Sorting 1 : Count Sort & Merge Sort

Hello everyone :)

Mipun Mittal Googler
BTech - 2019, MTech - 2021 (IIT Bombay)
~ 2 years of part-time teaching experience

Ques1: find the smallest no. that can be formed by rearranging the digits of a given no. in array.

$0 <= A[i] <= 9$

A = [3 2 4 1] → [1 2 3 4]

A = [6 3 4 2 7 2 0] → [0 2 2 3 4 6 7]

Soln: sorting in ascending order

Inbuilt sorting : TC = O(NlogN)

output will always look like :

0000... 111... 22 ... 33 ..... 88 ... 99....

freq(0) freq(1) ...

A: [9 8 9 7 1 1 9 0 0 2 9 1 3 7]

0 1 2 3 4 5 6 7 8 9 10 11 12 13

f: [0 0 0 0 0 0 0 2 1 4]

0 1 2 3 4 5 6 7 8 9

f(i) → freq of i (0-9) in the array

f = [2 3 1 1 0 0 0 2 1 4]

0 1 2 3 4 5 6 7 8 9

→ 0 0 1 1 1 2 3 7 7 8 9 9 9 9

Count Sort

∀i, f[i] = 0 // len = 10

```
for (i=0 to n-1) {
    f[A[i]] ++
}
```

calculating frequency

N iterations

```
for (d = 0 to 9) {
    for (i=1 to f[d]) {
        print(d)
    }
}
```

sort array using frequency

iterations → 10×N

or

N ✓

total TC : O(N)

SC = O(10) = O(1)

d	i	iteration
0	1...f[0]	f[0]
1		f[1]
⋮		⋮
9		f[9]

$\Rightarrow f[0] + f[1] + \dots + f[9] = N$

What if $0 <= A[i] <= 10^9$ → Is CountSort possible?

↓

length of freq array = 10^9

↓

int array

4B

$4B \times 10^3 \times 10^3 \times 10^3$ (10^9)

4KB

4MB

~4GB (Array space)

↓

Memory Limit Exceeded error

(MLE)

$[10^6 - 10^7]$

What if $-9 <= A[i] <= 9$ → Is CountSort possible?

len = $[-9, 9]$ $\Rightarrow 9 - (-9) + 1 = 19$ yes

A = [-2 3 8 -4 -2 3 0]
(0 1 2 3 4 5 6)

f = [1 2 1 2 1]
(0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18)
(-9 -8 -7 . . . -3 -2 -1 0 1 2 3 . . . . . 8 9)

∀i, f[i] = 0 // len = 19

for (i = 0 to n-1) {

~~f[A[i]]++~~ f[A[i] + 9]++ // f[A[i] - minElement]

in this case minElement = -9

}

for (d = 0 to len) {

for (i = 1 to f[d]) {

~~print(d)~~ print(d - 9) // print(d + minElement)

}

}

TC = O(N)

SC = O(range of array)

