Sorting

"Sorting is the foundation of organization, and organization is the key to efficiency" - Anonymous

Sorting? arrangement of data, in particular order, on the basis of some rule

sorted in asc order

sorted in dsc. order

Quiz

Sorted based on how many factors they have in increasing order.

count of factors: 1 2 3 4 6

Sort → Price, avg.

advanced

How ~~to~~ sort

why sorting?

- searching made easy

⇒ inbuilt sort function

sort( ) , T.C. (nlogn) // in advanced classes

S.C. O(N)

⊛ T.C. = $O(N^2)$

↓

would sorting help to reduce the time complexity?

Q. Given an array of N integers, minimise the cost to empty the array, where cost of removing an element is equal to sum of all elements left in the array.

ex. A = [2, 1, ~~4~~] , ans = 11

step I remove 4 ⇒ 4+2+1 = 7

II. remore 2 ⇒ 2+1 = 3

III remove 1 ⇒ 1 = 1

total cost = 11.

another way $A = [2, 1, 4)$

I remove 2 $\Rightarrow$ $2+1+4 = 7$

II remove 4 $\Rightarrow$ $1+4 = 5$

III remove 1 $\Rightarrow$ $1 = 1$

total cost = 13

Quiz $A = [4, 6, 1)$

I remove 6 $\Rightarrow$ $4+6+1 = 11$

II remove 4 $\Rightarrow$ $4+1 = 5$ $\Rightarrow$ 17

III remove 1 $\Rightarrow$ 1

another way $A = [4, 6, 1)$

I remove 4 $\Rightarrow$ $4+6+1 = 11$

II remove 6 $\Rightarrow$ $6+1 = 7$ $\Rightarrow$ 19

III remove 1 $\Rightarrow$ $1 = 1$

not minimum

Quiz: $A = [3, 5, 1, -3]$

(I) remove 5 ⇒ 3+5+1+(-3) = 6
(II) remove 3 ⇒ 3+1+(-3) = 1
(III) remove 1 ⇒ 1+(-3) = -2
(IV) remove -3 ⇒ -3 = -3

⇒ 2

Brute force: try all the possibilites ⇒ n! ✗ not a good idea

n! ← possibilities

§. optimal soln:

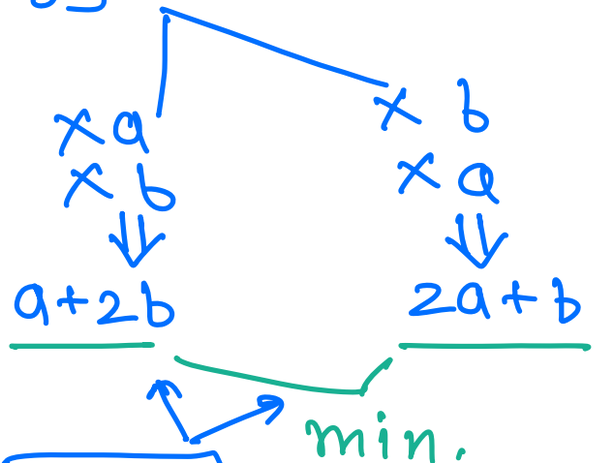
	N	A	cost
→	0	[ ]	0
→	1	[1]	1
→	1	[13]	13
→	1	[9]	9
→	2	[1, 5]	7

remove 5 → 1+5 ⇒ 7
remove 1 → 1

1+5 remove 1 → 11
5 remove 5

⇒ 2 [10, 30] 50

⇒ 2 [a, b]



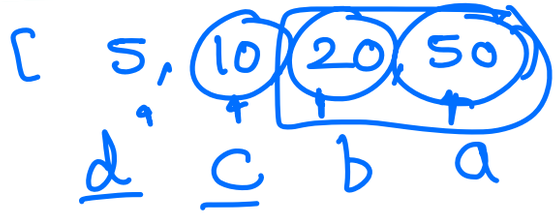
if $a < b$ ⇒ $2a+b$ would be smaller than $a+2b$.

⇒ [a, b, c, d]

remove a ⇒ $a+b+c+d$
remove b ⇒ $b+c+d$
remove c ⇒ $c+d$
remove d ⇒ d

total cost ⇒ $a+2b+3c+4d$

4 numbers minimise this



$\Rightarrow$ [50, 20, 10, 5]

0 1 2 3

$1*50 + 2*20 + 3*10 + 4*5$

idea: ① Sort the array (in dsc. order)

② cost = cost + A[i]*(i+1)

```
int minCost (A) {                    A = [2, 1, 4]

    int N = A.size();                N = 3
    reverse_sort (A); // O(NlogN)    A = [4, 2, 1]
```

```
int cost = 0;
for (i = 0; i < N; i++) {
        cost += A[i] * (i+1)
}
return cost;
}
```

cost = ~~0 4 28 11~~
i = ~~0 1 2~~ 3 4

T.C. = O(NlogN)
S.C = O(N)

Q. Find count of noble integers in the array of distinct elements.

(count of)

A[i] is noble, if elements smaller than A[i] is equal to A[i].

