Q1) Given arr[N] where all the elements are lying in the range 1-5. Sort the array.

Arr → [5 1 2 1 4 2 2 3 3 4 3 2 2 1 1]

⇓

[1 1 1 1 2 2 2 2 2 3 3 3 4 4 5]

① freq Hashmap.

element	freq
5	1
1	~~1~~ ~~2~~ ~~3~~ 4
2	~~1~~ ~~2~~ ~~3~~ ~~4~~ 5
4	1 2
3	~~1~~ ~~2~~ 3

② freq arr

0	4	5	3	2	1
0	1	2	3	4	5

Count sort

#code. →

```
int freq[K+1];
for( i = 0; i < N; i++){
    freq[arr[i]]++;
}
int idx = 0
for( i = 1; i <= K; i++){
    for( j = 1; j <= freq[i]; j++){
        arr[idx] = i;
        idx++;
    }
}
```

i	j	no. of iterations
1	[1,a]	a +
2	[1,b]	b +
3	0	1 +
4	0	1 +
5	0	1 +
6	[1,c]	c +
⋮		⋮ +
K	[1,x]	x

T.C → O(N+K)

S.C → O(K)

→ what if $1 \le arr[i] \le 10^9$, $1 \le N \le 10^5$

freq[10^9+1] ✗

freq Hashmap? ✓

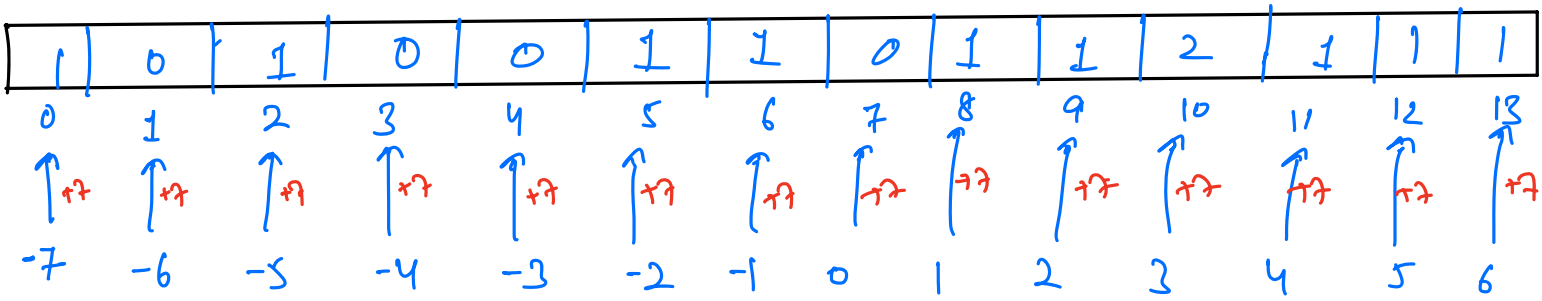
[∵ no. of keys in worst-case = 10^5]

• → -ve elements

$-7 \le arr[i] \le 6$, $1 \le N \le 10^5$

min = -7, max = 6

arr → [2, 1, 3, -5, -2, -7, -1, 5, 3, 6, 4]



code →

```
// iterate and get min and the max element.
freq[max - min + 1];   // ∀i, freq[i] = 0
for(i = 0; i < N; i++){
        freq[arr[i] - min]++;
}

for(i = min; i <= max; i++){
        for(j = 1; j <= freq[i-min]; j++){
            print(i);
        }
}
```

T.C → $O(N + max - min)$

S.C → $O(max - min)$

Q: int → 3 4 5 7 6 8
5 4 3 2 1 0

345768/1000 ⇒ 345 % 10 → 5.

N -> 368 , 10's digit

368/10 ⇒ 36 % 10 ⇒ 6.

i^{th} - digit of a number ⇒ $\left(\frac{N}{10^i}\right)$ % 10

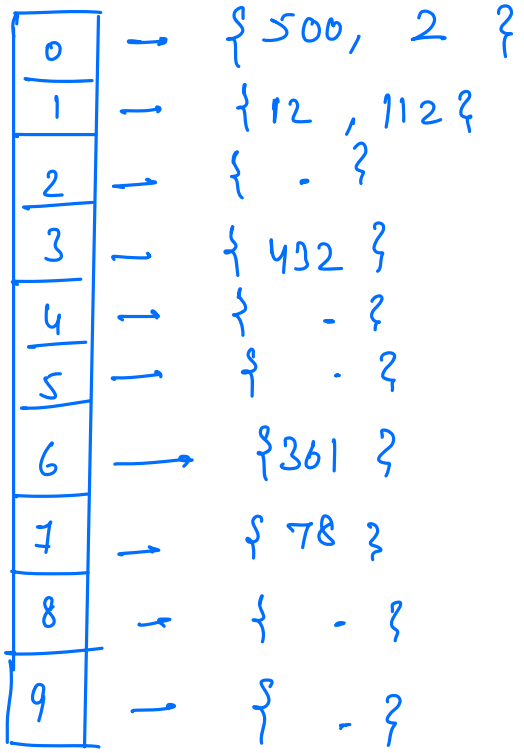
Q: Given +ve integers. Sort these integers on the basis of tens place. ⇒ [stable]

arr → [361, 432, 12, 78, 500, 112, 2]

o/p → [500, 2, 12, 112, 432, 361, 78]

digit - [0,9] count?