(max(A[i]) - min(A[i]) + 1) <= 10^6

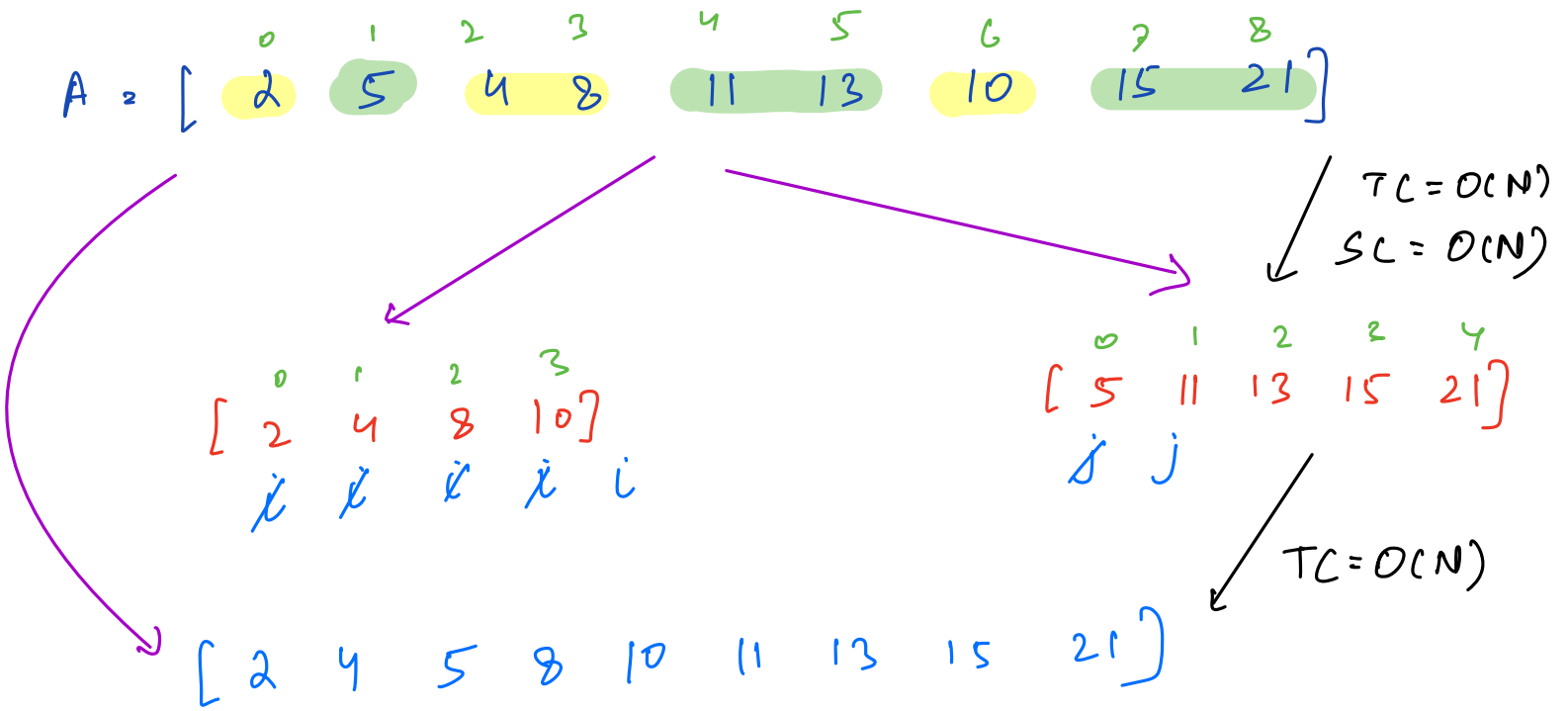
use CountSort

$[-10^3, 10^3]$

$\Rightarrow 10^3 - (-10^3) + 1$

$\Rightarrow 2 \times 10^3 + 1 \quad <= 10^6$ ✓

Ques Given an integer array where all odd elements are sorted & all even elements are sorted, sort the array.



Merge 2 sorted array of size N & M

TC = O(N+M)

SC = O(N+M)

```
int[] merge(A[], N, B[], M) {
    ans[N+M]
    i=0, j=0, k=0
    while(i<N && j<M){
        if(A[i] <= B[j]) {
            ans[K] = A[i]
            i++
        }
        else {
            ans[K] = B[j]
            j++
        }
        K++
    }
    while(i<N){
        ans[K] = A[i]
        K++, i++
    }
    while(j<M) {
        ans[K] = B[j]
        K++, j++
    }
```

