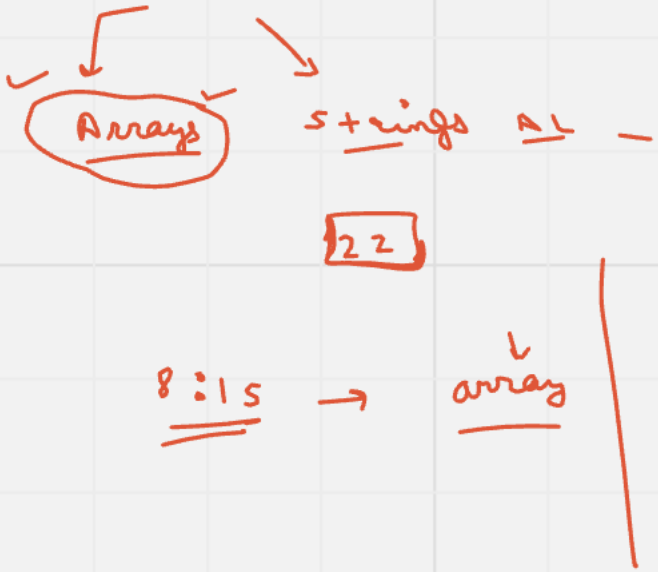
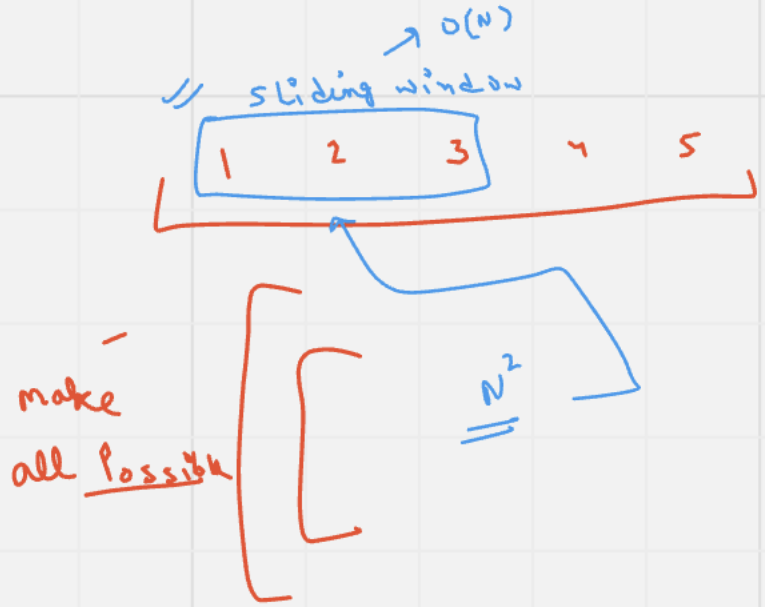
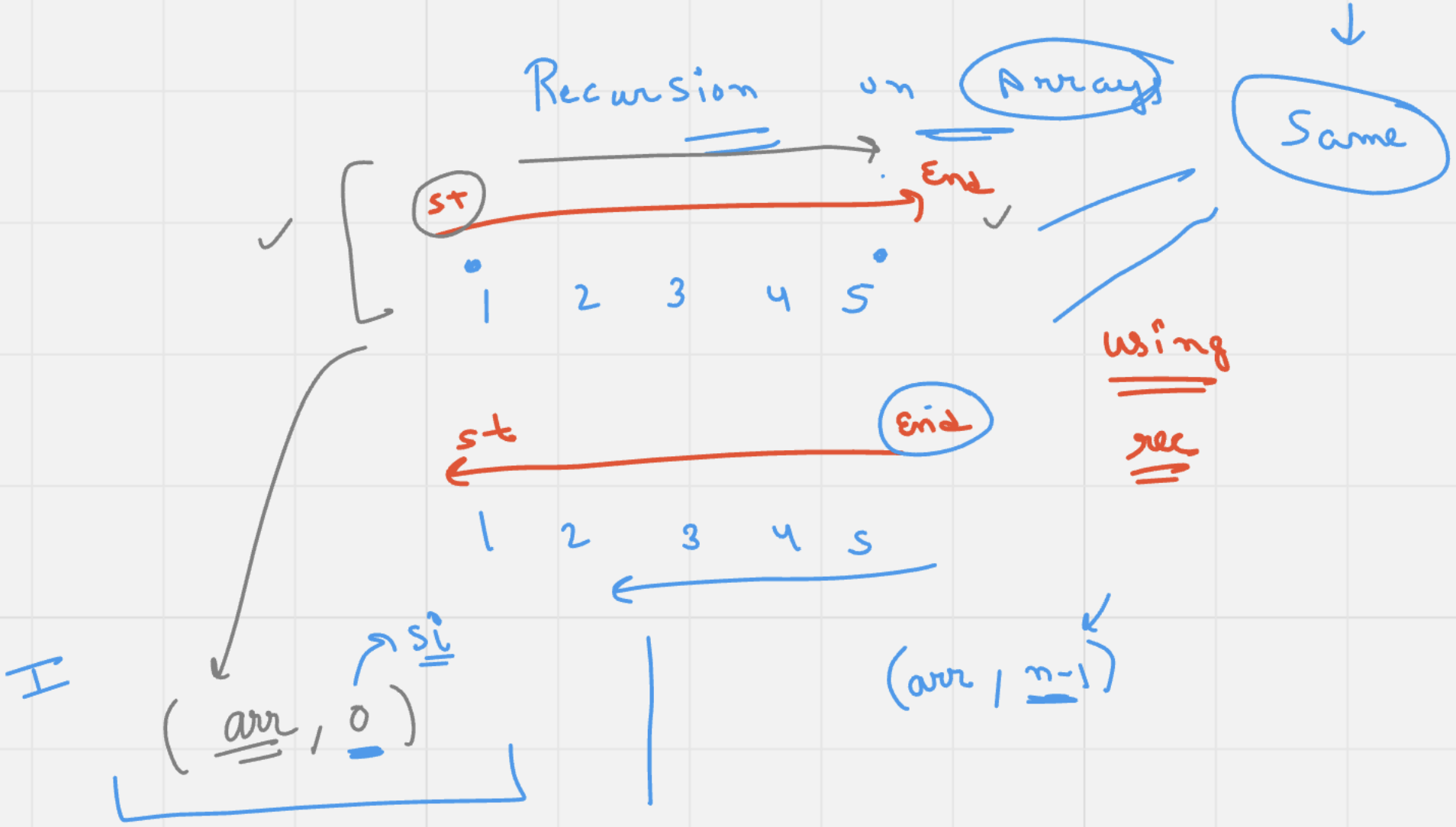
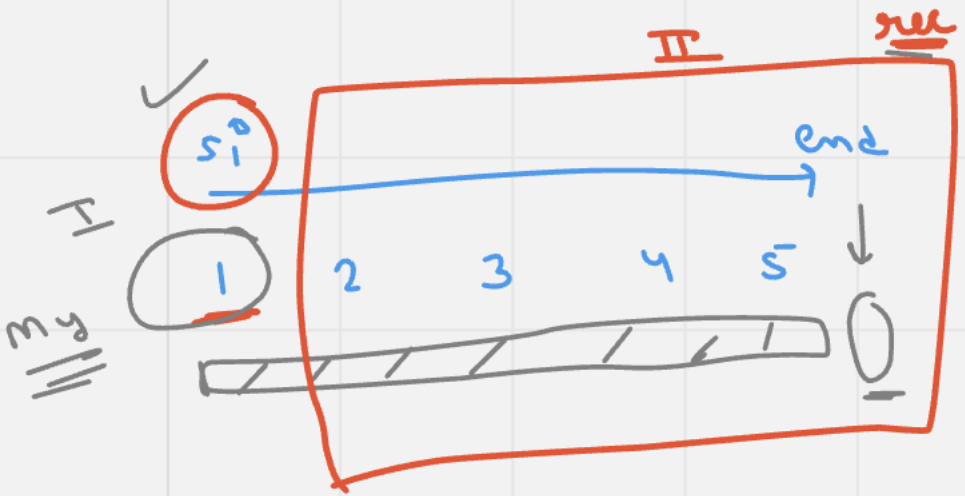

