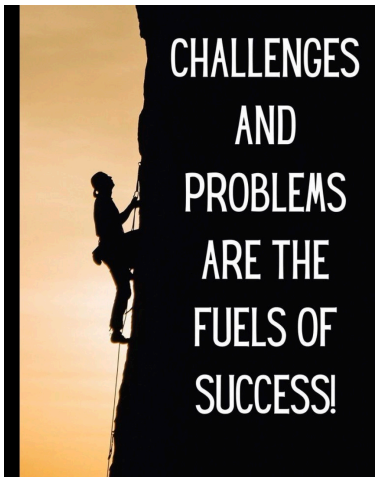
Today's Quote



Today's content

- understand sorting
- few problems on sorting
- 1 sorting algo
  - Comparator

Sorting ⇒ Arranging of data in inc/dec order based on a parameter

Eg: { 2 4 7 11 15 20 } → inc order, par → value

Eg: { 15 10 9 6 -2 } → dec. order, par → value

Eg: { 1 2 3 7 4 9 6 } → sorted in inc. order based on factors count.

factors: 1 2 2 2 3 3 4

Inbuilt library functions →

- sort(), in every language.
- How? logic? ⇒ {In advance module}
- T.C → NlogN, if N elements are present in the array.

Q) Elements Removal

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

[Cost to remove element = Sum of array elements present in array]

Find min cost to remove all elements.

Note: first add the cost of removal & then remove it.

eg: arr[3] = {2 1 4}
(indices: 2 → 0, 1 → 1, 4 → 2)

	cost
remove 2 :	7
remove 1 :	5
remove 4 :	4
total cost = 16	

	cost
remove 4 :	7
remove 2 :	3
remove 1 :	1
total cost = 11	

eg: arr[4] : {3 6 2 4}
(indices: 3 → 0, 6 → 1, 2 → 2, 4 → 3)

	cost
remove 6 :	15
remove 4 :	9
remove 3 :	5
remove 2 :	2
total cost = 31	

Observation ⇒ Delete elements in decreasing order to get min cost ?

arr - { a b c d }

cost.

remove a → a+b+c+d

remove b → b+c+d

remove c → c+d

remove d → d

Total cost :

a + 2b + 3c + 4d

1st max

2nd max

3rd max

4th max.

pseudo-code

```
int minCost ( arr, N ){
    sort ( arr, desc ) // sort the array in dec. order.
    cost = 0
    for ( i = 0 ; i < N ; i++){
        //How many times arr[i] will be present in total cost
        cost += arr[i] * (i+1)
    }
    return cost ;
}
```

T.C → O(NlogN)

S.C → O(1)

Q) Noble Integers {distinct data}

Given N array element, calculate no. of noble integers.
An element ele in arr[] is said to be noble if
{No. of elements < ele = ele itself}

Eg → { 1 (0), -5 (1), 3 (2), 5 (3), -10 (4), 4 (5) }
#less: 2, 1, 3, 5, 0, 4
ans = 3

Eg → { -3 (0), 0 (1), 2 (2), 5 (3) }
#less: 0, 1, 2, 3
ans = 1

observation:
[-ve integer can't be noble integers]

Eg → { -10 (0), -5 (1), 1 (2), 3 (3), 4 (4), 5 (5), 10 (6) }
#less: 0, 1, 2, 3, 4, 5, 6
[ans → 3]

Brute force solution: for every element, find count of smaller elements & then check if [count == current element].

```
int nobleIntegers(arr, N){
    ans = 0
    for(i=0; i<N; i++){
        count = 0
        for(j=0; j<N; j++){
            if(arr[j] < arr[i]){ count += 1 }
        }
        if(count == arr[i]){ ans += 1 }
    }
    return ans;
}
```

T.C → O(N²)
S.C → O(1)

Optimisation

sort the array in increasing order

```
int nobleIntegers ( arr, N) {
    sort ( arr, asc.)
    ans = 0;
    for ( i = 0 ; i < N : i++){
        if( arr[i] == i) { ans += 1 }
    }
    return ans;
}
```

T.C → O(N log N)
S.C → O(1)

Q: Noble integers : {data can repeat}

Eg: { 0 2 2 3 3 6 } {ans → 3}

#lem 0 1 1 3 3 5

Eg: { -10 1 1 1 4 4 4 7 10 } [ans → 7]

#lem. 0 1 1 1 4 4 4 7 8

Eg: { -3 0 2 2 5 5 5 5 8 8 10 10 10 14 }

#lem 0 1 2 2 4 4 4 4 8 8 10 10 10 13

[ans → 7]

Observations :

→ if element is repeating, (arr[i] == arr[i-1])

count of smaller elements is going to remain same.

