- Form the smallest number from single digits
- Count Sort
- Merge two sorted arrays
- Merge Sort
- Stable & Inplace Sort

↳ Terms

Q

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

A[ ] = {6, 3, 4, 2, 7, 2, 1}

Answer: {1, 2, 2, 3, 4, 6, 7}

A[ ] = {4, 2, 7, 3, 9, 0}

Answer: {0, 2, 3, 4, 7, 9}

1) sort the digit array in ascending order

⇒ Arrays.sort(A) → $N \log N$

2) The digits in ~~a number~~ array can only range from 0 to 9

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

Frequency :

how many times 0 is coming — A[0]

1 — A[1]

2. — A[2]

index → key

A[index] → value ⇒

3.	A[3]
4.	A[4]
5.	A[5]
6.	A[6]
7.	A[7]
8.	A[8]
9.	A[9]

[4, 2, 7, 3, 9, 0]

0	1
2	1
3	1
4	1
7.	1
9	1

⇒

0	2	3	4	7	9

frequency info ⇒ array → size 10

0	1	2	3	4	5	6	7	8	9

index = keys

A[index] = values

A[9]

$A = \{ \overset{0}{6}, \overset{1}{3}, \overset{2}{4}, \overset{3}{2}, \overset{4}{7}, \overset{5}{2}, \overset{6}{1} \}$ 7

	0	1	2	3	4	5	6	7	8	9
F =	0	~~0~~1	~~0~~2	~~0~~1	1~~0~~	0	~~0~~1	~~0~~1	0	0

$F[A[\overset{i}{\cancel{0}}]]$ ++

	0	1	2	3	4	5	6	7	8	9
F =	0	~~0~~1	~~0~~2	~~0~~1	1~~0~~	0	~~0~~1	~~0~~1	0	0

ans = { 1, 2, 2, 3, 4, 6, 7 } 7

	0	1	2	3	4	5	6	7	8	9
F =	0	~~0~~1	~~0~~2	~~0~~1	1~~0~~	0	~~0~~1	~~0~~1	0	0

A.size = 7

ans =

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

smallest number

K

10 int
40B

int [10] F → F → 10

```
for ( i = 0        i < N        i++) {
        F [A[i]] ++
}
```

N

int [N] ans output

```
int K = 0
for ( d = 0        d <= 9        d++) {
        count = F[d]
        for ( i = 1        i <= count        i++) {
                ans [K] = d
                K++
        }
}
```

N

return ans

SC = 40 + 4×4 = 56B
constant = O(1)

TC = O(N)

2N

0	1	2	3	4	5	6	7	8	9
d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9	d_{10}

d	i	Total
0	d_1	d_1
1	d_2	d_2
2	d_3	d_3
3	d_4	$\vdots$
$\vdots$	$\vdots$	$\vdots$
9	d_{10}	d_{10}
		$\sum = N$

input array

counting

count sort] arrays

A[i] digits

constant & fixed range

0 to 9

digits

all numbers

A = [1 , 100 , 1000]

F = [0 1 2 3 . . . 100 . . 200 . . . 1000]

[1 1 1]

1001

wastage of space

d = 0 to d = 1000

low high

1001]

[1, 100, 1000, 1000000]

1	1
100	1
1000	1
10^6	1

⇒

smallest

↓

next smallest

↓

X

constant & fixed range

↓

static range ⇒ count sort.

Range is : -9 to 9 ✓

A[i] is digit

↓

-9 to 9

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

$A[i]$ ·····> $F[A[i]]$

0 – 9

$F[-9]$ ✗

-9

value		index
-9	·····>	0
-8		1
-7		2
-6		3
-5		4
-4		5
-3		6
-2		7
-1		8
0		9
1		10
2		11
3		12
4		13
5		14
6		15

index = value + 9

7 16

8 17

9 18

0 — 18 indicis

F[19]

