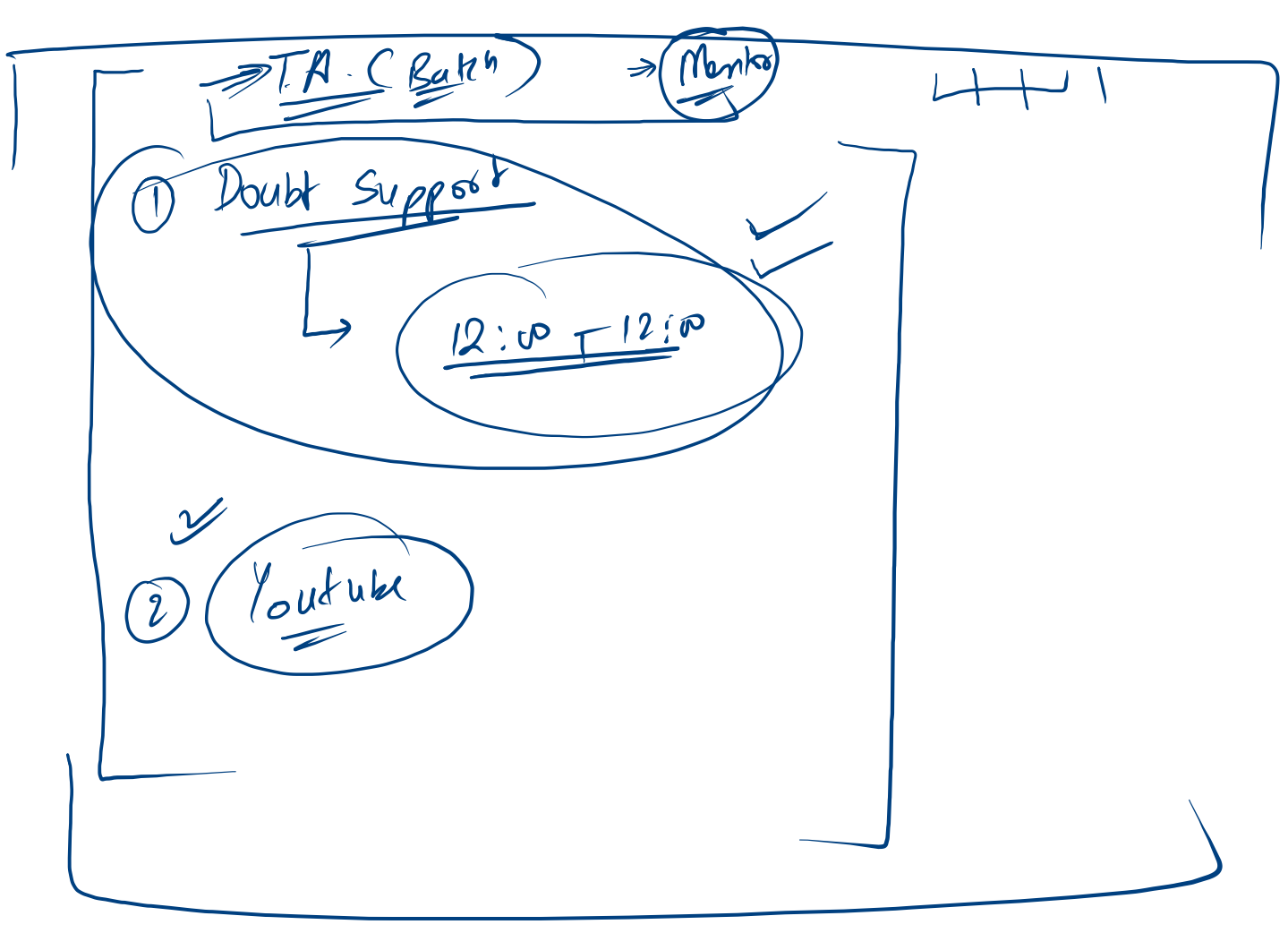
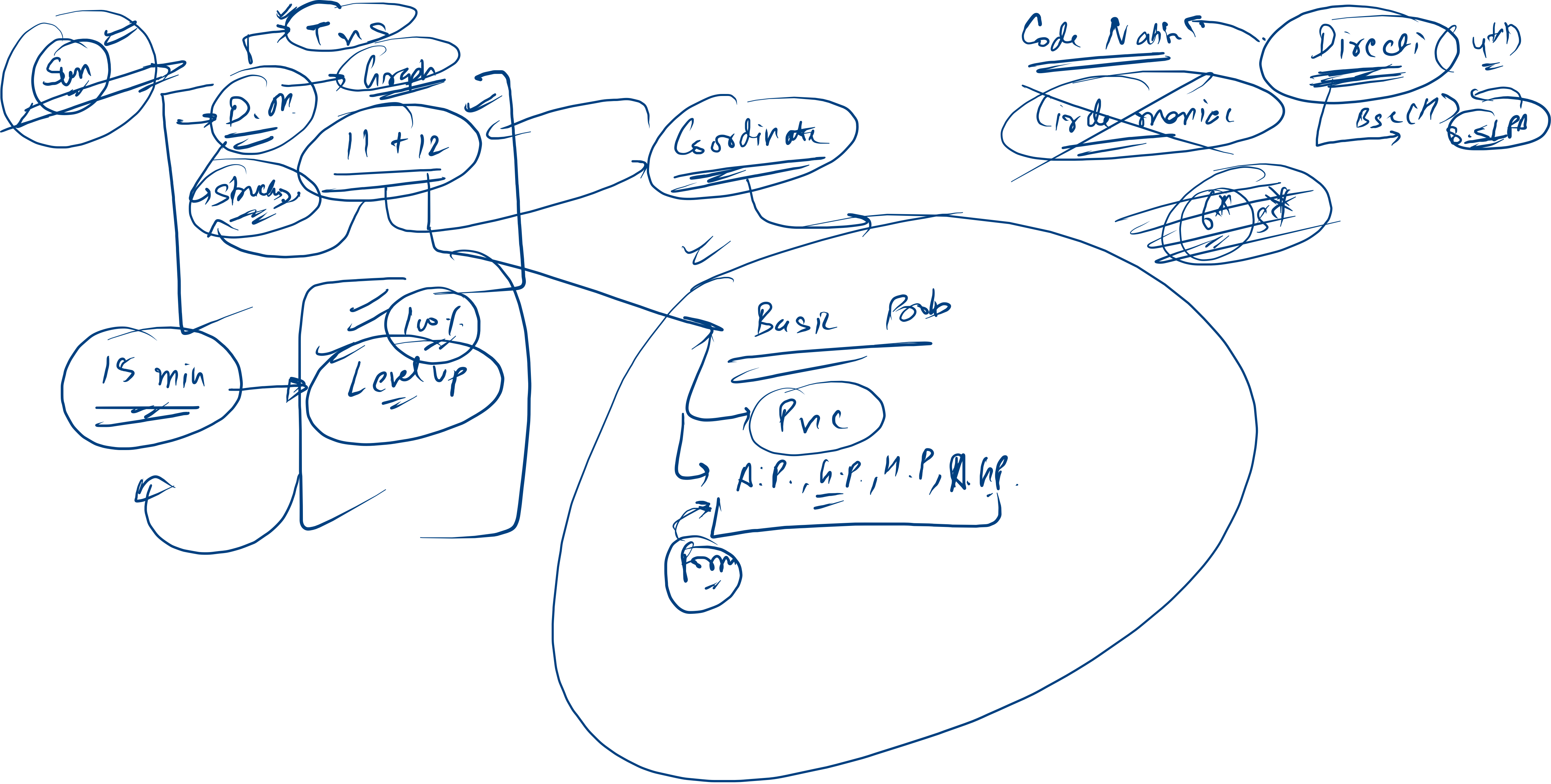
