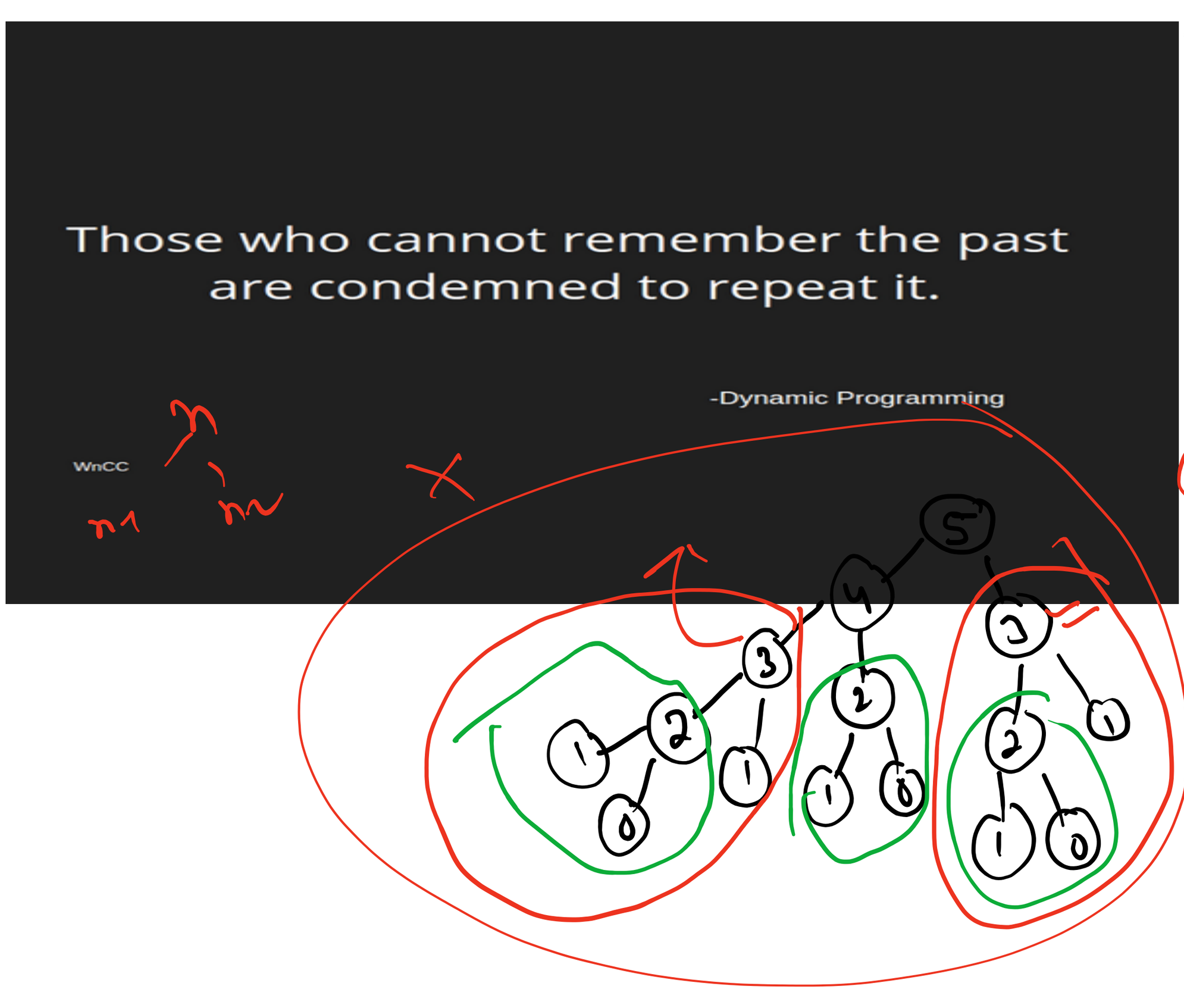
Lec-39

13/07/25 10:02 AM



Dynamic Programming

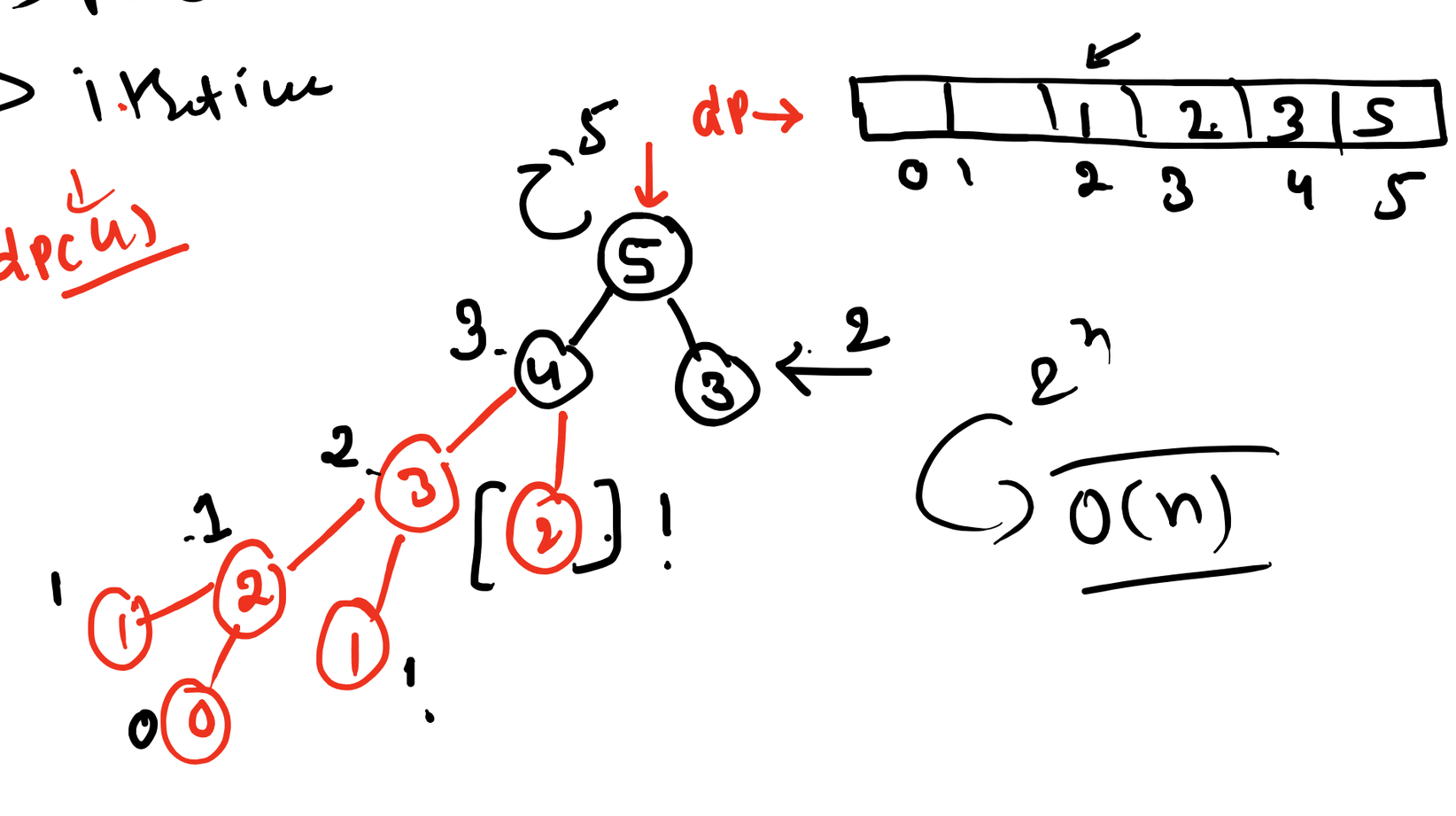
DP

(i) Optimal Sub str

(ii) Problem overlap

(i) Top-down → Recursive

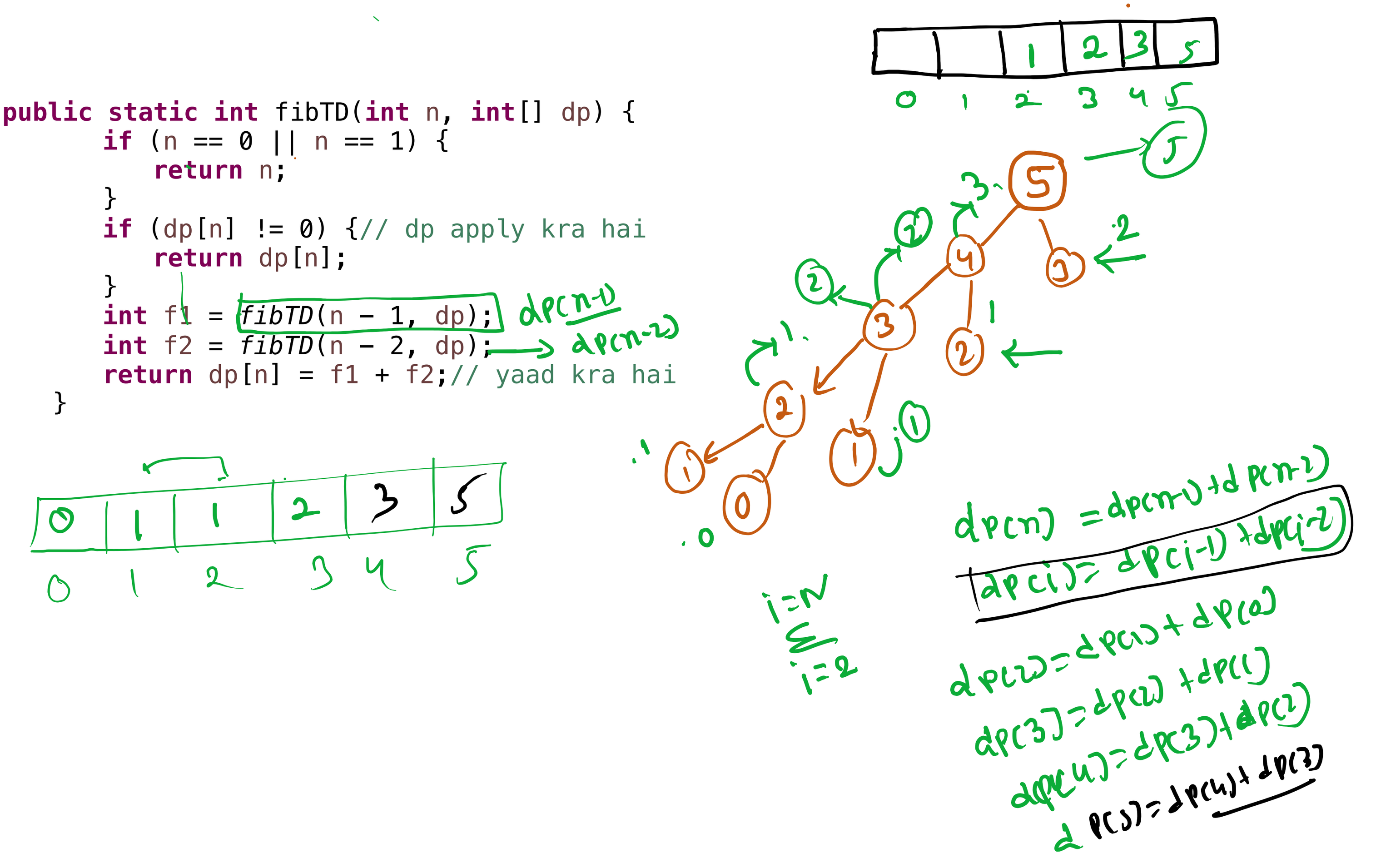
(ii) → Bottom-up → iterative



