Agenda

1. Smallest Number
2. Merge 2 sorted arrays
3. Merge Sort
4. ~~Sort 0-1 → (if time permits)~~

Count Sort

$0 \le arr[i] \le 9$

< Question > : Find the smallest number that can be formed by rearranging the digits of the given number in an array. Return the smallest number in the form of an array.

arr[ ] → [6 3 4 2 7 2 1] ⇒ [1 2 2 3 4 6 7]

arr[ ] → [4 2 7 3 9 0] ⇒ [0 2 3 4 7 9]

Idea -1 use inbuilt sort function.

T.C → O(NlogN)
S.C → O(depends on the algo)

Idea -2

arr[ ] → [$\overset{✓}{9}$ $\overset{✓}{1}$ $\overset{✓}{2}$ $\overset{✓}{5}$ $\overset{✓}{4}$ $\overset{✓}{2}$ $\overset{✓}{1}$ $\overset{✓}{2}$ $\overset{✓}{5}$ $\overset{✓}{8}$]

farr →	0	2	3	0	1	2	0	0	1	1
	0	1	2	3	4	5	6	7	8	9

arr[] → [1 1 2 2 2 4 5 5 8 9]

</> Code

```
int farr[10];

for( i=0; i<N; i++){
    val = arr[i];
    farr[val] ++;
}

K=0;
for( i=0; i<=9; i++){
    for( j=0; j< farr[i]; j++){
        arr[K] = i;
        K++;
    }
}
```

i	j	no. of iterations
i=0	-	0
i=1	[0,1]	2
i=2	[0,2]	3
i=3	-	0
i=4	[0,1)	1
i=5	(0,1)	2
i=6	-	0
i=7	-	0
i=8	[0,0]	1
i=9	[9,0)	1

$$\boxed{\begin{array}{l} T.C \to O(N) \\ S.C \to O(1) \end{array}}$$

N+10

$0 \le A[i] \le 10^9$

↳ farr[10^9] X

Count sort will only work if range is upto 10^6. => farr[10^6]

Think of applying count sort when range is very small.

How to handle negative numbers?

arr[] → (5, 2, -3, 4, -2, 9, -1, 2]

min → -3

max → 9

range → max - min + 1

farr →	1	1	1	0	0	2	0	1	1	0	0	0	1
	0	1	2	3	4	5	6	7	8	9	10	11	12
	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
	-3	-2	-1	0	1	2	3	4	5	6	7	8	9

</> Code

```
min = arr[0] ,    max = arr[0]
for( i = 1; i < N; i++){
        min =   Math.min( min, arr[i]);
        max =   Math.max( max, arr[i]);
}

int   farr [max-min+1];

for( i= 0; i < N ; i++){
        val = arr[i];
        farr[ val - min] ++;
}

K=0;
for( i= min; i ≤ max; i++){
    for( j=0; j < farr[i-min] ; j++){
            arr[K] = i;
            K++;
    }
}
```

R → range.
→ max-min+1

T.C → O(N+R)
S.C → O(R)

Merge Two Sorted Arrays?

a[ ] → [2 4 7 8 12] → N ($-10^9 \leq$ element $\leq 10^9$)

b[ ] → [3 5 6 7] → M

res →

2	3	4	5	6	7	7	8	12
0	1	2	3	4	5	6	7	8

Idea -1 Create res[N+m];

Copy all elements in res[] from a[] and b[].

Sort the res[] using inbuilt sort function.

T.C → $O((N+m) \cdot \log(N+m))$

S.C → O(depends on sorting algo.)

Idea -2

i i i i

a[ ] → [2 4 7 8 12]

b[ ] → [3 5 6 7]

j

c[] → [2 3 4 5 6 7 7 8 12]