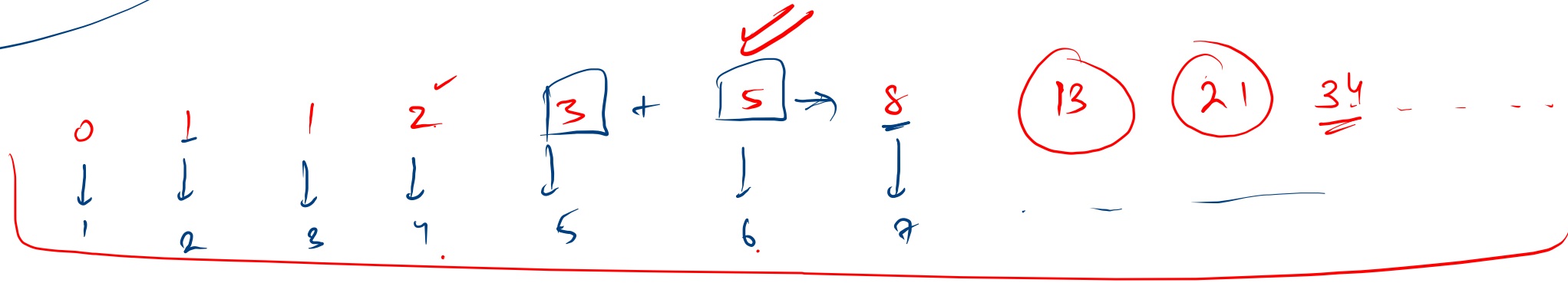
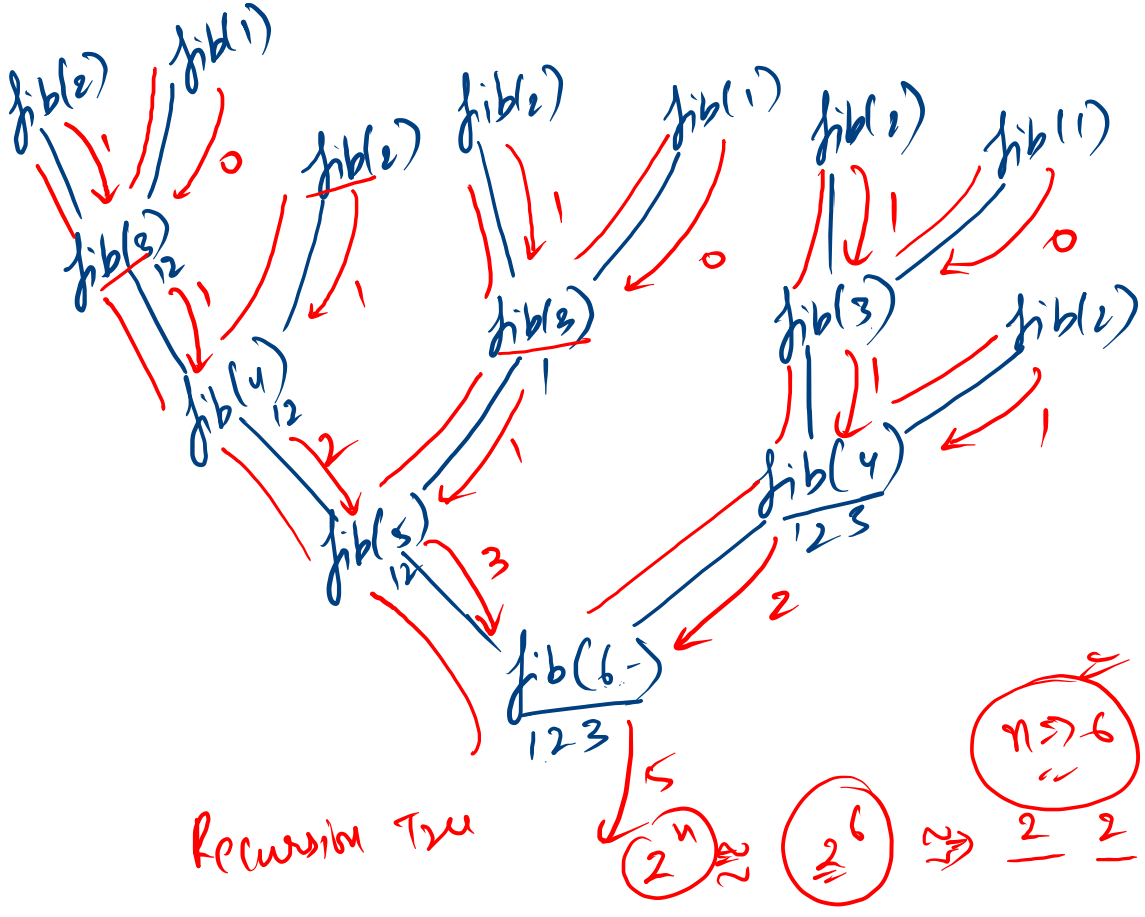

