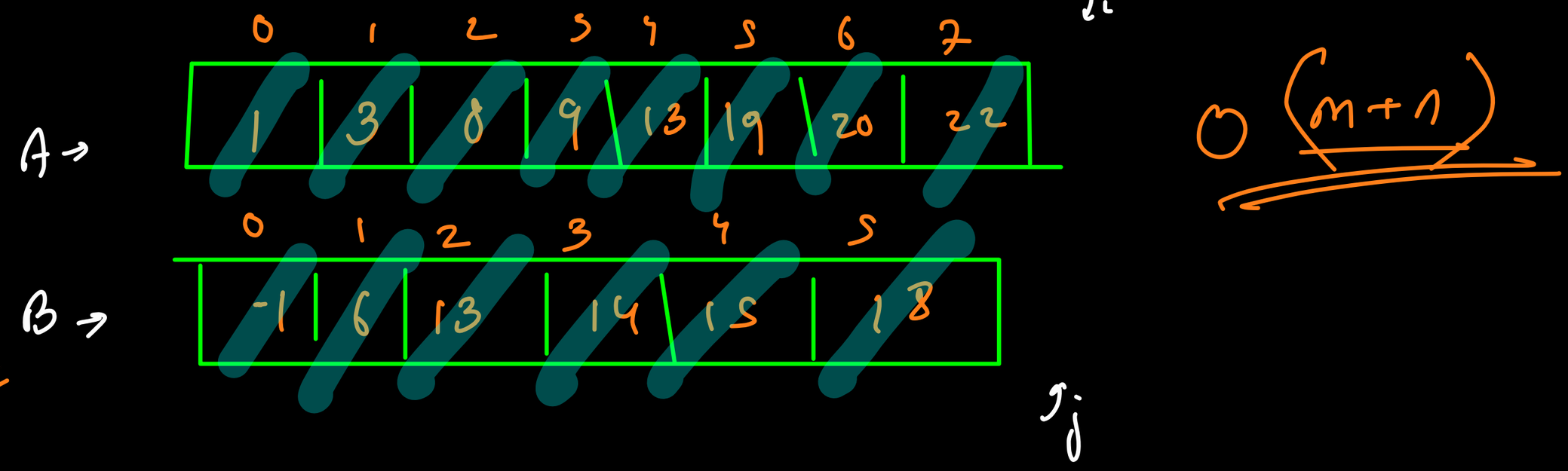
Q⇒ Given 2 sorted (asc) arrays, of length m and n. Merge the 2 sorted arrays into a new single array of length (m+n) such that the resultant array is also sorted.