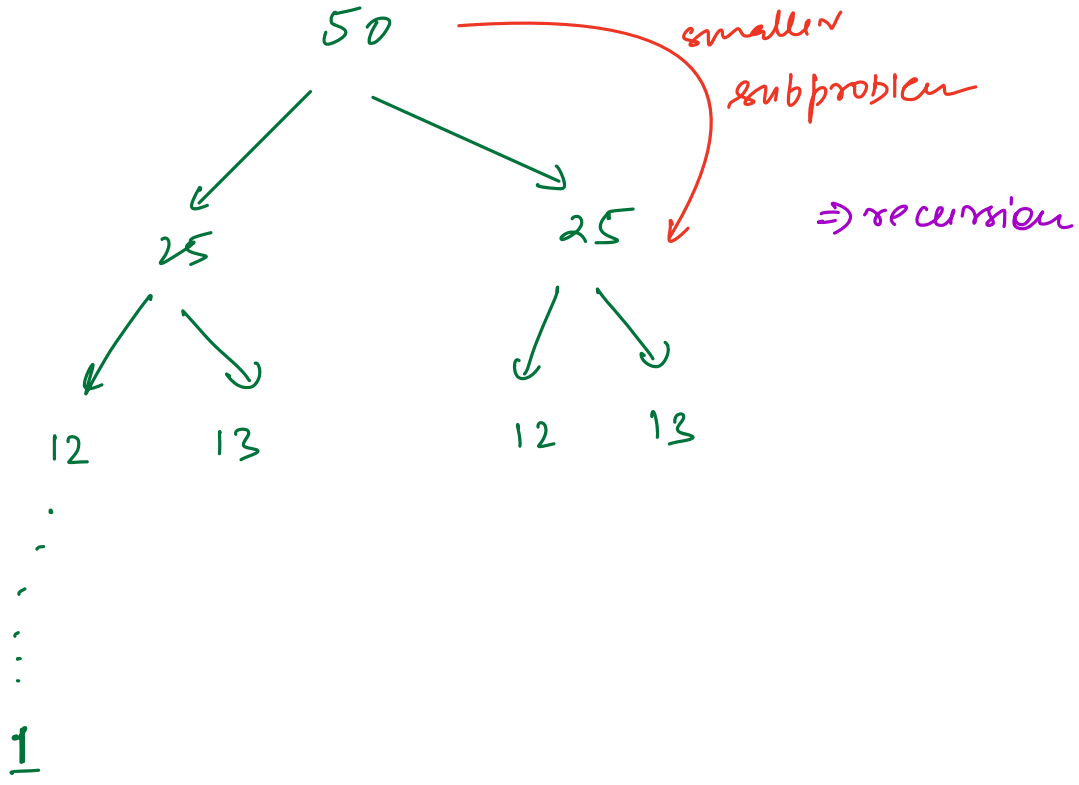
TC = O(N+M)

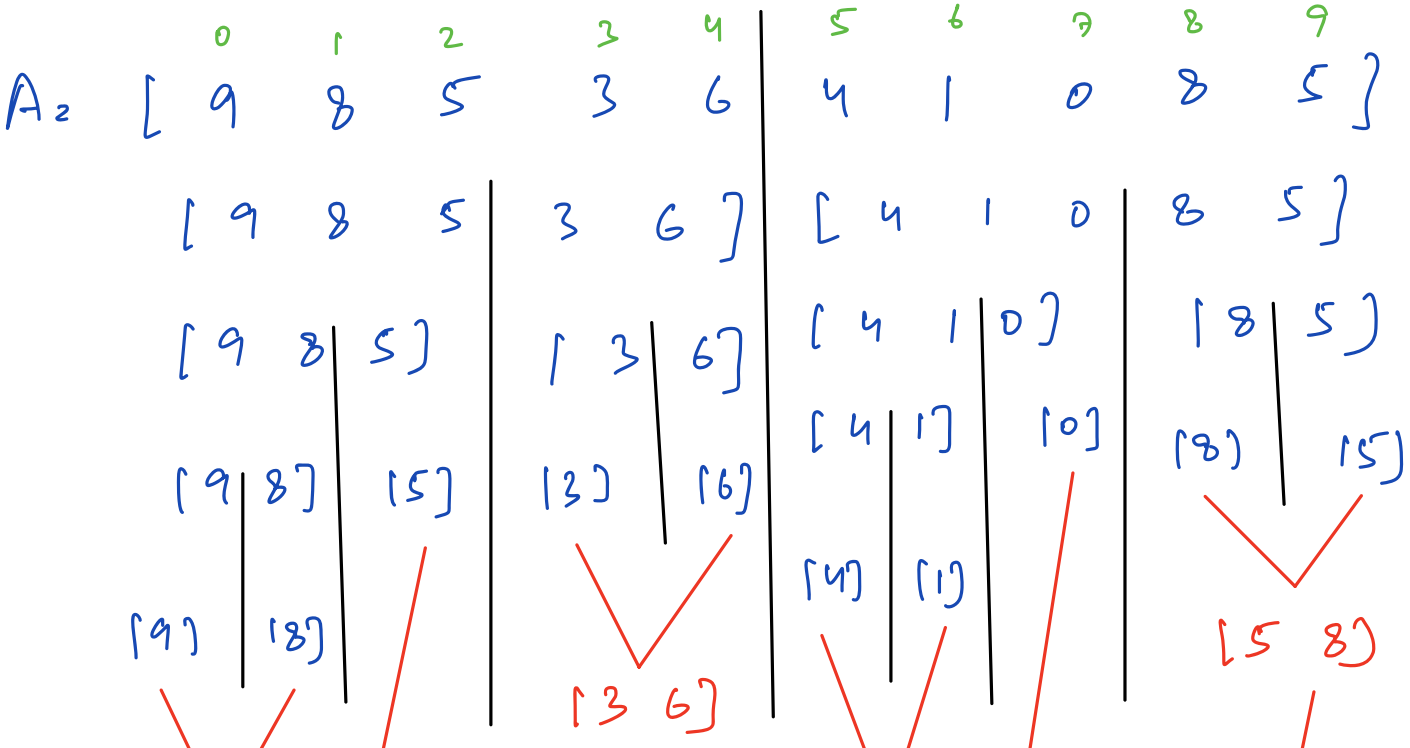
SC = O(N+M)