Arrays
strings AL
22
8:15 → array

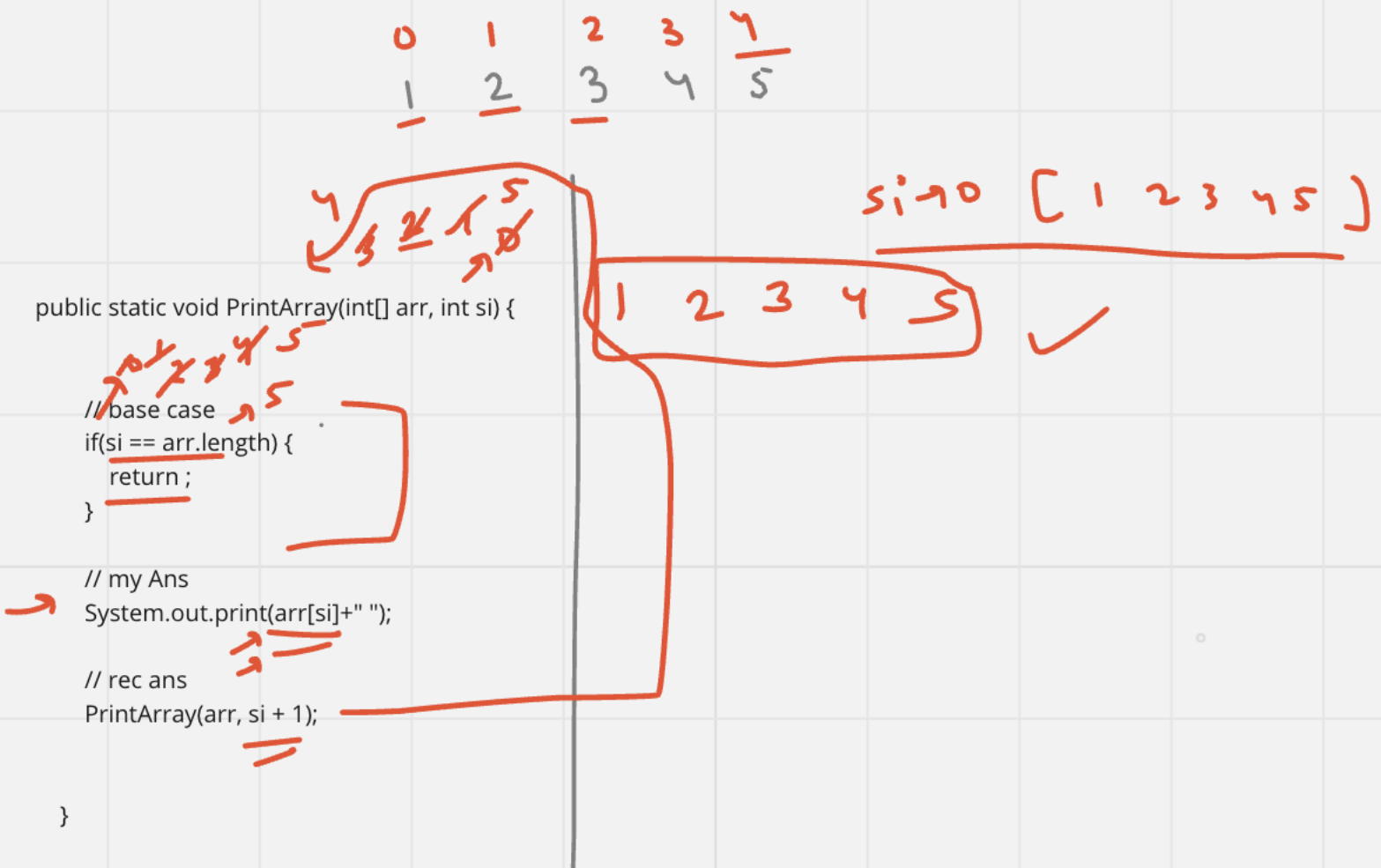

O(N)
// sliding window
1 2 3 4 5
make all possible
N²


Recursion on Array
Same
st
End
1 2 3 4 5
using rec
End
st
1 2 3 4 5
I
(arr, 0)
si
(arr, n-1)

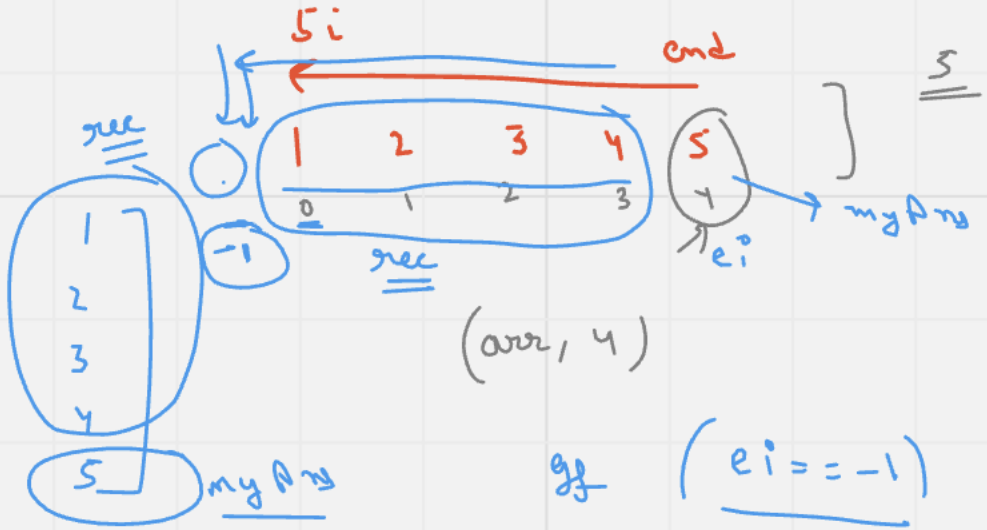


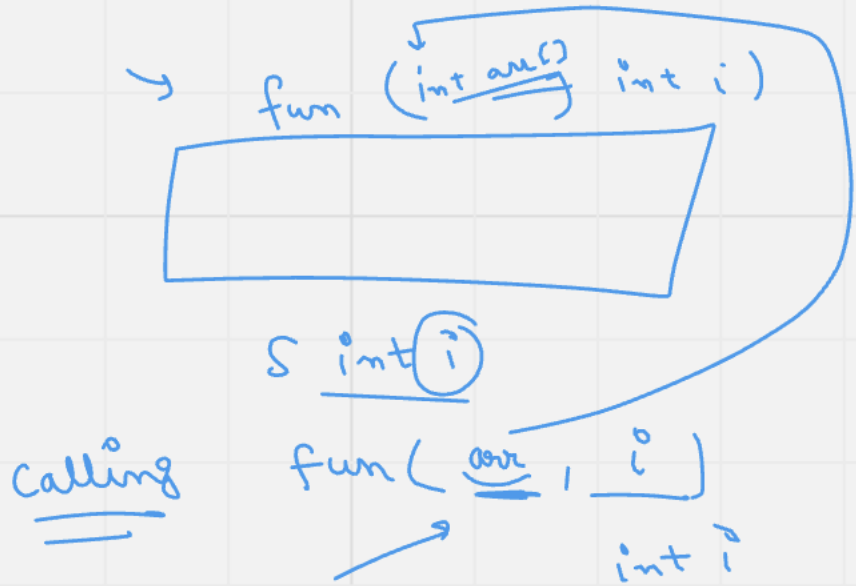
(arr, 0)

