1) Count Sort
2) Merge Sort

Q) find the smallest no. that can be formed by rearranging the digits of no. given in the form of an array.

Return in form of array

Eg) A = {6, 3, 4, 2, 7, 2, 1} = 6342721

O/P = {1, 2, 2, 3, 4, 6, 7} = 1223467

Sol^n 1) Sort the array & return it.

T.C. = O(N log N)

⇓ O(N) ??

Sol^n 2)

OBS : $0 \le A[i] \le 9$

Format of O/P = {0,0..., 1,1..., 2,2..., 3...., 4, ... 99}

i i i i i i i i i i i i i

I/P : {4, 2, 3, 2, 9, 0, 1, 6, 0, 1, 2, 5}

⇒ Store the freq of all elements

freq[10]: [2, 2, 3, 1, 1, 1, 1, 0, 0, 1]
0, 1, 2, 3, 4, 5, 6, 7, 8, 9
i i i i i i i i i i i

S.C. = O(10) = O(1)

i =

0 1 2 3 4 5 6 7 8 9 10 11
I/P : {0, 2, 3, 2, 1, 0, 2, 1, 3, 0, 1, 2, 5}
0 1 1 2 2 3 4 5 6
j j j j j j j j j j j j j

"Count Sort"

Code

```
int freq[10] = {0};

for( i=0; i<N; i++){
        freq[A[i]]++;
}                                   } O(N)

int index = 0;

for (i=0; i<10; i++){
```

```
for (j = 0; j < freq[i]; j++) {
        A[index] = i;
        index++;
    }
}
```

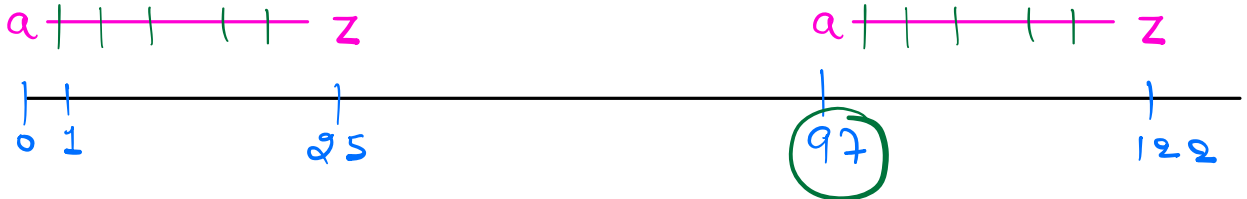
T.C. = O(N)
S.C. = O(1)

Q. will Count Sort work for large values

A[i] > 10^9 ??

Count sort can typically be applied for A[i] ≤ 10^6

Q. How to implement count sort for Ex. having -ve no's as well.



$a = 97 - 97 = 0$
$b = 98 - 97 = 1$
$c = 99 - 97 = 2$
:
$z = 122 - 97 = 25$

Approach

1) find the range of elements (min & mex)

the freq of any element X will be stored on the index (X − min)

Size of freq array = [min, mex]

⇓

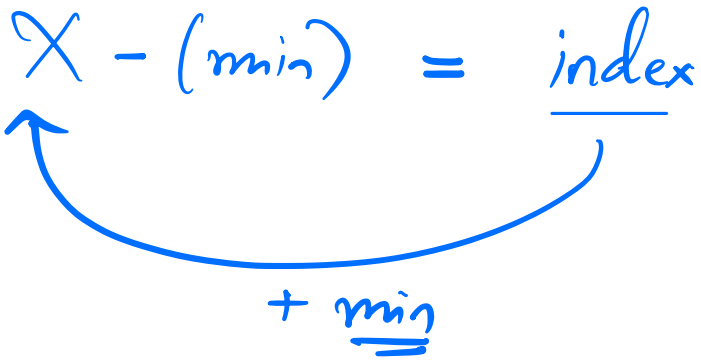
elements = mex − min + 1

Eg $A = [\underset{0}{-2}, \underset{5}{3}, \underset{10}{8}, \underset{5}{3}, \underset{0}{-2}, \underset{5}{3}, \underset{0}{0}, \underset{2}{4}]$ (indices 0 1 2 3 4 5 6 7)

min = −2
mex = 8
> Size = (8 − (−2) + 1) = 11

Freq = [2, 0, 1, 0, 0, 3, 1, 0, 0, 0, 1] (indices 0 1 2 3 4 5 6 7 8 9 10)

(−2, −1, 0, 1, 2, 3, 4, 5, 6, 7, 8)



Given an integer array where the odd valued elements are relatively sorted & even valued elements are relatively sorted.

