Intro to Sorting

Tim Sort ← array, vectors

python

Java : Collections.sort()

C++ : sort(vector.begin(), vector.end())

C : sort (arr, arr+n);

Quick Sort ←

T.C : $\boxed{O(n\log n)}$ ← $\boxed{O(n\log n)}$

Tim Sort
→ Insertion sort
→ Merge Sort

Question: Noble Integer

Given an array of integers, check if there is an element 'P' such that

No of elements > P = P

$O(n)$ $O(n)$ $O(n)$ $O(n)$

Ex 1   3   2   1   3 ✓

Let P = 1 ; 3 ✗

P = 3 ; 0 ✗

P = 2 ; 2 ✓   True

Ex 2   1   1   3   1

P = 3 ; 0 ✗

p = 1 ; 1 True

p in the Array

Ex 3

1 1 3 3

p = 1 ; 2 ✗

p = 3 ; 0 ✗

p = 2 ?

False

Brute Approach:

→ For every number of array, iterate and check no. of greater elements

Duplicate

T.C : n × O(n)

$\boxed{T.C : O(n^2)}$

(Sorting)

Approach 2 : (Sorted the array)

sorted, No Duplicate (N) = 7

0	1	2	3	4	5	6
1	2	4	5	6	8	10

(i)

N = 7

i = 2

4

n − i (7 − 2) = 5 ✗

$\boxed{n - i - 1 \quad (7 - 2 - 1) = 4}$

n − i + 1 (7 − 2 + 1) = 6 ✗

if(a[i] == n − i − 1)

True;

for (a[i], n − i − 1)

a[i] == n − i − 1)

(a[i] == n − i − 1)

→ O(n)

Ex

0	1	2	3	4
4	4	4	7	10

N = 5

(5 − 4 − 1) ≠ 0

False

[i, (4)

2

i, No. of elements to right = n − i − 1

(5 − 2 − 1) = 2

(n − i − 1 == A[i])

Return true

$\boxed{O(n^2)}$

if(a[i] == a[i+1])
SKIP

// Find last occurrence

(Hash Set)

2) 1 2 4 4

T.C: O(nlogn) + O(n)

=> $\boxed{O(nlogn)}$

Question: Find pairs of numbers having minimum difference.

-> No Duplicates => 0

-> Return all the pairs in sorted order

(a, b) => $\boxed{abs(a-b)}$

	0	1	2	3	4	5	6
A =	3	5	7	2	1	4	6

=> Ans = [{1, 2} {2, 3} {3, 4} {4, 5} {5, 6} {6, 7}]

vector<pairs>

(4, 3)

(1, 2) (4, 5)

(1, 2) (1, 3) = -

(1, 2), (1, 3)

5 6 1 3

=> (3, 5)
(3, 7)
(3, 2)

Brute Force:

A =	(3)	5	7	2	1	4	6

$\boxed{T.C: O(n^2)}$

Consider all the pairs

(3, 7)

[abs(a-b)]

indices

min ...

ans = ~~[(3,5)~~ [(3,2) , (3,4)]

$\Rightarrow$ T.C : (n-1) + (n-2) + (n-3) + ... 1

2) $\frac{n(n-1)}{2}$ = $\boxed{O(n^2)}$

map : { 10 : 0, 30 : 1, 14 : 2 ... }

0	1	2	3	4	5	6
10	30	14	17	19	27	11

Approach 2

(Greedy)

Sort :

E1

10	30	14	17	19	27	11
0	1	2	3	4	5	6

(10, 17) = 1

mp[14];

(10, 11)

arr = [<10,0> , <30,1> <14,2> <17,3> , <19,4> <27,5> <11,6>

(17, 19)

$\Rightarrow$ arr [<10, 0> , <11, 6> , <14, 2> . . . . >

index

(17), 2)

(17, 14) = 3

pair <a[i], index >

Min difference will occur only with closest elements $\rightarrow$ (11, 14)

$\Rightarrow$ O(n)

S = [10 11 14 17 19 27 30]

0 6 2 3 4 5 1

(10, 11) (11, 10)

(17, 19) (19, 27) (27, 30)

[1] (14, 17)

(10, 11)

10 : 11

11 : 10, 14

14 : 11, 17

(11, 14)

(14, 17)

sorted order

(sort the pairs)

ans = [ ]

