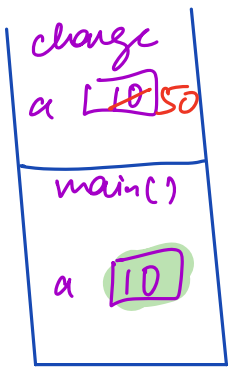
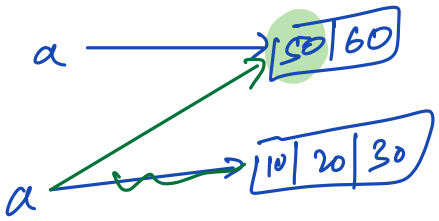
Sorting Basics

Quizzes





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

ex1 2 4 7 11 15 : sorted in Asc par = array values

ex2 15 9 6 6 2 0 : sorted in Desc par = array values

ex3 1 13 9 6 12 : sorted in Asc based on
#factors 1 2 3 4 6 #factors = par
(1,13) (1,3,9) (1,2,3,6) (1,2,3,4,6,12)

Sort in ascending order

- Java

  Arrays.sort(arr) - static array

  Collections.sort(arr) - ArrayList

- C++

  sort(arr, arr+n) - static array

  sort(arr.begin(), arr.end()) - vector

- Python

  arr.sort()

TC = O(NlogN)

Check out how to sort in descending order

Question 1

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

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

find min. cost to remove all elements.

eg a[] = [2 1 4]

		cost	
[2 1 4]	remove 2 :	7	(2+1+4)
[1 4]	remove 1 :	5	(1+4)
[4]	remove 4 :	4	
		16	

		cost	
[2 1 4]	remove 4 :	7	(2+1+4)
[2 1]	remove 2 :	3	(2+1)
[1]	remove 1 :	1	
		11	

eg a = [4 6 1]

remove 6	:	11	(4+6+1)
remove 4	:	5	(4+1)
remove 1	:	1	
		17	

eg $a = [3 \quad 5 \quad 1 \quad -3]$

remove 5 : 6 (3+5+1-3)

remove 3 : 1 (3+1-3)

remove 1 : -2 (1-3)

remove -3 : -3

2

---

observation : deleting ele by ele in decreasing order to get min cost ?

arr[4] = {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 $a + 2b + 3c + 4d$

$a \geq b \geq c \geq d$

$a + 4d$ (1, 4)

$1 + 4 \times 4 = 17$

$4 + 4 \times 1 = 8$

1. Sort in descending order
2. for each a[i], ans += (i+1) * a[i]

```
int totalCost(a[], n) {
    sort-desc(a)
    sum = 0
    for(i=0 to n-1) {
        sum += (i+1) * a[i]
    }
    return sum
}
```

→ TODO in your language
TC = O(NlogN)

TC = O(NlogN + N)
= O(NlogN)

SC = O(1)

a = [3 5 1 -3] → [5 3 1 -3]

sum = 1*5 + 2*3 + 3*1 + 4*-3
= 5 + 6 + 3 - 12
= 2

Question 2 — Noble Integers {Data is distinct}

Given N elements, calculate no. of noble integers.

An element in array is called noble iff

no. of elements < element = element itself

(no. of elements: count)

eg

	-1	-5	3	5	-10	4
#len	2	1	3	5	0	4

ans = 3

	-3	0	2	5
#len	0	1	2	3

ans = 1

Bruteforce

```
def countNoble(a[], N){
    ans = 0
    for(i=0 to n-1){
        c = 0
        for(j=0 to n-1){
            if(a[j] < a[i])
                ++c
```

TC = $O(N^2)$

SC = $O(1)$

```
    }
    if( c == a[i])
        ans++
  }
  return ans
}
```

Idea : sort the array in ascending order

sorted(a) : a[0] a[1] . . . . . a[i] . . . . . a[n-1]

all elements are [0,i-1] less than a[i]

count = i

if(a[i] == i) then a[i] is noble

```
def countNoble(a[], N){
    sort_asc(a)   → TODO
    ans=0
    for(i=0 to n-1){
        if(a[i] == i)
            ans++
    }
```

TC = O(NlogN)

SC = O(1)

return ans

}

dry run : -1 -5 3 5 -10 4

sort(a) -10 -5 -1 3 4 5

0 1 2 3 4 5

ans = 3

Question 3 Count Noble integers { Data can repeat }

eg -10 1 1 3 100

#len 0 1 1 3 4

ans = 3

eg -10 1 1 2 4 4 4 8 10

#len 0 1 1 3 4 4 4 7 8

ans = 5

eg

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
#len	-3	0	2	2	5	5	5	5	8	8	10	10	10	14
	0	1	2	2	4	4	4	4	8	8	10	10	10	13

ans = 7

obs1 : If ele. is coming for first time (if (a[i] != a[i-1]))

count of ele. len than = i

obs2 : If ele. repeats (if (a[i] == a[i+1]))

count of ele. len than will remain same

```
def countNoble (a[], N) {
    sort_asc(a)
    ans = 0
    count = 0   // ele. len than a[i]
    if (a[0] == 0) { ans++ }
    for (i = 1 to n-1) {
        if (a[i] != a[i-1]) ) {   // a[i] is coming first time
            count = i
```

```
}
else {   // a[i] is repeating
    // count will not change
}
```

