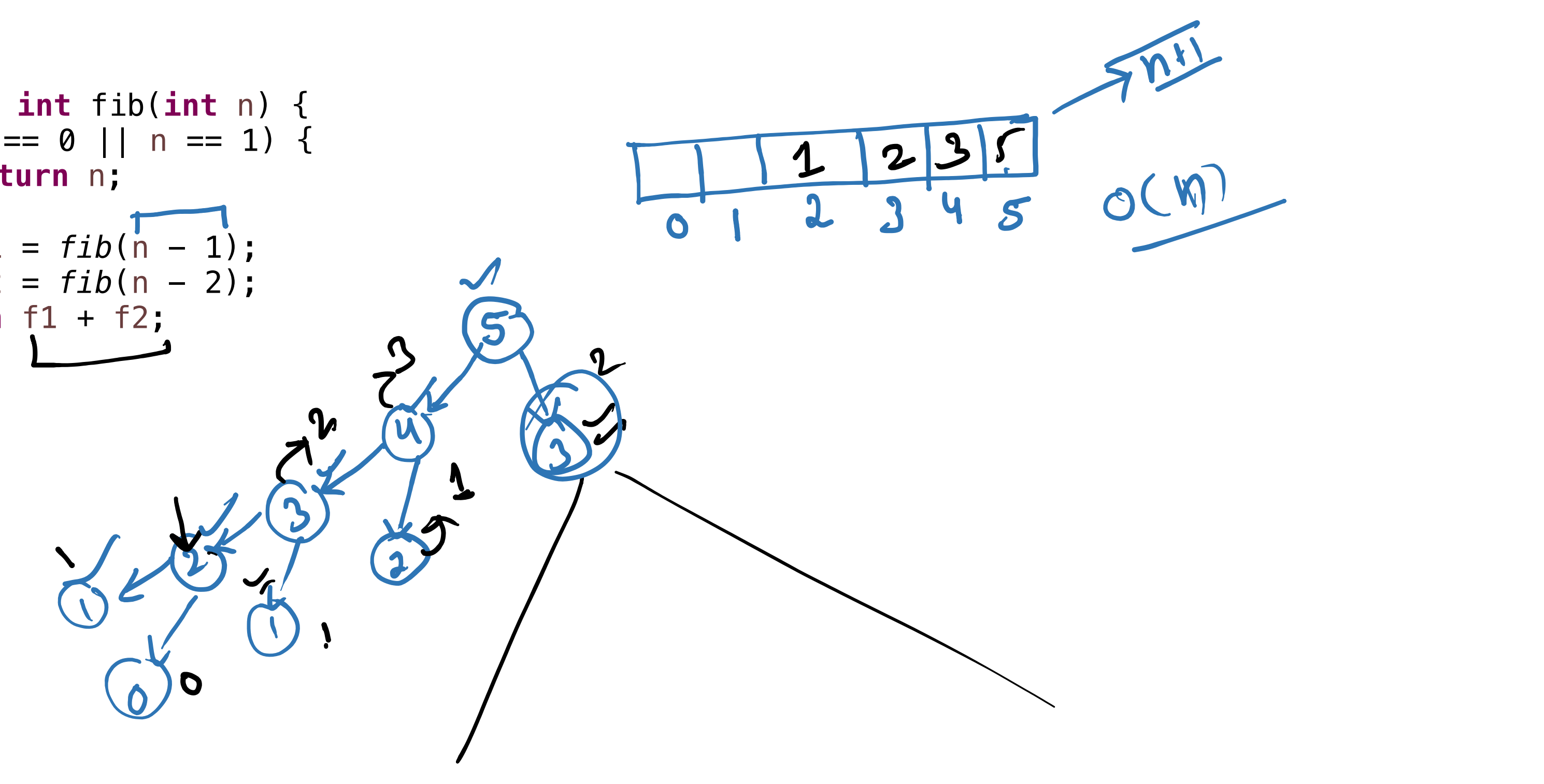


(i) Top-down → memoization

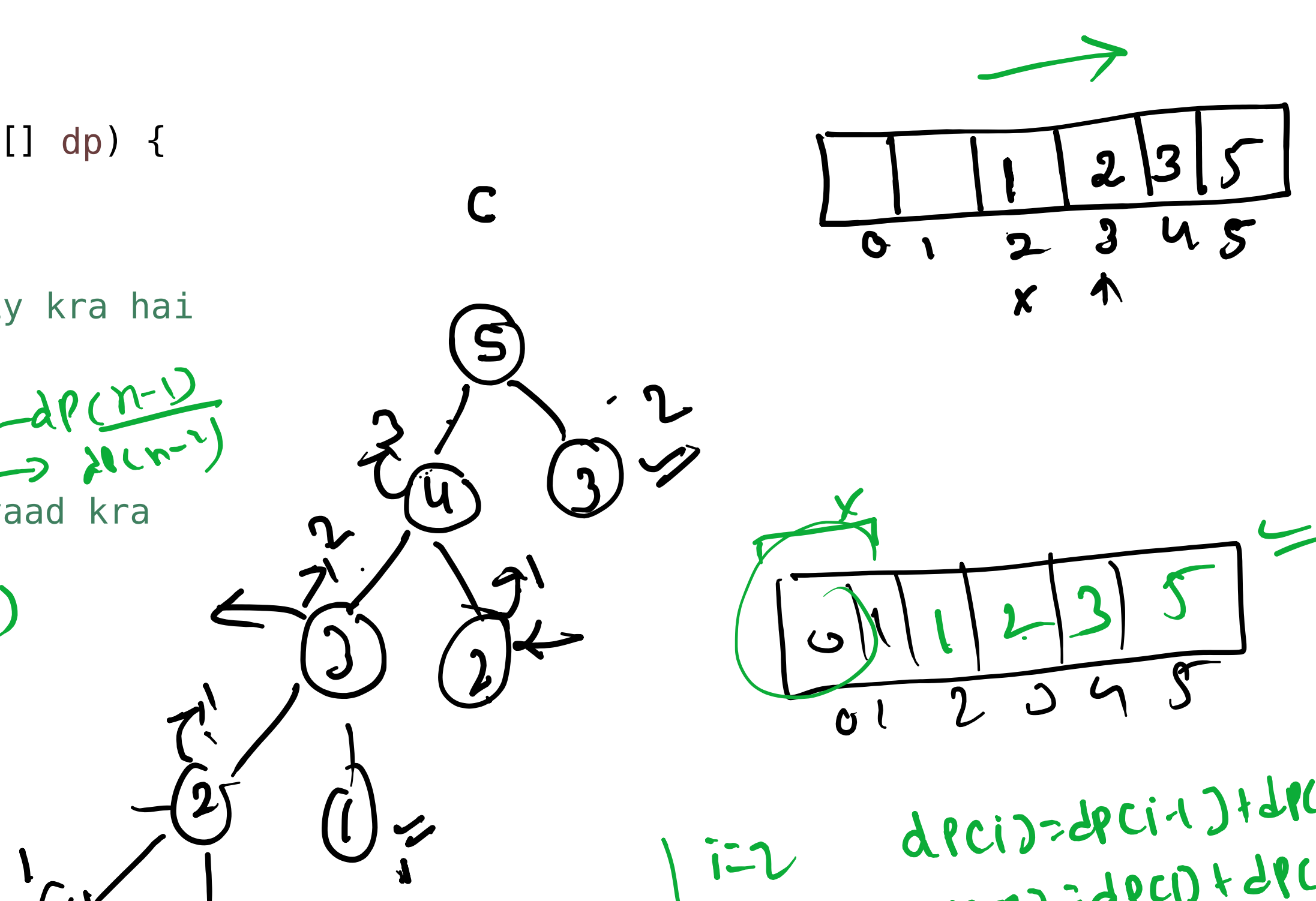
(ii) Bottom-up → Tabulation, Iterative

```java
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;
}
```

O(n)



```java
public static int fibTD(int n, int[] dp) {
    if (n == 0 || n == 1) {
        return n;
    }
    if (dp[n] != 0) {// dp apply kra hai
        return dp[n];
    }
    int f1 = fibTD(n - 1, dp);
    int f2 = fibTD(n - 2, dp);
    return dp[n] = f1 + f2;// yaad kra
}
```