arr → [361, 432, 12, 78, 500, 112, 2]



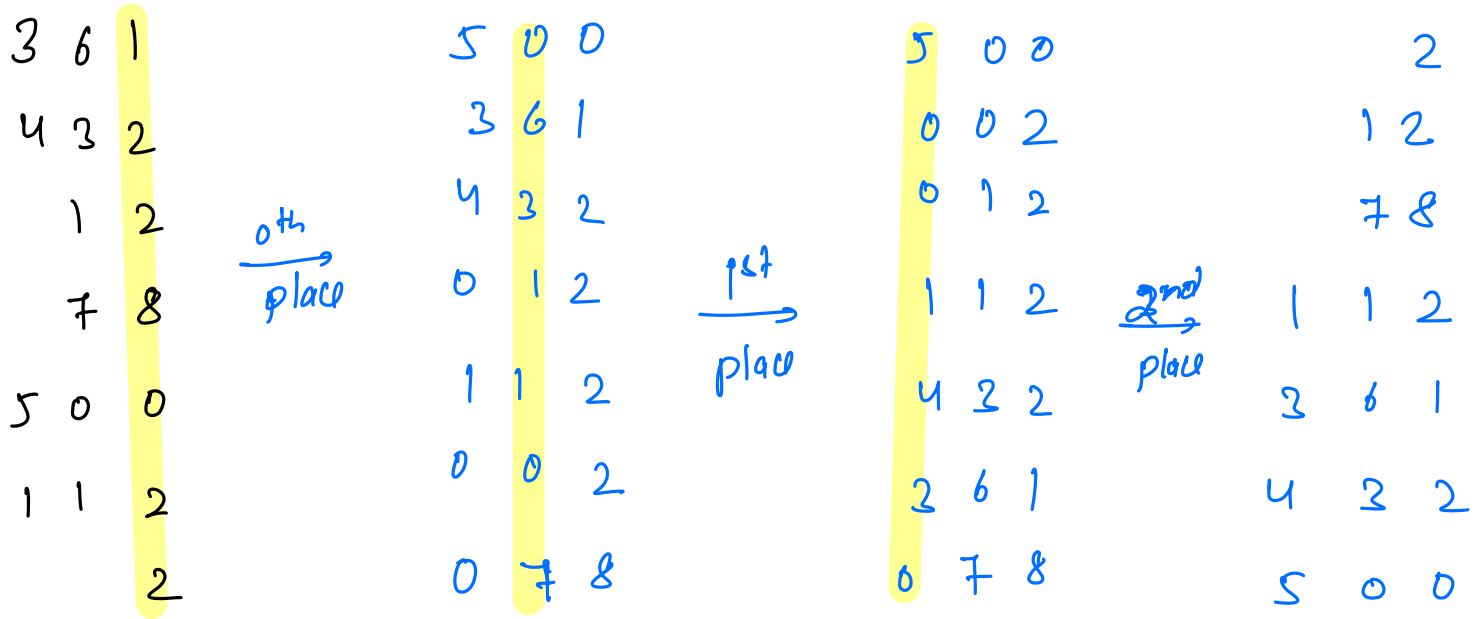
o/p → [500, 2, 12, 112, 432, 361, 78]

code →

```
List<Integer>[10] freq;
for( i=0; i<N; i++){
        tens = (arr(i)/10)%10;
        freq(tens].insert( arr(i));
}
for( i=0; i<10; i++){
    for( j=0; j<freq[i].size(); i++){
        print( freq[i].get(i));
    }
}
```

T.C → O(N)
S.C → O(N)

Radix Sort → sort the elements from least significant digit to most significant digit.



d → no. of digits in the maximum number.

T.C →	O(N*d)
S.C →	O(N)

d → ?

```
max = 0
for(i=0; i<N; i++){
    max = Max(max, arr[i])
}
int m = max; digits = 0
while(m != 0){
    m /= 10;
    digits++
}
```

Break → 10:31 → 10:38

Q) Given arr[N] and all elements are distinct. Find sum of (max - min) of all subsets.

arr → [3, 1, -4]

	max	min	max - min
[ ]	-	-	0
[3]	3	3	0 +
[1]	1	1	0 +
[-4]	-4	-4	0 +
[3, 1]	3	1	2 +
[3, -4]	3	-4	7 +
[1, -4]	1	-4	5 +
[3, 1, -4]	3	-4	7
	Σmax_s	Σmin_s	21