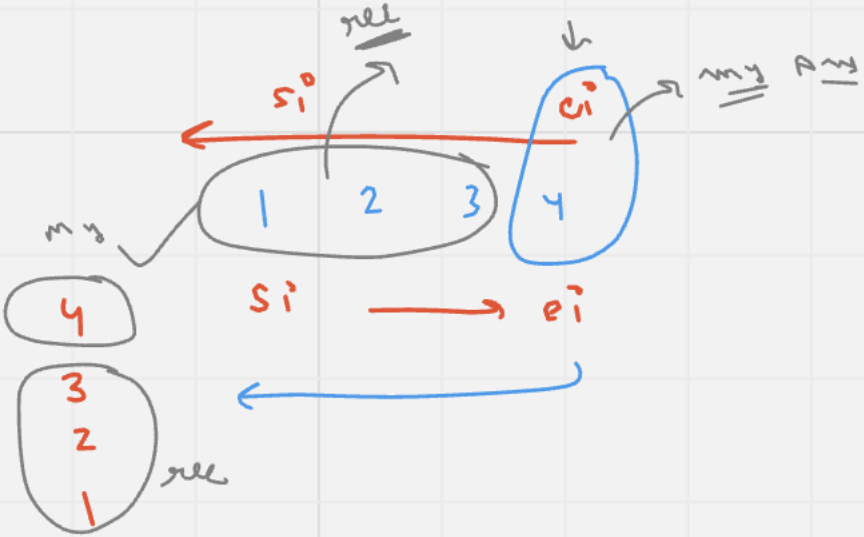
0	1	2	3	4
1	2	3	4	5

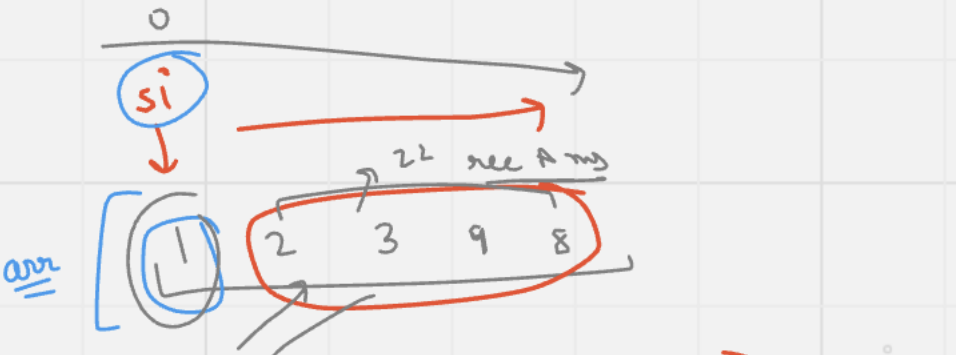


```
public static void PrintArray(int[] arr, int si) {

    // base case
    if(si == arr.length) {
        return ;
    }

    // my Ans
    System.out.print(arr[si]+" ");

    // rec ans
    PrintArray(arr, si + 1);


}
```


5 i
end
5
rec
1 2 3 4 5
0 1 2 3 4
ei
myArr
-1
rec
1
2
3
4
5
myArr
(arr, 4)
if (ei == -1)