Sort the entire array.

0	1	2	3	4	5	6	7	8
2,	5,	4,	8,	11,	13,	10,	15,	21

A = [2, 5, 4, 8, 11, 13, 10, 15, 21]

Soln 1) Use the default sorting algo.

T.C. = O(N log N)

↓ O(N) ??

Idea: Split the array into 2 arrays

Even [E] ≠ Odd [O]

Count of Even elements

Count of odd elements

	0	1	2	3	4	5	6	7	8	9
A =	2	5	4	8	11	13	10	14	15	21

	0	1	2	3	4
Odd[5] =	5	11	13	15	21

	0	1	2	3	4
Even[5] =	2	4	8	10	14

} O(N)

	0	1	2	3	4	5	6	7	8	9
A =	2	4	5	8	10	11	13	14	15	21

Code

1) Split s/w Odd & Even

2) Merge back into A.

```
int Od = 0;
for (i=0; i<N; i++) {
        if (A[i] % 2 != 0) {
                Od++;
        }
}
int E = N - Od;

// split
ind_even = 0;
ind_odd = 0;
Even [E];
Odd [Od];

for (i=0; i<N; i++) {

        if (A[i] % 2 == 0) {
                Even [ind_even] = A[i];
                ind_even++;
        } else {
                Odd [ind_odd] = A[i];
                ind_odd++;
        }
}

// Merge
i = j = 0;
```

```
ind = 0;
while ((i < Od) && (j < E)) {
    if (Odd[i] < Even[j]) {
        A[ind] = Odd[i];
        i++;
        ind++;
    }
    else {
        A[ind] = Even[j];
        j++;
        ind++;
    }
}

if (i < Od) {
    while (i < Od) {
        A[ind] = Odd[i];
        i++;
        ind++;
    }
} else {
    while (j < E) {
        A[ind] = Even[j];
        j++;
        ind++;
    }
```

b

b

---

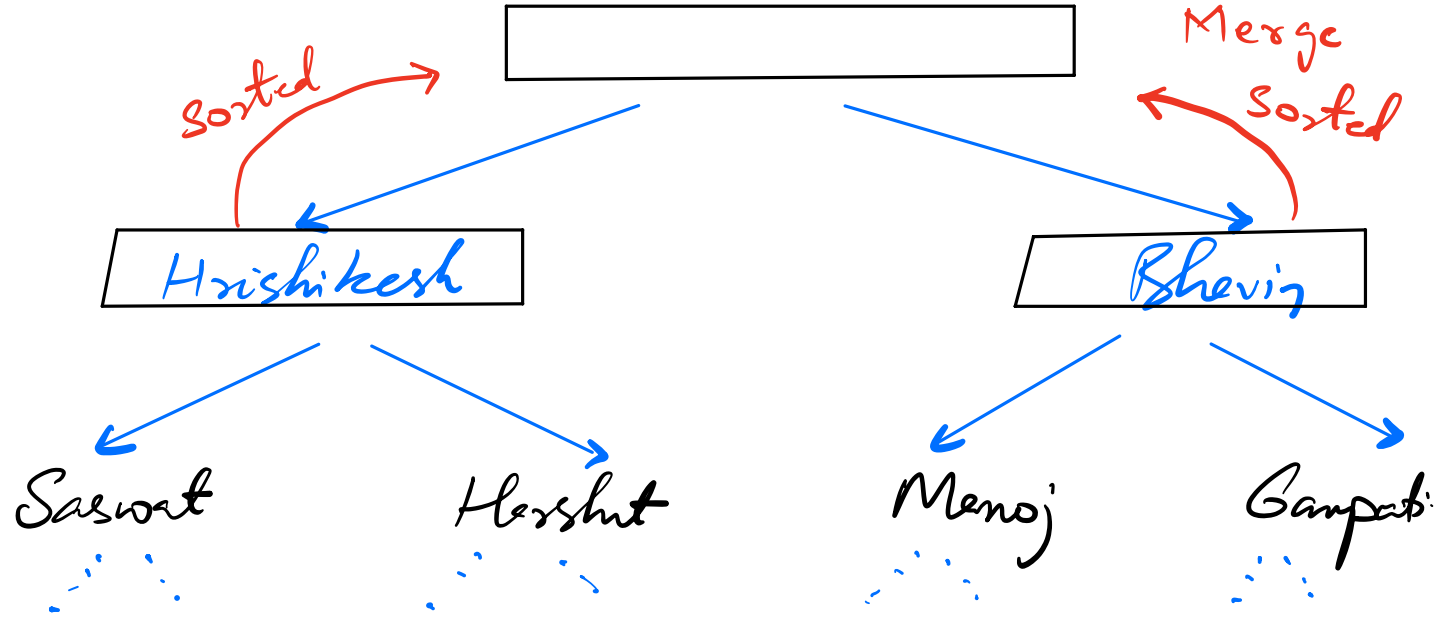
iteration = 2 × N.