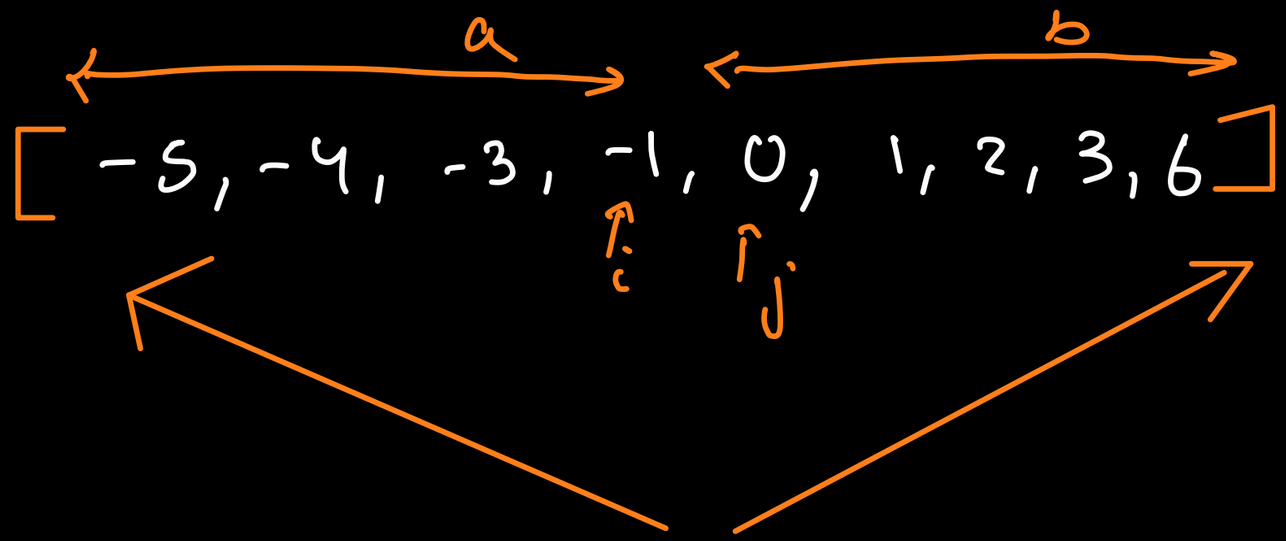
$n \leq 10^5$
$m \leq 10^5$

Ex → [1, 5, 7, 9]
[2, 3, 8, 11, 13, 15]

ans → [1, 2, 3, 5, 7, 8, 9, 11, 13, 15]
2
asc




```
if ( A[i] <= B[j] ) {
    C[k] = A[i]
    i++; k++;
} else {
    C[k] = B[j]
    j++; k++
}
```




O(n)

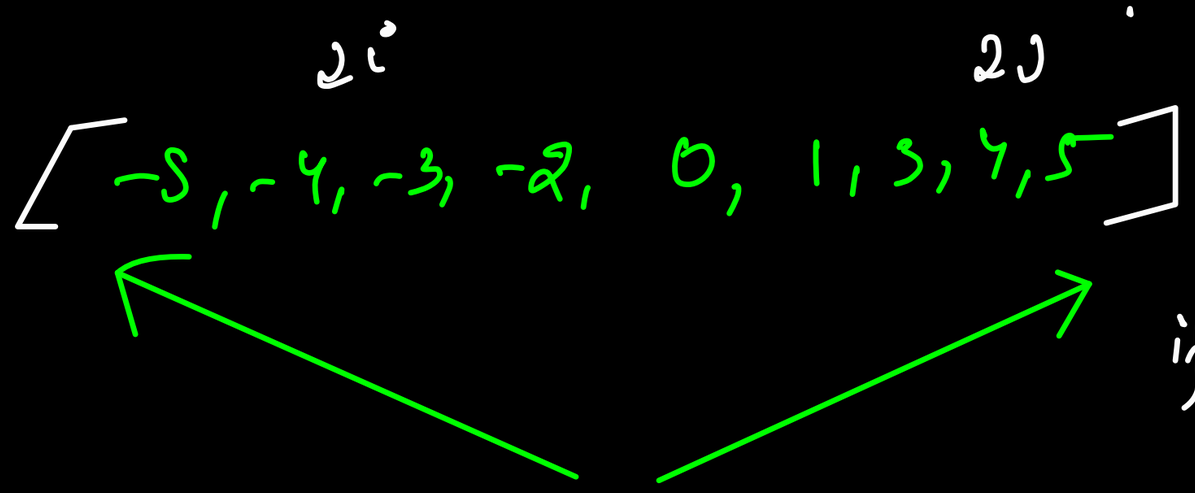
if (abs(a[i]) <= a[j])
c[k] = a[i];
i--
k++

[ 0, 1, 2, 3, 6 ]

[ 1, 3, 4, 5 ]

↳ ( 0, 1, 1, 2, 3, 3, 4, 5, 6 ]

( 0 1 4 9 9 16 25 36 ]