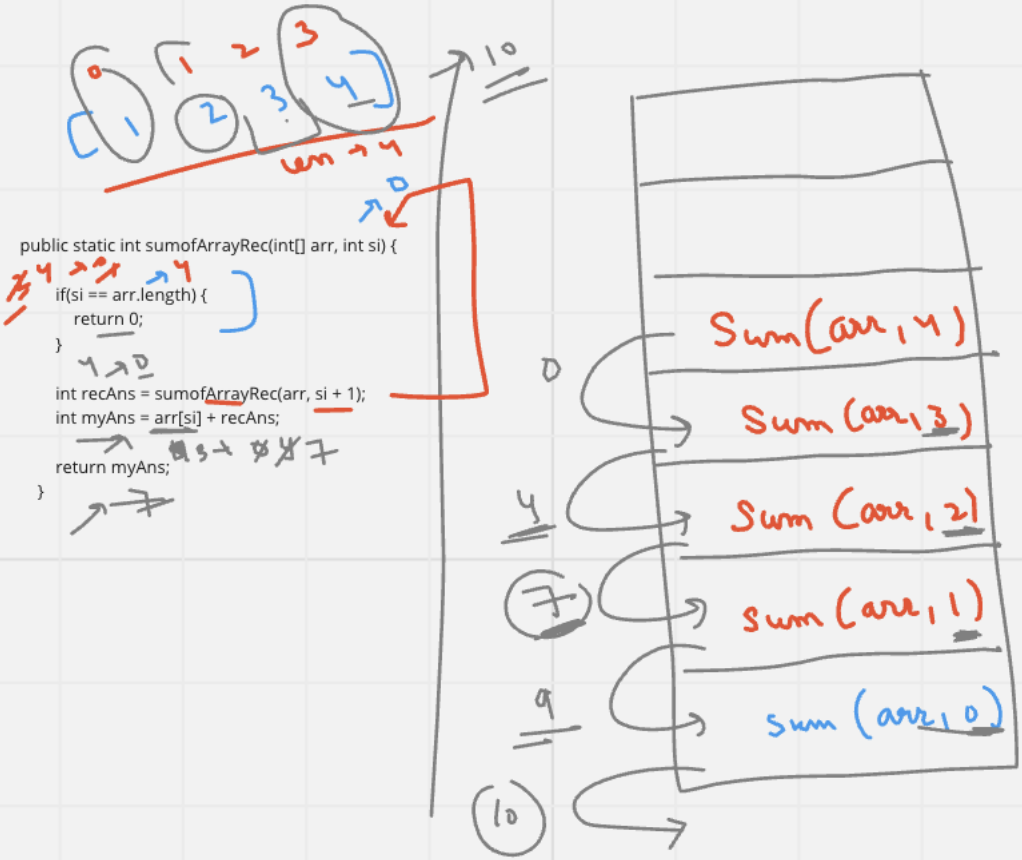

fun (int arr[] int i)
5 int i
Calling
fun(arr , i)
int i

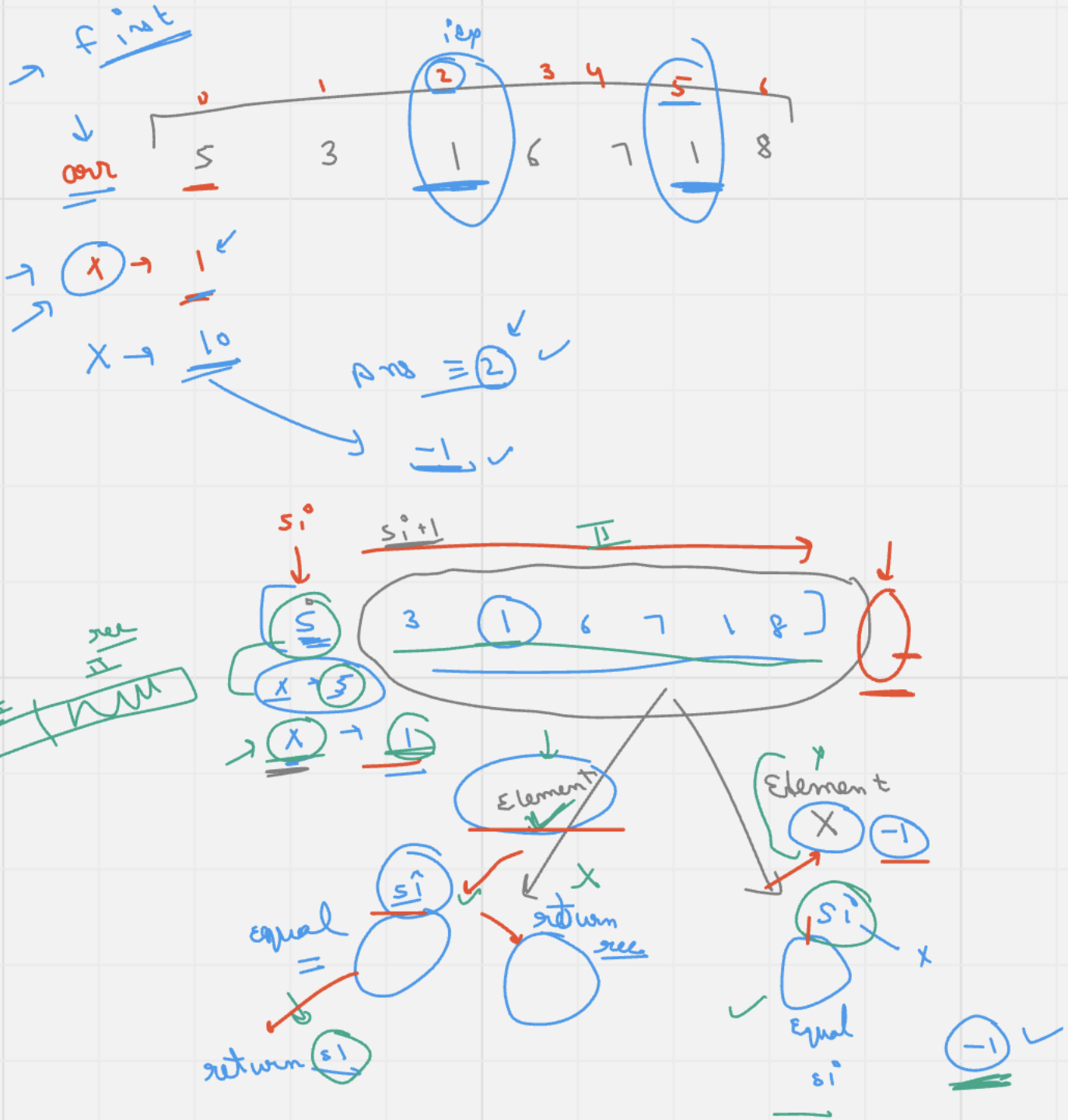


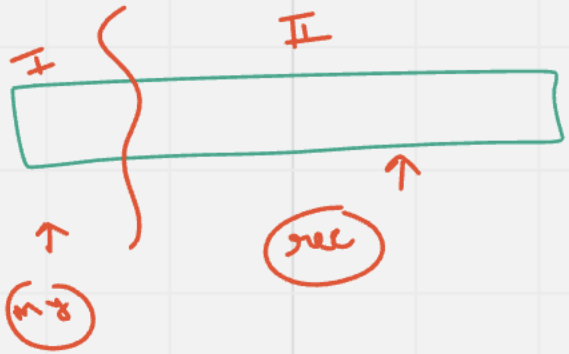


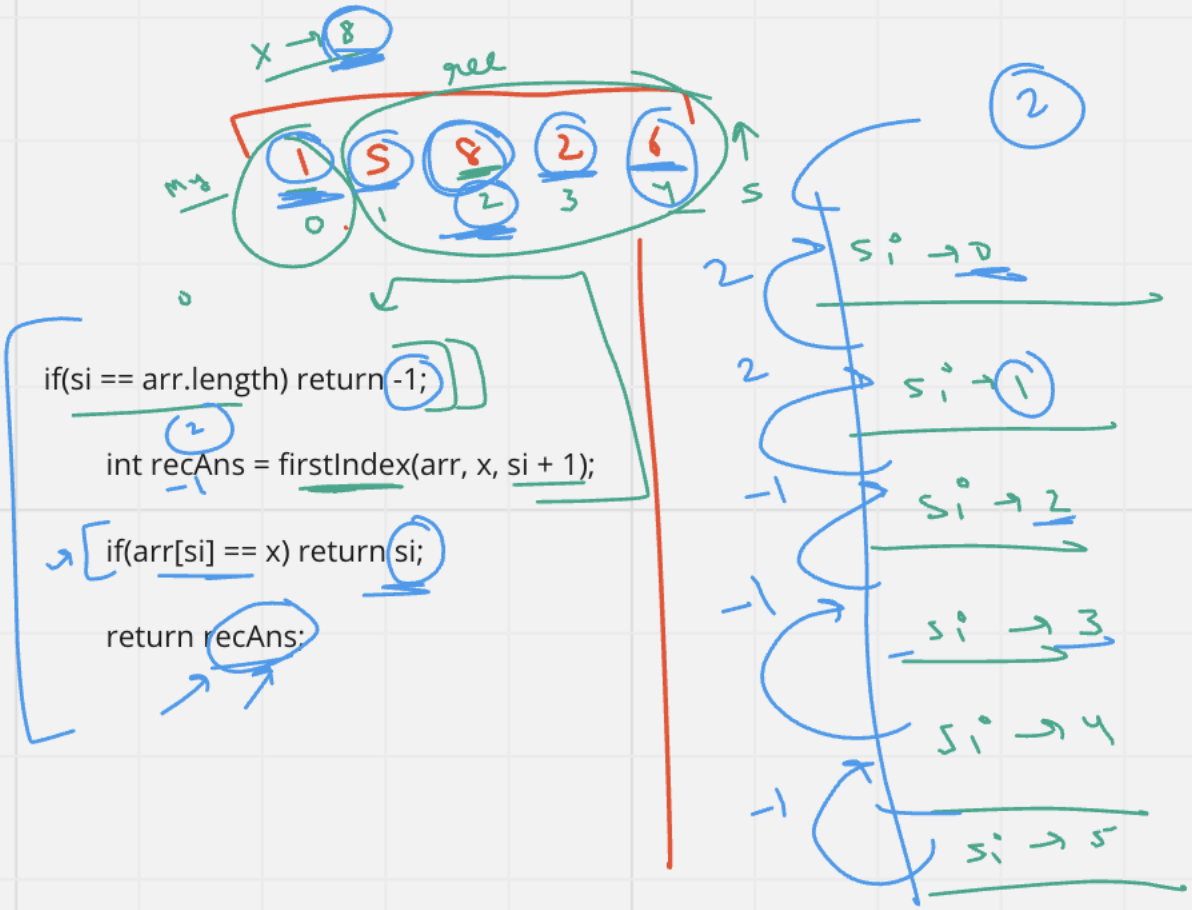
int rec → 22

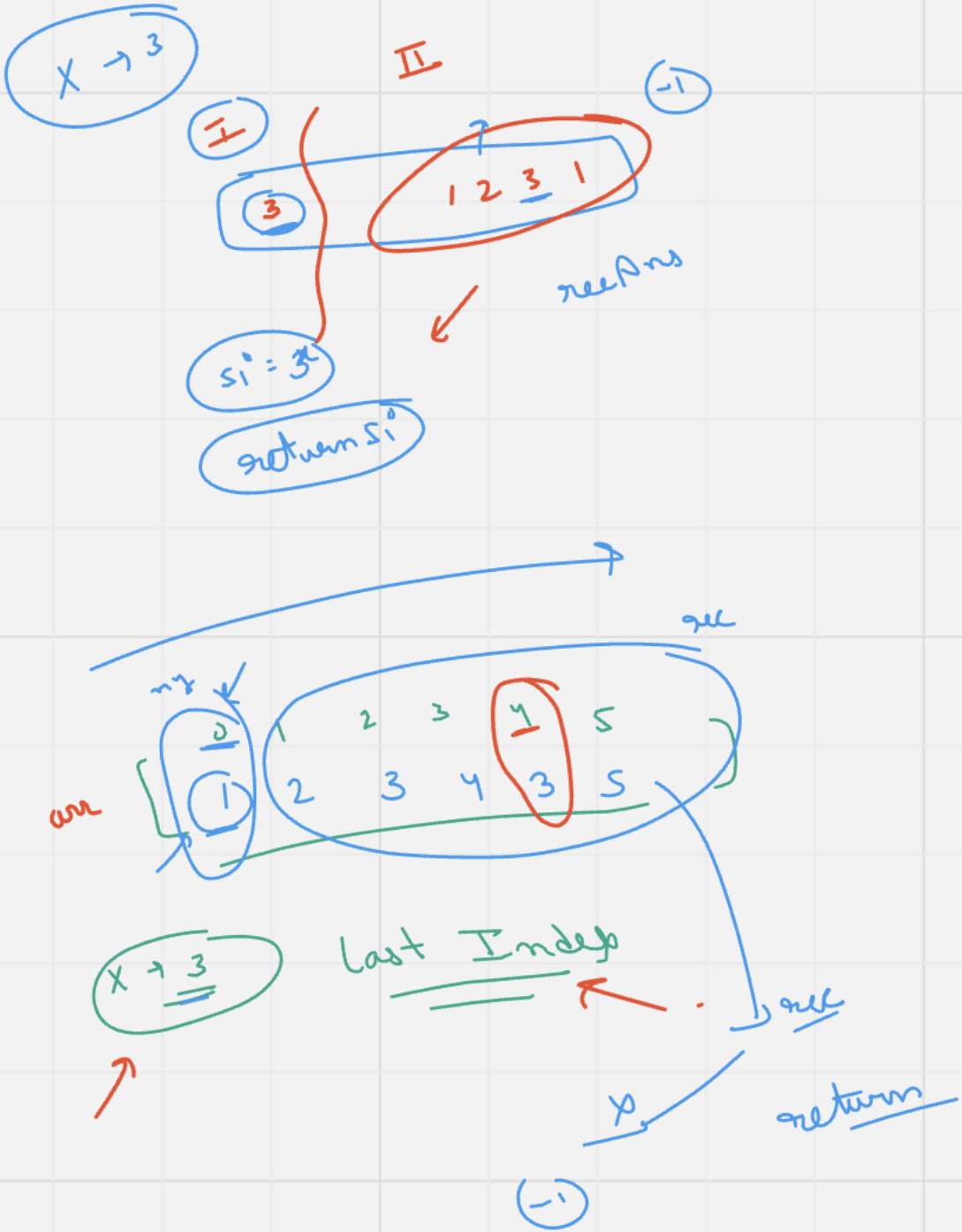