0 1 2 3 4 5 6 7 8

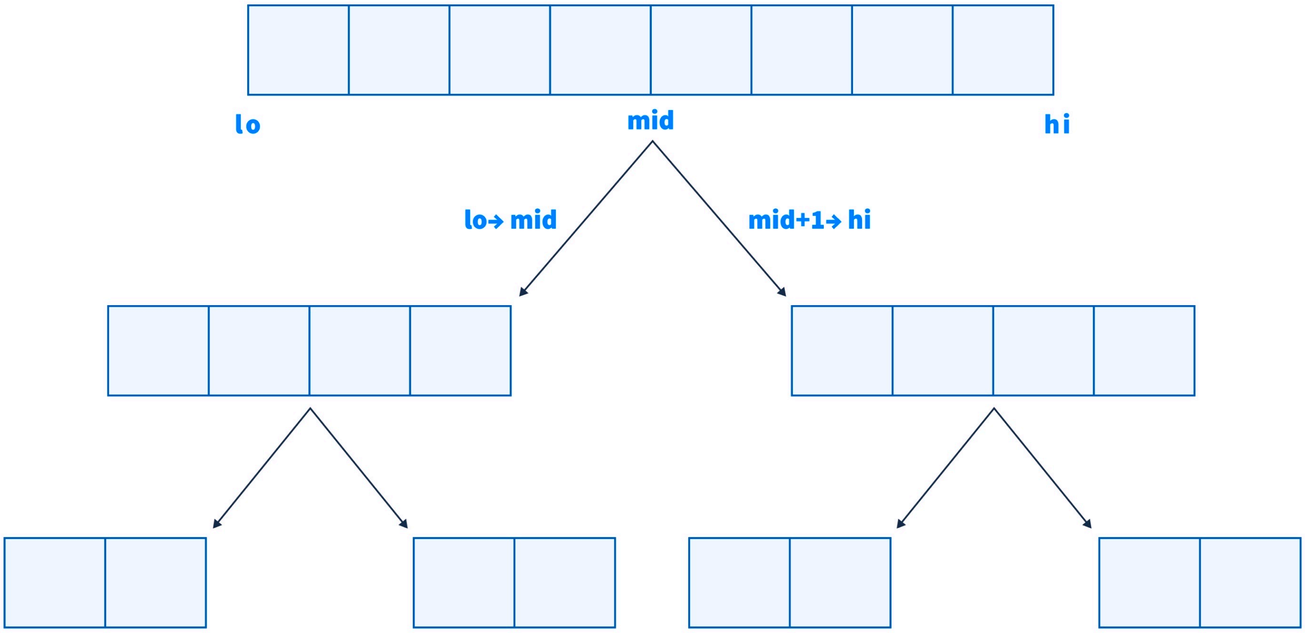
k

</> Code

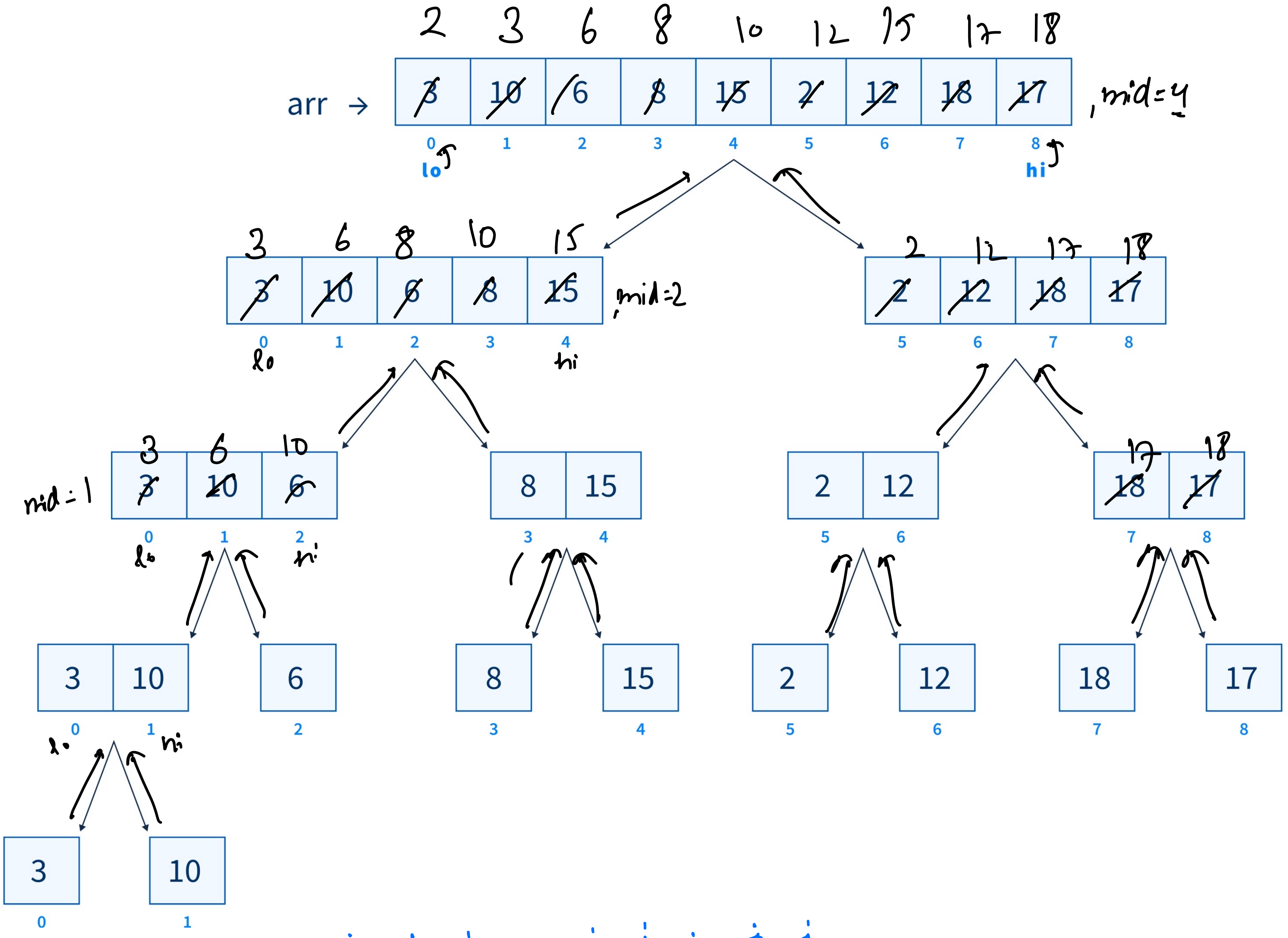
```
int c[N+m];

i=0, j=0

while( i<N && j<m){
        if(a[i] ≤ b[j]){
                c[k] = a[i];
                i++, k++;
        }
        else{
                c[k] = b[j];
                j++, k++;
        }
}

while( i<N){
        c[k] = a[i];
        i++, k++;
}

while( j<m){
        c[k] = b[j];
        j++, k++;
}

return c[];
```