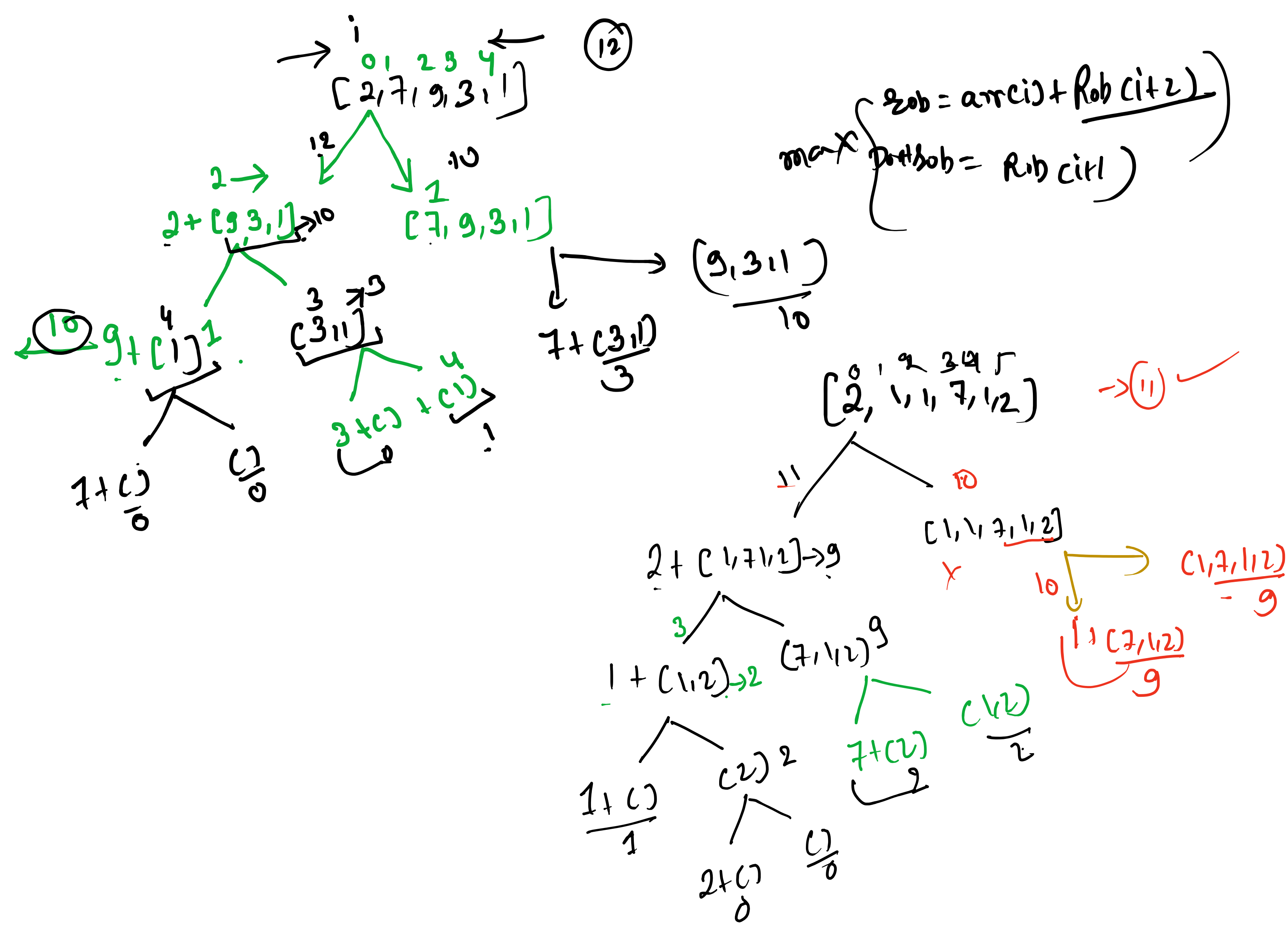
i=2 dp[i]=dp[i-1]+dp[i-2]

dp[2]=dp[1]+dp[0]

i=3 dp[3]=dp[2]+dp[1]