T.C. = O(N)
S.C. = O(N)

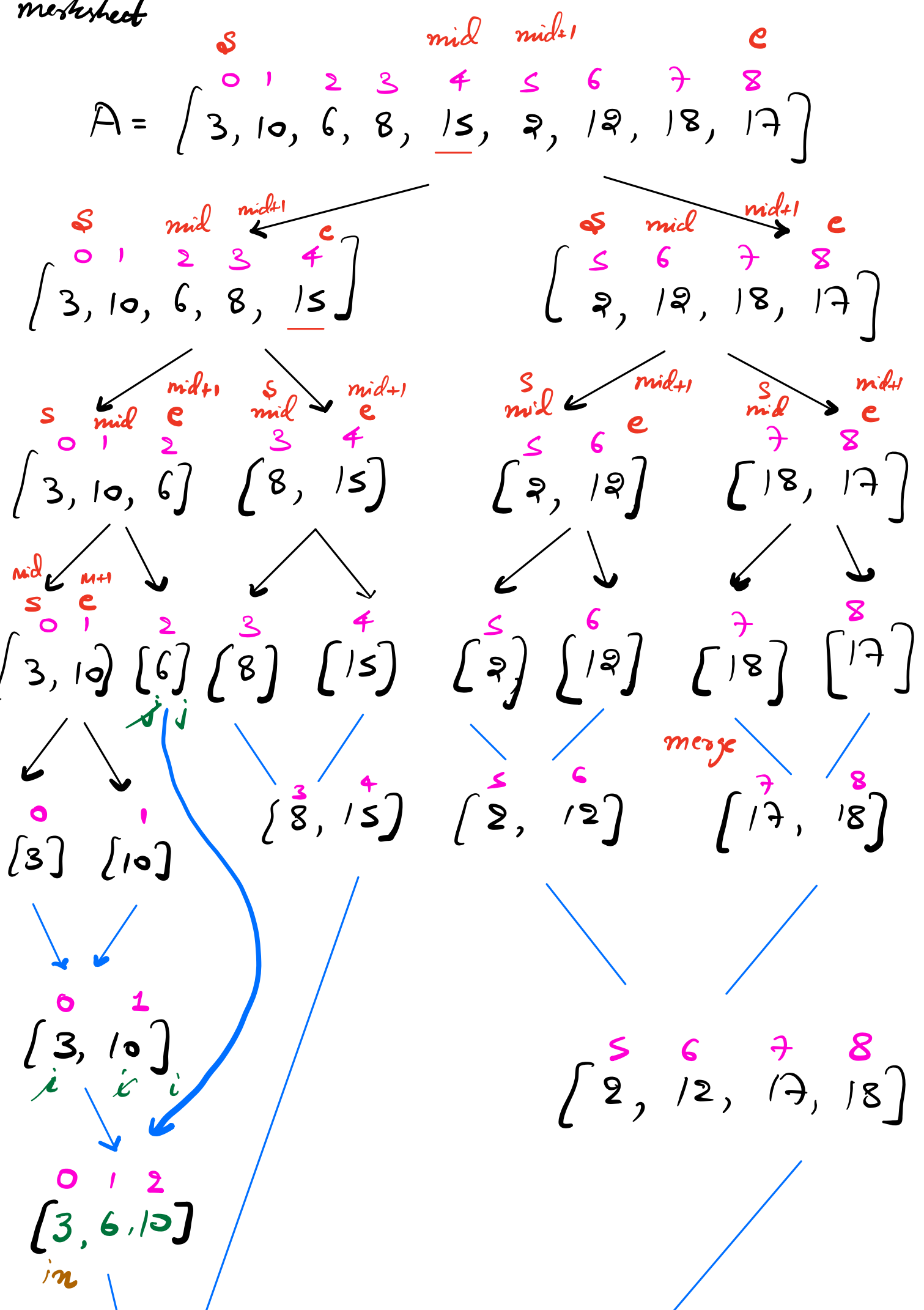
Merge Sort