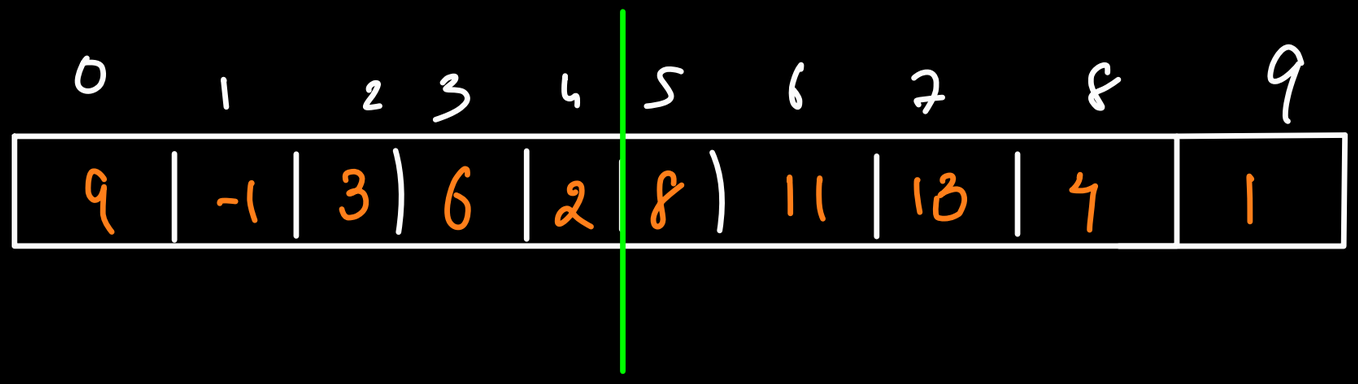



O(n)


```
if(abs(a[i]) >= abs(a[j])
  c[k] = abs(a[i])
  i++
  k--
}else{
  c[k] = abs(a[j])
  j--
  k--
}
```