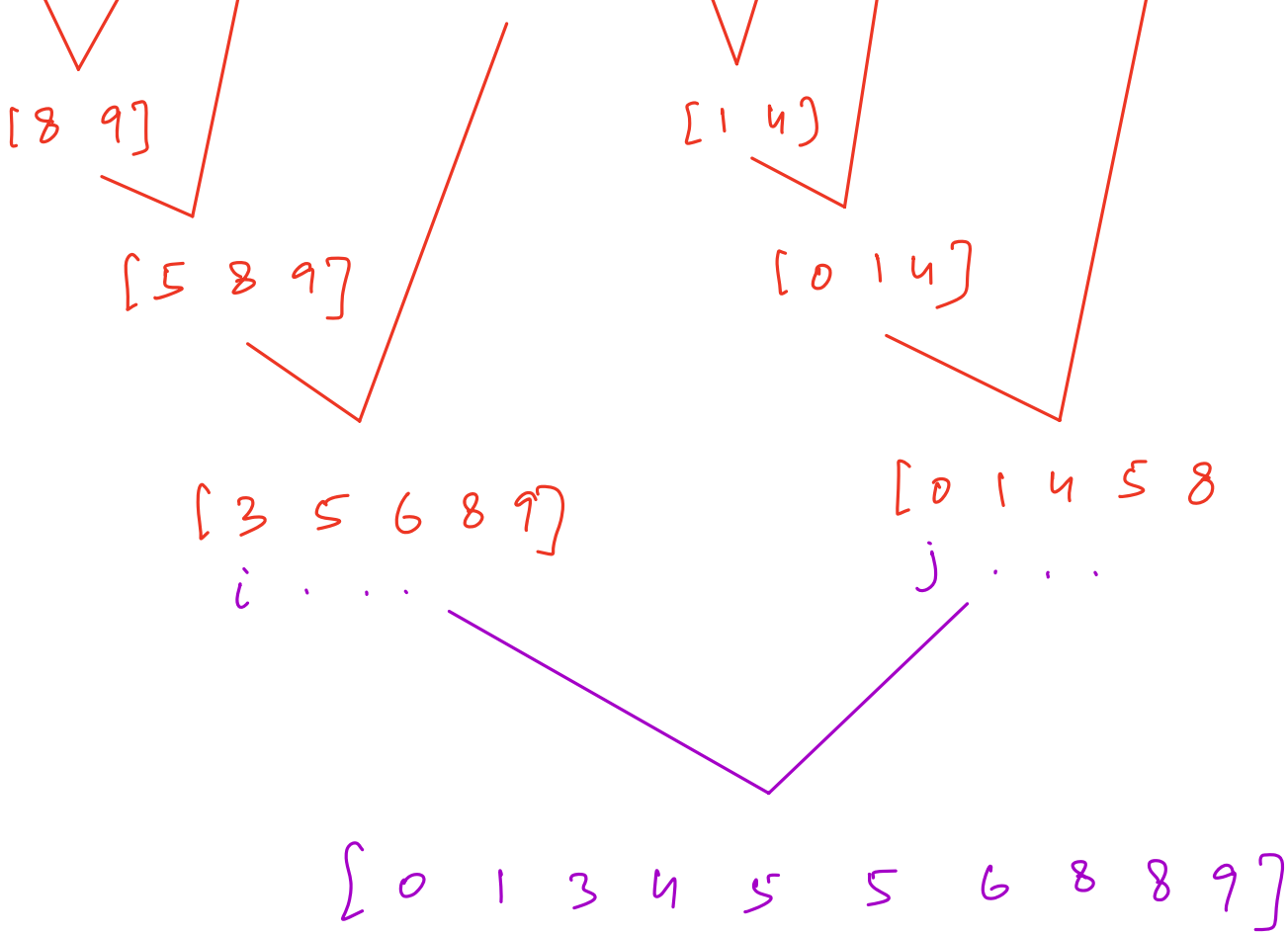
return ans

}

Merge Sort

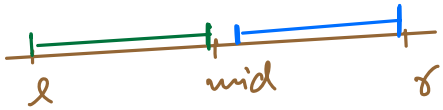




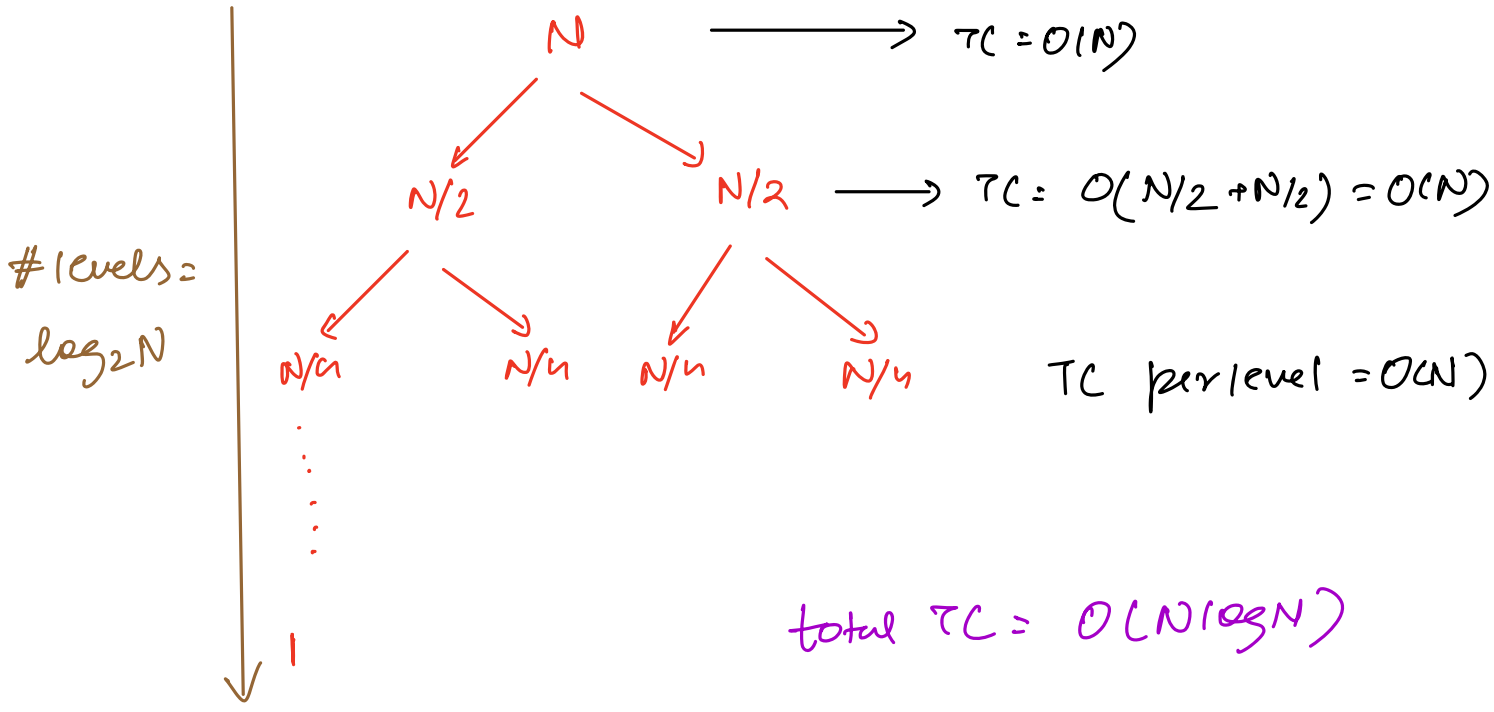



```
void mergeSort(A[], l, r) {
    if(l >= r)  return;
    mid = (l+r)/2
    mergeSort(A, l, mid)
    mergeSort(A, mid+1, r)
    merge(A, l, mid, r)
}
```

