Sorting Basics

Content

- Introduction
- Minimize cost to empty array
- Noble Integers
- sorting Algos.

Introduction

Sorting — Arranging the data in a particular order based on a parameter

Eg : 2 3 9 12 17 19 } ascending order

values increase

parameter = value

10 5 3 1 } descending order

values decrease

parameter = value

Quiz 1 13 9 6 12 ?

factors 1 2 3 4 6

parameter = # of factors.

Note : Do not assume the data can only be sorted via value as parameter.

ie ascending or descending.

Minimize the cost to empty array

Given int[n], minimize the cost to empty given array where

cost of removing an element = sum of elements left in the array

Eg A[] = [2 1 4]

way 1

		cost
Remove 1	⟶	2+1+4 = 7
[2 4]		
Remove 4	⟶	2+4 = 6
[2]		
Remove 2	⟶	2 = 2

Total cost = 15

way 2

[2 1 4]

	cost
Remove 4	2+1+4 = 7
[2 1]	
Remove 2	2+1 = 3
[1]	
Remove 1	1

11

[4 6 1] Cost

Remove 6 → 4 + 6 + 1 = 11

[4 1]

Remove 4 → 4 + 1 = 5

[1]

Remove 1 → 1 = 1

17

[3 5 1 -3] Cost

Remove 5 → 3 + 5 + 1 - 3 = 6

[3 1 -3]

Remove 3 → 3 + -3 + 1 = 1

[1 -3]

Remove 1 → 1 - 3 = -2

[-3]

Remove -3 → -3

2

[a b c d] → cost

Remove a → a+b+c+d

[b c d]

Remove b → b+c+d

[c d]

Remove c → c+d

[d]

Remove d → d

$1.a + 2.b + 3.c + 4.d$

minimize

Note to minimize $1.a + 2.b + 3.c + 4.d$ the value of a should be max of A[].

Algo steps

1> sort the array in decending order

4	3	2	1
a	b	c	d

$4 + 2.3 + 3.2 + 4.1$

20.

NOTE → Contribution of i^{th} value → $i+1$

Pseudocode

[3 5 1 -3]

// sort the array in descending order

[5 3 1 -3]

0 1 2 3

```
cost = 0
for ( i ⟶ 0 to N-1 ) {
|       cost += (i+1) * A[i]
}
print (cost)
```

1.5 + 2.3 + 3.1 + 4.-3

5 + 6 + 3 − 12

14 − 12 = 2

TC : O(N + TC for sorting) = O(N log N)

SC : O(1)

TC for sorting ⟶ O(N log N)

Count Noble Integers

Given an int[n] of distinct elements. Find the count of noble integers

A[i] is noble if count of elements smaller than A[i] is equal to A[i]

Eg A[] = [1 -5 3 5 -10 4]
smaller 2 1 3 5 0 4

Output = 3

Eg A[] = -3 0 2 5
smaller 0 1 2 3

output = 1

Bruteforce

For all element x. Iterate through array to find of count of < x = p

if x == p

ans += 1

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

index	0	1	2	3
A[] =	-3	0	2	5
# smaller	0	1	2	3

Note → If data is sorted then index == # of smaller

Pseudocode

```
// sort the A[] in ascending order
ans = 0
for i → 0 to N-1 {
    if (i == A[i]) {
        ans += 1
    }
}
print(ans)
```

TC : O(N log N + N) = O(N log N)

SC : O(1)

Count Noble Integers

Given an int[n] of elements. Find the count of noble integers

A[i] is noble if count of elements smaller than A[i] is equal to A[i]

	0	1	2	3	4
A[] =	-10	1	1	3	10
# smaller =	0	1	1	3	4

	0	1	2	3	4	5	6	7	8
A[] =	-10	1	1	2	4	4	4	8	10
# smaller	0	1	1	3	4	4	4	7	8

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
A[] =	-3	0	2	2	5	5	5	5	8	8	10	10	10	14
# small.	0	1	2	2	4	4	4	4	8	8	10	10	10	13

Observation

If pval != val then # smaller = index

Pseudocode

```
// sort in ascending order.
smaller = 0
ans = 0

if( A[0] == 0 ) ans = 1        //   A = [0]

for i ⟶ 1 to N-1 {
    pval = A[i-1]
    val = A[i]
    if (pval != val) smaller = i

    if( val == smaller ) {
        ans += 1
    }
}

print (ans)
```

