Sorting?

→ Arranging data in a particular order!

Eg:

3	7	8	2	1	5
2	2	4	2	1	2

ASC

1	2	3	5	7	8

DESC

8	7	5	3	2	1

INC order of no. of factors

1	3	7	2	5	8

Sort() f^n → [INBUILT]

TC: $O(N \log N)$

SC: $O(N)$ → $O(1)$ (Heap Sort)

int vs int → $O(1)$

str vs str → $O(m)$

0	1	2	—	—	N-1
~	~	~	~	~	~

M

TC = $O(M N \log N)$

→ ~~pankaj~~

→ ~~prabhas~~

Q Given N elements, at every step remove an element!

Cost of removing an element: Sum of array elements present in the array!

Find the MIN cost to remove all the elements!

A : [2, 1, 4]

		Cost
		7
-2	[1, 4]	+ 5
-1	[4]	+ 4
-4	x	$\boxed{16}$

A : [2 1 4]

		7
-4	[2, 1]	+ 3
-2	[1]	+ 1
-1	x	$\boxed{11}$

Observation

(a, b, c, d)

$-a$ LARGE $\quad (b, c, d)$

$-b \quad (c, d)$

$-c \quad (d)$

$-d$ SMALL

Cost

$a + b + c + d$

$+ b + c + d$

$+ c + d$

$+ d$

X

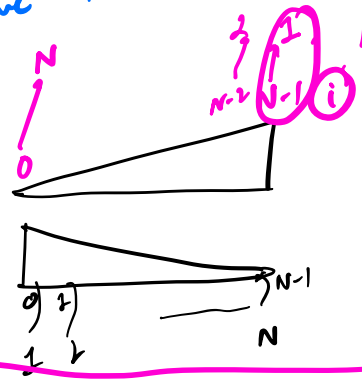
$$\boxed{\text{Total Cost} : \underbrace{a}_{\text{LARGE}} + 2b + 3c + \underbrace{4d}_{\text{SMALL}}}$$

Idea: Remove the elements in DESC order!

// A[]

Sort(A);

~~rev(A);~~



$N-(N-1) = 1$

$N-0 = N$

$i \to$ index

$i+1$ times

```
cost = 0;
for( i = 0; i<N; i++) {
    cost += A[i] x (i+1);
}
ret cost;
```

```
for( i = N-1 ---> 0) {
    sum += (N-i) x A[i];
}
```

$\boxed{TC : O(N \log N)}$

$\boxed{SC : O(1)}$

Q

Noble Integer [DISTINCT]

Given N array elements. Calculate the no. of Noble integers!

Noble integer : $\boxed{A[i] \{ \text{No of elements} < A[i] \} == A[i]}$

A : [1, -5, 3, 5, -10, 4]

#cnt lesser: 2, 1, 3, 5, 0, 4

ANS → 3

1) BF

```
ANS = 0
f( i = 0 → N-1) {
    cnt = 0;
    f( j = 0 → N-1) {
        if( A[j] < A[i]) {
            cnt++
        }
    }
    if (A[i] == cnt) {
        ANS ++;
    }
}
ret ANS;
```

TC : $O(N^2)$

SC : $O(1)$

2) A : [1, -5, 3, 5, -10, 4]

SORT

ind:	0	1	2	3	4	5
:	-10	-5	1	3	4	5
# cnt less	0	1	2	3	4	5

i

A(i)

```
//A[]
Sort(A);        → NlgN
ANS=0;
for( i=0 → N-1){        N
    if (A[i]==i){
        ANS++;
    }
}
ret ANS;
```

$\boxed{TC : O(N \lg N)}$

$\boxed{SC : O(1)}$

Q SAME AS PREV [elements could repeat]

Ex:

A : [0, 2, 2, 4, 4, 6] → 1

ind:	0	1	2	3	4	5
cnt →	0	1	1	3	3	5

→ 7

A : [−3, 0, 2, 2, 5, 5, 5, 5, 8, 8, 10, 10, 10, 14]

ind:	0	1	2	3	4	5	6	7	8	9	10	11	12	13
cnt:	0	1	2	2	4	4	4	4	8	8	10	10	10	13

1) if (A(i) != A[i-1])
→ A(i) is the 1st occ of A(i)

2) if (A(i) != A(i-1))
cnt = i;

else → no change to cnt

```
ANS = 0;                    sort(A);
cnt = 0;
for (i = 0; i < N; i++) {
    if ((i == 0) || (A[i] != A[i-1])) {
        cnt = i;
    }
    if (A[i] == cnt) {
        ANS++;
    }
}
ret ANS;
```

$\boxed{TC: O(N \log N)}$

$\boxed{SC: O(1)}$

⊙ Comparator

A f^n that helps sorting algo to decide the order!

Given an array!
Sort it in INC order of no. of factors!
If 2 elements have SAME # of factors then smaller element should come first!

A : [1, 21, 4, 6]

#f → 1, 4, 3, 4

SORT

[1, 4, 6, 21]

(A) CUSTOM COMPARATOR →

```
sort( A, comp );

bool Comp( int a, int b) {
    // Return true if a before b
    //        false otherwise
    if(a < b) ret true;
    ret false;
}
```

ASC!

a > b → DESC!

```
bool comp( int a, int b) {
    int fa = countFactor(a);
    int fb = countFactor(b);
    if((fa < fb) || ((fa == fb) && (a < b))) ret true;
    ret false;
}

sort(A, comp);
```

$O(K)$

$\boxed{TC: O(KN \log N)}$

$\boxed{SC = O(1)}$

Q Given A[]
Sort in INC order of no. of digits!
Tie: bigger no. before smaller no!

A: [93, 2, 45, 813]

Sort

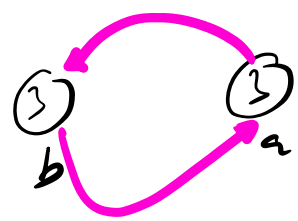
[2, 93, 45, 813]

```
bool comp( int a, int b) {
    int da = digits(a);
    int db = digits(b);

    if( (da < db) || ( (da == db) && (a > b) )) {
        ret true;
    }
    ret false;
}
```

A[] **ASC**

```
bool comp( a, b) {
    if( a <= b) ret true;
    ret false;
}
```



NOTE: ret true ONLY when you strongly want a before b

→ for equal priority elements
↳ ALWAYS ret false!