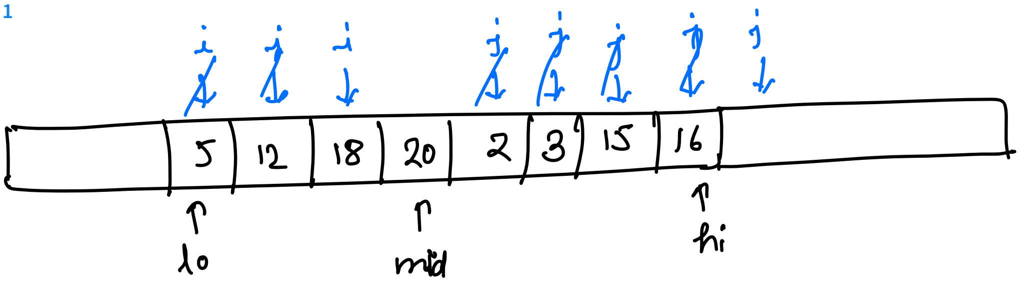
T.C → O(N+m)
S.C → O(1)

Merge Sort



```
void   mergeSort( int[] arr,  int lo,  int  hi){
|      if(lo = = hi) { return }
|      int   mid   =   (lo+hi)/2;
|      mergeSort( arr, lo , mid);
|      mergeSort( arr,  mid+1, hi);
|      merge2 sortedSubarrays ( arr, lo, mid, hi);  →  O(N),O(N)
}
```