→ if element is coming for the first time (arr[i] != arr[i-1])

count of smaller elements = i

→ idea 1 of previous question ✓

→ idea 2 [we need to make some changes]

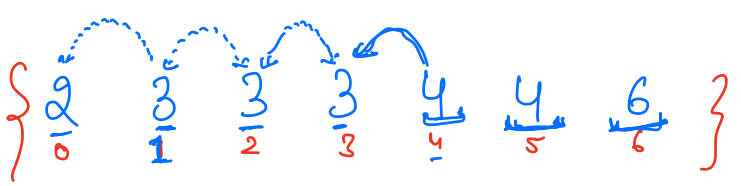
pseudo-code

```
int noble Integers ( arr , N ) {
    sort ( arr, asc. )
    ans = 0 , count = 0
    if ( arr[0] == 0 ) { ans += 1 }      ⇒ Edge case.
    for ( i = 1 ; i < N ; i++ ) {
        // for every arr[i], we need to find count of smaller elements

        if ( arr[i] != arr[i-1] ) {
            count = i
        }
        else {
            // that element is repeating
        } // do nothing

        if ( arr[i] == count ) { ans += 1 }
    }
    return ans ;
}
```

T.C → O(NlogN)
S.C → O(1)

arr[i] == arr[i-1]

Eg:



count = ~~0~~ ~~1~~ ~~4~~ 6. ans = ~~0~~ ~~1~~ ~~2~~ 3

→ Give enough time while reading the qn
→ take some egs.

Sorting algo

Bubble Sort

arr → { 3 8 6 2 4 }

iteration 1 : { 3 6 2 4 8 } : 8 at correct position.

iteration 2 : { 3 2 4 6 8 } : 6, 8 at correct position.

iteration 3 : { 2 3 4 6 8 } : 4, 6, 8 at correct position [whole array is sorted].

observation

After 1 iteration → last 1 element at correct position

After 2 iteration → last 2 elements " "

After 3 " → last 3 elements " "

After N-1 iteration → last N-1 elements at correct position

⇓

last N elements at correct position

idea & pseudo-code

We need N-1 iterations →
in every iteration, traverse from left to right and check the adjacent elements.

```
bubbleSort ( arr , N ) {
    for( i = 1 ; i < N ; i++) {
        for ( j = 0 ; j < N-1 ; j++){   // small optimisations are possible
            if (arr[j] > arr[j+1]){
                //swap arr[j] with arr[j+1]
            }
        }
    }
}
```

T.C → O(N²)
S.C → O(1)

// Decreasing order.

```
bubbleSort ( arr , N) {
    for( i = 1 ; i < N ; i++) {
        for ( j = 0 ; j < N-1 ; j++){
            if (arr[j] < arr[j+1]){
            // swap arr[j] with arr[j+1]
            }
        }
    }
}
```

[has power to control the inc/dec. order.]

idea -

```
bubbleSort ( arr , N) {
    for( i = 1 ; i < N ; i++) {
        for ( j = 0 ; j < N-1 ; j++){
            if ( comp(arr[j],arr[j+1]) ){
            // swap arr[j] with arr[j+1]
            }
        }
    }
}
```

```
boolean comp(int a, int b){
    // Just implement this function to get your own sorted order
    // if a is going to come before b or vice-versa.
}
```

Comparator in Java / Python

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

```
def compare ( a , b ) :
    if you want a to come before b : { return -1 }
    if you want a & b same         : { return 0 }
    if you want b to come before a : { return 1 }
```

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

⟹ {check the syntax in your language.}

Q1 Given N elements. Sort them in increasing order of their no. of factors.

Note: If two elements have same no. of factors, element with less value should come first.

→ No extra space is allowed.

Eg → arr: { 9 (0) 3 (1) 4 (2) 8 (3) 16 (4) 37 (5) 6 (6) 13 (7) 15 (8) }

factors: { 3 2 3 4 5 2 4 2 4 }

sorted order → { 3 13 37 4 9 6 8 15 16 }
(0) (1) (2) (3) (4) (5) (6) (7) (8)

Idea:

→ sort(arr, comp)

{your own sorted order}

{T.C → in adv.}

```
int comp(int a, int b){
    int fa = factor(a)
    int fb = factor(b)
    if(fa < fb){
        // a should come first
        return -1
    }
    else if(fa == fb && a<b){
        // a should come first
        return -1
    }
    else{
        // b should come first
        return 1
    }
}
```

Doubts :

arr :- { ~~3~~ ~~6~~ ~~4~~ ~~2~~ }
0 1 2 3

Cost of removal = sum of all the elements before removing

cost:

remove 6 : 6 + 4 + 3 + 2

remove 4 : 4 + 3 + 2

remove 3 : 3 + 2

remove 2 : 2

[6·1 + 4·2 + 3·3 + 2·4]

arr → [6 4 3 2]
0 1 2 3

6*1 4*2 3*2 2*4

[ans += arr[i] * (i+1)]

✓ Arrays.

✓ Modular Arithmetic

B.M.

→ watch recording → 1.25x 1.5x

→ [try to attempt H.W].

→ 25.30 minutes to every question

→ connect with T.A ✓