Day 7

15 November 2022 20:36

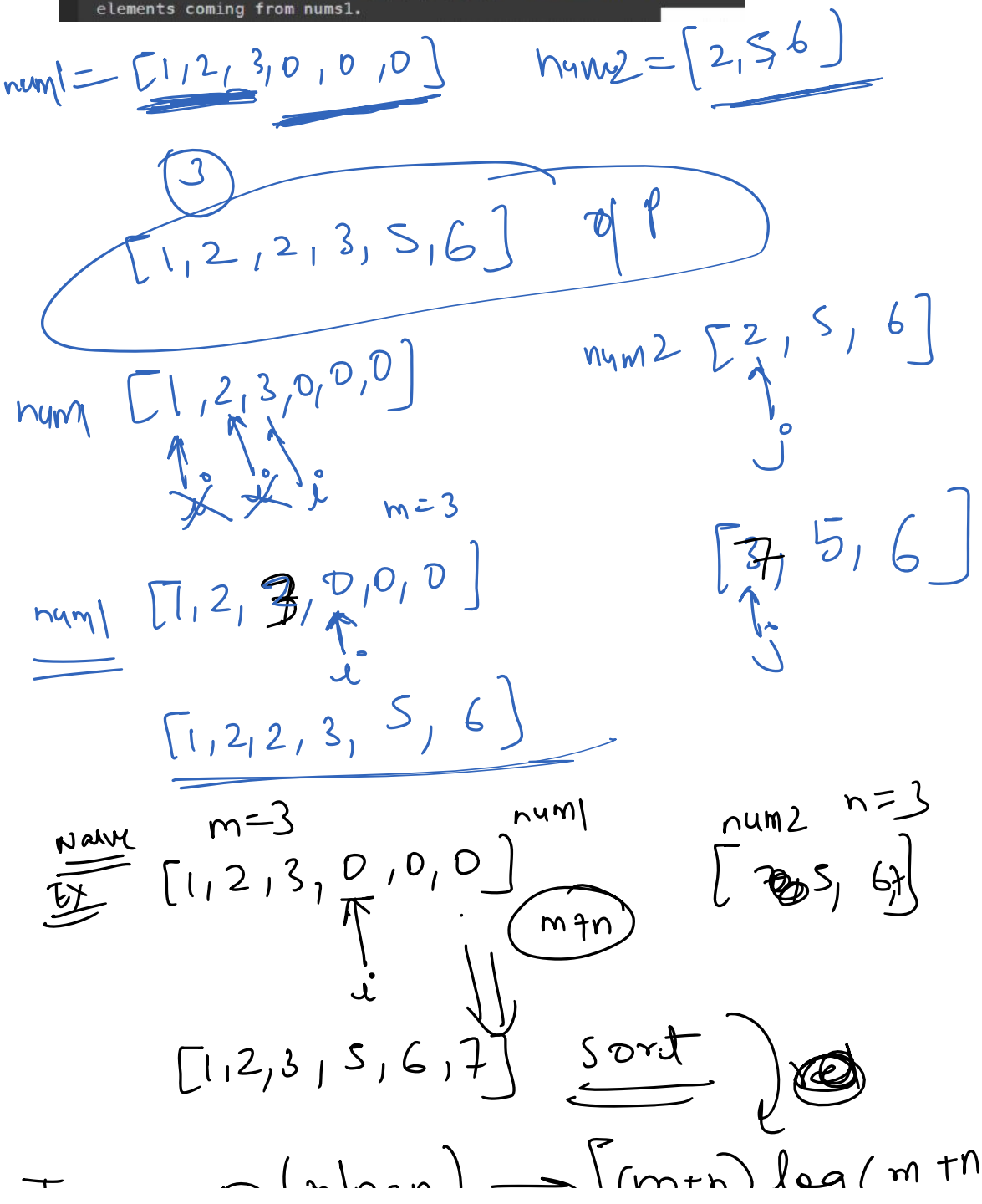
You are given two integer arrays `nums1` and `nums2`, sorted in **non-decreasing order**, and tw integers `m` and `n`, representing the number of elements in `nums1` and `nums2` respectively.