a[] →

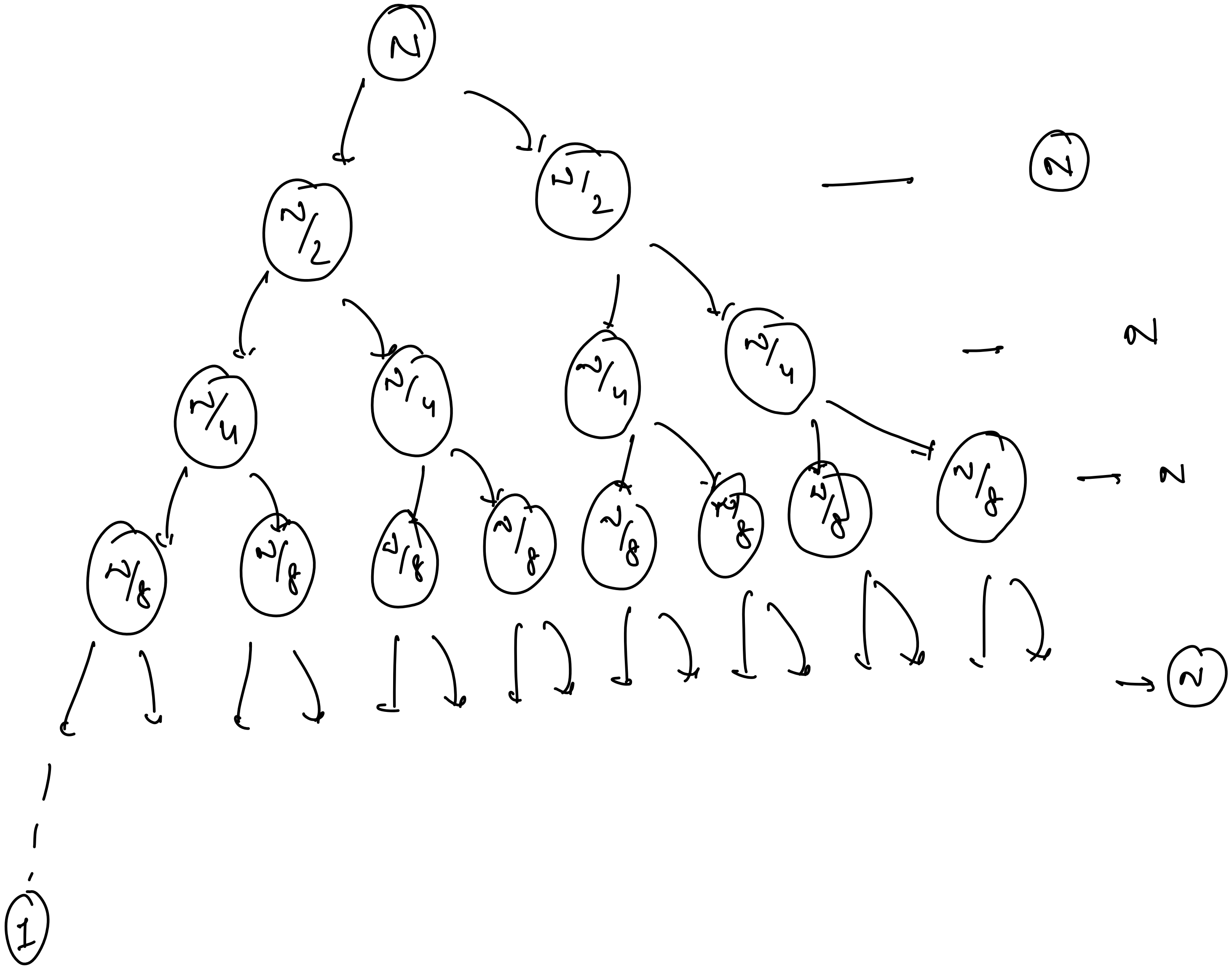
2	3	5	12	15	16	18	20
0	1	2	3	4	5	6	7

```
void merge2sortedSubarrays( int[] arr, int lo, int mid, int hi){
    int [] a = new int [hi-lo+1];
    int i = lo, j = mid+1, k = 0;

    while (i ≤ mid && j ≤ hi){
        if (arr[i] ≤ arr[j]){
            a[k] = arr[i];
            i++, k++;
        }
        else{
            a[k] = arr[j];
            j++, k++;
        }
    }

    while( i ≤ mid){
        a[k] = arr[i];
        i++, k++;
    }

    while( j ≤ hi){
        a[k] = arr[j];
        j++, k++;
    }

    for( i=0; i < a.length; i++){
        arr[i+lo] = a[i]
    }
}
```

T.C → O(N)

S.C → O(N)

Time Complexity Analysis [Merge Sort]



T.C → O(N log N)
S.C → O(N)

In-place

↳ Space complexity must be O(1).

Stable.

Naman → 75

Varun → 37

Priya → 90

Vikas → 37

Kapil → 75

$\xrightarrow[\text{inc. order of marks}]{\text{sort on}}$

Varun → 37

Vikas → 37

Naman → 75

Kapil → 75

Priya → 90

(stable)

$\xrightarrow{\text{sort}}$

Varun → 37

Vikas → 37

Kapil → 75

Naman → 75

Priya → 90

(Not stable)

⇒ Merge Sort is stable or not ? [H.W]

Scenerio

Google's Gmail offers an "All Inboxes" feature that allows users to view emails from multiple email accounts in one seamless interface. This is particularly useful for users managing personal and professional communications through separate accounts.

The feature ensures that emails from all accounts are merged into a single feed sorted by date and time, facilitating better email management and access.

Problem

Develop a function to emulate the "All Inboxes" feature of Gmail.

You are given two sorted arrays that represent timestamps of emails from two different email accounts. Each element in the array is an email object.

Your task is to merge these two arrays into a single list, ensuring that the resulting list is sorted by the timestamp, allowing the user to view emails in a chronological order from both accounts combined.

Example

Input:

ACCOUNT Email Times

A

[

1

,

5

$A[\,] \rightarrow [1, 5, 6, 9]$

$B[\,] \rightarrow (2, 4, 8)$

$\underline{A[i] - A[j] = -B}$

$\Downarrow$

$\boxed{A[j] - A[i] = B}$ $\underline{i \,!= j}$