Count & Merge Sort

AGENDA:

- Count Sort
- Merge two sorted arrays.
- Merge Sort
- Inversion Count.
- Stable / Inplace

Current PSP

67% ⟶ 70%

Rules

⟶ Qs in question tab

⟶ Answers in private

Count sort

Find the smallest no. that can be formed by re-arranging the digits of given no. in an array

Return the smallest no. in the form of an array.

A[i] → [0,9]

A = 6 3 4 2 7 2 1

ans = 1 2 2 3 4 6 7

A = 4 2 7 3 9 0

ans = 0 2 3 4 7 9

Brute force sort the array TC: O(N log N)

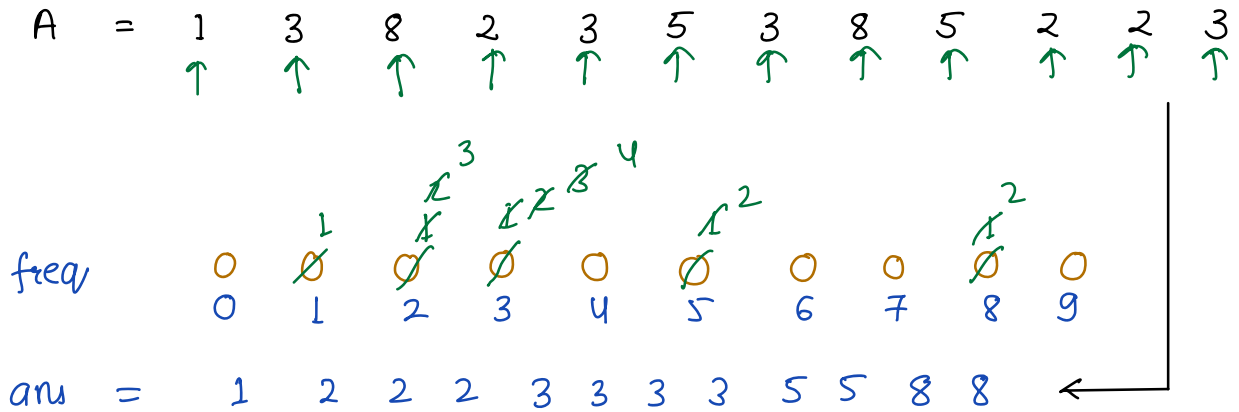
A 0 1 2 9 5 1 4 0 6 .....

ans = 0 00 ..... 1 ..... 2 .... 8 .... 9 ....

freq of 0 freq of 1 ..... freq 8 freq 9

Idea ⟶ use freq array to sort A.

HM as well



Pseudocode

```
// calculate freq
freq [10]   // ∵ digits ⟶ [0, 9]

for i ⟶ 0 to N-1 {
    freq [A[i]] ++
}
```

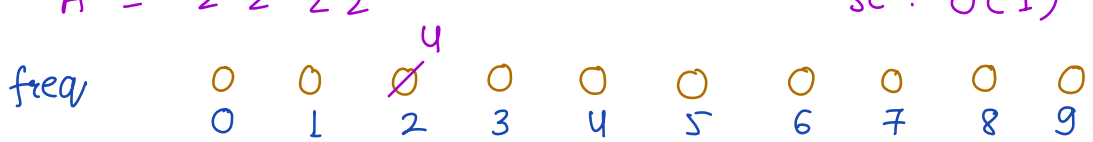
TC : O(N)

```
for digit ⟶ 0 to 9 {
    times = freq [digit]
    for i ⟶ 1 to times {
        print (digit)
    }
}
```

TC : O(N)

Overall TC : O(N)

SC : O(1)



Q> Will count sort work if A[i] is around 10^9

$[0, 10^9]$

wont work for large A[i] values.

∵ we cannot create freq $[10^9]$

$4 * 10^9$

4GB

HM will not work for $10^{\wedge 9}$ scenario think why ?

---

can count sort work on negative no. ?

A = -2 , 3 , 8 , 3 , -2 , 3

Yes count sort can work for negative value

we need to shift the range to non-negative values.