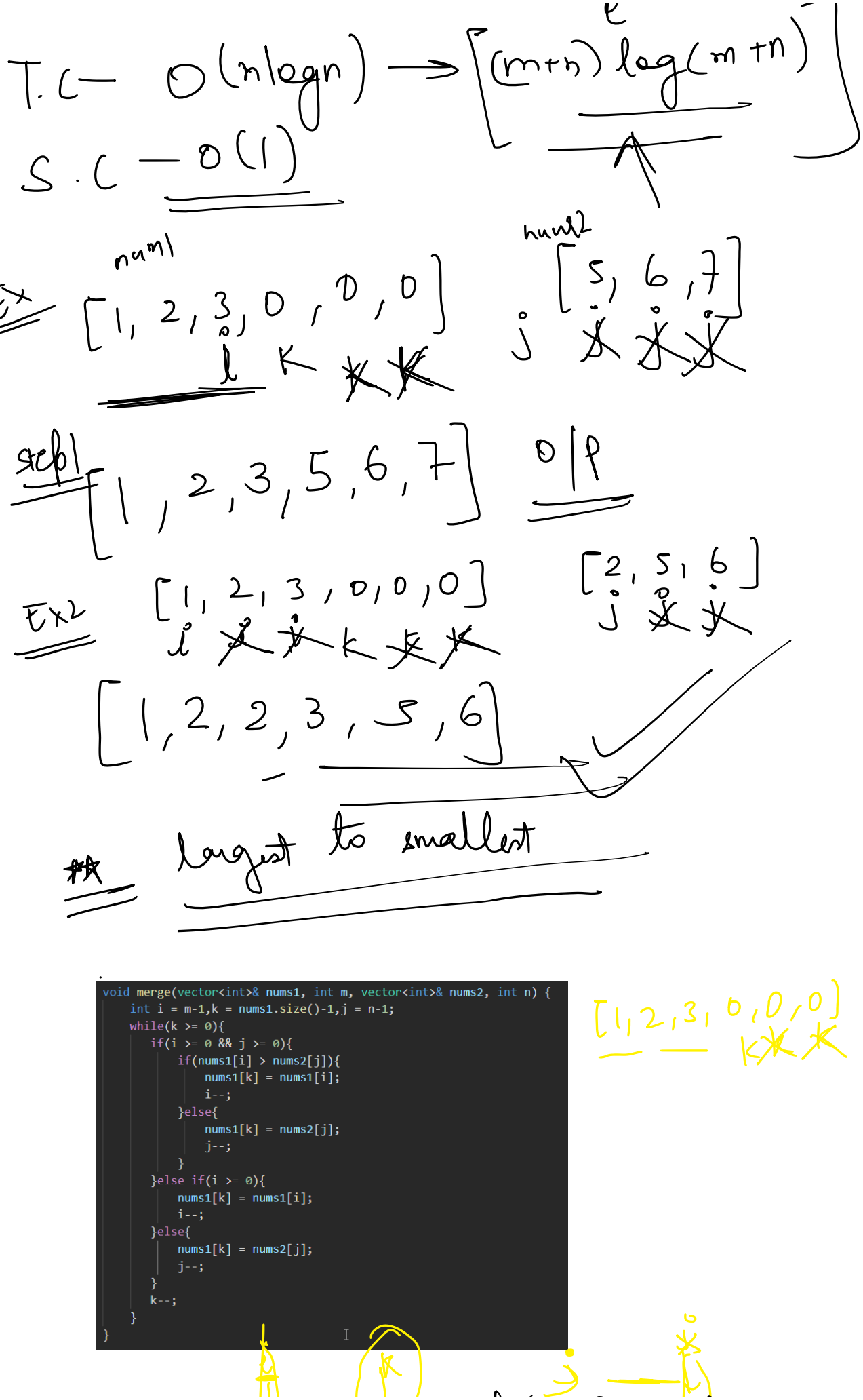
Merge `nums1` and `nums2` into a single array sorted in **non-decreasing order**.

The final sorted array should not be returned by the function, but instead be *stored inside the array* `nums1`. To accommodate this, `nums1` has a length of `m + n`, where the first `m` elemen denote the elements that should be merged, and the last `n` elements are set to `0` and shou be ignored. `nums2` has a length of `n`.