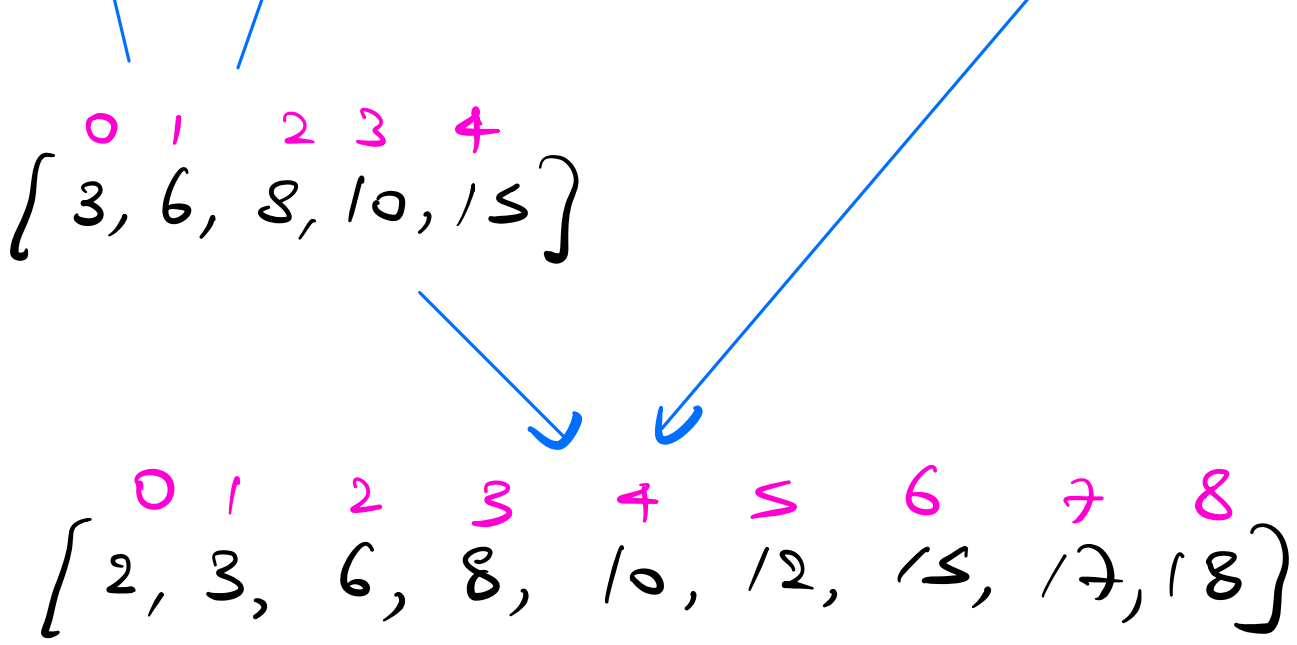
Teacher ⇒ Mark Sheet of all students random order