myAns = arr[si] + recAns

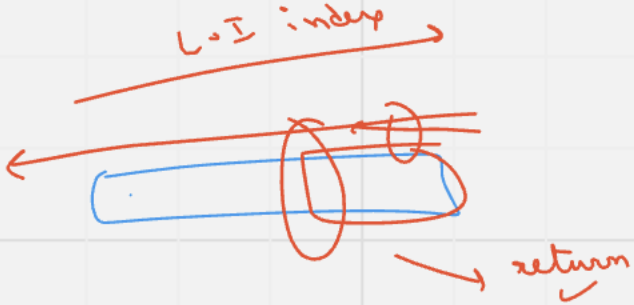

public static int sumofArrayRec(int[] arr, int si) {
if(si == arr.length) {
return 0;
}
int recAns = sumofArrayRec(arr, si + 1);
int myAns = arr[si] + recAns;
return myAns;
}
Sum(arr, 4)
Sum (arr, 3)
Sum (arr, 2)
sum (arr, 1)
sum (arr, 0)

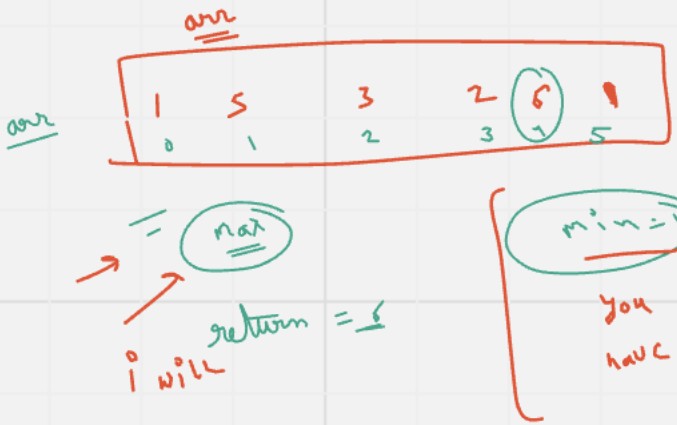

int
cour
0 1 2 3 4 5 6
5 3 1 6 7 1 8
X → 1
X → 10
-1
Si
Si+1
Element
return
equal
-1


I
II
my
rec

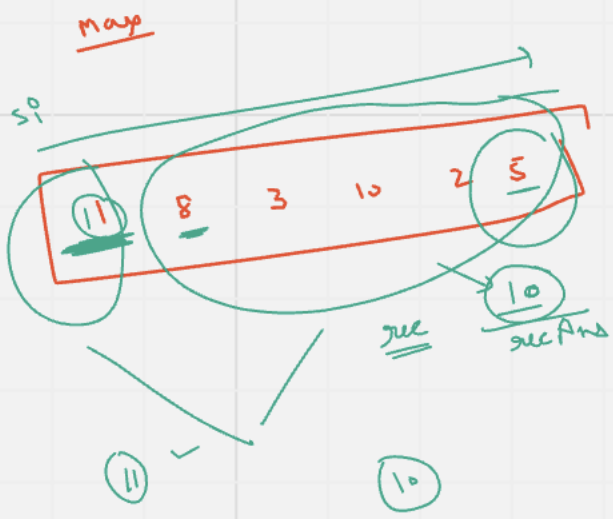

x → 8
rec
my
1
5
8
2
6
0
1
2
3
4
s
0
if(si == arr.length) return -1;
