02/02/22

DSML Intermediate - DSA

Sorting

Today's Content :

1. Intro - Sorting
2. Problem 1 : Min Cost To Remove All Elements
3. Problem 2 : Noble Integer 1 & 2
4. Problem 3 : Minimum Difference
5. Comparator - What, How, Where?
6. Problem 4 : Largest Number

Sorting = Arranging data in an orderly manner (1)

↙ asc. ↘ desc.

based on some parameter (2)

$L_1 = [5, 8, 12, 14, 16, 17]$ ✓

Based on values in asc. order!

$L_2 = [14, 11, 5, 3, 1]$

Based on values in desc. order!

$L_3 = [0, 0, -1, -1, 1, 1, 1]$ => $[0, 0, 1, 1, 1, 1, 1]$

Based on abs(vals) in asc. order.

$L_4 = [1, 5, 3, 9, 6, 10, 12]$ → 1, 2, 3, 4, 6, 12

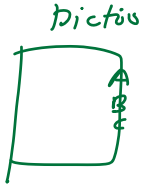
parameter = # factors.

$[1, 2, 2, 3, 4, 4, 6]$

L_4 is sorted based on # factors in asc. order

(1) Why need sorting?

- easy access for data.
- reduce access time.



Binary search

Python

Focus on Applications of Sorting.

3 sessions in Adv Batch.

① $l = [2,1,4,5,7,3]$

* $l.sort()$ → input list l itself is sorted. In-place sorting

$l \rightarrow [1,2,3,4,5,7]$

* $l.sort(reverse = True)$

$l \rightarrow [7,5,4,3,2,1]$

$l.sort(key = factors)$

def factors
...
return cnt.

② $l = [2,1,3,4]$

$l_2 = sorted(l)$ # return a sorted

$l_2 = sorted(l, reverse = True)$

$l_2 \rightarrow [1,2,3,4]$

$l \rightarrow [2,1,3,4]$ has not changed

T.C. = $O(N \log N)$

Q1 Amazon, DE Shaw

Given list of size N,

→ At every step remove 1 element.

Cost of removing = sum of all elements present till now (include element being deleted)

Find the min cost to remove all elts one by one.

---

l = [2,1]

Remove 1 → cost = 3, [2] → Remove 2 → cost = 2

Total cost = 3+2 = 5

Remove 2 → cost = 3, [1] → Remove 1 → Cost = 1

Total Cost = 3+1 = 4

Eg 2 [4,6,1]

[4,6,1] (6) : 6+4+1 = 11

[4,1] 4 : 4+1 = 5

[1] 1 : 1

11+5+1 = 17

* Observation (1):

Total Cost depends on order of operations.

* (2):

Element never contributes after its is deleted.

a_1, a_2, a_3, a_4

Delete a_4 : $a_1 + a_2 + a_3 + a_4$

a_3 : $a_1 + a_2 + a_3$

a_2 : $a_1 + a_2$

a_1 : a_1

Minimize this total cost

Total cost = $4 * a_1 + 3 * a_2 + 2 * a_3 + a_4$

a_1 → 4th max → 1st min

a_2 → 3rd max

a_3 → 2nd max

a_4 → 1st max

[2 4 1]

[4, 2, 1]

1* 2* 3*

Remove 4 : 4 + 2 + 1

2 : 2 + 1

1 : 1

$4 + 2*2 + 3*1$

$= 4 + 4 + 3 = 11$

[4,2,2]

4*1 + 2*2 + 3*2

= 4 + 4 + 6

= 14

Idea

O(NlogN) ← ① Sort array in desc. order

② Compute the min cost ← O(N)

$a_1, a_2, a_3, \ldots a_n$

$*_1 \; *_2 \; *_3 \; \ldots *_n$

$res = \sum_{i=1}^{n} (a_i) * i$

T.C. $O(N + N\log N)$

$= O(N \log N)$

S.C. $O(1)$

Q2. Google

Given N array elements, / list 'l'

Return cnt of noble integers in l.

l[i] is noble if:

(count of elements < l[i]) = l[i]

NOTE: All the elements are distinct.

---

i

	0	1	2
l =	-10	1	3
smaller than l[i]:	0	1	2

ans = 1

---

l = [3, 5, 0, 1]

	0	1	2	3
smaller than l[i]	2	3	0	1

ans = 2

l.sort() ascending order.

<5

l = [0, 1, 3, 5]

0 1 2 3

0, 1, 3, 5

Brute Force

For every elt

Check if noble or not.