3

merge 2 sorted arrays




merge sort

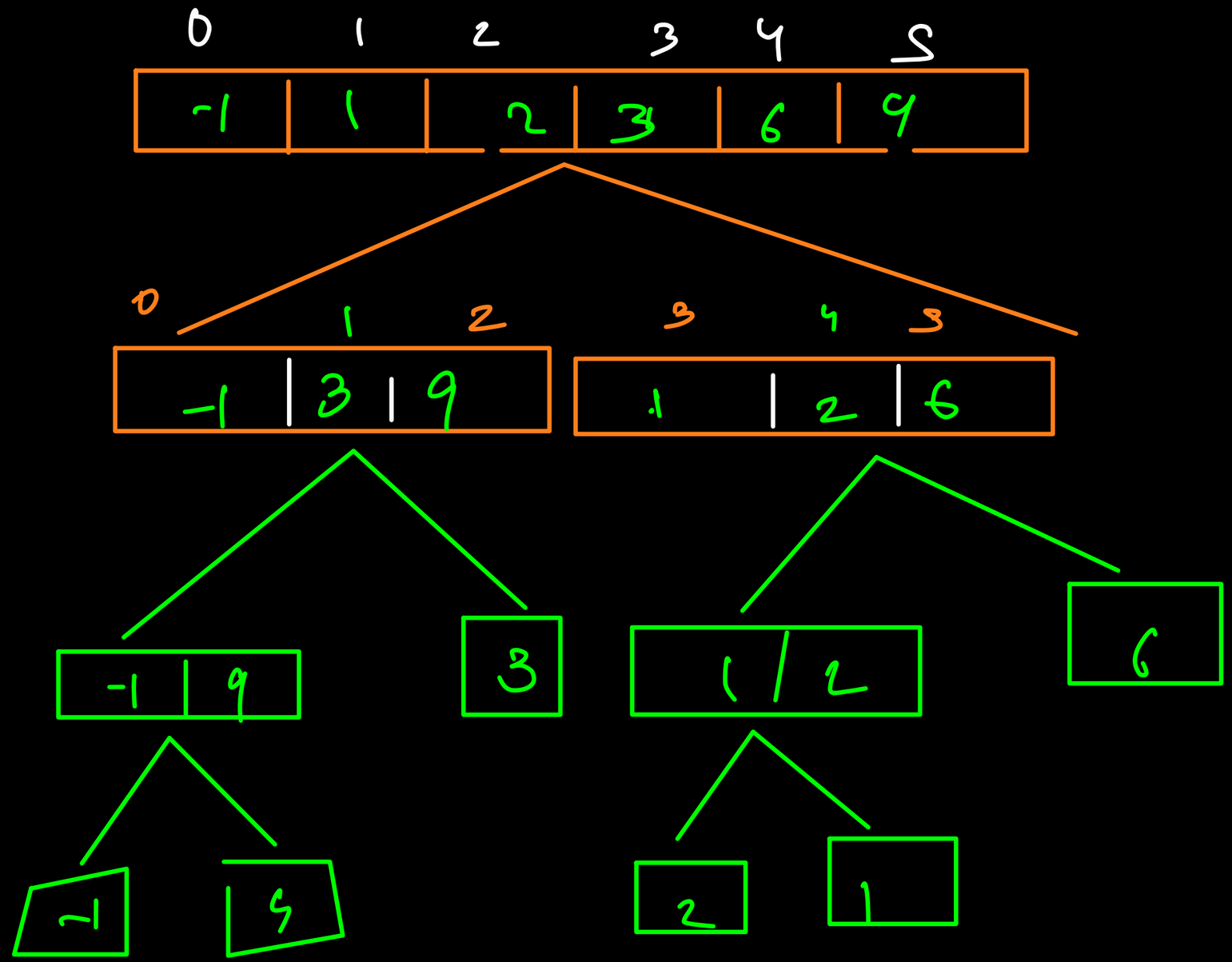
↓

recursive algo

-1, 2, 3, 6, 9 1, 4, 8, 11, 13

↳ Recursively sort the left & right half

↳ merge the 2 sorted halfs



$f(arr, i, j)$ = $f(arr, i, mid)$ $f(arr, mid+1, j)$ merge()

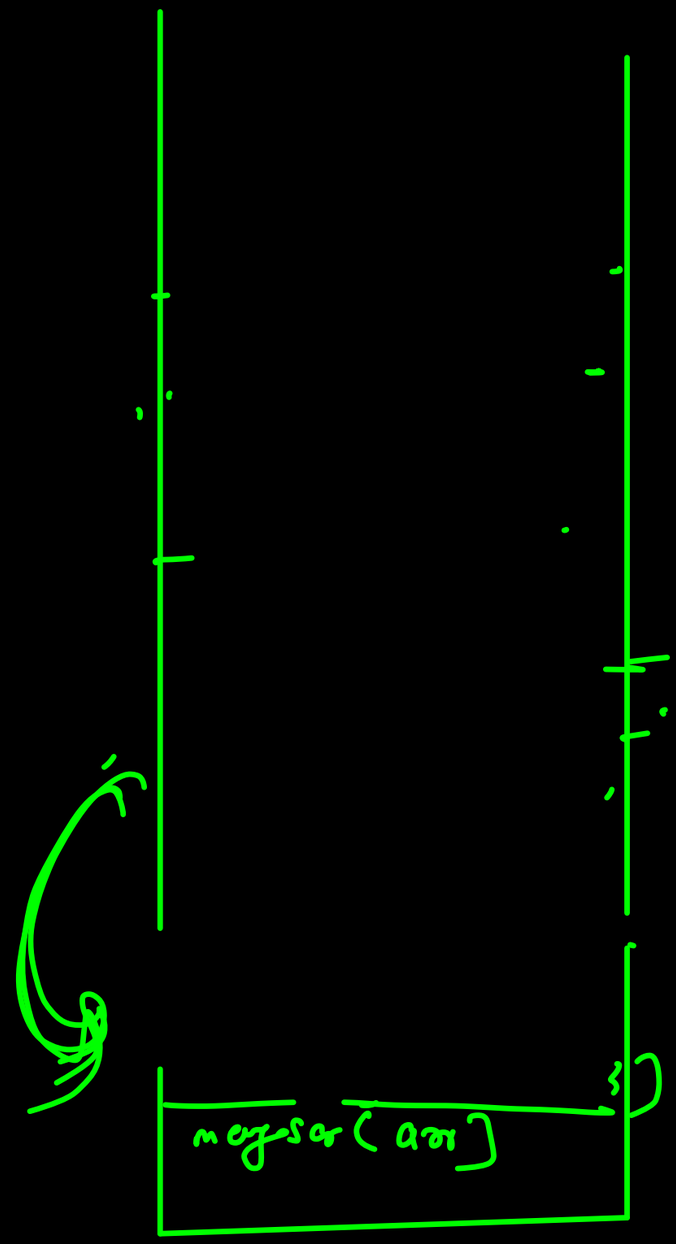
2 nested on the array in the range (i, j)

```cpp
std::vector<int> f(std::vector<int> &arr, int i, int j) { // this returns a sorted array
    if(i == j) {
        // single length array
        return std::vector<int> {arr[i]};
    }
    int mid = (i + j) / 2;
    std::vector<int> left = f(arr, i, mid);
    std::vector<int> right = f(arr, mid+1, j);
    std::vector<int> result = mergeTwoSortedArrays(left, right);
    return result;
}

void merge_sort(std::vector<int> &arr) {
    arr = f(arr, 0, arr.size() - 1);
}
```

arr = [5, 4, 3, 2]