int recAns = firstIndex(arr, x, si + 1);
if(arr[si] == x) return si;
return recAns;
2
si → 0
si → 1
si → 2
si → 3
si → 4
si → 5
-1

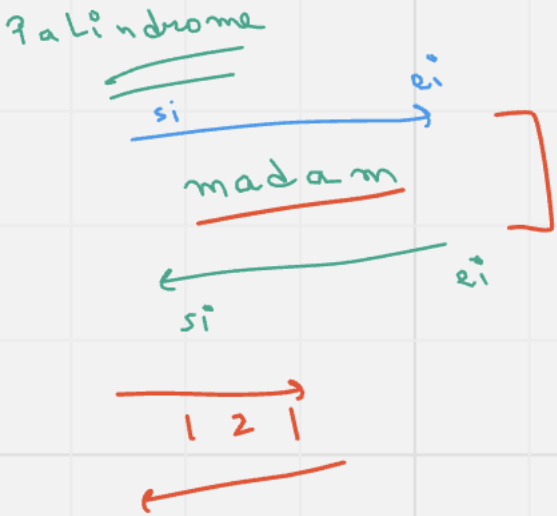

X → 3
II
I
-1
3
1 2 3 1
recAns
s1 = 3
return s1
rec
mx
2
1
1 2 3 4 5
2 3 4 3 5
arr
X → 3
Last Index
rec
X
return
-1