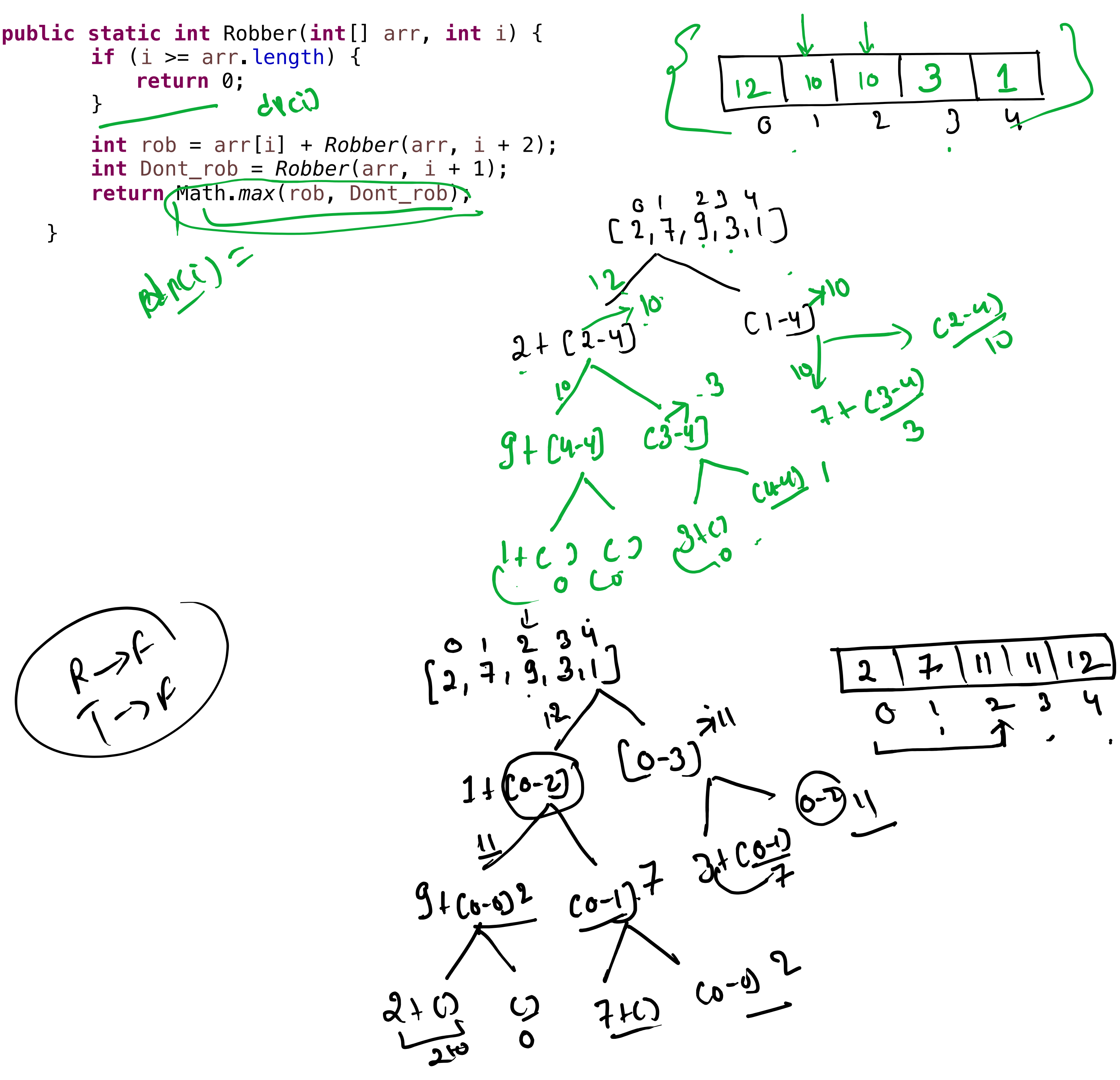
i=4 dp[4]=dp[3]+dp[2]

i=5 dp[5]=dp[4]+dp[3]