```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;
    }
```

return → dp(n)=f1+f2

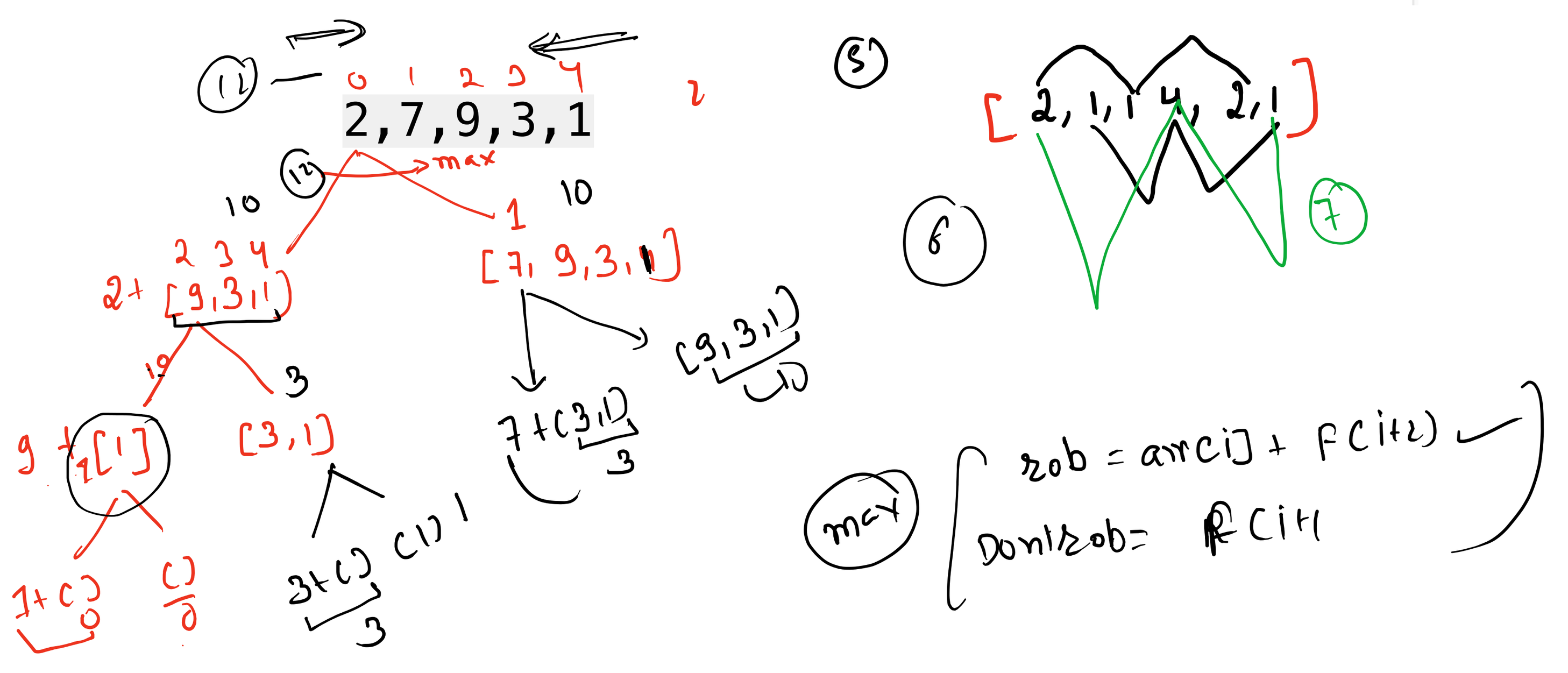
```java
public static int fibTD(int n, int[] dp) {
        if (n == 0 || n == 1) {
            return n;
        }