TA - C Batch
Mentor
1 Doubt Support
12:00 - 12:30
2 Youtube


Sum
Graph
D.M.
11 + 12
Coordinate
15 min
Level up
100%
Basic Prob
A.P., G.P., H.P., A.G.P.

Fibonacci Series



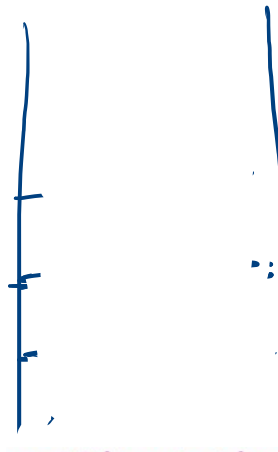
$$fib(n) = fib(n-1) + fib(n-2)$$



Program stack

fib(2) = 1
fib(1) = 0

Given



```java
public static int fib(int n){
    if(n == 1){
        return 0;
    }
    if(n == 2){
        return 1;
    }
    int fibNm1 = fib(n-1);
    int fibNm2 = fib(n-2);
    int fibN = fibNm1 + fibNm2;

    return fibN;
}
```

$2^n \approx 2^6 \Rightarrow 2\ 2\ 2\ 2\ 2\ 2$

n => 6

Input1 -> 1
Output1 -> 1 1 1

Input2 -> 2
Output2 -> 2 1 1 1 2 1 1 1 2

Input2 -> 3
Output3 -> 3 2 1 1 1 2 1 1 1 2 3 2 1 1 1 2 1 1 1 2 3

Code

$1 <= n <= 10$

```java
public static void main(String[] args) throws Exception {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    pzz(n);
}

public static void pzz(int n){

}
```