(l: 0, r: N-1)



merge(A, l, mid, r) → TC = O(N), SC = O(N)



total TC: $O(N \log N)$

SC: $O(N + \log N) = O(N)$

because of recursion stack size

Question

Given an array, find count of inversion pairs.

inversion pair (i, j) $\Rightarrow$ $i < j$ && $A[i] > A[j]$

	0	1	2	3	4
A = [	10	3	8	15	6]

i < j		A[i] > A[j]	
0	1	10 > 3	✓
0	2	10 > 8	✓
0	3	10 > 15	✗
0	4	10 > 6	✓
1	2	3 > 8	✗
1	3	3 > 15	✗
1	4	3 > 6	✗
2	3	8 > 15	✗
2	4	8 > 6	✓
3	4	15 > 6	✓

ans = 5

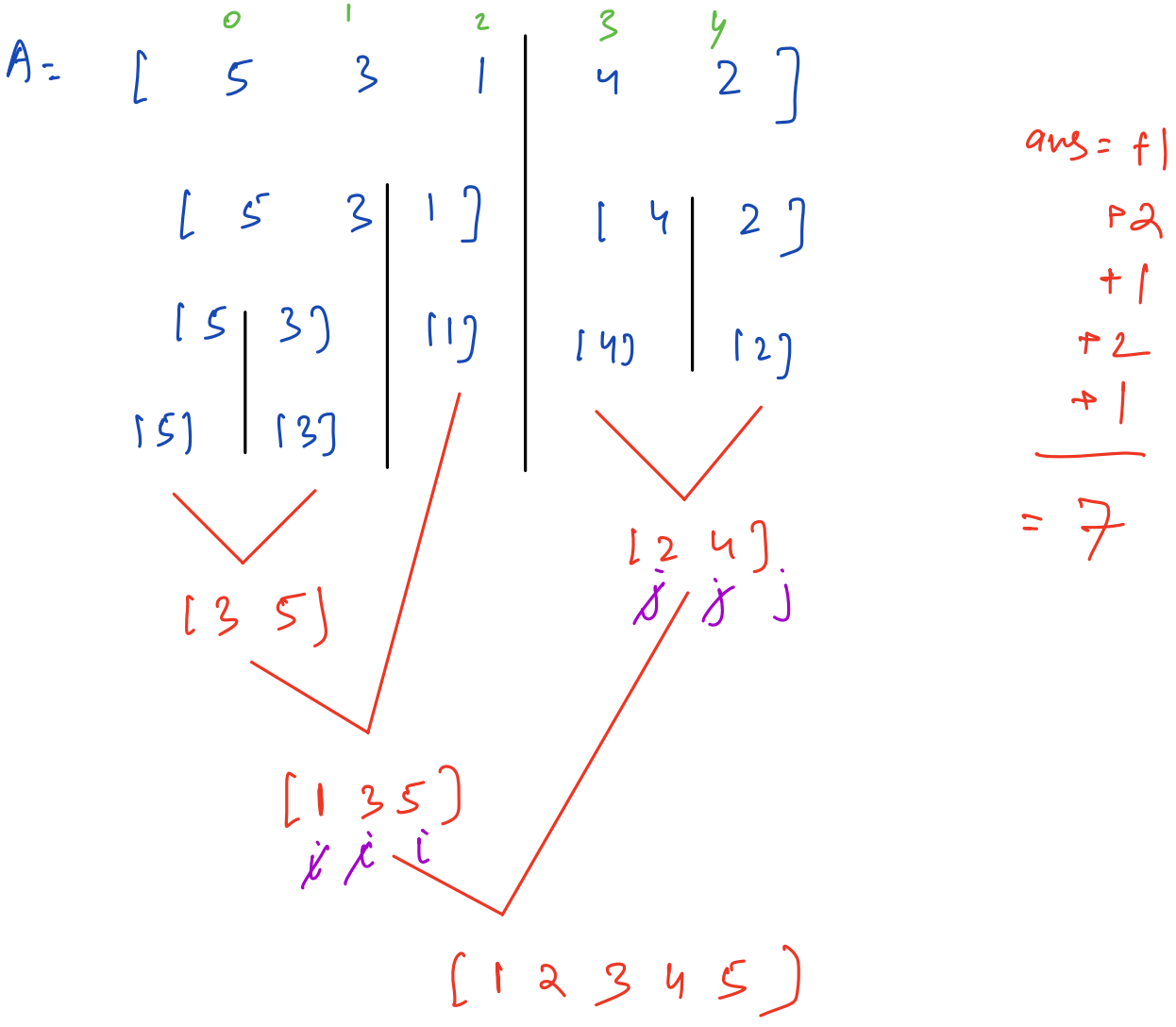
Bruteforce ⇒ $TC = O(N^2)$

$SC = O(1)$

A = [5 3 1 4 2]

ans = 0 + 1 + 2 + 1 + 3
= 7

i < j && A[i] > A[j] ⟶ count # of greater elements in left



ans = f1
+2
+1
+2
+1
= 7

if selecting right side element while merging

ans += # remaining elements in left

TC = O(N log N) SC = O(N)

```
   0  1  2  3
[  5  2  6  1  ]
   ↓  ↓  ↓  ↓
   0  1  0  3      => 1+3 = 4
```

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