A = -2 , 3 , 8 , 3 , -2 , 3 $\quad$ Ai $\longrightarrow [-9, 9]$

+2 +2 +2 +2 +2 +2

0 5 10 5 0 5

freq

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

$\Rightarrow$

0 0 5 5 5 10

-2 -2 -2 -2 -2 -2

-2 -2 3 3 3 8

A = -2 -3 3 5 9 -9 -2 -3

1

make the smallest value pointing to index 0

A = -2 -3 3 5 9 -9 -2 -3

+9 +9 +9 +9 +9 +9 +9 +9

7 6 12 14 18 0 7 6

Apply count sort here and convert back the output by subtracting 9

freq [max − min + 1]

Approach 2 ⟶ Use hashmap and no need for any conversion of input & output.

A = -2 3 8 3 -2 3

HM = -2 : 2
3 : 3
8 : 1

} Create freq HM

```
for val ⟶ min val to max val {
    times = hm.getOrDefault(val, 0)
    for i ⟶ 1 to times {
        print(val)
    }
}
```

NOTE : Only we count sort if $max - min + 1 <= 10^6$

Always avoid count sort and use in-built sort.

Merge two sorted arrays

Given an integer array where all odd elements are sorted and all even elements are sorted.
Sort the entire array { better than nlogn }

A = 2 5 4 8 11 13 10 15 21

A

2 4 8 10

i i i i i

B

5 11 13 15 21

j j

C = 2 4 5 8 10 11 13 15 21

Pseudocode

```
// step 1  segregate even into A[]
                   odd into B[]

int[] merge ( A[N] , B[M] ) {
    C [N+M]
    i = 0   // A
    j = 0   // B
    k = 0   // C

    while ( i < N && j < M) {
        if (A[i] <= B[j] ) {
            C[k] = A[i]
```

```
            i++
            k++
        } else {
            C[k] = B[j]
            j++
            k++
        }
    }

    // while A or B contains elements put them
       in C
    while (i < N) {
        C[k] = A[i]
        i++
        k++
    }
    while (j < M) {
        C[k] = B[j]
        j++
        k++
    }

    return C
}
```

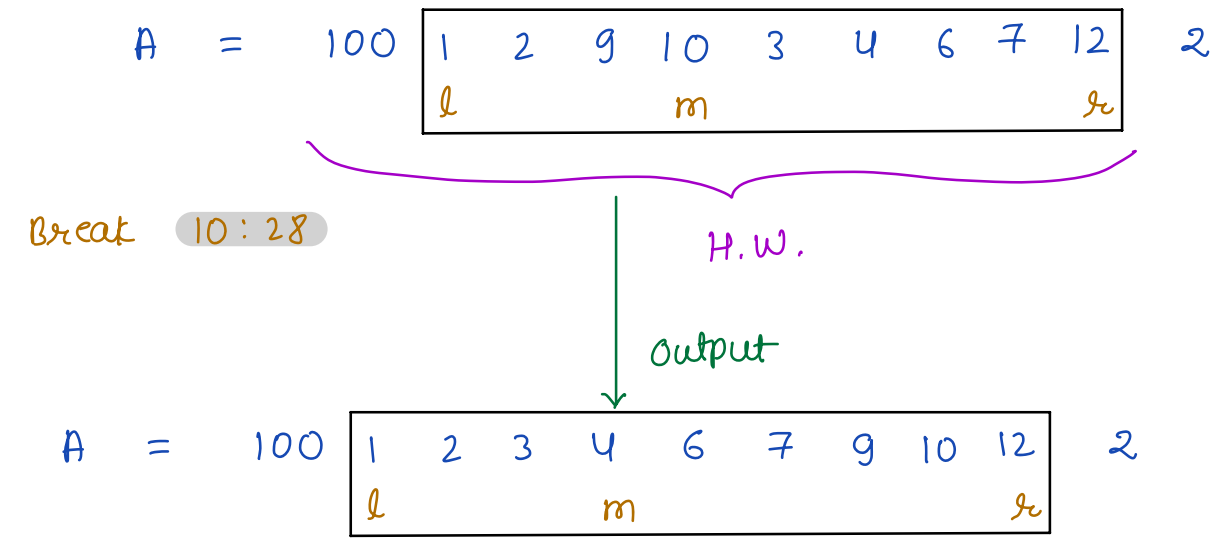
TC : O(N+M)