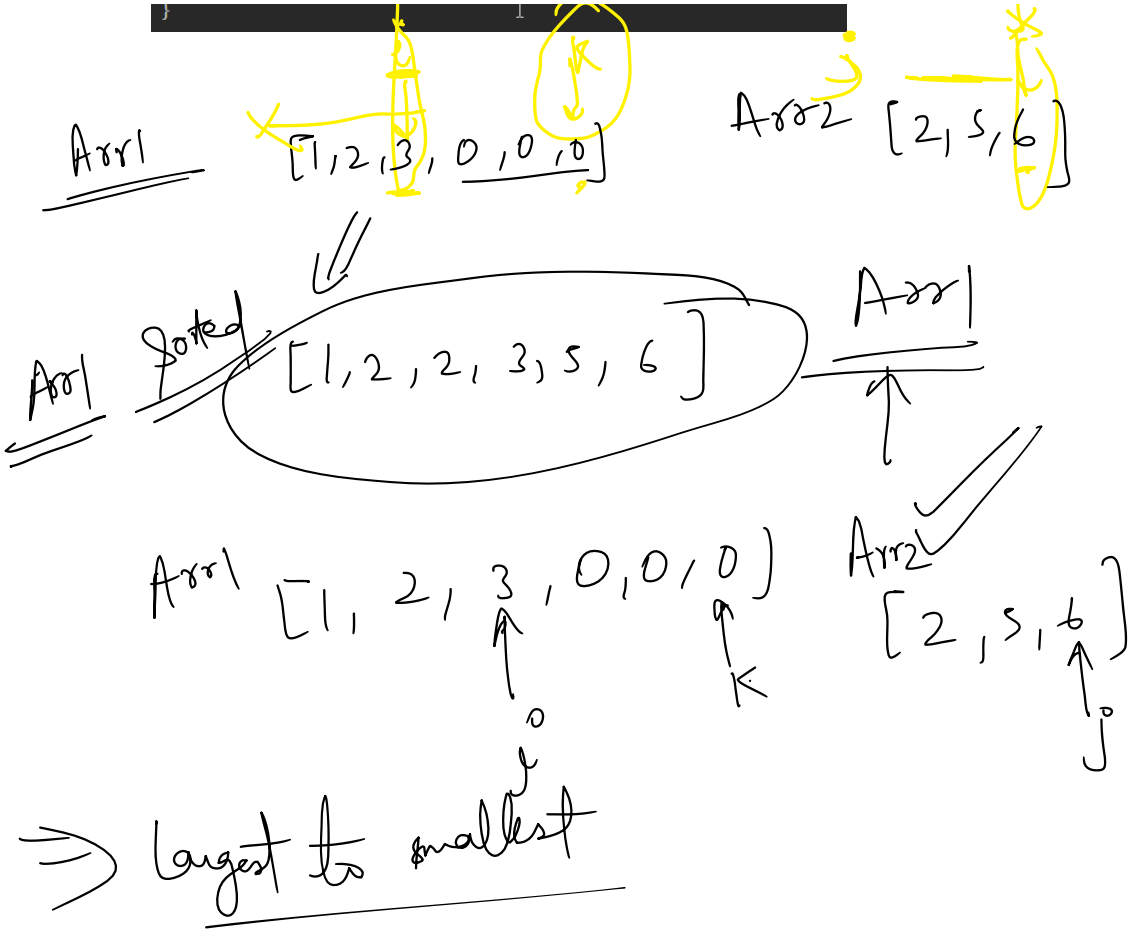
Example 1:

```
Input: nums1 = [1,2,3,0,0,0], m = 3, nums2 = [2,5,6], n = 3
Output: [1,2,2,3,5,6]
Explanation: The arrays we are merging are [1,2,3] and [2,5,6].
The result of the merge is [1,2,2,3,5,6] with the underlined
elements coming from nums1.
```






```
void merge(vector<int>& nums1, int m, vector<int>& nums2, int n) {
    int i = m-1,k = nums1.size()-1,j = n-1;
    while(k >= 0){
        if(i >= 0 && j >= 0){
            if(nums1[i] > nums2[j]){
                nums1[k] = nums1[i];
                i--;
            }else{
                nums1[k] = nums2[j];
                j--;
            }
        }else if(i >= 0){
            nums1[k] = nums1[i];
            i--;
        }else{
            nums1[k] = nums2[j];
            j--;
        }
        k--;
    }
}
```



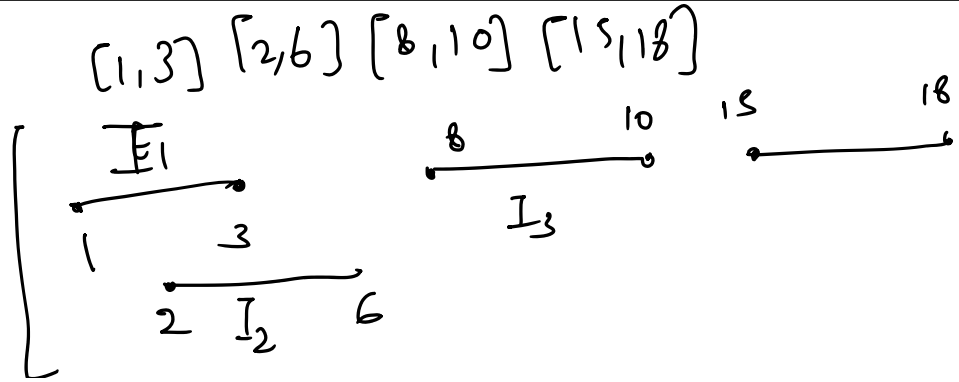
Given an array of `intervals` where `intervals[i] = [start_i, end_i]`, merge all overlapping intervals, and return *an array of the non-overlapping intervals that cover all the intervals in the input.*

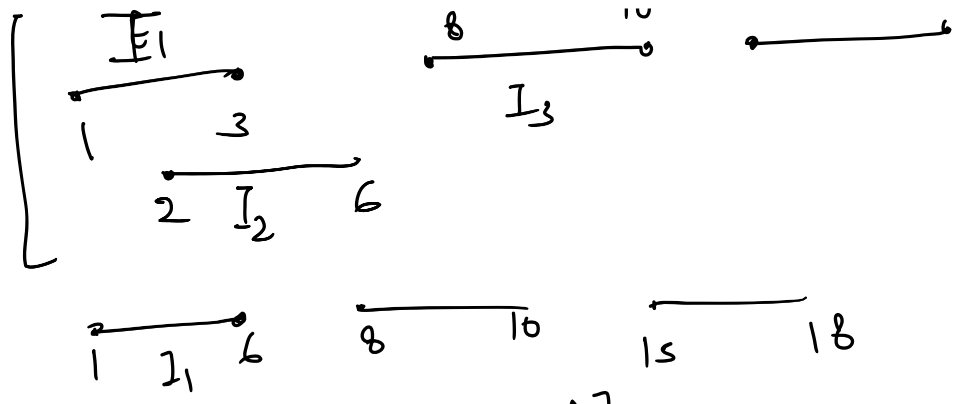
Example 1:

```
Input: intervals = [[1,3],[2,6],[8,10],[15,18]]
Output: [[1,6],[8,10],[15,18]]
Explanation: Since intervals [1,3] and [2,6] overlap, merge them into [1,6].
```