max { rob = arr[i] + Rob[i+2] ; Dont_rob = Rob[i+1] }

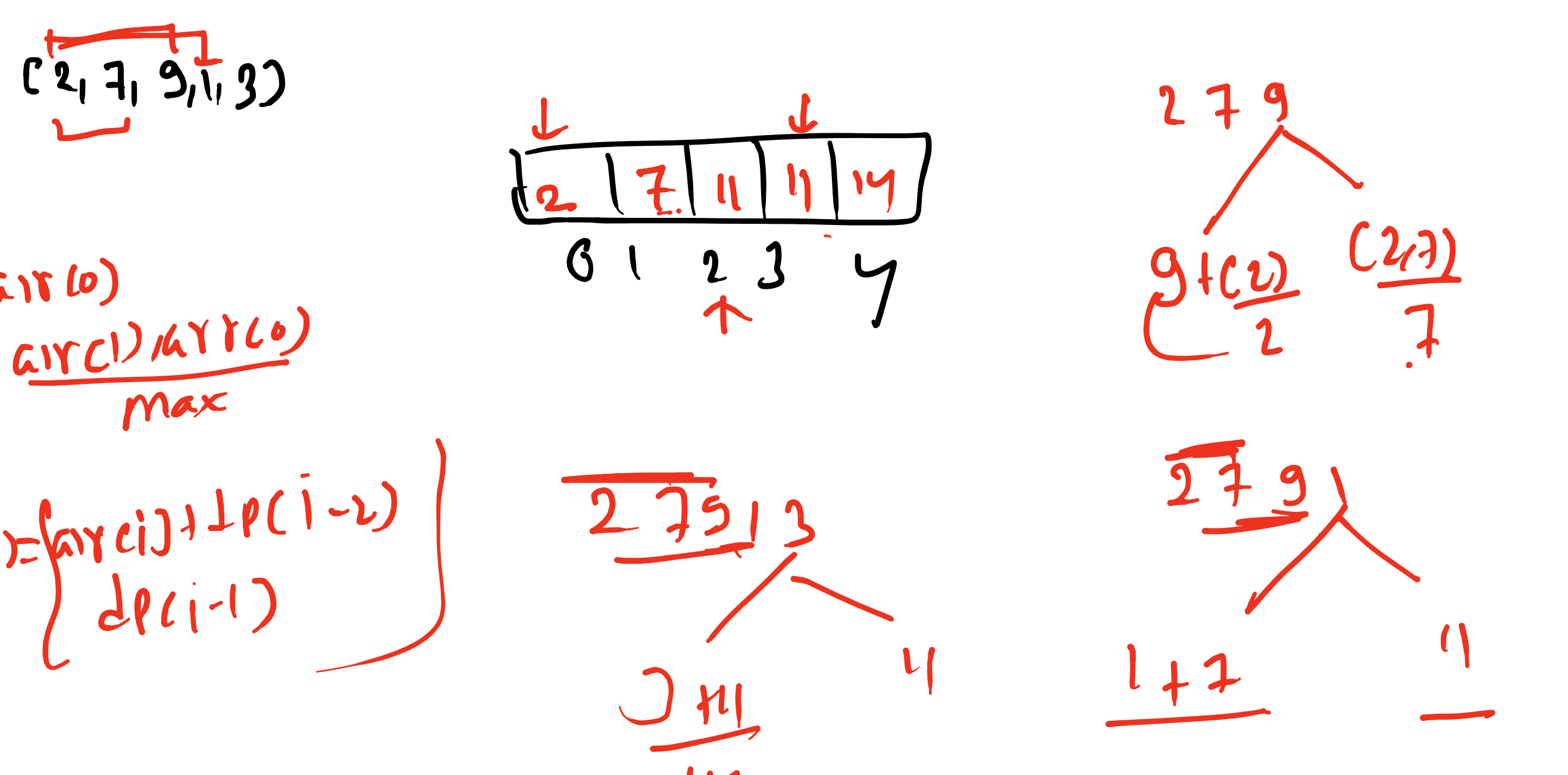
```java
public static int Robber(int[] arr, int i) {
    if (i >= arr.length) {
        return 0;
    }
    int rob = arr[i] + Robber(arr, i + 2);
    int Dont_rob = Robber(arr, i + 1);
    return Math.max(rob, Dont_rob);
}
```