SC : O(N+M)

---

void merge (A[N], l, m, r)

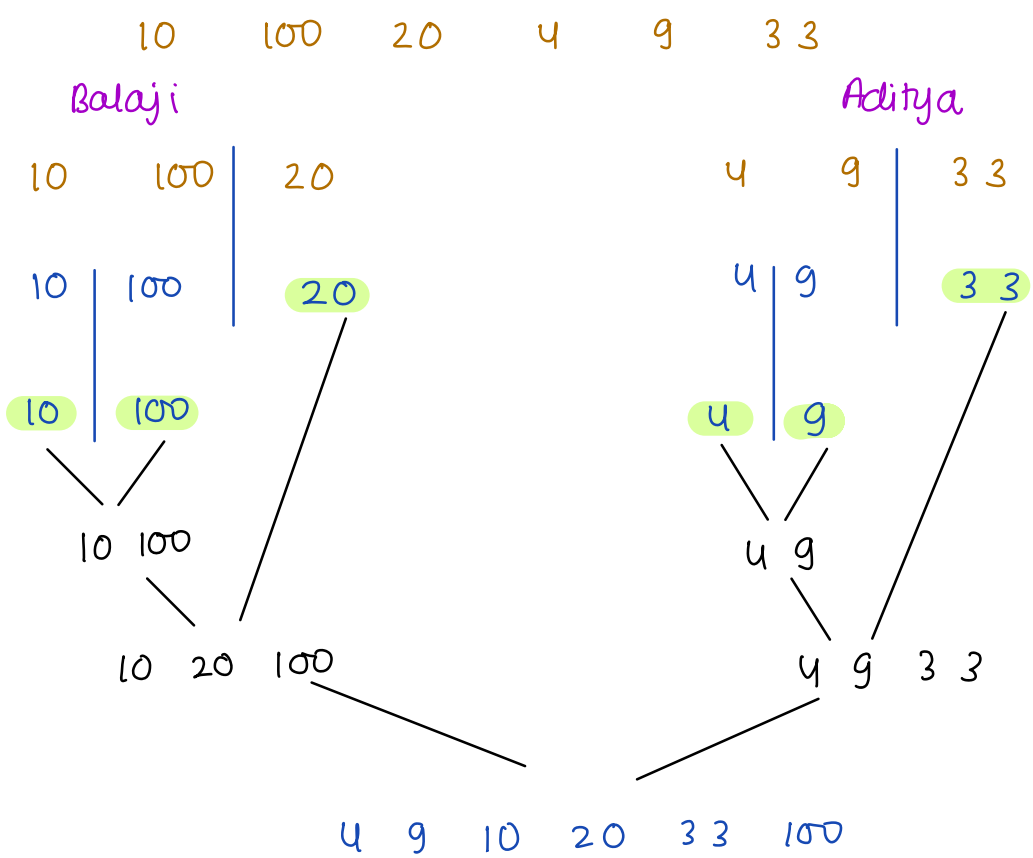
merge A[] from l to m (sorted) and m+1 to r (sorted).