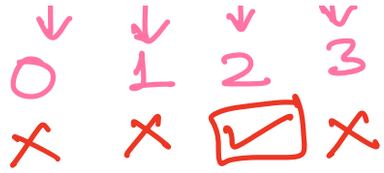
ex: A = [1, -5, 3, 5, -10, 4]

0	1	2	3	4	5
1	-5	3	5	-10	4
↓	↓	↓	↓	↓	↓
2	1	3	5	0	4
✗	✗	✓	✗	✗	✗

Count = 3

Quiz: A = [-3, 0, 2, 5]

0	1	2	3
-3	0	2	5
1	1	1	1



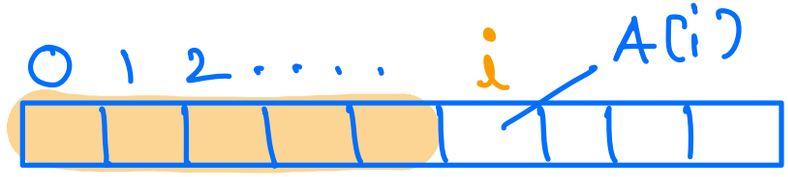
count = 1.

Brute force: ① for every element countsmallerelement

② if countSmallerElements == A[i]
ans++

T.C. = $O(N^2)$
S.C. = $O(1)$

Optimal soln: Sort the data. asc. order



(L, R)
(R-L+1)

smaller than A[i]

count = (0 to i-1)

↓ (i-1-0+1)

$= i$

Obsn: if i == A[i] ⇒ A[i] is noble.

idea: ① Sort the array
② count for i == A[i] for all i(s)

```
// Sort the data
count = 0
for (i = 0; i < N; i++) {
        if (A[i] == i) {
                count++
        }
}
```

T.C. = O(N log N)
S.C. = O(N)
(merge sort)

Ⓐ understanding the problem.

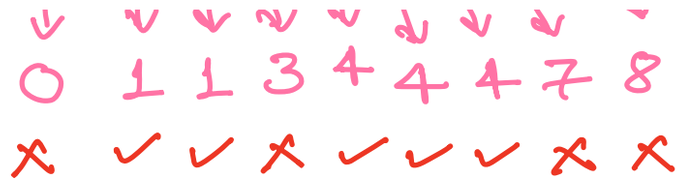
Ⓐ what if there are duplicates.

quiz: A = [-10, 1, 1, 3, 100]
0 1 2 3 4
↓ ↓ ↓ ↓ ↓
0 1 1 3 4
✗ ✓ ✓ ✓ ✗

Ans = 3

Quiz: A = [-10, 1, 1, 2, 4, 4, 4, 8, 10]
0 1 2 3 4 5 6 7 8

Ans = 5



count = 7

Quiz: A = [-3, 0, 2, 2, 5, 5, 5, 5, 8, 8, 10, 10, 10, 14]

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

0 1 2 2 4 4 4 4 8 8 10 10 10 13

Smaller element

$\neq i$

if current element != previous element

countSmallerElements) = i

idea:

1. sort the array
2. if (current element != previous element) {
   if (A[i] == i)
   count++;
3. else remember / calculate smaller elements

10:36

Code

Idea: ① Sort the array

② Keep record of countSmallerElements

③ If A[i] != A[i-1]
the countofsmallerElements = i

④ else: don't update countofsmallerElements

```
int countnobles(A){
    // sort the data ⟶ // T.C. ⇒ O(NlogN)
    int countofsmallerElements = 0
    int nobleElements = 0
    if (A[0] == 0) { nobleElements++; }
    for (i=1; i<N; i++){
        if (A[i] != A[i-1]{
```

T.C. = O(NlogN)

S.C = O(N)
(merge sort)

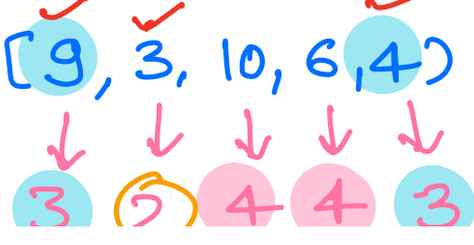
```
            countOfSmallerElements = i
        }
        if (countOfSmallerElements == A[i]){
            nobleElements++;
        }
    }
    return nobleElements;
}
```

Q. we have used library sort fn.

Q. sort the numbers based on count of factors. if count of factors are equal sort the number on the basis of magnitude.

A = [9, 3, 10, 6, 4]

⇒



aw = [3, 4, 9, 6, 10]

Que: How we can use the in-built library function to sort based on some rule.