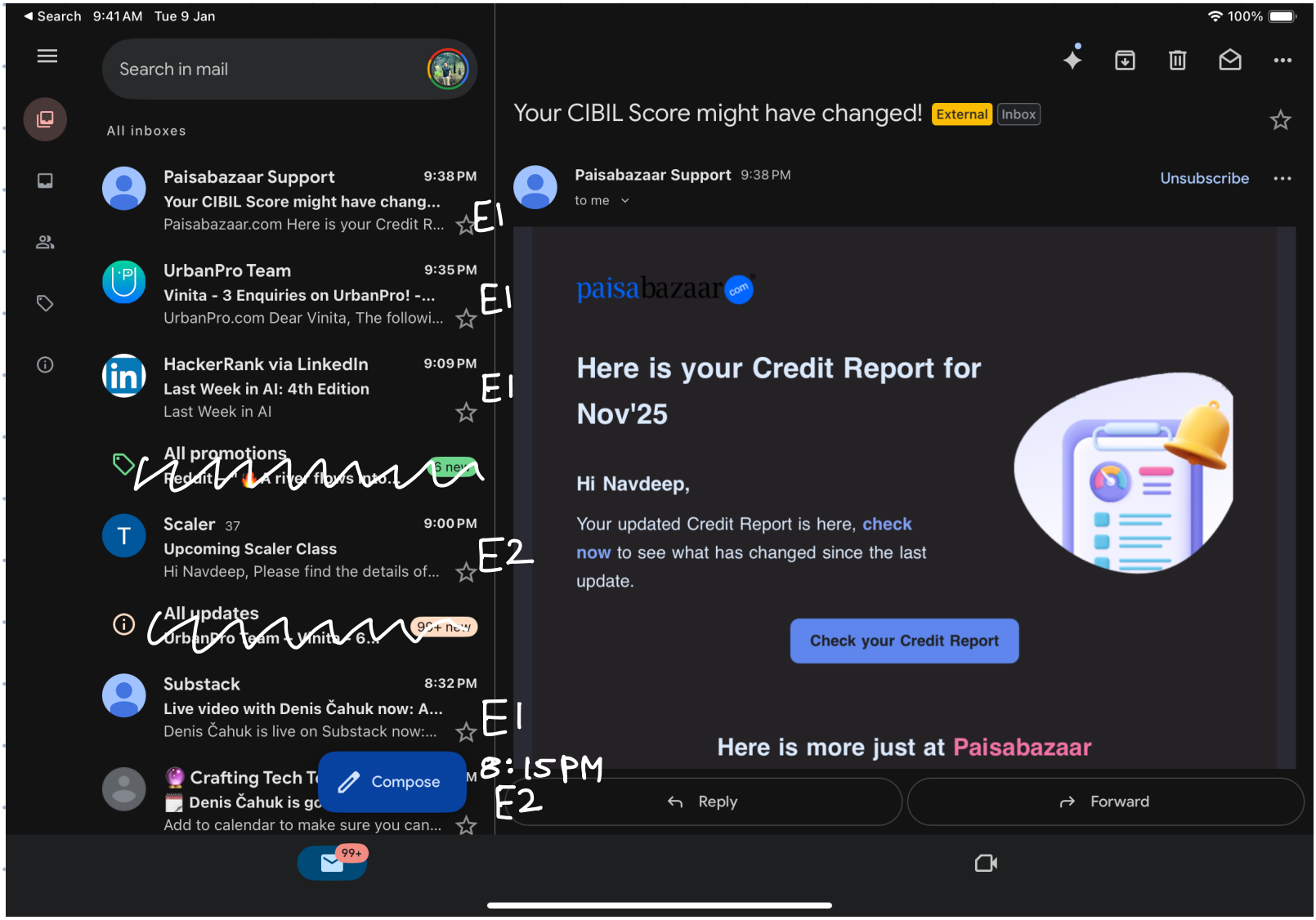
```
int F[19]

for ( i=0        i<N        i++) {
        index = A[i] + 9
        F[index]++
}
```