Merge Sort { divide and conquer }

Sujoy { Profenor }

{ sort exam papers }



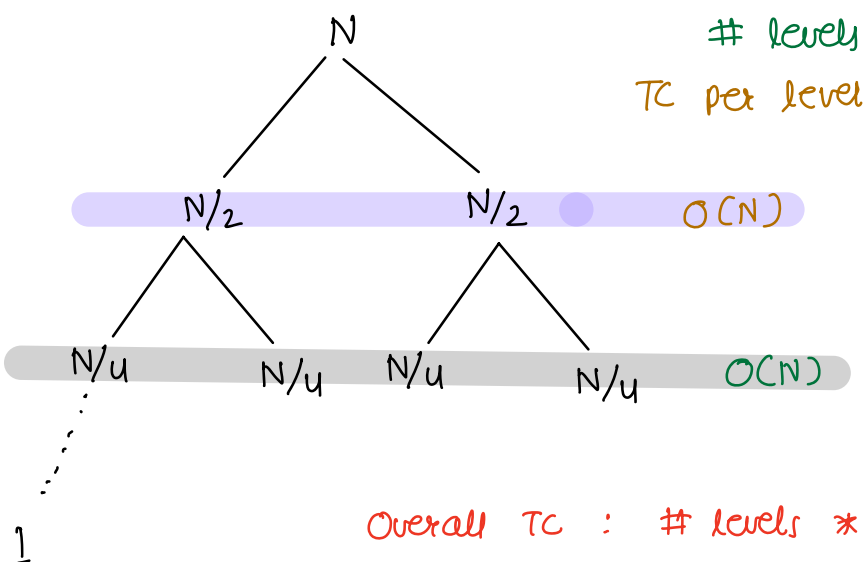
Pseudocode

```
// sort A[] from index l to index r
void mergeSort(A[], l, r) {
    // Base condition
    if (l == r) return

    m = (l + r)/2

    mergeSort(A[], l, m)
    mergeSort(A[], m+1, r)

    merge(A[], l, m, r)  // merges two sorted array l to m
                         // m+1 to r
}
```



levels = log N

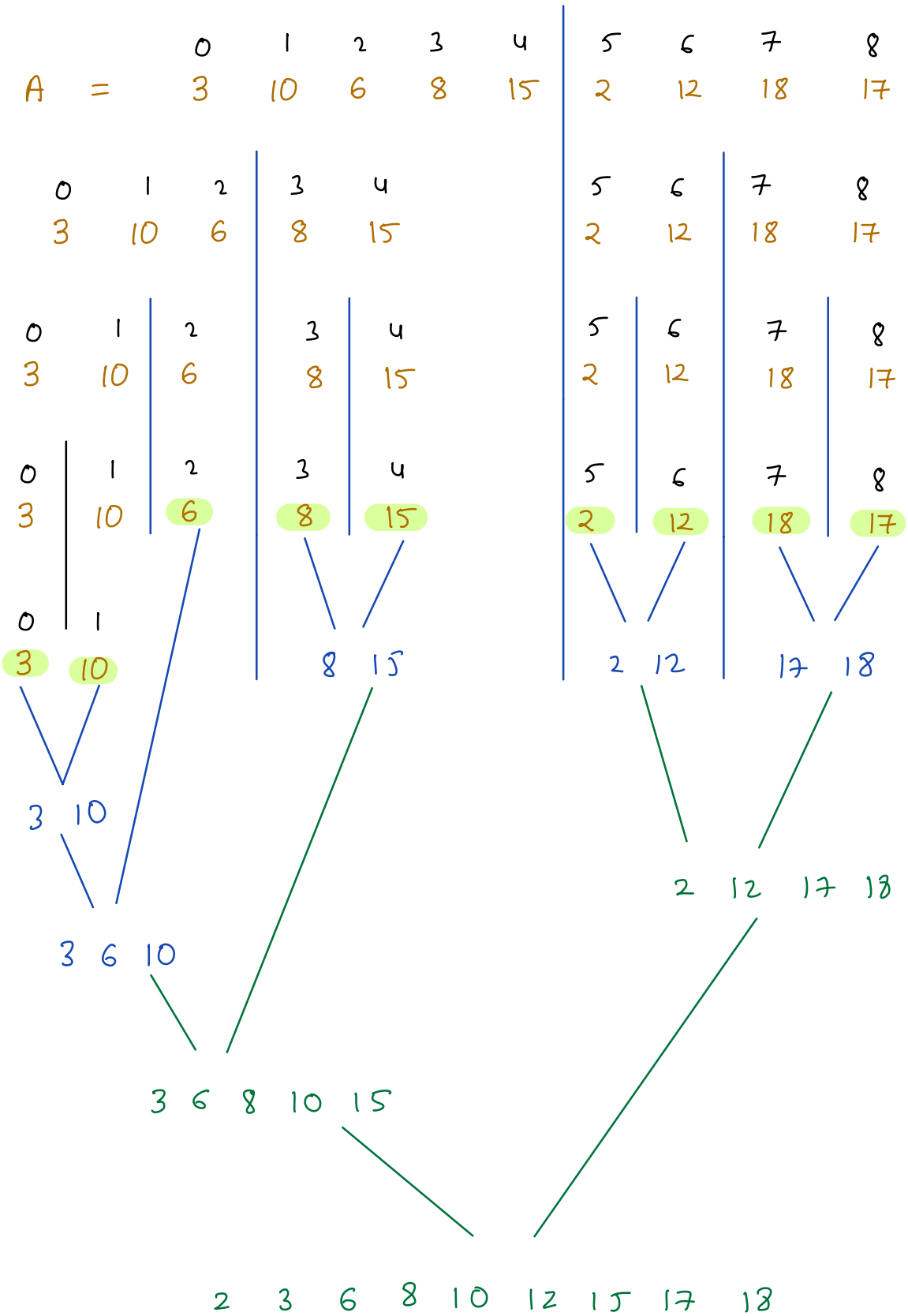
TC per level = O(N)