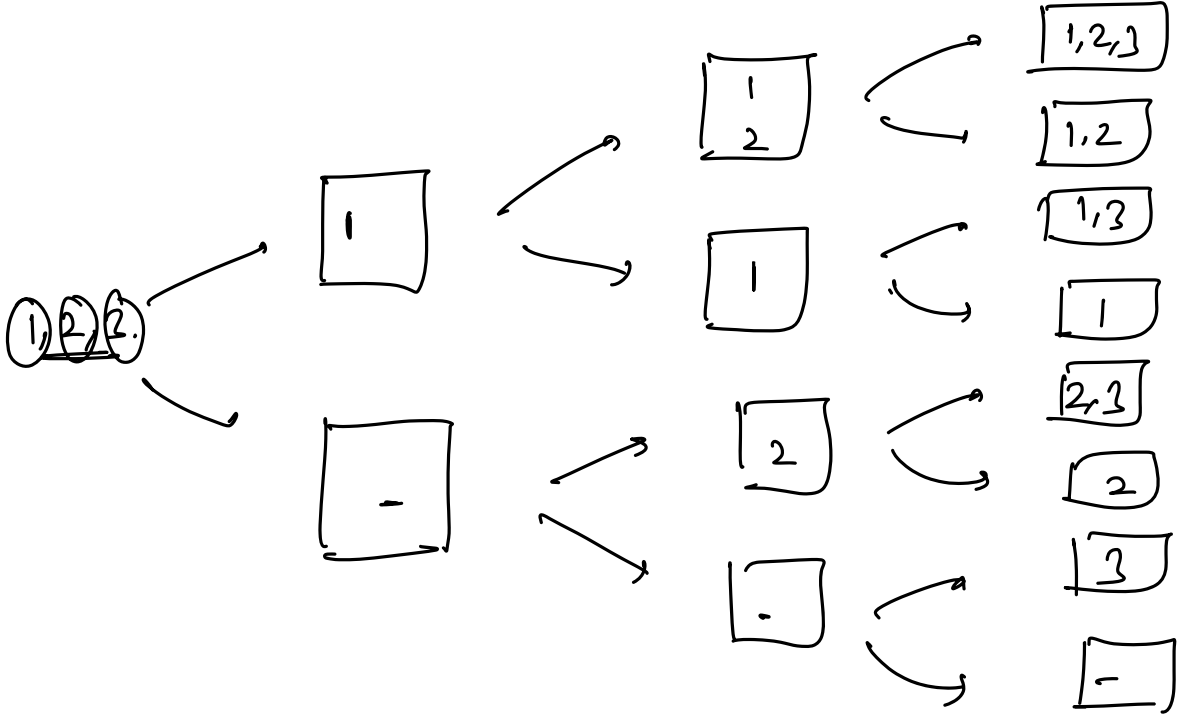
subsets → set of elements attained by deleting 0 or more elements.

arr → [1, 7, 9, 3, 4]

[7, 3, 4] ✓ [7, 34] ✗

[1, 3] ✓

[-] ✓



B.f idea → Generate all subsets, and for every subset find the minimum & maximum element & take their difference.

T.C → $O(2^N * N)$ ⇒ exponential T.C

Contribution?

$$\Sigma(max-min)_s = \Sigma max_s - \Sigma min_s$$

Contribution = arr[i] * no. of subsets in which arr[i] will be the maximum element

arr → [3, 2, 8, 7, 4, 6]

① In how many subsets, 6 is the maximum element.

[6]

⇒ $2^{\text{no. of smaller elements than 6.}}$

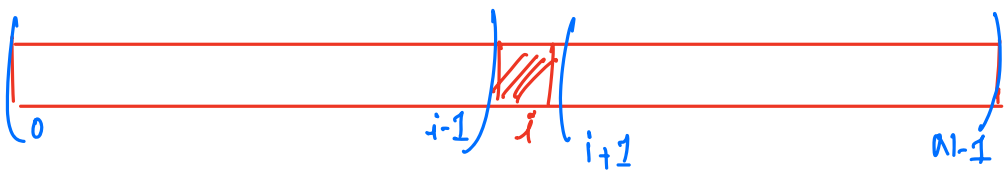
⇒ $2^3 = 8$.

② In how many subsets, 6 is the minimum element.

⇒ $2^{\text{no. of greater elements than 6}}$

⇒ $2^2 = 4$.

Sorting



Count of smaller elements than arr[i] → i

Count of greater elements than arr[i] → $N-i-1$

$[a,b] \Rightarrow b-a+1$

$[i+1, N-1] \Rightarrow N-1-(i+1)+1$

$\Rightarrow N-1-i-1+1$

∴ Contribution of arr[i] in Σmax_s = arr[i] * 2^i

Contribution of arr[i] in Σmin_s = arr[i] * 2^{N-i-1}

code..

```
sort(arr), long max_s = 0, long min_s = 0, mod = 10^9+7
for( i = 0; i < N; i++){
    max_s = max_s + arr[i] * fastpower(2, i, mod);
    min_s = min_s + arr[i] * fastpower(2, N-i-1, mod);
}
return (int)((max_s - min_s) % mod)
```

T.C → O(N log N)

S.C → O(log N)

$1 \le N \le 10^4$

$1 \le arr[i] \le 10^3$

take modulo with 10^9+7

{Doubt Session → 11:00 A.M → 1:30 P.M} → Inversion count

→ Doubts on sorting