        if (dp[n] != 0) {// dp apply kra hai
            return dp[n];
        }
        int f1 = fibTD(n - 1, dp);
        int f2 = fibTD(n - 2, dp);
        return dp[n] = f1 + f2;// yaad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



You are a professional robber planning to rob houses along a street. Each house has a certain amount of money stashed, the only constraint stopping you from robbing each of them is that adjacent houses have security systems connected and **it will automatically contact the police if two adjacent houses were broken into on the same night.**

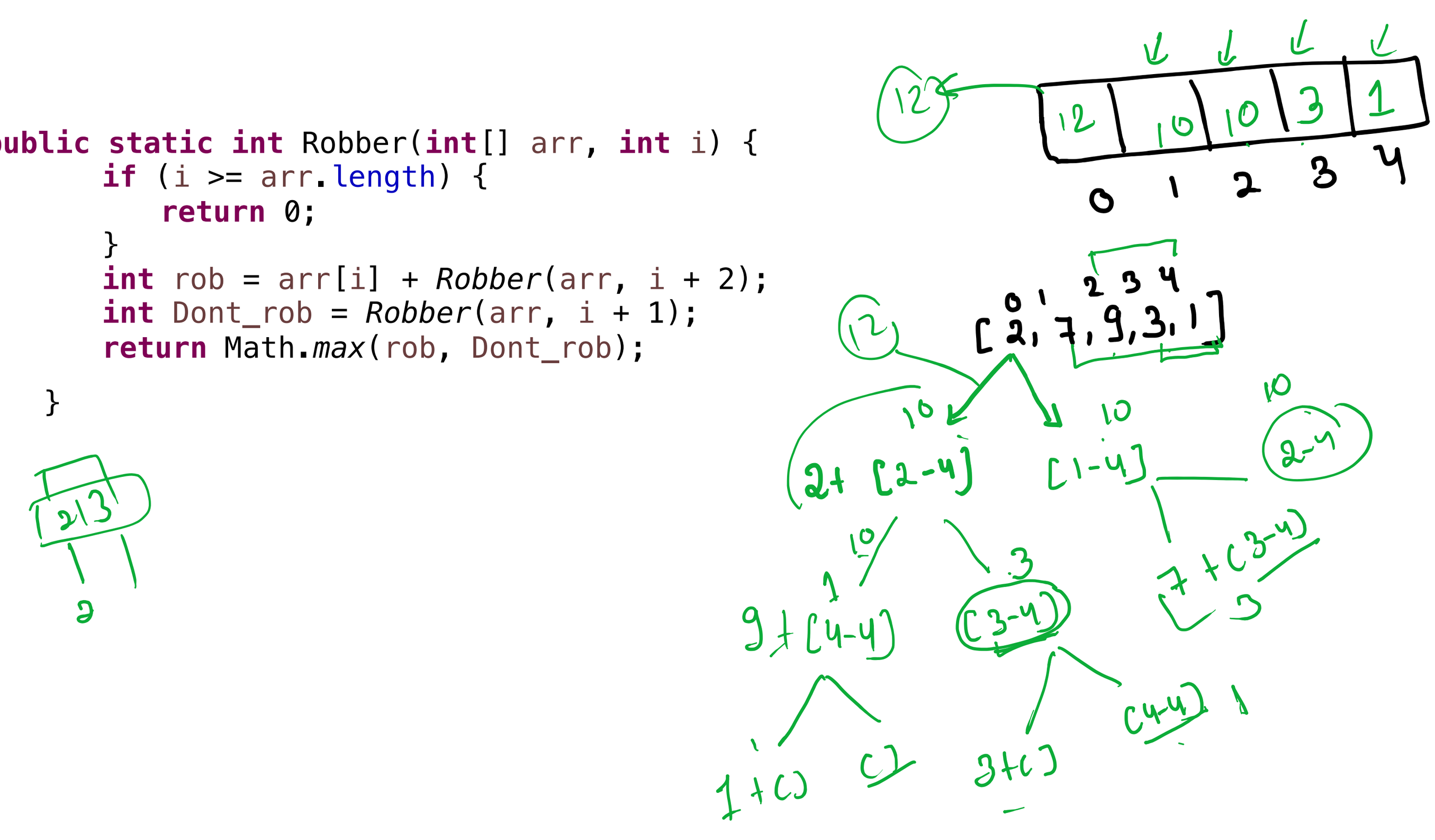
Given an integer array `nums` representing the amount of money of each house, return *the maximum amount of money you can rob tonight* ***without alerting the police***.