ans = ~~1~~ ~~2~~ ~~3~~ 4

	0	1	2	3	4
	0	1	1	3	10

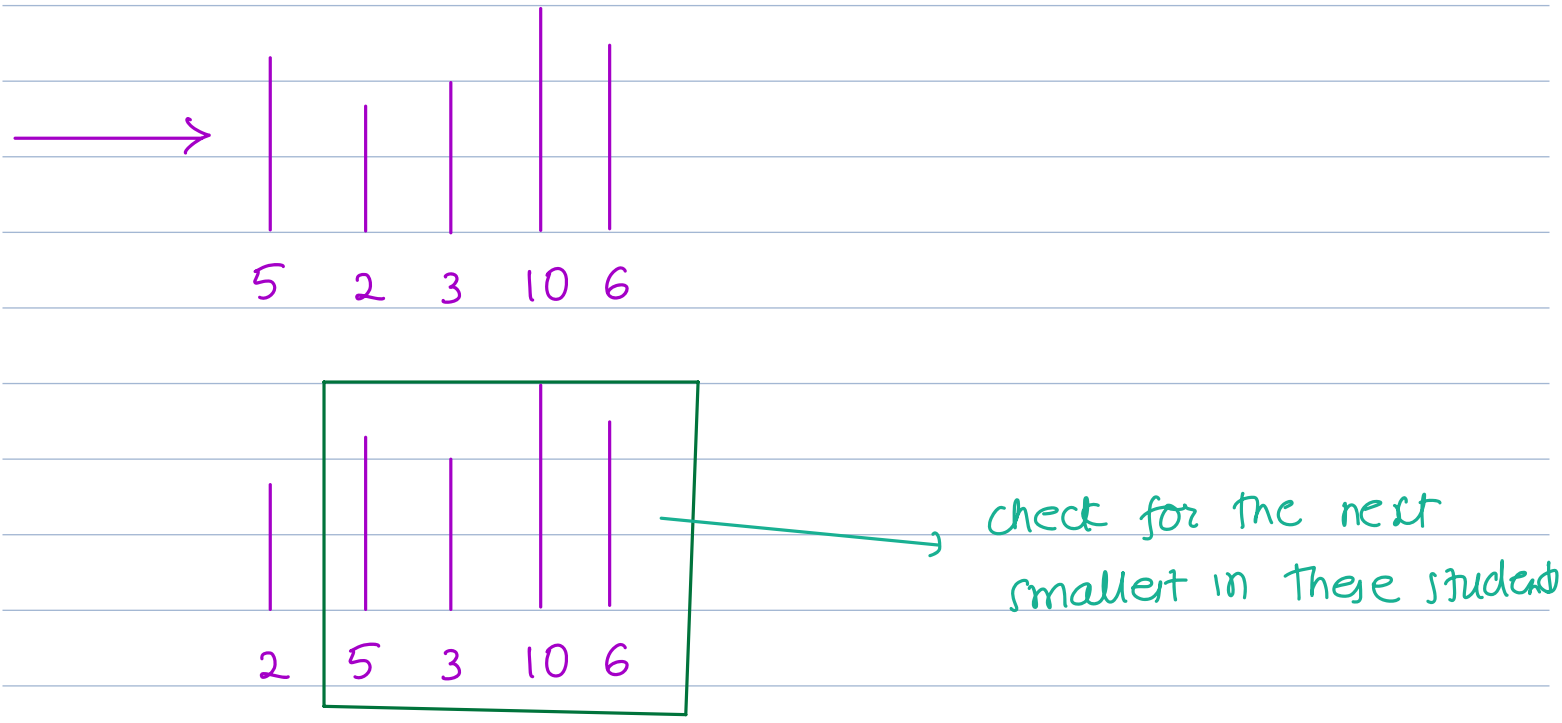
smaller = ~~0~~ ~~1~~ ~~3~~ 4

	0	1	2	3	4
	0	1	1	3	10
	0	1	1	3	4

TC : O(N log N)

SC : O(1)

Break : 10 : 37



---

values	5	2	3	10	6
index	0	1	2	3	4

Range [0 – 4]

minval = ~~5~~ 2

minIndex = ~~0~~ 1

values	2	5	3	10	6
index	0	1	2	3	4

Range [1 – 4]

minval = ~~5~~ 3

minIndex = ~~1~~ 2

values	2	3	5	10	6
index	0	1	2	3	4

Range [2 – 4]

minval = 5

minIndex = 2

values	2	3	5	10	6
index	0	1	2	3	4

Range [3 – 4]

minVal = ~~10~~ 6

minIndex = ~~3~~ 4

values	2	3	5	6	10
index	0	1	2	3	4

Pseudocode { Selection Sort }

```
for i ⟶ 0 to N-1
    minVal = A[i]
    minIndex = i

    // Find min in range [i to N-1]
    for j ⟶ i+1 to N-1 {
        if (A[j] < minVal) {
            minVal = A[j]
            minIndex = j
        }
    }

    // swap index i with minIndex
    temp = A[i]
    A[i] = A[minIndex]
    A[minIndex] = temp
```

3

TC : $O(N^2)$

SC : $O(1)$

Insertion Sort

A[] = 3 10 6 2 4

3 10 6 2 4

Iteration I → 3] 10 6 2 4

→ 3 10] 6 2 4

→ 3 6 10] 2 4

→ 2 3 ~~6~~ 10] 4

4

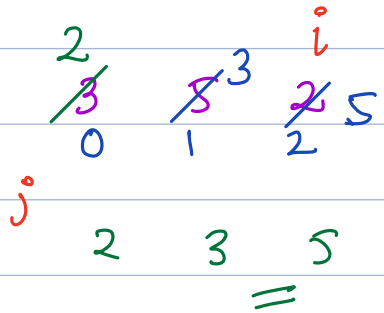
6 10

3 !> 4

Note

↓ all values greater than 4 will be shifted right.

Pseudo code




```
for i ⟶ 0 to N-1 {
    X = A[i]
    j = i - 1

    // shifting logic
    while (j >= 0 && A[j] > x) {
        A[j+1] = A[j]
        j--
    }

    A[j+1] = x
}
```

TC: $O(N^2)$

SC: $O(1)$

***

NOTE : Always use inbuilt sorting.