→ else part not needed only written for understanding

```
if (a[i] == count)
    ans++
}
return ans
}
```

TC = O(NlogN)

SC = O(1)

Sort → Selection Sort

1. Pick smallest & place in front.
2. Pick second smallest & place in 2nd position.

⋮

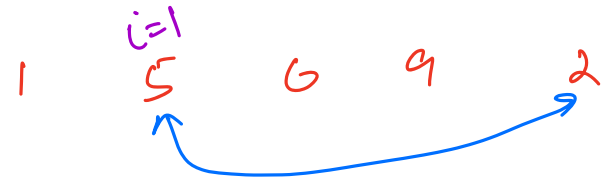
```
void selectionSort (a[], N) {
    for (i=0 to n-1) {
        minIndex = i
```

```
for (j = i+1 to n-1) {
    if (a[j] < a[minIndex]) {
        minIndex = j
    }
}
swap(a[minIndex], a[i])
}
```

$TC = O(N^2)$

$SC = O(1)$

minIndex = 0 1

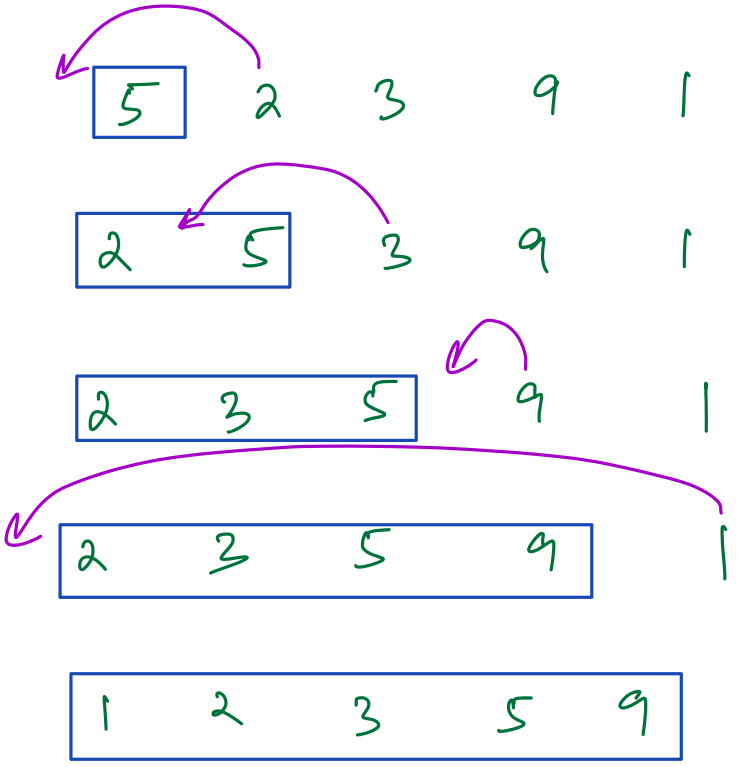


minIndex = ✗ 4

minIndex = ✗ 4

minIndex = ✗ 4

Sort – Insertion sort



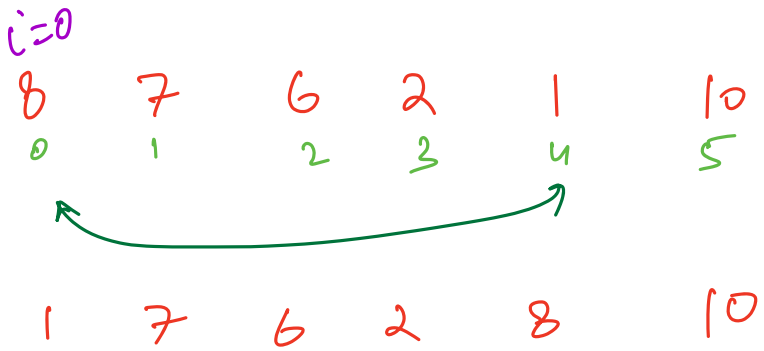

```
void insertionSort(a[], N) {

    for(i=1 to n-1) {

        j=i

        while(j>0 && a[j] < a[j-1]) {

            swap(a[j], a[j-1])

            j--

        }

    }

}
```

$TC = O(N^2)$

$SC = O(1)$

best case $TC = O(N)$

Selection sort



min = ~~0~~ ~~1~~ ~~2~~ ~~3~~ 4