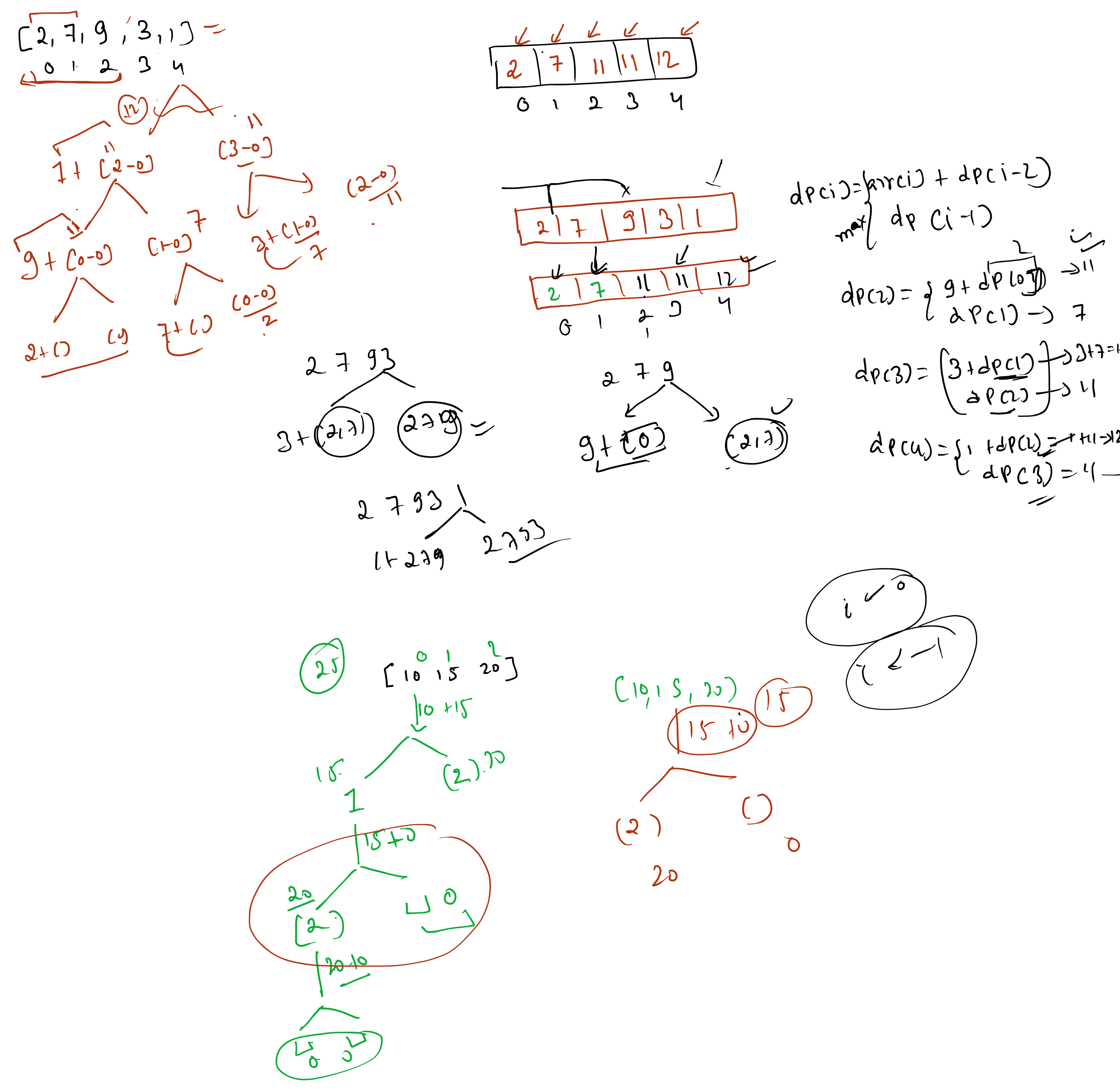
2,7,9,3,1



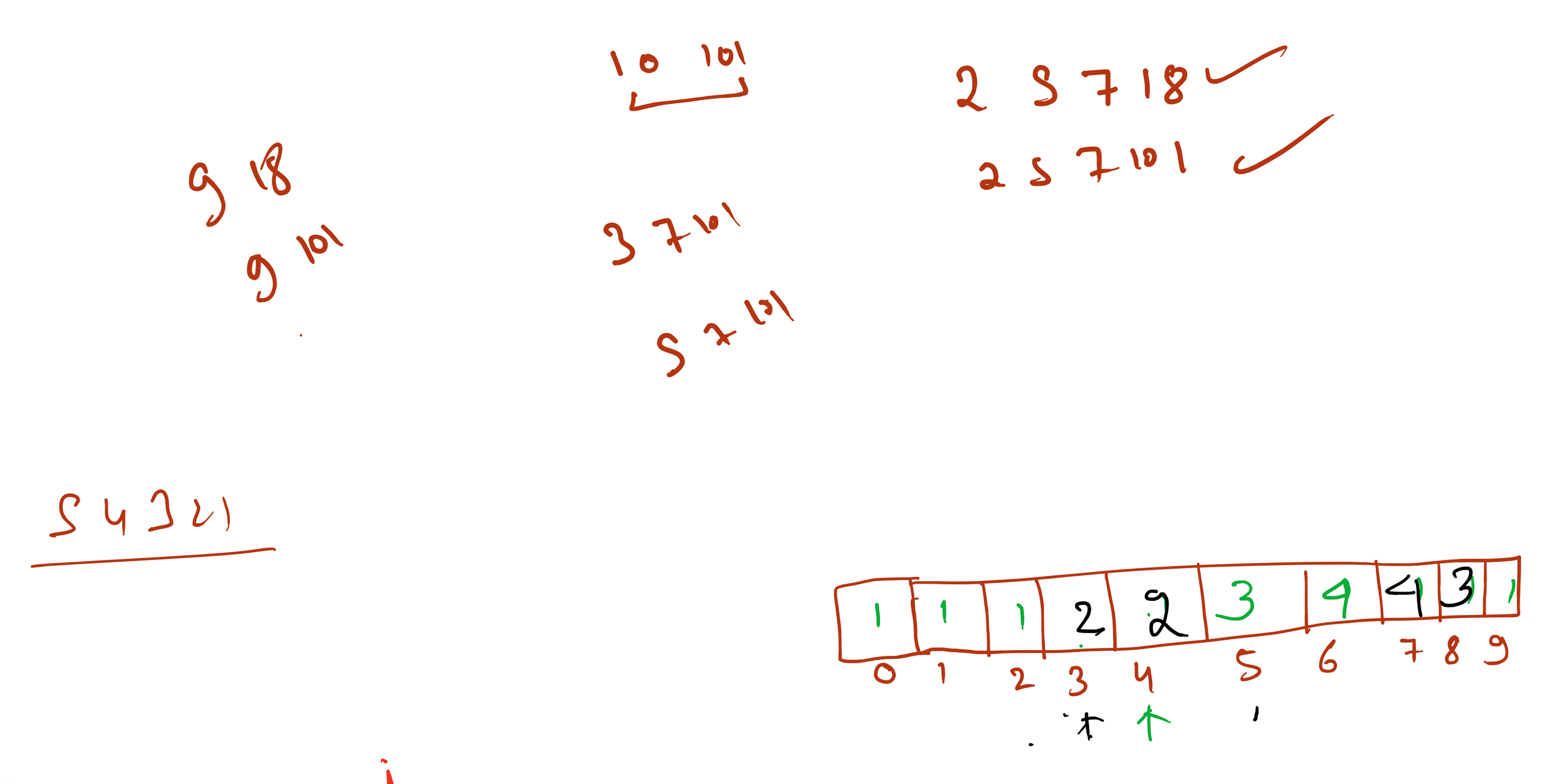
```java
public static int Robber(int[] arr, int i) {
        if (i >= arr.length) {
            return 0;
        }
        int rob = arr[i] + Robber(arr, i + 2);
        int Dont_rob = Robber(arr, i + 1);
        return Math.max(rob, Dont_rob);

    }
```





Input: nums = [10,9,2,5,3,7,101,18]
Output: 4
Explanation: The longest increasing subsequence is [2,3,7,101], therefore the length is 4.



10,9,2,5,3,7,101,18,6,1

```java
if (arr[j] < arr[i]) {
                int x = dp[j] + 1;
                dp[i] = Math.max(x, dp[i]);
            }
```

int[] arr = { 0, 8, 4, 2, 12, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15 };