n → min input = 1

Input1 -> 1

Output1 -> 1 1 1

Input2 -> 2

Output2 -> 2 1 1 1 2 1 1 1 2

```
if (n == 1) {
    print (111);
}
```

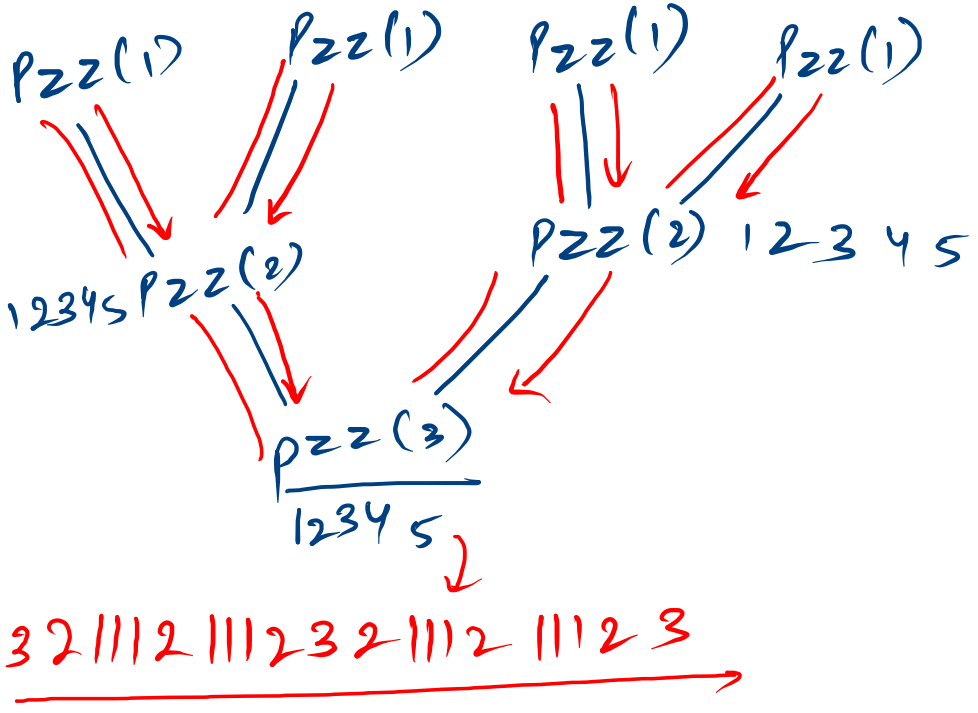
pzz(2) => print(2)

pzz(1)

print(2)

pzz(1)

print(2)



Input2 -> 3
Output3 -> 3 2 1 1 1 2 1 1 1 2 3 2 1 1 1 2 1 1 1 1 2 3

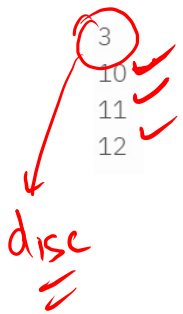
```
public static void pzz(int n){
    if(n == 1){
        System.out.print("111");
        return;
    }

    System.out.print(n); -1
    pzz(n-1); -2
    System.out.print(n); -3
    pzz(n-1); -4
    System.out.print(n); -5
}
```

Faith / Expectation → TOH

Src (10) — dest (11) — Helper (12)

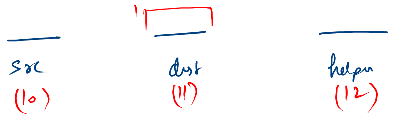
1[10 -> 11]
2[10 -> 12]
1[11 -> 12]
3[10 -> 11]
1[12 -> 10]
2[12 -> 11]
1[10 -> 11]



To Print steps for moving n discs from src → dest such that

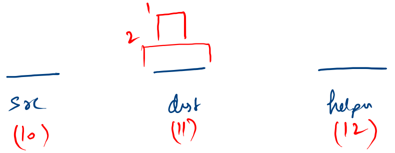
1. only one disc can be moved at a time
2. order must be followed.

n = 1



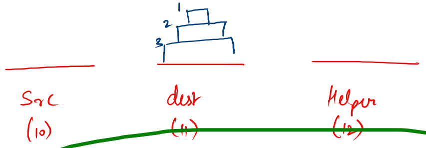
⇒ 1 [10 → 11]

n = 2



⇒
1 [10 → 12]
2 [10 → 11]
1 [12 → 11]