Example 2:

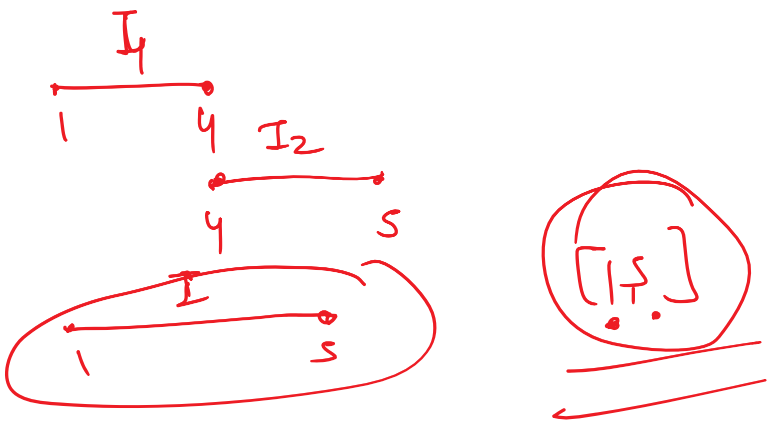
```
Input: intervals = [[1,4],[4,5]]
Output: [[1,5]]
Explanation: Intervals [1,4] and [4,5] are considered overlapping.
```





[[1,6], [8,10], [15,18]]

[[1,4], [4,5]]



[[1,3] [2, 6] [8, 10] [15,18]]

[[1,3] [2, 6] [8, 10] [15, 18]]

start = 1
ending = 6

start = 8
end = 10

start = 1
ending = 3

[[1,6] [8,10] [15, 18]]

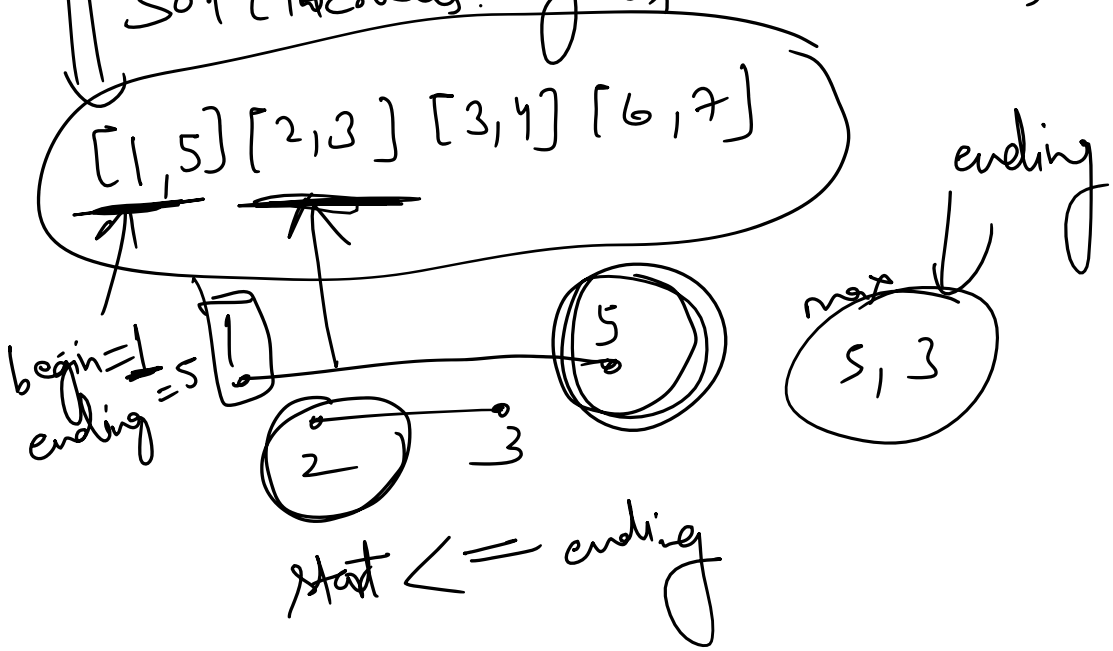
4 Hours Confusion

[[5,1], [6,3]]