{ O(nlogn) }

T. C : O(nlogn) + O(n)

= $\boxed{O(nlogn)}$

pair <int, int >

0	1	2	3
10	23	14	14

{ 10 : 0, 23 : 1, 14 : <2 > }

... Increasing order of Indices

Return ... ,

Question: check if we can reorder the array such that they form an Arithmetic Progression

AP: Series of numbers where diff between any 2 consecutive numbers is same

Ex: $\underbrace{2 \quad 4}_{2} \quad \underbrace{\quad 6}_{2} \quad \underbrace{\quad 8}_{2} \quad \underbrace{\quad 10}_{2}$

Ex: $\underbrace{11 \quad 14}_{3} \quad \underbrace{\quad 17}_{3} \quad \underbrace{\quad 20}_{3} \quad \underbrace{\quad 23}_{3} \quad \underbrace{\quad 25}_{2}$

$a \quad a+d \quad a+2d \quad a+3d \quad \dots \quad a+(n-1)d$

$d < 0$

$7 \quad 4 \quad 1 \quad -2 \quad -5 \quad -8 \quad \boxed{-11}$ 7th element

$a = 7$

$d = -3$

$d < 0$

$a + (n-1)d$

$7 + (7-1)(-3)$

$7 + (-18) = \boxed{-11}$

$d > 0$

$-11 \quad -8 \quad -5 \quad -2 \quad 1 \quad 4 \quad 7$

$a = -11$

$d = -3$

Solution 1:

2) Sort the Array $\rightarrow O(n)$

Ex : 8 6 10 4 2

Sort : 2 4 6 8 10 — sorted Array

(2)

A = 0 0 0 0 0

Increasing

strictly Increasing

T.C : $\boxed{O(n\log n)} + O(n)$

$= \boxed{O(n\log n)}$

0 0 0 0 0

a = 0

d = 0

AP

d > 0

Solution 2 :

a   a+d   a+2d   a+3d . . . . . .

Min element   2nd Min

O(n) → 1st Min & 2nd Min

(N elements)

a = min1

d = min2 − min1

$\boxed{\text{min} = a}$

max = a+(n−1)d

a = min1

d = (max−min)/(n−1)

a, a+d, a+2d, a+3d, . . . a+(n−1)d

(a, a+d, a+2d, a+3d ..)

(Hash set) ⇒ O(1)

d

→ Compare Two Array : O(n)

→ Hash the Array ← O(n)

T.C : O(n) + O(n) + O(n) + O(n)

Put into hashset   min1   min2

O(n log n)

O(n^2)

$\boxed{\text{T.C : } O(n)}$

A1 = 20 14 11 17 23

min1 = 11
min2 = 14

a = 11
d = 3 } → O(n)

A2 → 11 14 17 20 23 } O(n)

$\boxed{O(n^2)}$

→ sorting
→ Heap
→ B.ST

Ex 0 0 0 0

Duplicate
Frequency

$\boxed{a = 0, d = 0}$

a a a a
$\boxed{d = 0}$

(Distinct Numbers)

0 11 2 0, 1, 1, 2 × d

Question: Sort Colors
Given array of 0's, 1's, 2's
Sort the array.

A = 1 1 0 0 2 1 0 0 2

count_0 = 0
count_1 = 0
count_2 = 0

Approach 1:
sorting Algorithm
O(nlogn)

$\boxed{0\ 1\ 2}$

Approach 2:

→ Count No of 0's, 1's, 2's O(n)
to find count O(n)

2 Traversal
→ 1 Traversal

→ 0's | 1's | 2's

T.C: O(n) + O(n)

S.C: O(1)

T.C = O(n)

1 Traversal . <=> Two-Pointer Technique

Question: k^{th} Min element. (Duplicates)
Array is not sorted

A = 2 0 5 7 3 8 4 1
Sort = 0 1 2 3 4 5 7 8
0 1 2 3 4 5 6 7
K = 3, 2
K = 5, Ans = 4

$\boxed{K = 3}$ A[2]

K = 5, A[4]

A[4]

Naive Approach:

=> Sort array
=> a[K] or a[K-1]

(Distinct Elements)

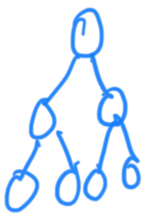
→ Heap (MaxHeap)

→ B.S.T

(Swaps) Sorting Algo.