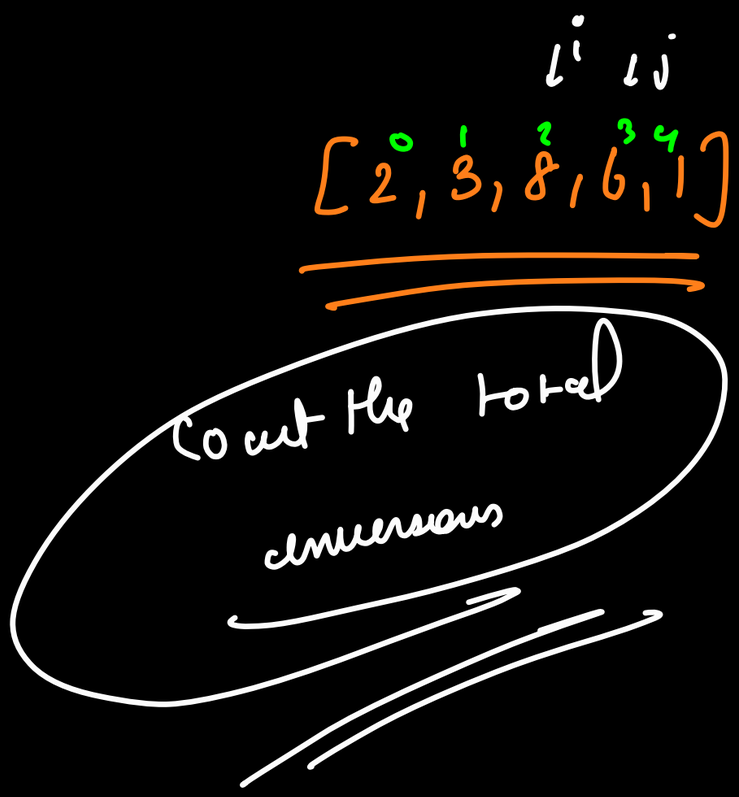
[2, 3, 4, 5]