TC. $O(N^2)$

No duplicates

1. Sort in asc. order ⟹ O(N log N)
2. count = 0
3. for i in range(0, N):
   if A[i] == i:
   count += 1 — O(N)
4. return count

Tc : O(N log N)

A = [-10, -5, 1, 3, 5, 6, 10]
(0 1 2 3 4 5 6)

[-10, -5, 1, 3, 4, 5, 10]
(0 1 2 3 4 5 6)

ans = 3

Q2 ii) With duplicates

TC : O(N log N)
SC : O(1)

0	1	2	3	4
-10	1	1	3	100

Ans = 3

cnt of elements < 1 = 1

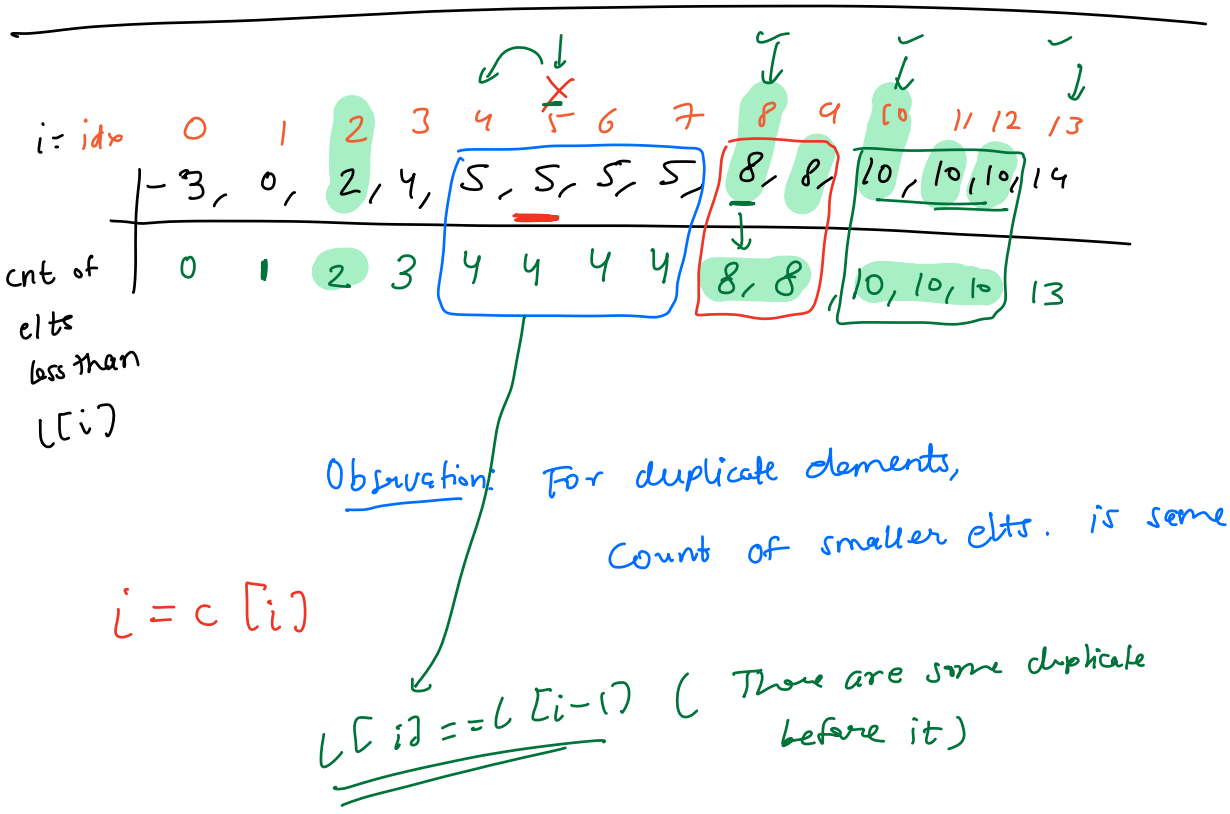
cnt of elements < 3 = 3

[-10, 1, 1]

