Sorting

Content

→ Understanding Sorting
→ few problems on sorting
→ 1 sorting algo
→ Comparator function

Sorting ?

Arranging data in increasing / decreasing order based on some parameter.

Eg 2 4 7 11 15 : sorted in Asc, parameter = array values

Eg 15 9 6 2 0 : sorted in Desc, par = array values

Eg 1 2 3 7 4 9 6 : sorted in Asc based on # factors.

Eg	1	2	3	7	4	9	6
# factors	1	2	2	2	3	3	4

par = # factor of array values

Inbuilt libary

↳ sort() , in every language sort() is present.
↳ How? logic ? ⇒ { In Advanced Batch }

TC : O(NlogN) , N = no. of elements to sort

Question 1 : Elements Removal

Given N elements, at every step remove an array element.

Cost to remove element : sum of all elements present in array

find min. cost to remove all elements.

Note : first calculate the cost, then remove the element.

eg a[3] = {2, 1, 4}

	cost
remove 2 :	2+1+4 = 7
remove 1 :	1+4 = 5
remove 4 :	4

total cost : 7+5+4 = 16

	cost
remove 1 :	7
remove 2 :	6
remove 4 :	4

total cost = 17

	cost
remove 4 :	7
remove 2 :	3
remove 1 :	1

total cost = 11

eg a[4] = {3 6 2 4}

	cost
remove 6 :	15
remove 4 :	9
remove 3 :	5
remove 2 :	2

total cost = 31

a[3] = {6 -3 4}

remove 6 :	7
remove 4 :	1
remove -3 :	-3

total cost = 5

Observation : deleting element by element in decreasing order to get min cost ?

$a[4] : \{ \overset{0}{a}, \overset{1}{b}, \overset{2}{c}, \overset{3}{d} \}$

$$\left.\begin{array}{lll} \text{remove } a & : & a+b+c+d \\ \text{remove } b & : & b+c+d \\ \text{remove } c & : & c+d \\ \text{remove } d & : & d \end{array}\right\} \xrightarrow[\text{cost}]{\text{total}} a+2b+3c+4d$$

a ↓ 1st max, $2b$ ↓ 2nd max, $3c$ ↓ 3rd max, $4d$ ↓ 4th max . . . .

Code

```
int minCost (a[]) {
    n = a.length
    sort (a, desc);  → sort a[] in desc order
                     ↳ TODO in your own language
    ans=0
    for (i=0; i<n; ++i ){
        ans = ans + (i+1)*a[i]
    }
    return ans
}
```

TC: ~~O(N)~~

TC: O(NlogN + N)

: O(NlogN)

SC : O(1)

dry run : a[4] = 3 6 2 4

sort(a) = 6 4 3 2

ans = 6x1 + 4x2 + 3x3 + 2x4

= 6 + 8 + 9 + 8 = 31

Question 2 : Noble Integers { Data is distinct }

Given N array elements, calculate no. of noble integers.

An ele in a[] is said to be Noble iff

$$\{ \underbrace{\text{No. of element} < \text{ele}}_{\text{count}} = \text{ele itself} \}$$

eg { -1 -5 3 5 -10 4 } Ans = 3

eg	-1	-5	3	5	-10	4
# less	2	1	3	5	0	4

eg { -3 0 2 5 } Ans = 1

eg	-3	0	2	5
# less	0	1	2	3

Bruteforce

For every ele. in a[], iterate & get # ele < ele and compare with ele. itself.

```
int ans=0
for (i=0; i<n; ++i) {
    count=0   // # ele < a[i]
    for (j=0; j<n; ++j) {
        if (a[j] < a[i])
            ++count
    }
    if (a[i] == count)
```

TC: $O(N^2)$

SC: $O(1)$

```
        ++ans
    }
```

Idea: Sort the array in asc. order.