↓
count of factors.

→ Sort ( ) ← by default 1 parameter → Array

→ sort (— —), (array index i, array index J)

to sort the numbers b/n index i to J.

→ sort (—, —, ——)

comparator function

rule for starting

```
int x, int y

if (x < y)
    put x first
else
    put y first
```

by default, it arranges the integers based on its magnitude (asc)

sort ⇒ asc order, based on magnitude

```
int x, int y

if(countfactors(x) < countfactors(y))
        ⇒ put x first

if (countfactors (y) < countfactors(x))
        ⇒ put y first
```

```
if (countfactors(x) == countfactors(y)) {
        if (x < y)
            put x first
        else
            put y first
}
```

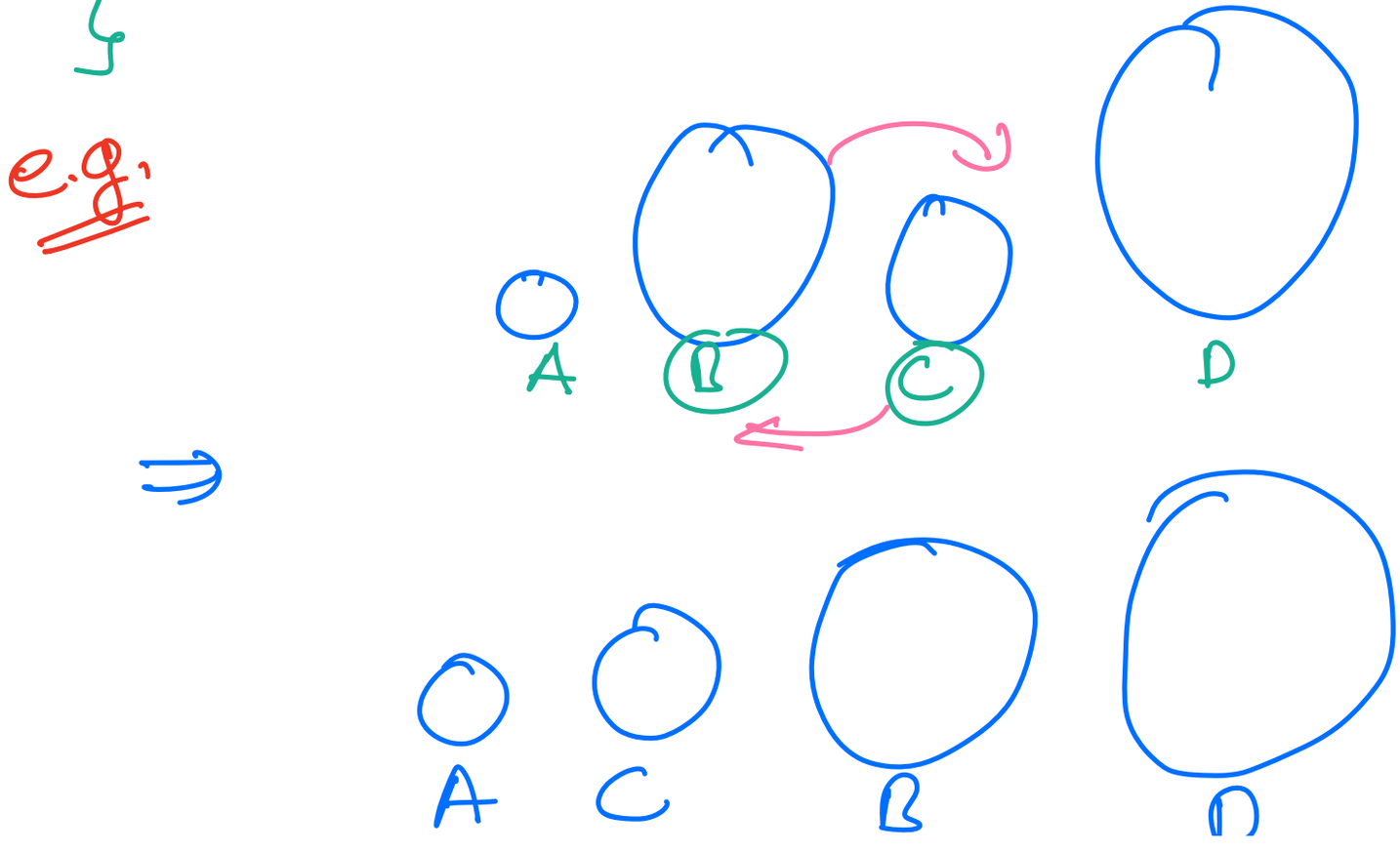
datatype datatype → if true put x first

```
bool comp (int x, int y){ → if false put y first
    int countX = countOfFactors(x)
    int countY = countOfFactors(y)

    if (countX < countY){
        return true
    }
    else if (countY < countX){
        return false
    }
```

```
else {    // both are equal
    if (x < y)
        return true
    else
        return false
}
}
```

e.g.