A = {5 1 1 1 2 2 3}
A = 1 1 1 2 2 3

$\boxed{K = 2}$ => $\boxed{k^{th} \text{ element in sorted Array}}$



T.C: O(n log n)

Constraint: Find the k^{th} element such that No. of swaps (1 swap) is ≤ K

≤ K

A = ~~3~~1 4 5 ~~1~~3 2

a[K-1]

A = 1 2 3 4 5

A = 5 1 2 3 4 (5 swaps)

if K = 1 — 1st min [0]

(K = 2) — 2nd min element [1]

O(n) swaps = 1 2 O(n)

3rd Selection sort

A = 2 0 5 7 3 8 4 1

1 iteration?

0 | 2 5 7 3 8 4 1 — K = 1

0 1 | 5 7 3 8 4 2 — K = 2

K times

(n)

0 1 2 7 3 8 4 5 — K = 3

(K times) Kth element in sorted Array

$\boxed{T.C: O(n \times K)}$

Swaps = K

ith iteration, ith smallest element in right place.

what if K = n? Sorted Array

$\boxed{n \log k}$

BBST: $\boxed{O(\log n)}$

Algorithm: $\boxed{\text{Selection sort}}$

Selecting the min element and place it in right position

if K = n.

K = 3

$\boxed{\log k}$ ⇒ $\boxed{\text{Heap}}$

(Heap)

10 15 17 16 11

17 16 14

Question: Given array, Remove all elements ...

Cost of removing an element = sum of all elements present before removing it.

Task: Remove all elements with min cost.

$A = \quad 2 \quad 3 \quad 1 \quad 4$ — Minimise the Cost

[2, 3, 1, 4] $\xrightarrow{\text{Remove 2}}$ [2+3+1+4] (Max) 10

[3, 1, 4] $\xrightarrow{\text{Remove 3}}$ 3+1+4 (Max) 8

[1, 4] $\xrightarrow{\text{Remove 1}}$ [1+4] (Max) 5

[4] $\xrightarrow{\text{Remove 4}}$ 4 (Max) 4

$\boxed{27}$ → Minimise this Cost

$\boxed{DP}$ → $\boxed{Greedy}$

1st step: Sum of array element.

2nd step:

[2, 3, 1, 4] $\xrightarrow[2+3+1+4]{\text{Remove 4}}$ 10

[2, 3, 1] $\xrightarrow[2+3+1]{\text{Remove 3}}$ 6

[2, 1] $\xrightarrow[2+1]{\text{R 2}}$ → 3

[1] $\xrightarrow[1]{}$ → 1

$\boxed{20}$

$\boxed{\text{cost} = 10 + 6 + 3 + 1}$

Sum = ~~10~~ ~~6~~ ~~3~~

A 2 3 1 4

A = [1 2 3] 4

cost = 20

T.C : nlogn + n

= $\boxed{O(n\log n)}$

Paradigm : Greedy Paradigm

Where can you think of applying sorting.

Order of index does not matter

A = 3 5 1 6 7

A = 1 3 5 6 7

Subarray, Subsequence ⇒ Order of elements

A = 2 3 1 4

A = 1 2 3 4

Arrays ⇒ Last Leetcode

cost = 0+10+6+3+1

sum = 10

6 3 10

T.C: O(nlogn) + O(n)

$\boxed{O(n\log n)}$

O(n)

cost = 1

1+2+3+6+7

(0,0) → (3,3)

Right

working

Min cost

Down

Greedy ✗ W

A =

6

3 × 3

Cost = 1 + 5 + 1 + 1 + 7

optimisation problems

Allocat

Arrays in Python ⇒ Dynamic Array

8 bytes

[1 , 'abc' , 3.24 , c]

4+3

4+3+8

[Dynamic Array]

1 char → 1 byte

0 1 2 3

Mapping

A =

T.C: O(n)

Array ⇒ N element

P ⇒ [0 . . . . N-1]

N = 10

A = 4 2 3 1 9 6 7 0 18 14

0 1 2 3 4 5 6 7 8 9

count = 0 0 0 0 0 0 0 0 0 1

> 9

greater = 12 + 1 = 3 4 5

N = 10

(9 = = 2)

② F↑

(≤ N-1)

P > N-1 ∧

element > 9

if (A[i] > N-1)

if (A[i] ≤ N-1)

[0 . . . N-1]

P = [0 . . . . N-1]

[0 . . . 9]

No-of

A = 1 2 3 4 5 6 7 8

(3)

(n-1)

p : (=> p)

[0 ... N-1]

=> (Advanced Model)

A = -10 80 14 17 19 27 11

A = -10 0 1 2 3

min = ~~INT_MAX~~

~~10~~ 1

Ans = [(0,1) , (1,2) , (2,3)]

(0,1)