n = 3

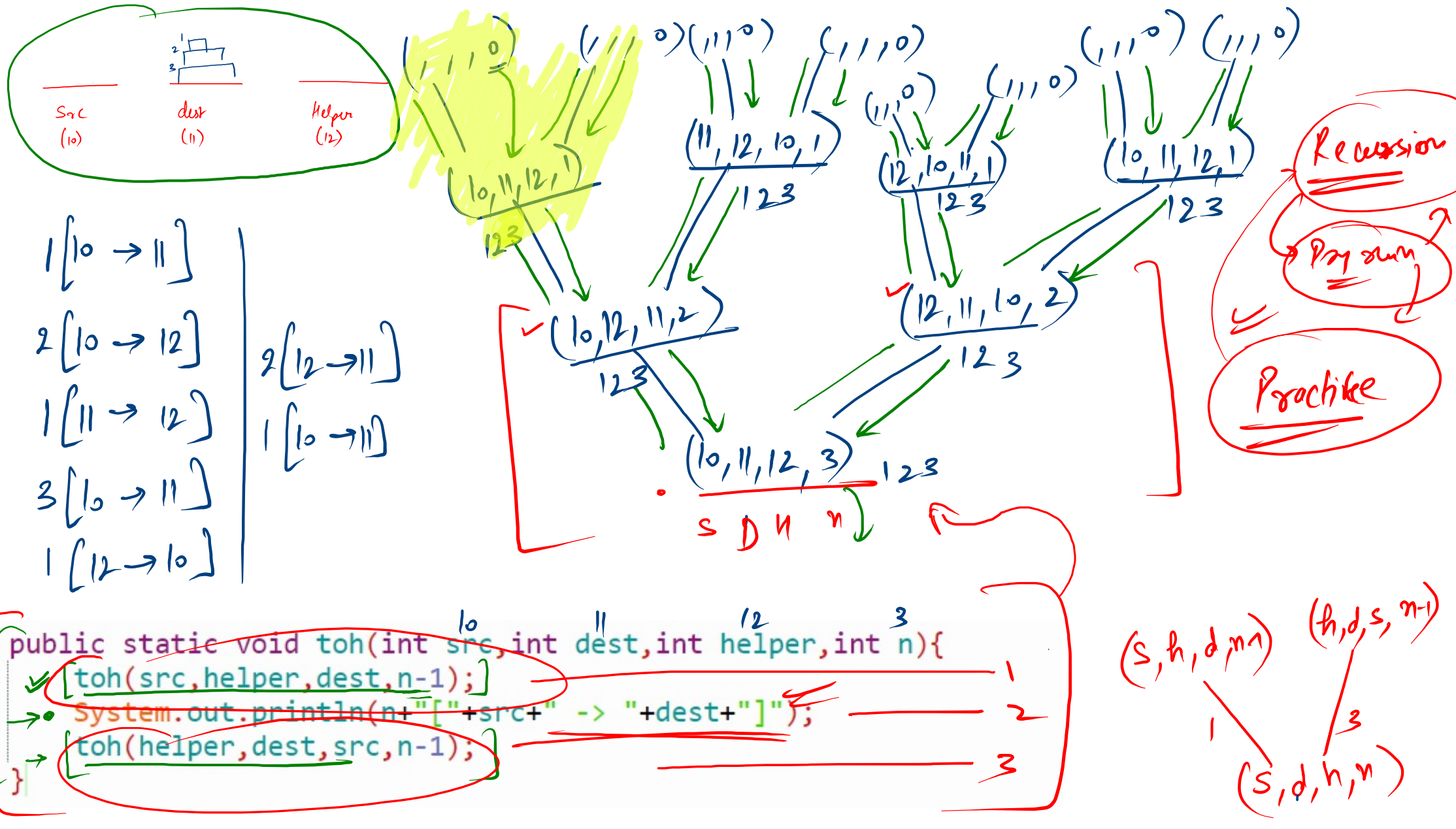


s d H n

Expectation ⇒ TOH(10, 11, 12, 3) ⇒ To print valid steps to move all disc from 10 → 11.

Faith : ⇒ TOH(10, 12, 11, 2) ⇒ program is capable to printing valid steps to move n(2) disc from 10 → 12.

TOH(12, 11, 10, 2) ⇒ " " " " " " " " 12 → 11



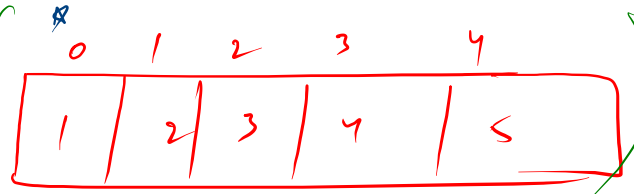

```
public static void toh(int src,int dest,int helper,int n){
    toh(src,helper,dest,n-1);
    System.out.println(n+"["+src+" -> "+dest+"]");
    toh(helper,dest,src,n-1);
}
```

10 11 12 3

1 2 3

(s,h,d,n) → (h,d,s,n-1) → (s,d,h,n)

idx = 0



displayArr(arr, idx) (idx → len-1)

displayArr(arr, idx+1) (idx+1 → len-1)

displayArr(arr, idx) ⇒ Print all values from idx block to last block of the arr.