Overall TC : # levels * TC per level

= O(N * log N)

SC : O(N)

↓ merge function


A = 3 10 6 8 15 2 12 18 17
0 1 2 3 4 5 6 7 8
3 10 6 8 15 2 12 18 17
0 1 2 3 4 5 6 7 8
3 10 6 8 15 2 12 18 17
0 1 2 3 4 5 6 7 8
3 10 6 8 15 2 12 18 17
0 1 2 3 4 5 6 7 8
3 10 6 8 15 2 12 18 17
0 1 3 10 8 15 2 12 17 18
3 10 2 12 17 18
3 6 10
3 6 8 10 15
2 3 6 8 10 12 15 17 18

Calculate no. of pairs such that A[i] > B[j]

Given two array, A[n] & B[m]
Calculate the no. of pairs i, j such that A[i] > B[j]

A = 7 3 5
B = 2 0 6

(7 2) (7 0) (7 6) (3 2) (3 0) (5 2) (5 0)

7 pairs

Bruteforce

```
count = 0
for i ——→ 0 to N-1 {
    for j ——→ 0 to M-1 {
        if ( A[i] > B[j] )  count ++
    }
}
```

A = 7 3 5
B = 2 0 6

sort A & B

A = 3 5 7
    i
B = 0 2 6
    j

any = +3
      +3
      +1

Pseudocode

```
count = 0
i = 0
j = 0
sort array A & B

while (i < N && j < M) {
    if (A[i] > B[j]) {
        count += N - i
        j++
    }
    else {
        i++
    }
}
print(count)
```

TC : $O(N \log N + M \log M + N+M)$

SC : $O(1)$

Inversion Count

Given A[n], calculate no. of pairs (i, j) such that

$i < j$ && $A[i] > A[j]$

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

i	j	Ai	Aj	Ai > Aj
0	1	10	3	✓
0	2	10	8	✓
0	3	10	15	X
0	4	10	6	✓
1	2	3	8	X
1	3	3	15	X
1	4	3	6	X
2	3	8	15	X
2	4	8	6	✓
3	4	15	6	✓

ans = 5

Bruteforce

TC : $O(N^2)$

```
count = 0
for i ——→ 0 to N-1 {
    for j ——→ i+1 to N-1 {
        if ( A[i] > A[j] )   count++
    }
}
```