$$\text{sorted}(a):\ \underbrace{a[0]\ \ a[1]\ \ \ldots\ \ a[i-1]}_{\text{count} = i-1-0+1 = i}\ \ a[i]\ \ a[i+1]\ \ \ldots\ \ a[n-1]$$

↳ all these elements are less than a[i]

count = i-1-0+1
= i

⇒ if (a[i] == i)
then
a[i] is noble

Code

```
int noble(a[]) {
    n = a.length
    sort(a, asc)  → TODO
    ans = 0
    for(i=0; i<n; ++i) {
        if(a[i] == i)
            ++ans
    }
    return ans
}
```

TC: O(N log N)

SC: O(1)

dry run:

	0	1	2	3	4	5
	-1	-5	3	5	-10	4
sort:	-10	-5	-1	3	4	5

Ans = 3

Question 3 : Count Noble integers : { Data can repeat }
↓
Duplicate values

We can solve using Bruteforce:

TC: $O(N^2)$ SC: $O(1)$

eg

	0	1	2	3	4	5
	0	2	2	3	3	6
# less	0	1	1	3	3	5

Ans: 3

Above Indexing approach will not work

eg

	0	1	2	3	4	5	6	7	8
	-10	1	1	1	4	4	4	7	10
#less	0	1	1	1	4	4	4	7	8

Observation 1 : index = #less for only the first occurance of every element

Observation 2 : if an element is noble, all occurance are noble.

Idea : if ele comes for the first time i.e, if(a[i] != a[i-1])

count of ele. less than a[i] = i.

If ele repeats i.e, if (a[i] == a[i-1])

count of ele. than a[i] will be same as prev. one

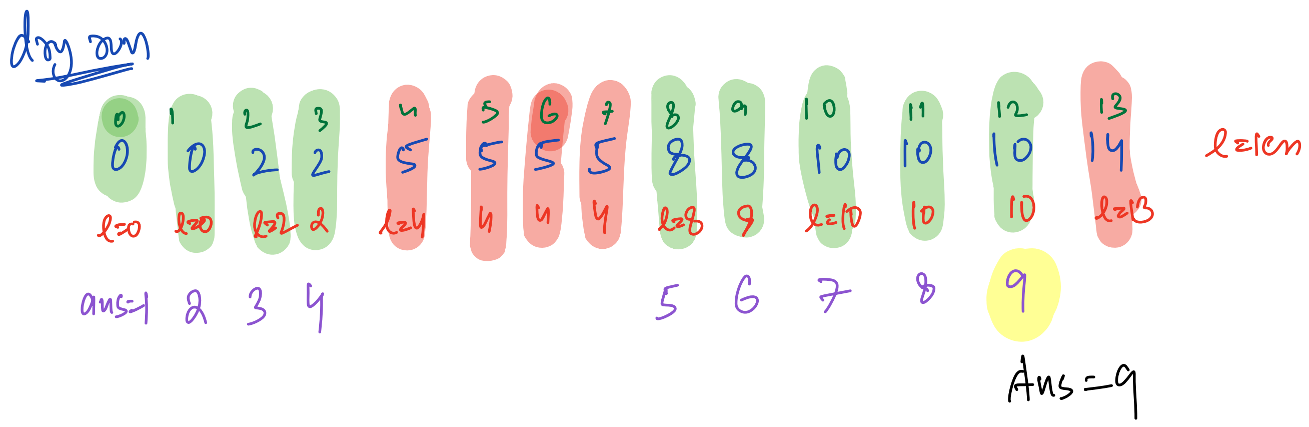
Code

```
int noble (a[]) {
    n= a.length
    ans=0                        → sort(a, asc);
    if (a[0]==0) { ans=1 }

    len=0   # ele less than a[i]

    for (i=1; i<n; ++i) {
        if (a[i] != a[i-1])   # a[i] coming for first time
            len = i
        else {
        }
        if (a[i] == len)
            ++ans
    }
    return ans
}
```