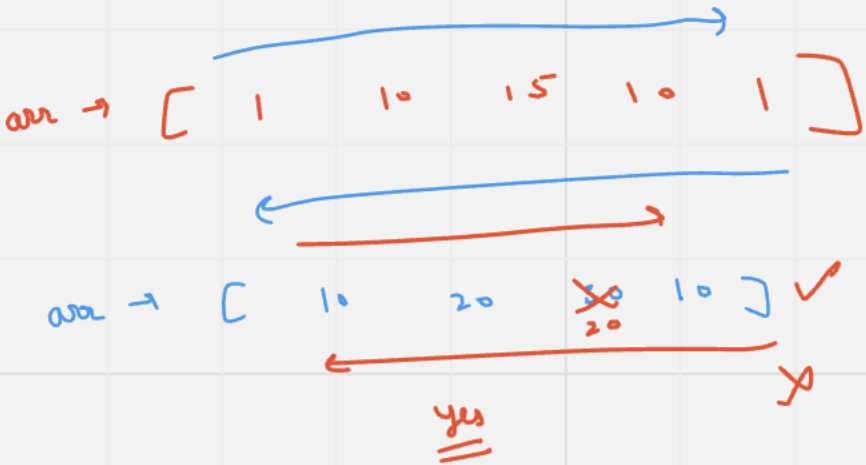

L.I : index
return

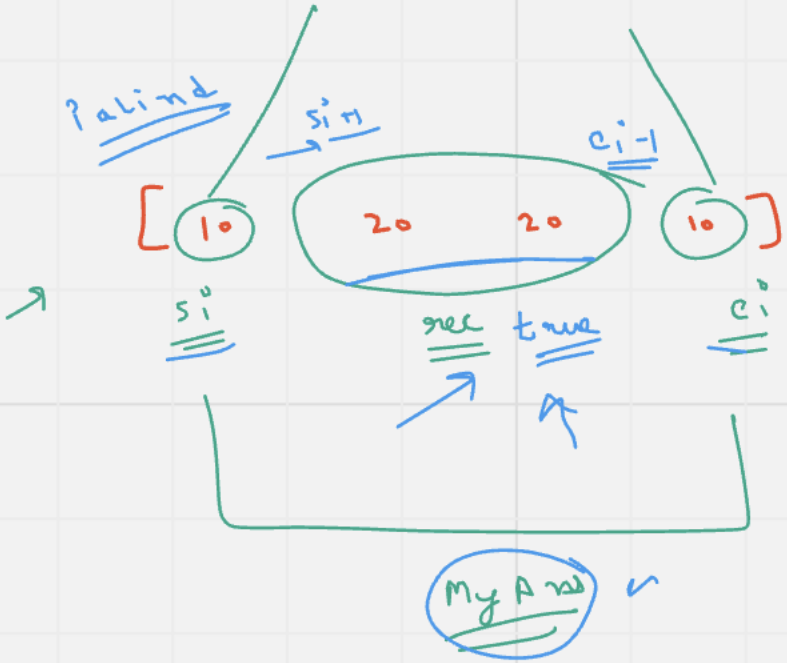

arr
arr
1 5 3 2 6 1
0 1 2 3 4 5
max
return = 5
I will
min = 1
You have

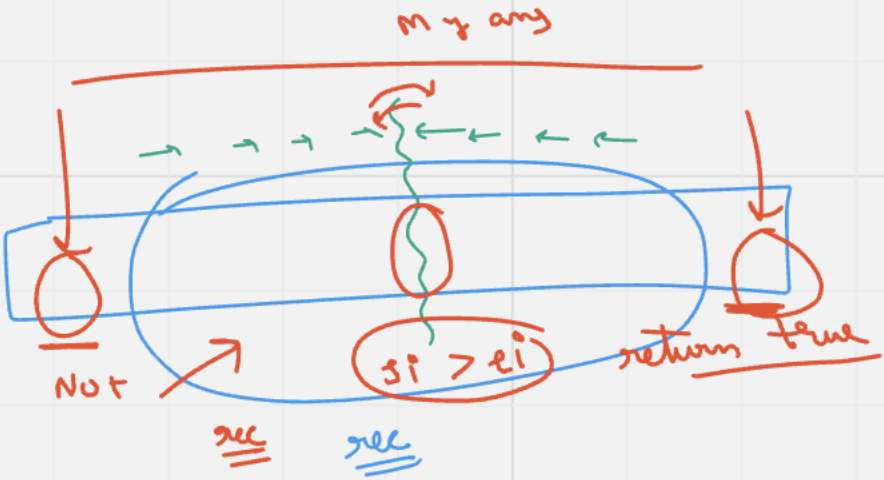
Sounds fair


map
11 8 3 10 2 5
10
rec
rec Ans
11
10





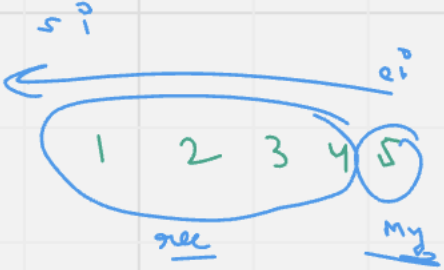




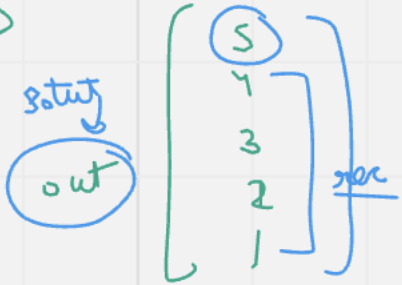
true

return false

Doubts ?.?



Solve



My Ans ✓