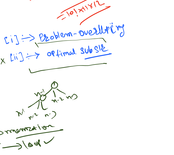
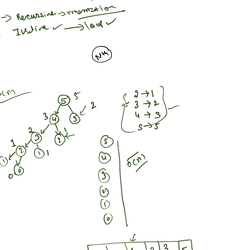
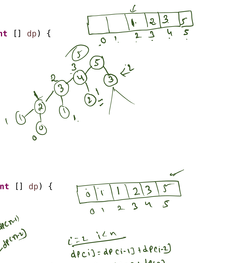
Lec-42



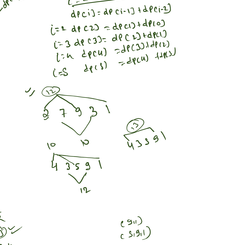

```
public static int fib(int n) {
    if (n == 0 || n == 1) {
        return n;
    }
    int f1 = fib(n - 1);
    int f2 = fib(n - 2);
    return f1 + f2;
}
```

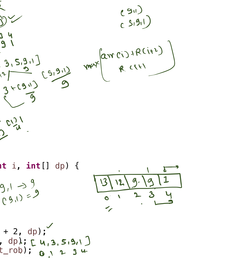


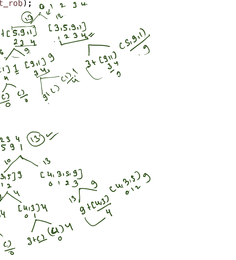
```
public static int fibTD(int n,int [] dp) {
    if (n == 0 || n == 1) {
        return n;
    }
    if(dp[n]!=0) {
        return dp[n];
    }
    int f1 = fibTD(n - 1,dp);
    int f2 = fibTD(n - 2,dp);
    return dp[n]=f1 + f2;
}
```



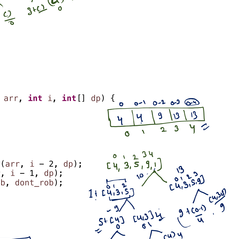
```
public static int fibTD(int n,int [] dp) {
        if (n == 0 || n == 1) {
            return n;
        }
        if(dp[n]!=0) {
            return dp[n];
        }
        int f1 = fibTD(n - 1,dp);
        int f2 = fibTD(n - 2,dp);
        return dp[n]=f1 + f2;
    }
```



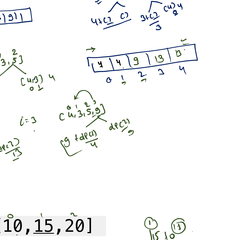




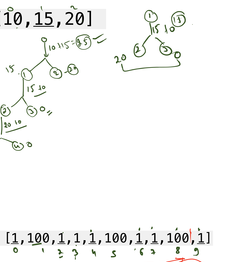
```
public static int Robber(int[] arr, int i, int[] dp) {
    if (i >= arr.length) {
        return 0;
    }
    if (dp[i] != -1) {
        return dp[i];
    }
    int rob = arr[i] + Robber(arr, i + 2, dp);
    int dont_rob = Robber(arr, i + 1, dp);
    return dp[i] = Math.max(rob, dont_rob);
}
```



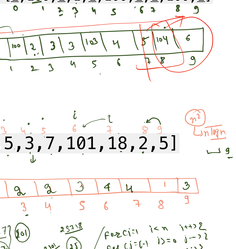
```
public static int Robber2(int[] arr, int i, int[] dp) {
        if (i <0) {
            return 0;
        }
        if (dp[i] != -1) {
            return dp[i];
        }
        int rob = arr[i] + Robber2(arr, i - 2, dp);
        int dont_rob = Robber2(arr, i - 1, dp);
        return dp[i] = Math.max(rob, dont_rob);
    }
```



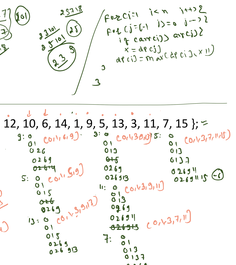
cost = [10,15,20]



Input: cost = [1,100,1,1,1,100,1,1,100,1]



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



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