Intervals को sort

[2,3], [1,5], [6,7], [3,4]

sort(intervals.begin(), intervals.end());



start <= ending

[1,5]

begin=1

ending=5

X

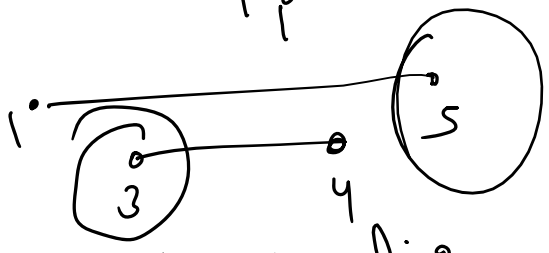
[2,3], [3,4], [6,7]

starting <= ending

2 <= 5

[3,4], [6,7]

p



starting <= ending

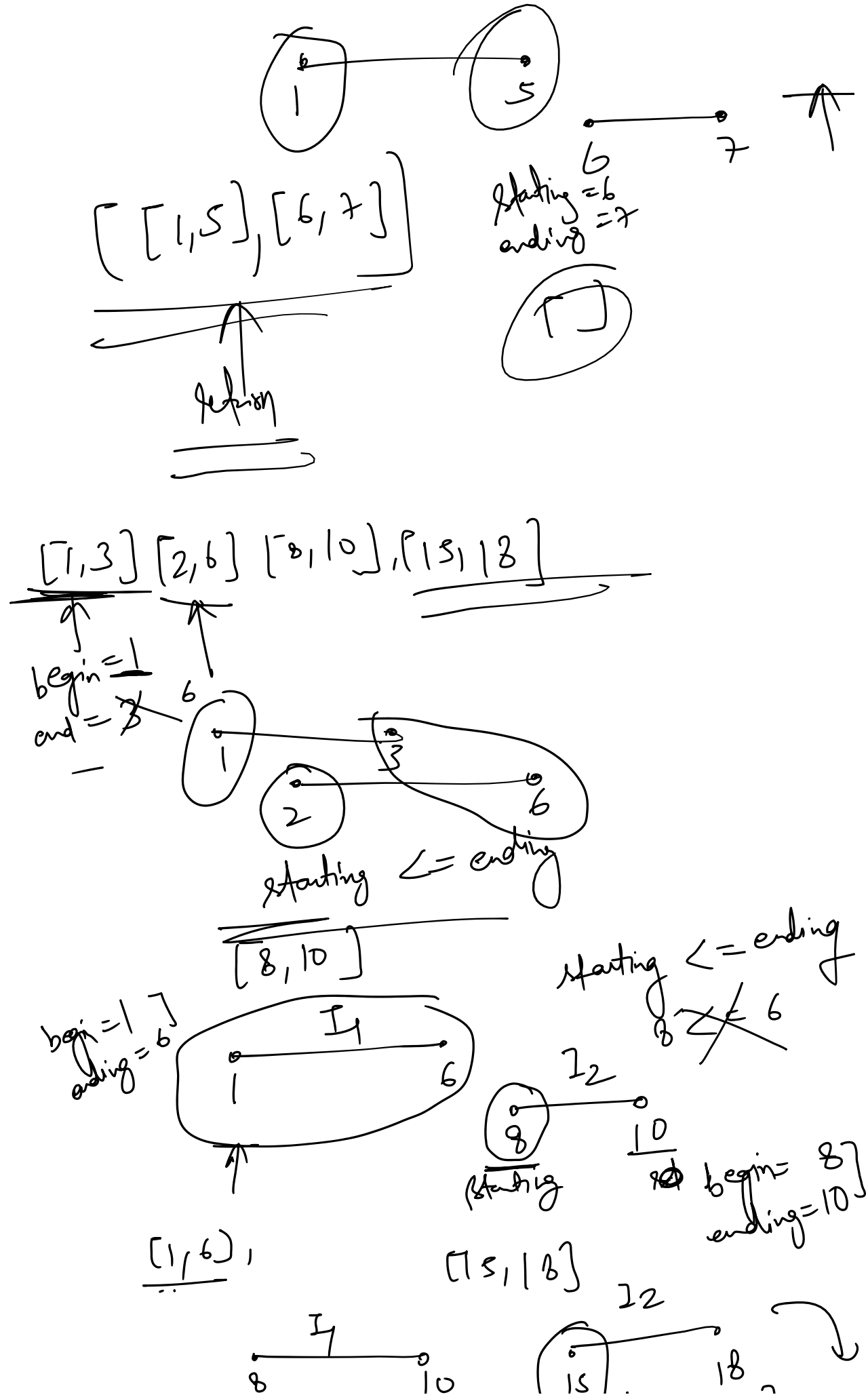
starting=1

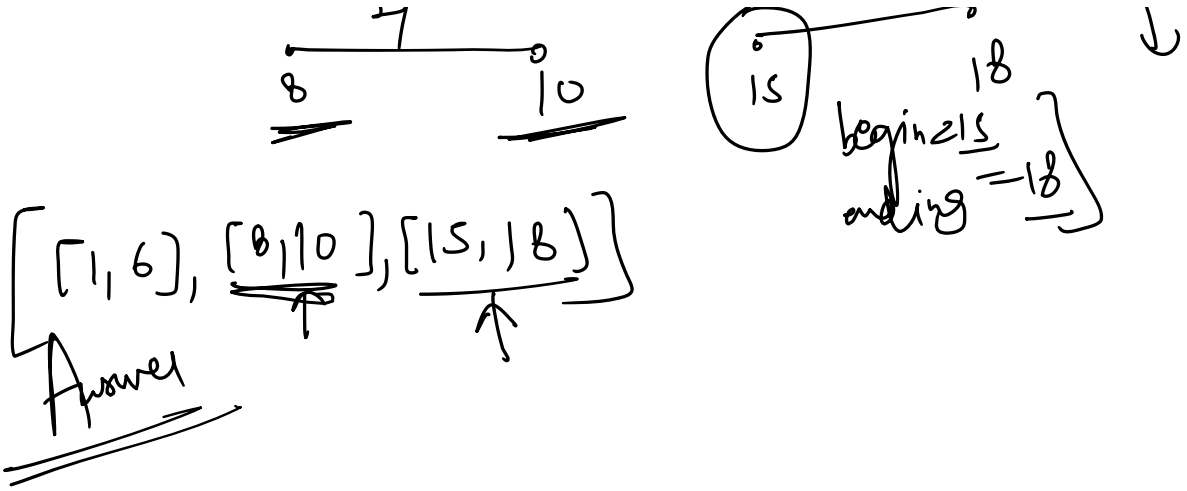