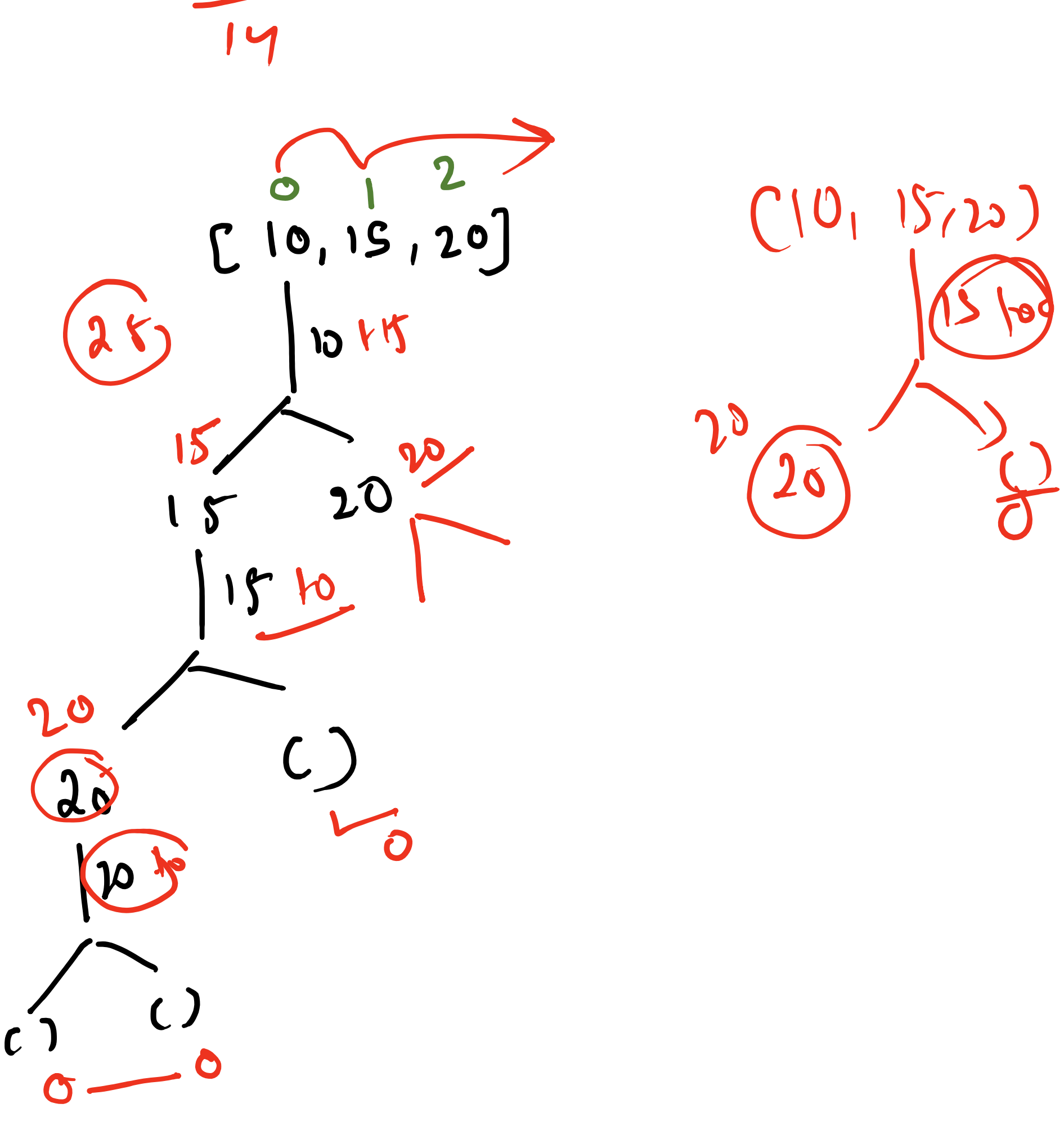
dp[0]=arr[0]

dp[1]=arr[1], arr[0] (max)