Think
2 mins

0	1	2	3	4
-10	2	2	3	100

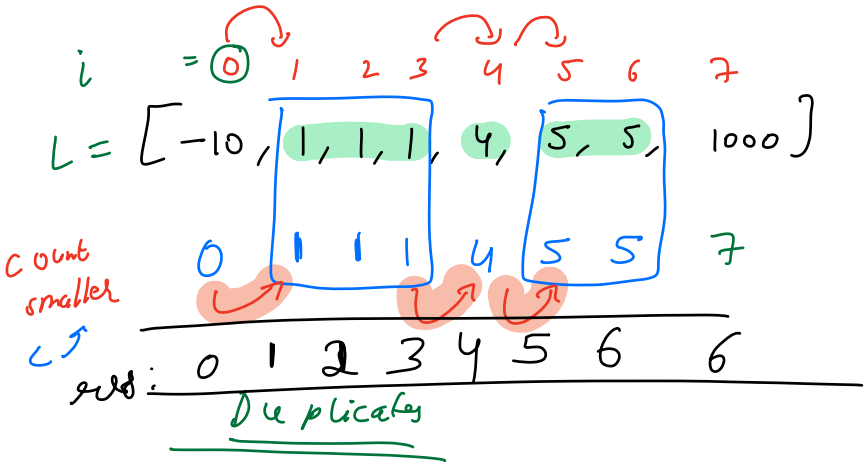
ans = 1



Observation: For duplicate elements, count of smaller elts. is same.

i = c[i]

L[i] == L[i-1] (There are some duplicate before it)



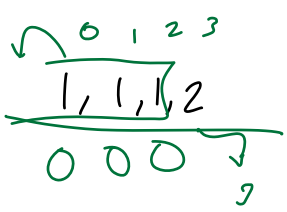
⇒ ① Sort in asc. order ⟹ O(N log N)

② res = 0
c = 0
if (arr[0] == 0): res = 1

③ for i in range(1, N):

if e[i] != e[i-1]: # non-duplicate
c = i
if c == e[i]:
res += 1

④ return res



Time = O(N log N)

Space = O(1)

Q 4. Minimize Difference

NOTE: $i \neq j$

Given N array elements, find the pair of indices (i, j) such that $|a[i] - a[j]|$ is minimized.

Return the min. value

$|x| = abs$

$|-1| = 1$

$|3| = 3$

$a = [5, 1, 2]$

$i = 0 \rightarrow n$

$j = i+1 \rightarrow n$

Brute Force

$(5, 1) = 4$ | $(1, 2) = 1$

$(5, 2) = 3$

All pairs

$O(N^2)$

ans = 1

Ex 2 $[9, 14, 21, 7, -3, 4, 26, 10]$

ans = 1.

$|9 - 10| = |-1| = 1$

Observation:

$[9, 14, 21, 7, -3, 4, 26, 10]$

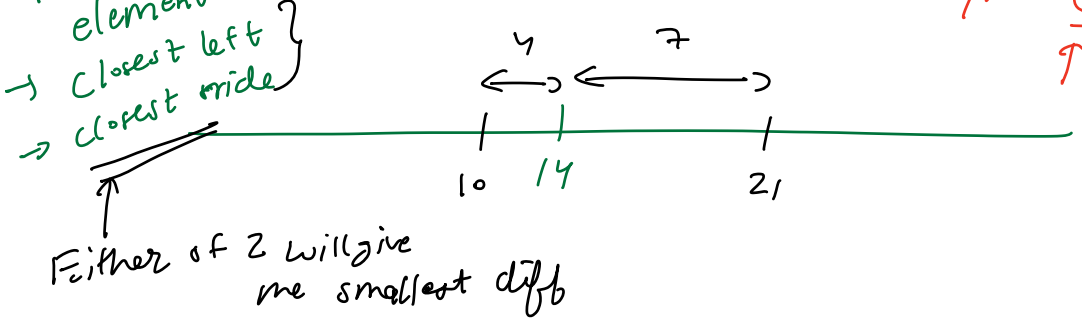
Fix one element

→ closest left

→ closest right

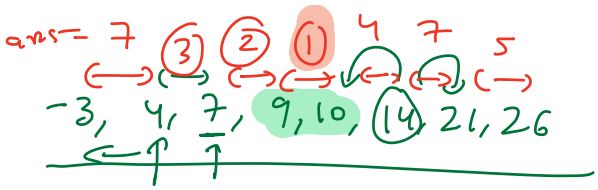
10/21

$|14 - 0|$

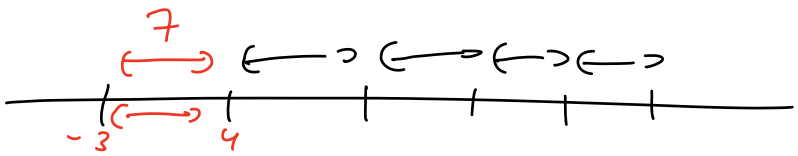


Either of 2 will give me smallest diff

① sort



② Find the min diff blw every adj. element.




```
→ l.sort()
→ ans = l[1]-l[0]      # assuming N>=2
    for i in range(2, N):
        ans = min(ans, l[i]-l[i-1])
    return ans
```