ending=5

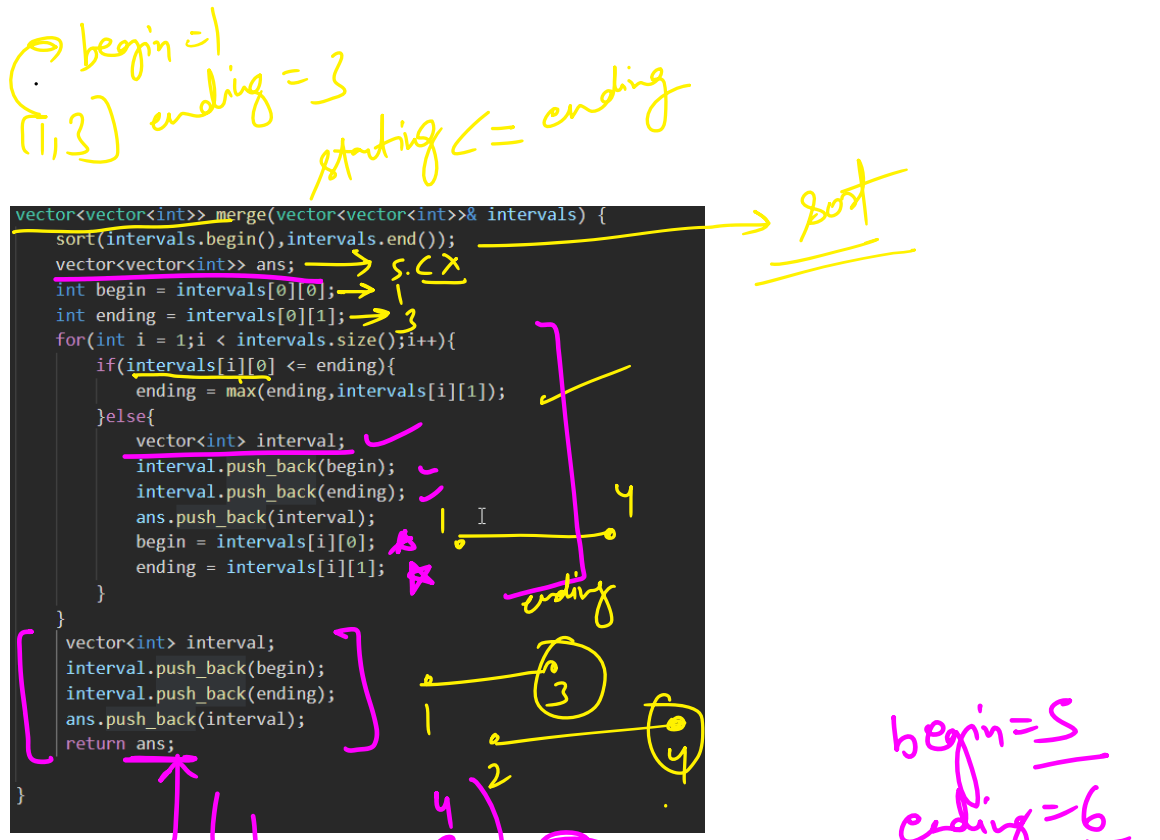
[6,7]

starting=6

6 <= 5


1
5
6
7
[[1,5], [6,7]]
starting = 6
ending = 7
[ ]
[1,3] [2,6] [8,10], [15,18]
begin = 1
end = 3
6
1
3
2
6
starting <= ending
[8,10]
begin = 1
ending = 6
I1
1
6
starting <= ending
8 <= 6
I2
8
10
starting
begin = 8
ending = 10
[1,6),
[15,18]
I1
8
10
I2
15
18