TC: O(NlogN)

SC: O(1)

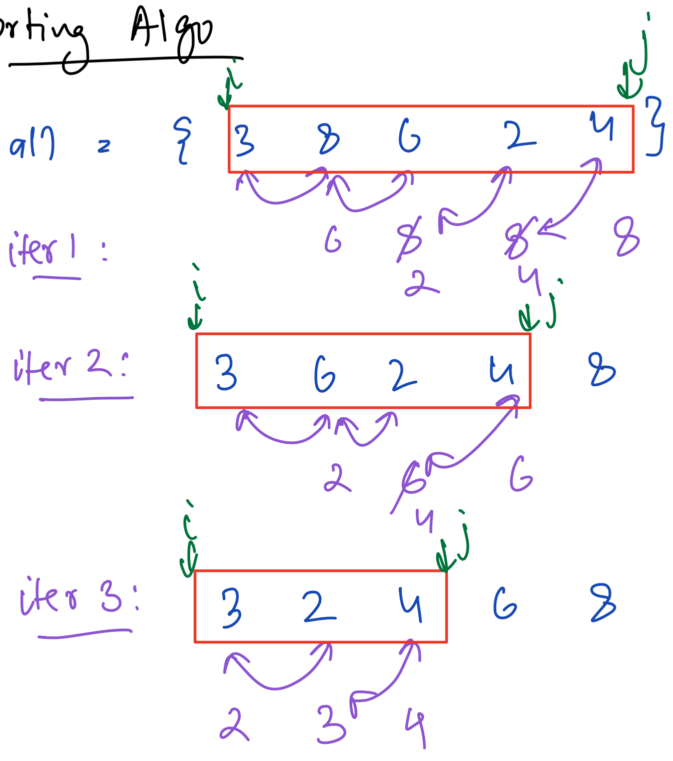
dry run



0	1	2	3	4	5	6	7	8	9	10	11	12	13	
0	0	2	2	5	5	5	5	8	8	10	10	10	14	
l=0	l=0	l=2	2	l=4	4	4	4	l=8	9	l=10	10	10	l=13	l=len
ans=1	2	3	4					5	6	7	8	9		

Ans=9

Sorting Algo



sort in asc

: 8 is at correct position

: 6,8 are at correct position

: completly sorted

Code

```
sort ( a[] ) {
    n = a.length
    for ( j = n-1 ; j >= 0 ; --j ) {
        for ( i = 0 ; i < j ; ++i ) {
            if ( a[i] > a[i+1] )
                swap ( a[i] , a[i+1] )
        }
    }
}
```

for decreasing a[i] < a[i+1]

⇒ Bubble Sort

TC: O(N²)

SC: O(1)

iteration = $n-1 + n-2 + n-3 + \dots + 1$

$= \frac{(n-1)n}{2} = O(N^2)$

for any type of sorting like sort based on # factors ,
we use comparator function.

```
for (j=n-1; j>=0; --j){
    for(i=0;  i<j ; ++i) {
        if ( comp( a[i] , a[i+1]))
            swap(a[i], a[i+1])
    }
}
```

just implement this function, to get your desired sorting. (custom)

In Java / JS

```
Comparator customComparator = ( Integer a, Integer b) → {
  # If you want a to come before b    : return -1
  # If you want a & b are same        : return 0
  # If you want b to come before a    : return 1
}

sort ( a , customComparator);

// sort on asc order of # of factors

        fa = factors(a)
        fb = factors(b)
        if ( fa < fb)  return -1
        else  return 1
```

In Python

```
def compare(a,b) :
    # If you want a to come before b      : return -1
    # If you want a & b are same          : return 0
    # If you want b to come before a      : return 1

a.sort(key= cmp_to_key(compare))
```

In C / Cpp

```
bool compare( int a, int b) {
    # If you want a before b     : return true
    # Else                       : return false
}
 sort(a, compare)
```