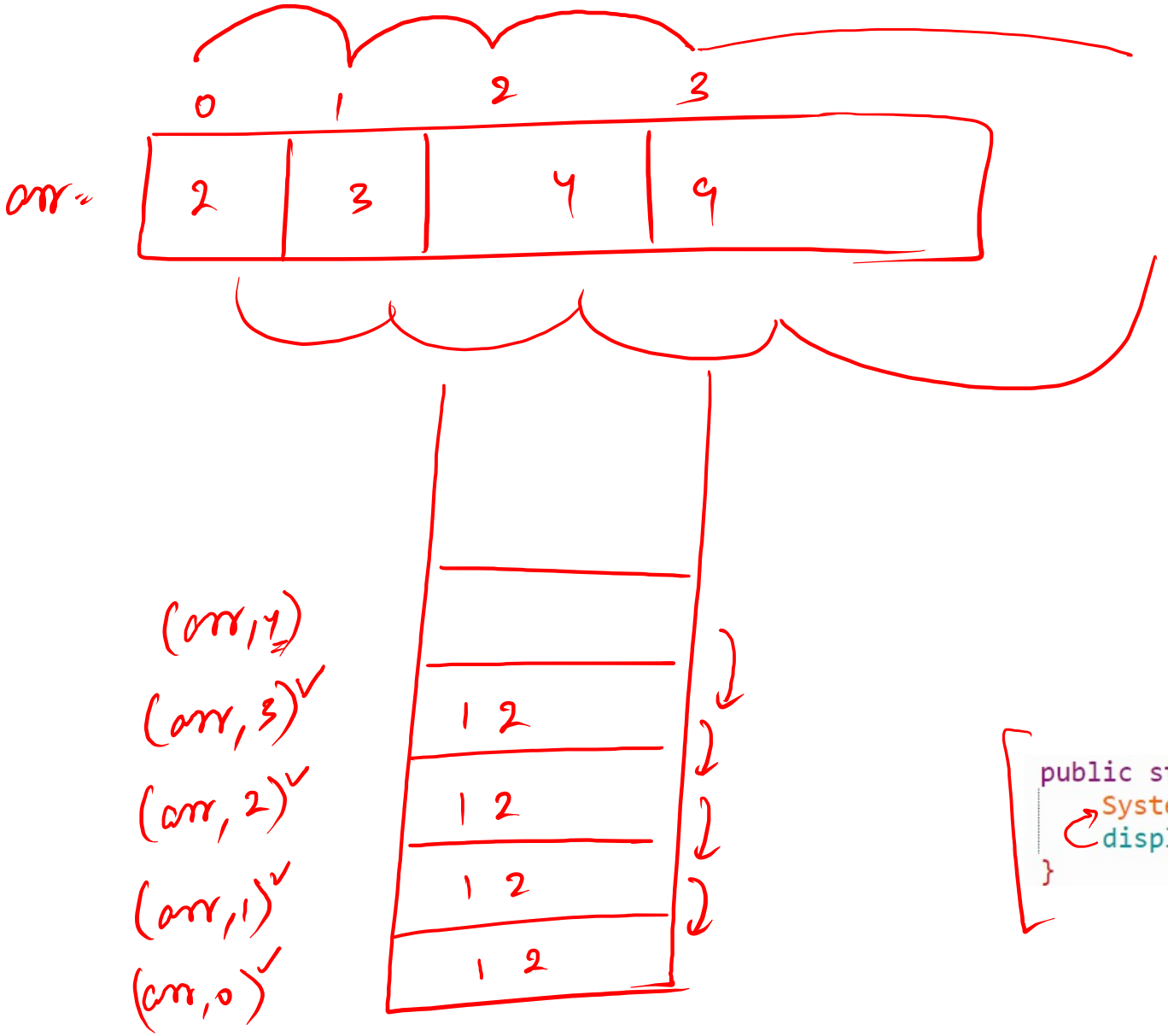
1 ⇒ arr[0]

2 ⇒ arr[1]

3 ⇒ arr[2]

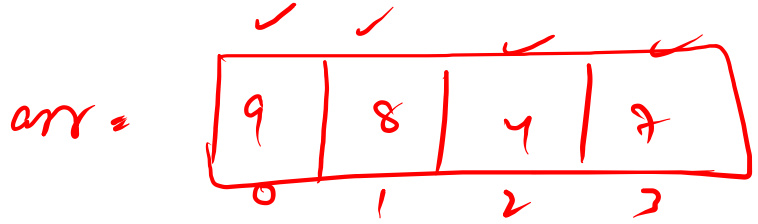
4 ⇒ arr[3]

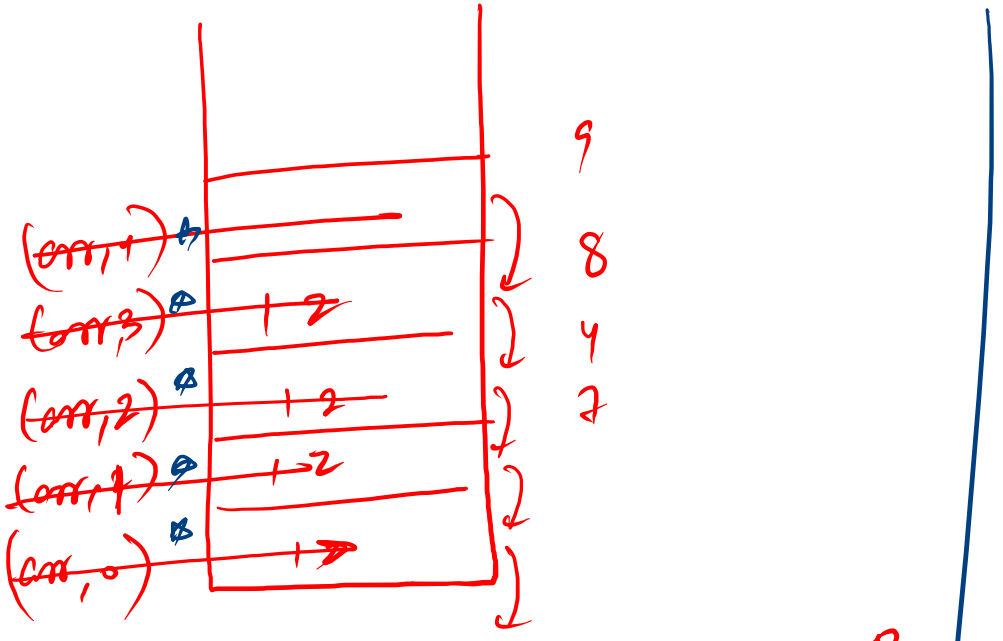
5 ⇒ arr[4]