8
10
15
18
begin=15
ending=18
[1,6], [8,10], [15,18]
Answer


begin=1
ending=3
[1,3]
starting <= ending
sort
vector<vector<int>> merge(vector<vector<int>>& intervals) {
sort(intervals.begin(),intervals.end());
vector<vector<int>> ans;
int begin = intervals[0][0];
int ending = intervals[0][1];
for(int i = 1;i < intervals.size();i++){
if(intervals[i][0] <= ending){
ending = max(ending,intervals[i][1]);
}else{
vector<int> interval;
interval.push_back(begin);
interval.push_back(ending);
ans.push_back(interval);
begin = intervals[i][0];
ending = intervals[i][1];
}
}
vector<int> interval;
interval.push_back(begin);
interval.push_back(ending);
ans.push_back(interval);
return ans;
}
ending
begin=5
ending=6
[1,4] [5,6]
5 <= 4 X
[1,3] [2,6] [8,10], [15,18]
b=1
e=3 6
2<=3
8 <= 6
b=8
e=10
15<=10
b=15
e=18

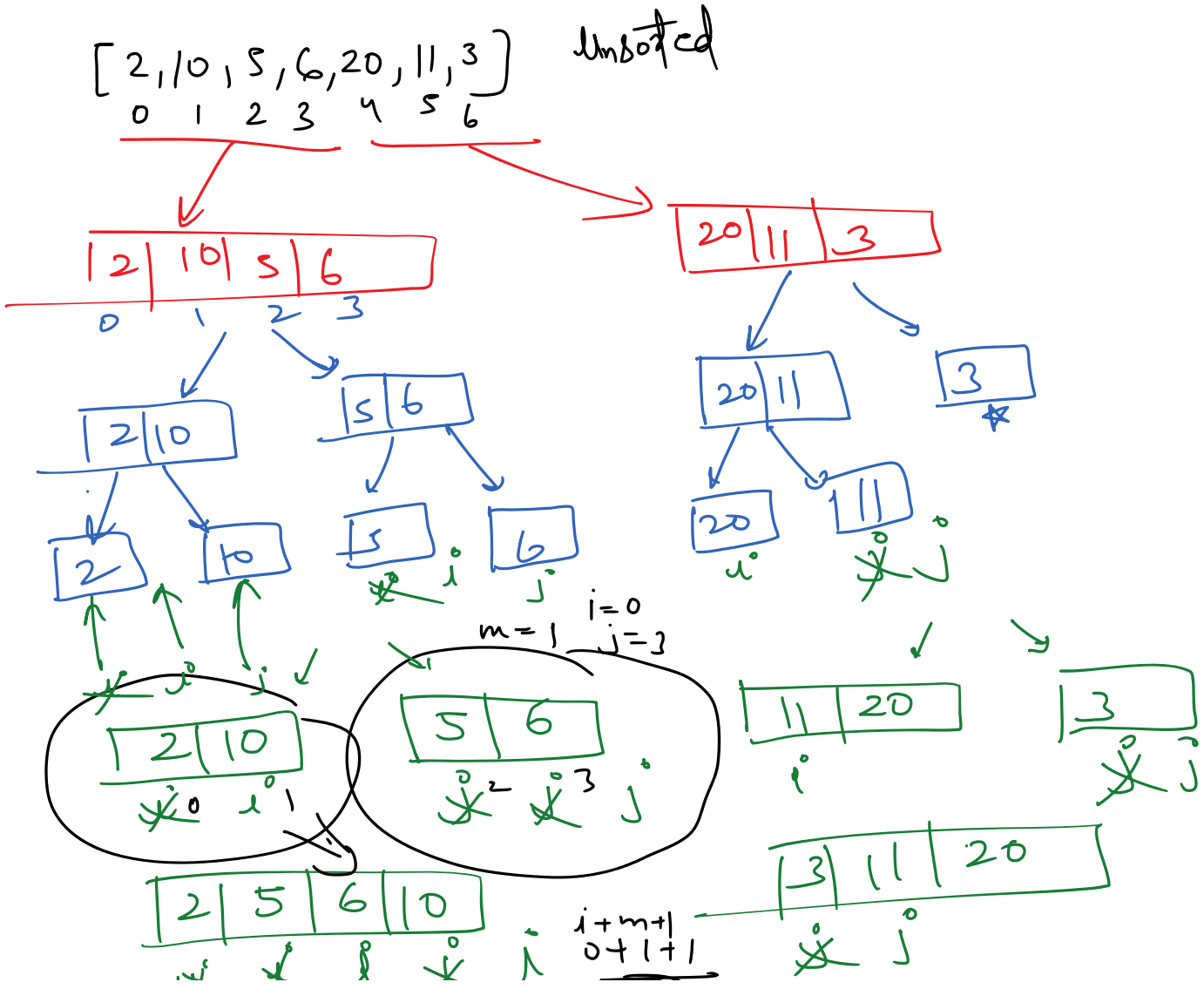
[[1,6],[8,10],[15,18]]