```
   0  1  2  3  4
[  5  3  1  4  2  ]      ans = 7
```

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

[l, mid]
[mid+1, r]

int[] merge(C[], l, mid, r) {

N = mid - l + 1 , M = r - (mid+1) + 1 = r - mid

A[0...N-1] = C[l .... mid]
B[0...M-1] = C[mid+1 .... r]

} create A[] & B[]

i = 0 , j = 0 , k = 0

while(i < N && j < M) {

if(A[i] <= B[j]) { → preserves stable order

C[k] = A[i]

i++

}

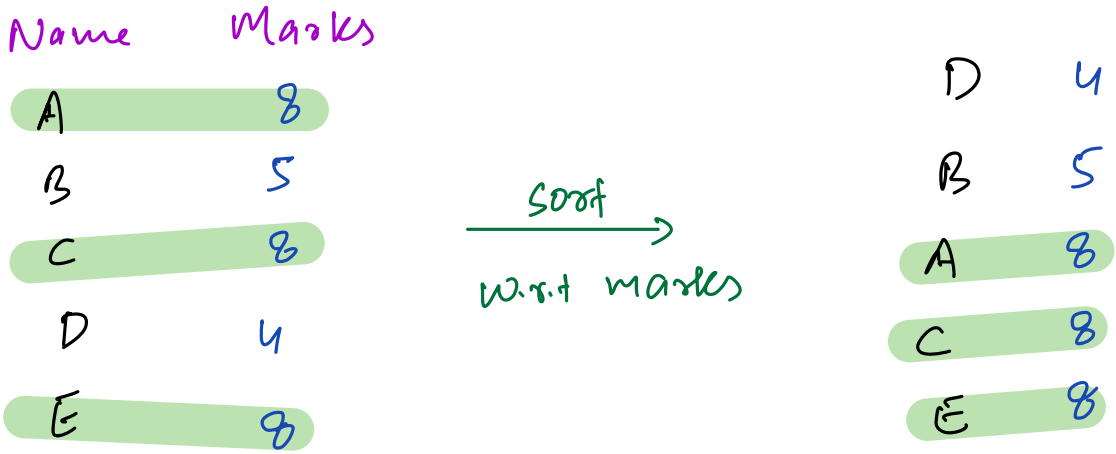
else { // inversion

```
            C[K] = B[j]
            j++
            count += N - i
        }
    K++
}
while (i<N){
    C[K] = A[i]
    K++, i++
}
while ( j<M) {
    C[K] = B[j]
    K++, j++
}
```

Stable sorting → Relative order of equal elements should not change which sorting w.r.t a parameter.

A = [6 5 3 5]

↳ [3 5 5 6]



Inplace sorting

if no extra space is needed to sort, it is called inplace sorting.

if $SC = O(1)$ $\Rightarrow$ inplace