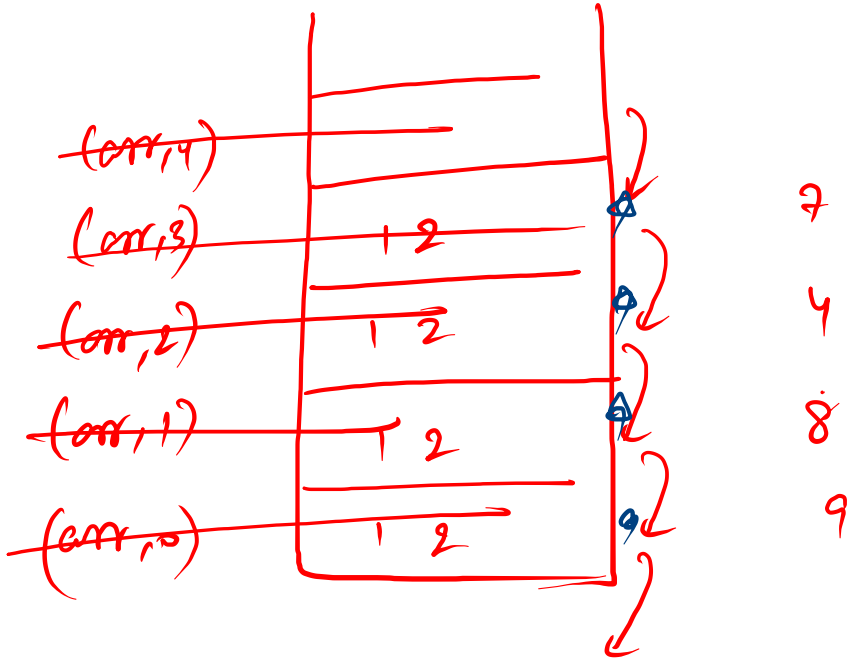


2	9
3	4
4	3
9	2

```java
public static void displayArr(int[] arr, int idx){
    System.out.println(arr[idx]);
    displayArr(arr,idx+1);
}
```

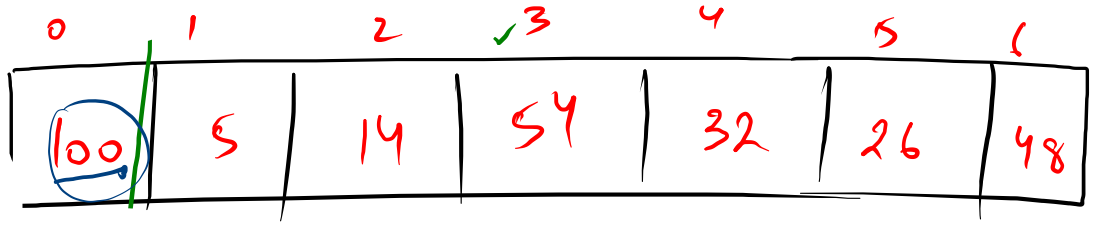







```java
public static void displayArr(int[] arr, int idx){
    if(idx == arr.length){
        return;
    }
    System.out.println(arr[idx]);
    displayArr(arr,idx+1);
}
```