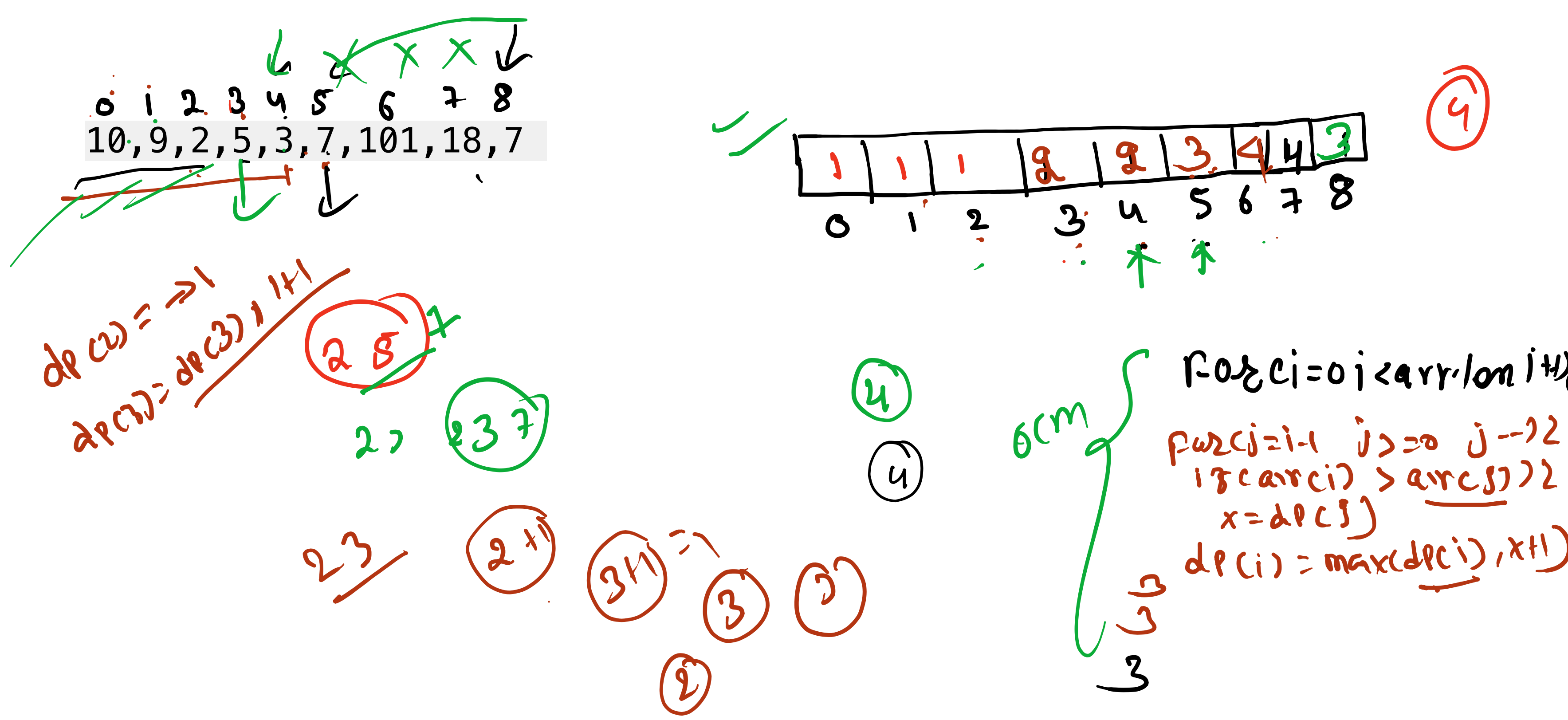
dp[i] = max { arr[i]+dp[i-2] ; dp[i-1] }



[10,15,20]



10,9,2,5,3,7,101,18,7



for(i=0; i<arr.length; i++){

for(j=i-1; j>=0; j--){

if(arr[i] > arr[j]){

x=dp[j]

dp[i] = max(dp[i], x+1)