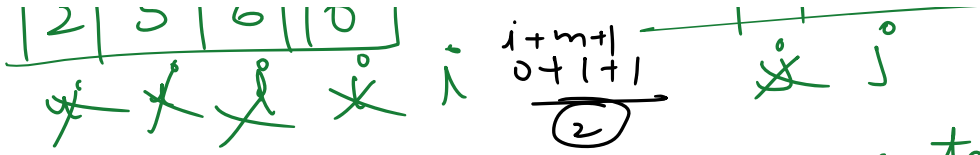
T.C - O(N)

S.C - O(1)

Merge Sort

Divide & Conquer





j - m

1 5 6 10 11 20 sorted Array

T.C - O(n log n)

S.C - O(n)

[2, 10, 5, 6, 20, 11, 3]
0 1 2 3 4 5 6

n = 7

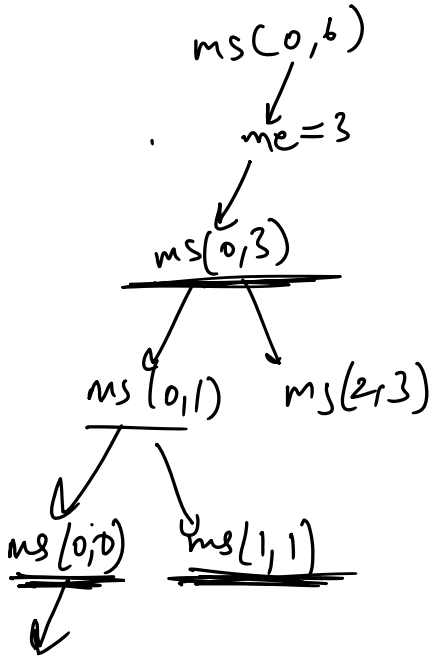
Recursive

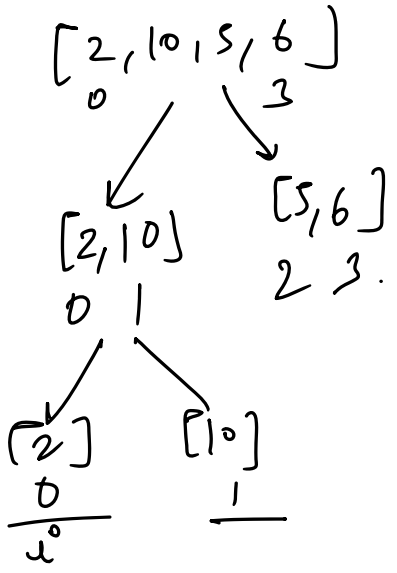
```
int main() {
    merge_sort(arr, 0, 6);
}

// Divide
void merge_sort(array, int i, int j) {
    if (i < j) {
        int me = (i + j) / 2;
        ✓ merge_sort(array, i, me); → i to me
        ✓ merge_sort(array, me+1, j);
        Merge(array, i, me, j);
    }
}
```

ms(0,0)
ms(1,1)

i = 0
me = 0
j = 1

[2, 10, 5, 6, 20, 11, 3]
 0 1 2 3 4 5 6
 i   j






```
Merge( a array, i, j, m ) {        i=0
    LeftArray [i+m+1];
    RightArray [j-m];
    for(int p=0; p< i+m+1; p++){
        LeftArray[p] = Array [p+i];
    }
    for(int p=0; p<j-m; p++){
        RightArray[p]= Array [p+m+1];
    }
```

```
    {
    int i1=0, j1=0, k=i;
    while( i1<i+m+1 or j1<j-m )
    { if (i1<i+m+1 && j1<j-m){
        if( LA[i1] < RA[j1] ){
            Arrg[k] = LA[i1];
            i1++;
        }else{
            Array[k] = RA[j1];
            j1++;
        }
        k++;
    }else if( i1<=i+m+1){
        Array[k]= LA[i1];
        i1++; k++;
    }else if( j1<=j-m){
        Array[k] = RA[j1];
        j1++; k++;
    }
    }
}
```

[2, 10, 5, 6, 20, 11, 3]
 0 1 2 3 4 5 6

ms(0,6)
me=3

ms(0,3)
me=1

ms(0,1)
me=0

ms(0,0)
i=j

ms(1,1)

ms(2,3)
[5] [6]

i=0 m=1 j=3

LA[2] → [2, 10]
RA[2] → [5, 6]

i=0 m=0 j=1

merge(Array, i, m, j)

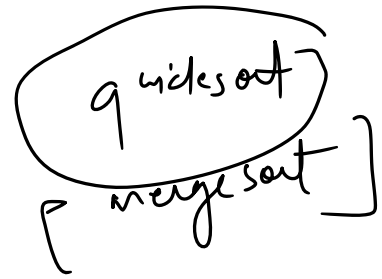
[2, 10, 5, 6, 20, 11, 3]

[2, 5, 6, 10, 20, 11, 3]

LA[i+m+1] ⇒ LA[1] → [2]
RA[j-m] ⇒ RA[1] → [10]

[2, 10, 5, 6, 20, 11, 3]

★ Divide Array function]
★ Merge two Array]
★ Changes in original array]



Count Inversion (100%)