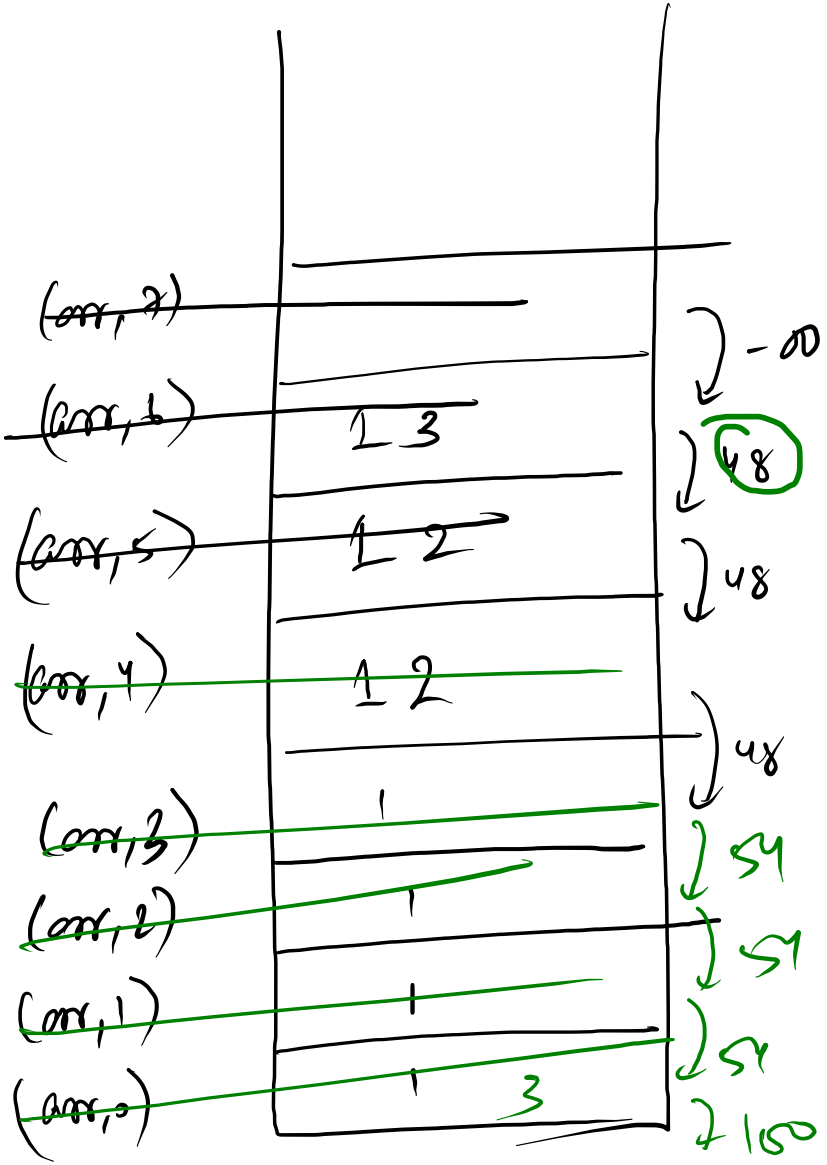
```java
public static void displayArrReverse(int[] arr, int idx){
    if(idx == arr.length){
        return;
    }
    displayArrReverse(arr,idx+1);
    System.out.println(arr[idx]);
}
```

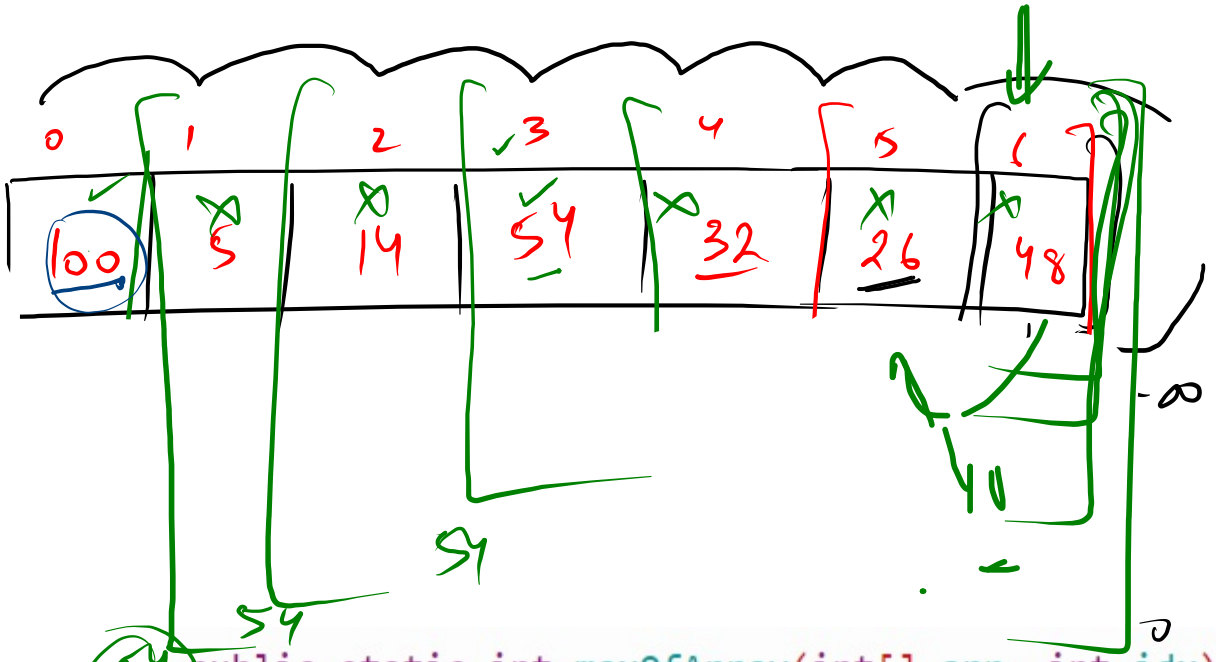


max(arr,0) :- func. is expected to return maximum value from arr.

Expectation

= max(arr,1) → better






```java
public static int maxOfArray(int[] arr, int idx){

    int res = maxOfArray(arr,idx+1);  ①
    if(res > arr[idx]){
        return res;                   ②
    }else{
        return arr[idx];              ③
    }
}
```