Goal: Arrange in sorting order



One


merge



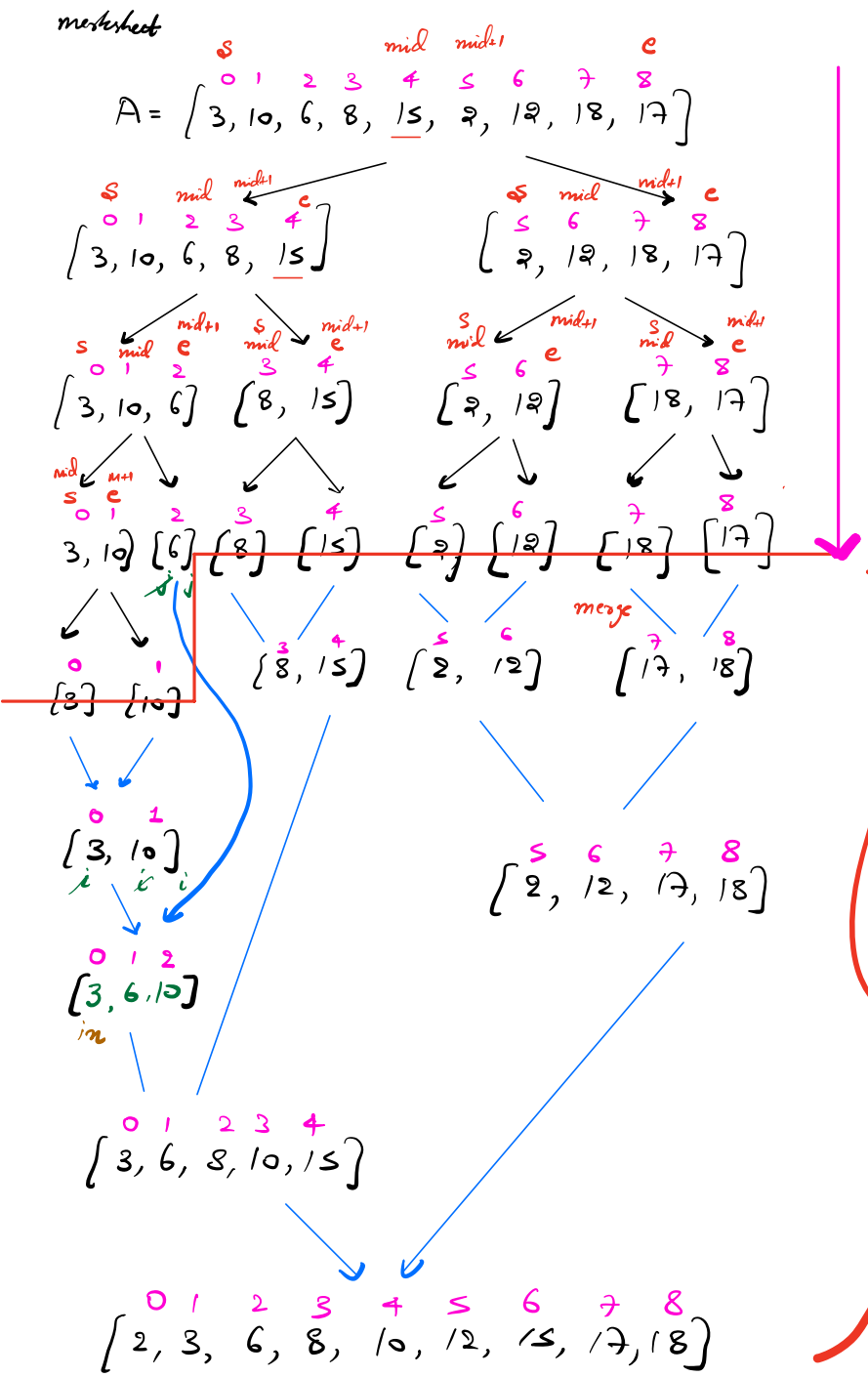
Code

```
                                    0      N-1
                                    ↑      ↑
void mergeSort (A[], s, e) {
    if (s >= e) { return; }
    int mid = (s+e)/2   ( s + (e-s)/2 )
    mergeSort (A, s, mid);      A[s, mid] is sorted
    mergeSort (A, mid+1, e);    A[mid+1, e] is sorted
    merge (A, s, mid, e); //O(N)
}
```

```
void merge (A, s, mid, e) {
    int n1 = mid - s + 1;        [s, mid]
    int n2 = e - mid;            [mid+1, e]

    int A1[n1];
    int A2[n2];
    int ind = 0;

    for (i = s; i <= mid; i++) {
        A1[ind] = A[i];
        ind++;
    }
    ind = 0;
    for (i = mid+1; i <= e; i++) {
        A2[ind] = A[i];
        ind++;
    }

// Merge

ind = s;
i = j = 0;

while (i < n1 && j < n2) {
```

```
        if (A1[i] <= A2[j]) {
            A[ind] = A1[i];
            i++;
            ind++;
        }
        else {
            A[ind] = A2[j];
            j++;
            ind++;
        }
    }
    if (i < n1) {
        while (i < n1) {
            A[ind] = A1[i];
            i++;
            ind++;
        }
    } else {
        while (j < n2) {
            A[ind] = A2[j];
            j++;
            ind++;
        }
    }

}
```

T.C. =



$O(\log N + N \times \log N)$

T.C. = $O(N \log N)$

S.C. = $O(N)$

{ Divide & Conquer }

$$T(N) = 2T\left(\frac{N}{2}\right) + N$$

Assign ⇒ Inversion Count

1) Brute force
2) Dry
3) Hint

$$A = [\ 2, 3, 4, 5, 7, 2, 3, 8, 11, 13, 10, 15, 14, 21\ ]$$

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

i i j