INT_MAX
(INF)
float('inf')

Q5:

* Largest Number >0

(+ve numbers)

Given an array of numbers, arrange the numbers such that they form the largest num.

l = [2, 3, 9, 0]

{ => 9320

[9, 3, 2, 0]

1) Sort in desc. order

2) Join the numbers

Eg 2. l = [3, 30, 34, 5, 9]

0 1 2 3 4

l.sort(reverse = True)

[34, 30, 9, 5, 3]

343095 X

4 3 2 0 1

9, 5, 34, 3, 30

954 3330

9534330

9434303

Input:

0 1

3, 34

0 1

3 34

1 0

34 3

✓ Comparator ⇒ allows to sort based on custom parameter

$l = [1, 2, 4, 10, 5, 6, 3]$

1, 2, 3, 4, 2, 4, 2

Given

```
def factors:
    return cnt
```

⇒ l.sort(key = factors)

1) order
2) parameter

$l = [1, 2, 5, 3, 4, 10, 6] \Rightarrow [1, 2, 3, 5, 4, 6, 10]$

1 2 2 2 3 4 4

⇝ If 2 elements have same cnt factors
smaller elt should come before

Remember

```
def myCmp(x, y):
    c1 = factors(x)
    c2 = factors(y)
    if (c1 < c2):
        return -1
    if (c2 < c1):
        return 1
    return 0
```

x < y ⟵ if (c1 < c2): return -1 } x, y

y < x ⟵ if (c2 < c1): return 1 } y, x

return True | -1 when x < y

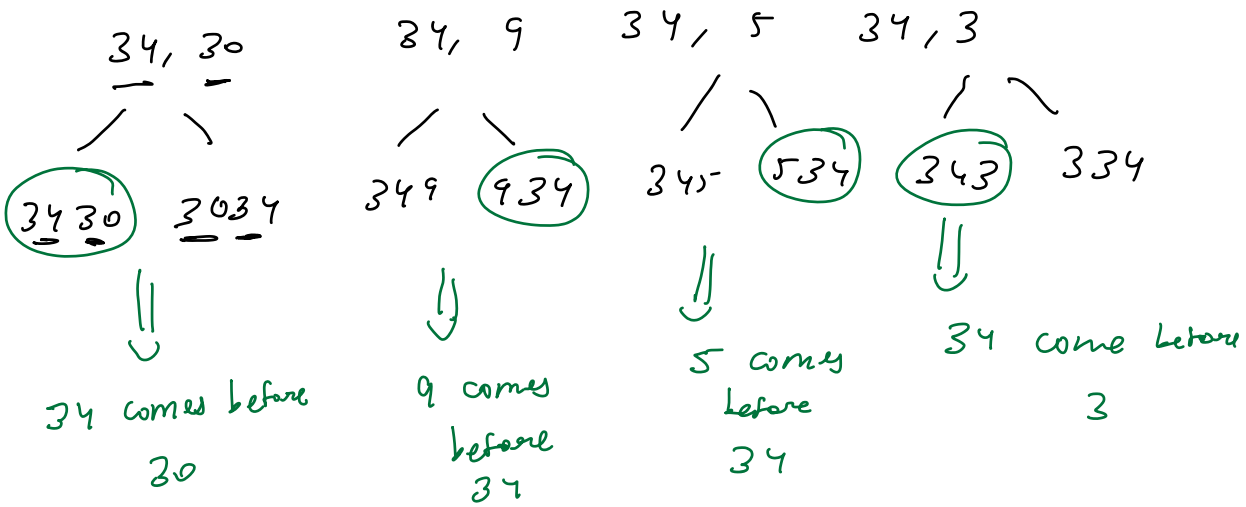
return False | 1 when x > y = y < x

0 | x = y

1st num comes before 2nd

1st num after 2nd num.

[34, 30, 9, 5, 3]



3, 3

3 3 3 3.

```
def myCmp(x, y):                # x & y are strings
    if x+y > y+x:               # xy > yx
        return -1
    if y+x > x+y:               # yx > xy
        return 1
    return 0
```

l = [3, 30, 34, 5, 9] (0 1 2 3 4) ⟹ O(NlogN)

⇒ l.sort(key = [ ] myCmp)

[9, 5, 34, 3, 30]

Code

l = [2, 1, 3, 4, 5, 6]

l.sort() ⟶ O(NlogN)

Merge sort

Quick sort

[1, 2, 3, 4, 5, 6]

By default

```
def comp(x, y):
    if x < y:
        return -1
    if x > y:
        return 1
    return 0
```