Time → $O(n \log n)$

Space → $O(n)$

HW

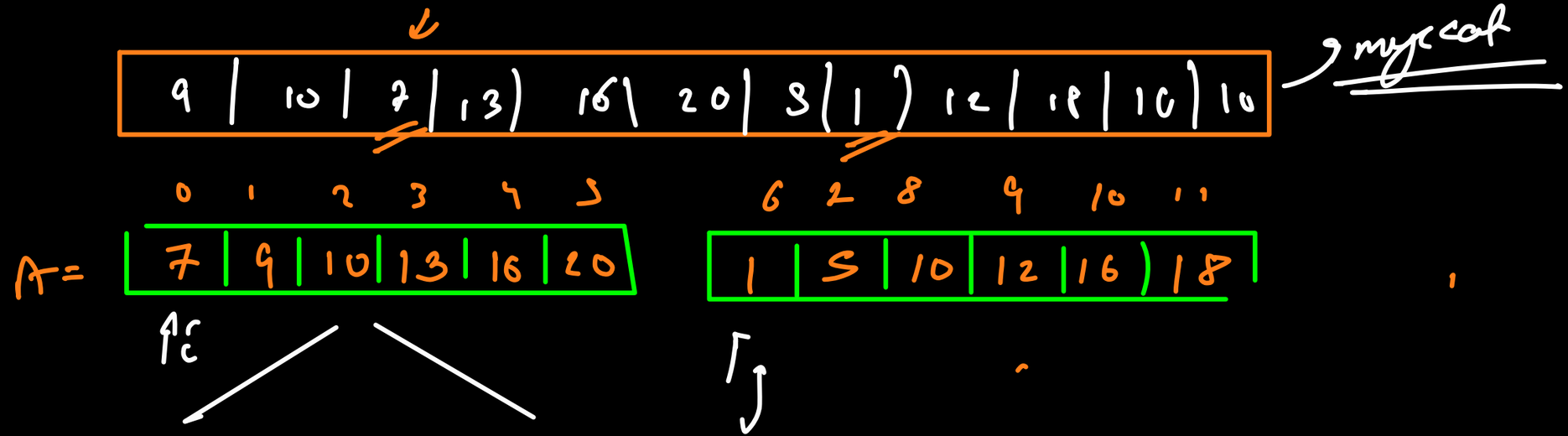


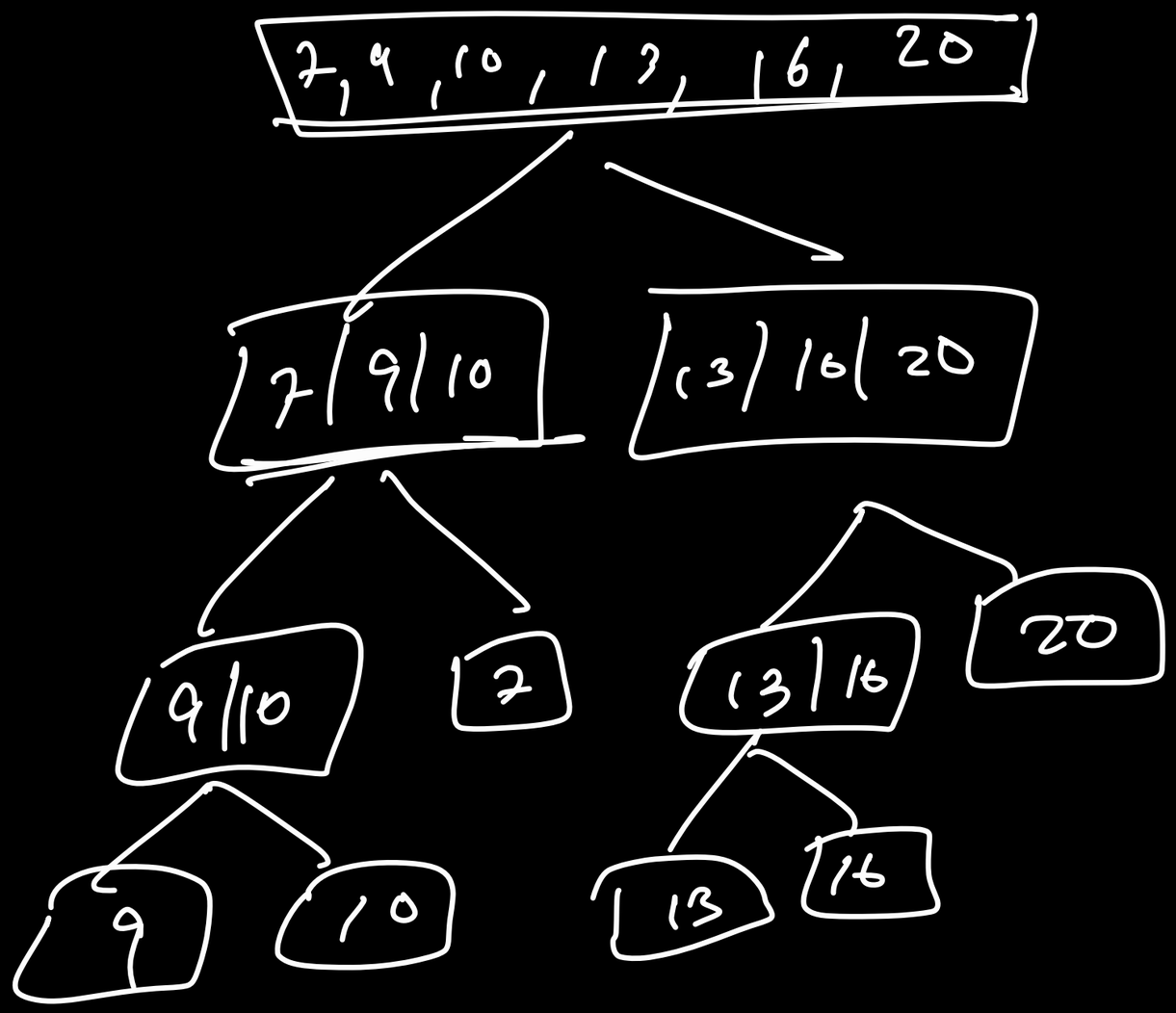

(0,4)
(1,4)
(2,4)
(3,4)
(2,3)

5

Brute force → $O(n^2)$

$n \leq 2 \times 10^5$






[2,4,3,5,1]

2+

2,3,4,

i

1,5

j

1 2 2 3 ?

1 2 5 1 3