Comparator fn in languages:

Java, python, JS, C#, ruby

→ if first parameter (x) (should come) before second (y)
-ve value should be returned

⇒ if second parameter (y) should come before first
+ve value should be returned.

⇒ if both are same, then 0 should be returned

in C++

⇒ in sorted form, if first parameter should come first return true

⇒ else return false.

ex: [10 −5 2 5 8]

Sort them in dsc ↓ order.

default library fn sort() returns in asc ↑ order.

⇒ we need to write our own comparator fn

code.C++

```
bool comparator (int x, int y) {
    if (x < y) {
        // put y first
        return false
    }
    else {
        return true:
    }
}
```

⇒ sort (A.begin() , A.end() , comparator);

Q. write the comparator fn to sort N strings. based on string length

ex. ['aa', 'b', 'ccc', 'aaa']

⇒ ['b', 'aa', 'ccc', 'aaa']

Comparator function ?

```
bool comparator (string x, string y){
    int xlen = x.size()
    int ylen = y.size()

    if (xlen < ylen){
        return true;
    }
    else
```

```
        else
            return false;
    }
}
```

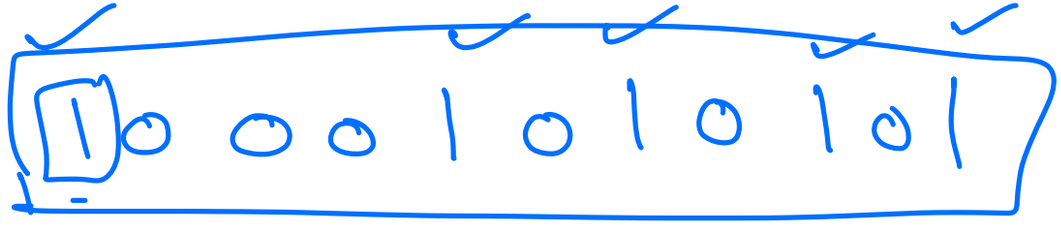
Doubt seniors

```
while (A > 0)
    A = A & (A-1) // unset the last set bit from the right
    count++
}
```

2 : 1 0 → 0

1 : 0 1 → 0

0 0

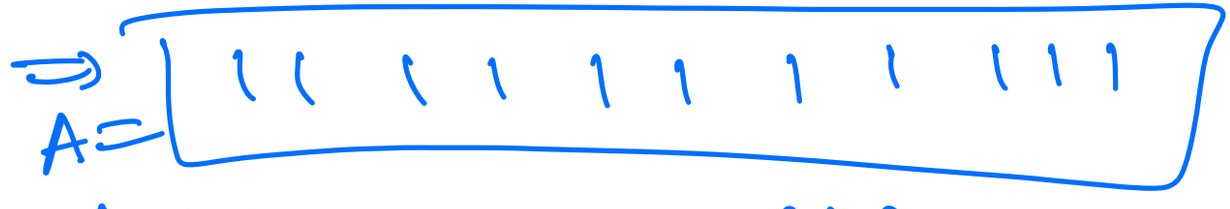


3

$\Rightarrow \log_2 3$

$= \lceil 1.\overline{58} \rceil$

$\Rightarrow 2$



worst case

$= \log_2(A)$

count of set bits

$T.C. = \log_2(A)$

int 32 bits

$T.C. = O(1)$

$\Rightarrow$ (11)

$\rightarrow$ (5) $\log_2 5$

$2^{\sim}$

$\Rightarrow$ 101

(*) intuition behind bit-manipulation:

Q

1 → 5
2 → 25
3 → 30
4 → 125
5 → 130
⋮
A ?

5^1
5^2
$5^2\ 5^1$
5^3
$5^3\ \ 5^1$

$5^3 + 5^2 \Rightarrow$ (150)

$5^A + 5^X$

$\Rightarrow$ (1)
$\Rightarrow$ 2
$\Rightarrow$ 2+1
$\Rightarrow$ 3
$\Rightarrow$ 3+1
$\Rightarrow$ 3+2
$\Rightarrow$ 3+2+1
$\Rightarrow$ (4)
$\Rightarrow$ 4+1

	s^4	s^3	s^2	s^1
⇒	0	0	0	1
⇒	0	0	1	0
⇒	0	0	1	1
⇒	0	1	0	0
⇒	0	1	0	1
⇒	0	1	1	0
⇒	0	1	1	1

⇒ 4+2

⇒ 4+2+1

⇒ 4+3

⇒ 4+3+1

⇒ 4+3+2

⇒ 4+3+2+1

poly^n

$a_n x^n + a_{n-1} x^{n-1}$