	0	1	2	3
A =	5	2	6	1

$i < j$ && $A_i > A_j$

5 2 2 1 6 1 ans = 4

5 1

	0	1	2	3	4
A =	5	3	1	4	2

5 3 3 1 4 2 ans = 7

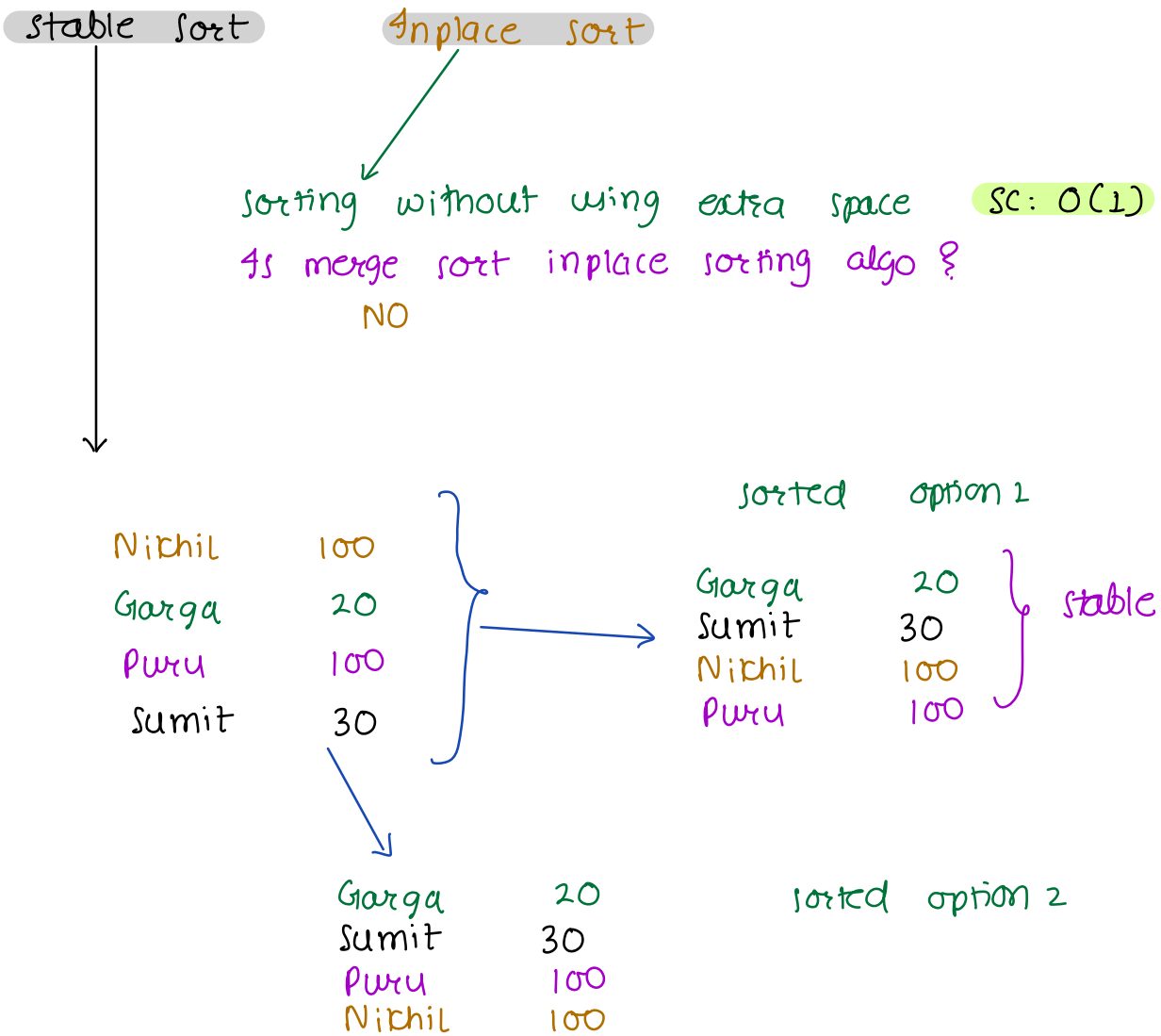
5 1 3 2

5 4

5 2

Hint : Exactly same as merge sort.
During the merge function use logic from previous question to count inversions.

H.W.



If the relative order b/w input and output is maintained for same marks

Eg. Nikhil & puru

HW ⟶ figure out if merge sort is stable or not.

Doubt session

A = 2 2 2 2

0 ~~1~~ ~~2~~ ~~3~~ 4

freq	0	1	2	3	4	5	6	7	8	9

TC : O (10 + N