K=0

```
int [N] ans

for ( d = -9        d <= 9        d++) {
        index = d + 9
        count = F[index]
        for ( i=1        i <= count        i++) {
                ans[K] = d
                K++
}
}
return ans
```



sorted

E1	E1	E1	E1
8:32 PM	9:09 PM	9:35 PM	9:38PM

E2	E2
8:15PM	9:00PM

sorted

merge 2 sorted feed

merge 2 sorted arrays

A1 = [2, 4, 8, 11, 13] N

A2 = [3, 7, 8, 15] m

combined & sorted array → 2

[2 3 4 7 8 8 11 13 15]

ans

i

A1 = [2, 4, 8, 11, 13] N=5

A2 = [3, 7, 8, 10] m = 4

j

i = 0 j = 0

C

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

A1 A2 K

C = N+M

```
while ( i < N && j < M) {
    if (A1[i] < A2[j]) {
        C[K] = A1[i]
        K++
        i++
    } else if ( A1[i] >= A2[j]) {
        C[K] = A2[j]
        K++
        j++
    }
}
```

```
if (j == M) {
    // i
    while (i < N) {
        C[K] = A1[i]
        i++
        K++
    }
}
if (i == N) {
    // j
    while (j < M) {
        C[K] = A2[j]
        j++
        K++
    }
}
return C ⟶ C
```

Iterations => in every it →

either

A1[i] or A2[j]

N + M

both arrays are of same size

N

2N => O(N)

SC = O(1)

Q. Merge sort

$$A = \begin{array}{ccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \end{array}$$
$$A = [3, 10, 6, 8, 15, 2, 12, 18, 17]$$

```
int[] sort ( A, s, e ) {

}
```

$$P = \text{sort}(A, s, e)$$

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

$$m = \frac{s+e}{2}$$

$P = \text{sort}(A, s, e)$

$S1 = \text{sort}(A, s, m) = fh[]$

$S2 = \text{sort}(A, m+1, e) = sh[]$

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

fh[]

0	1	2	3	4
3	6	8	10	15

sh[]

2	12	17	18

```
C[] = merge (fh, sh)
return C

int[] sort (A, s, e) {
    if (s == e) {
        int arr[1]
        arr[0] = A[s]
        return arr;
    }
    m = (s + e)/2
    fh[] = sort (A, s, m)
    sh[] = sort (A, m+1, e)
    C[] = merge (fh, sh)
    return C
}
```

A =

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

int[] sort (A, s, e) {

[2 3 6 8 10 12 15 17 18]

$m = \frac{s+e}{2}$

1

2 fh[] = sort (A, s, m)

3 sh[] = sort (A, m+1, e)

4 C[] = merge (fh, sh)

5 return C

0, 8

m = 4

fh = [3 6 8 10 15]

sh = [2, 12, 17, 18]

C = [2 3 6 8 10 12 15 17 18]

0, 4

m = 2

fh = [3, 6, 10]

sh = [8, 15]

C = [3, 6, 8, 10, 15]

0, 2

m = 1

fh = [3, 10]

sh = [6]

C = [3, 6, 10]

0, 1

m = 0

[3] [10]

fh sh

C = [3, 10]

2, 2

[6]

3, 4

m = 3

fh = [8]

sh = [15]

3, 3 C = [8, 15] 4, 4

[8] [15]

5, 8

m = 6

fh = [2, 12]

sh = [17, 18]

C = [2, 12, 17, 18]

5, 6

m = 5

fh [2]

[2, 12]

sh = [12]

5, 5 [2]

6, 6 [12]

7, 8

fh = [18]

sh = [17]

C = [17, 18]

7, 7 [18]

8, 8 [17]

~~s 0, e 0~~

~~1, 1~~

[10]

arr[1] = A[s]

return arr

Homework → TC and SC

max depth of stack × SC of 1 call

↳ No of recursive calls × TC of 1 recursive call

Stable Sort → relative order of equal elements is maintained

[6 5 3 5]

↓

[3 5 5 6]

- Bubble
- Insertion
- Selection

Inplace sort → sorting in the input array

- Bubble
- Insertion
- Selection