Revisit last quest^n:

Q.6 Given array of +ve ints, find length of longest consequtive sequence which can be formed using nos. from array.

eg. a = [9, 3, 10, 4, 20, 2]

2 3 4 $\xrightarrow{9\ 10}$

Ans = 3.

① Brute: → Sort the array, linear traversal

sort → [2 3 4 9 10 20] → O(N log

linear count → O(N)

⇒ T.C ⇒ O(N log N).

② ⇒ O(N)

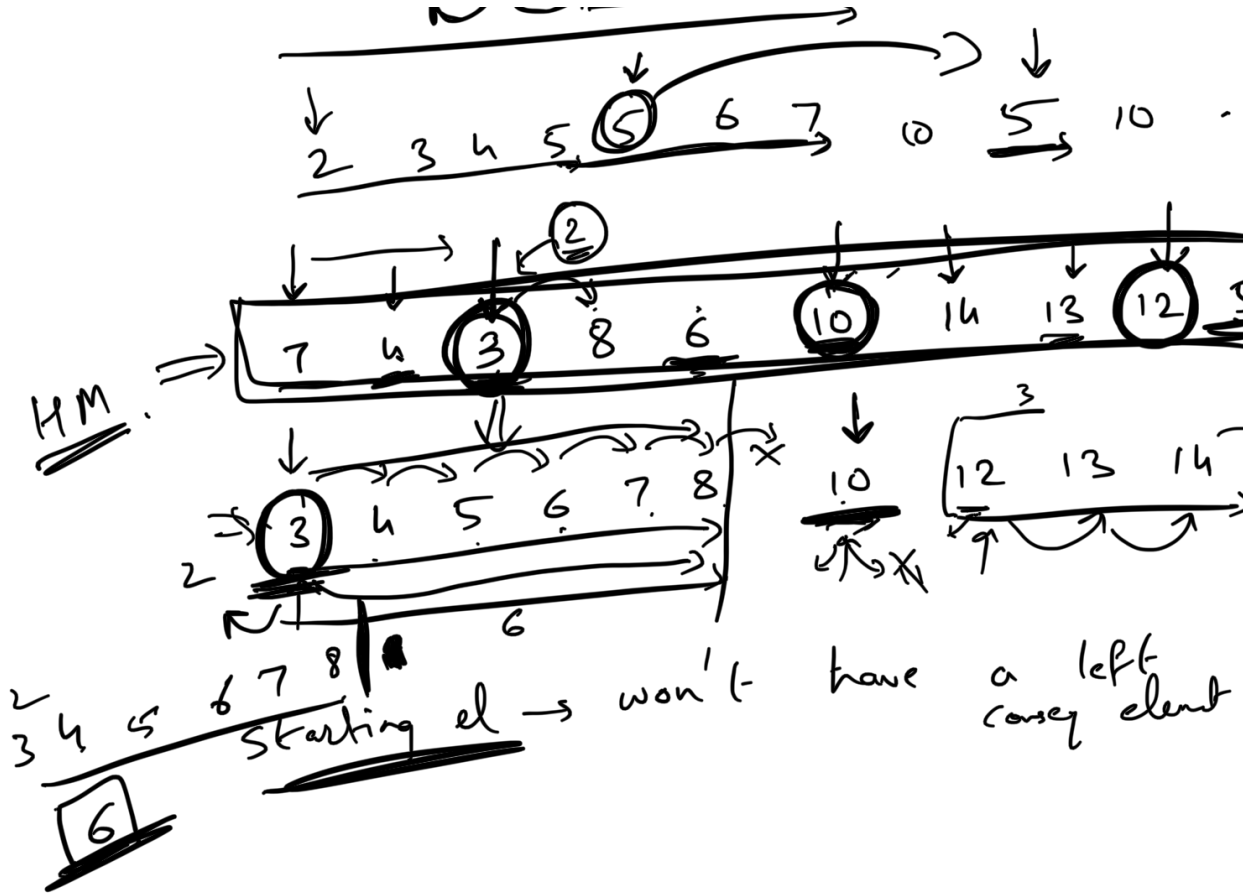
Obs: [(2 3 4) (9 10) (20)] 3

3

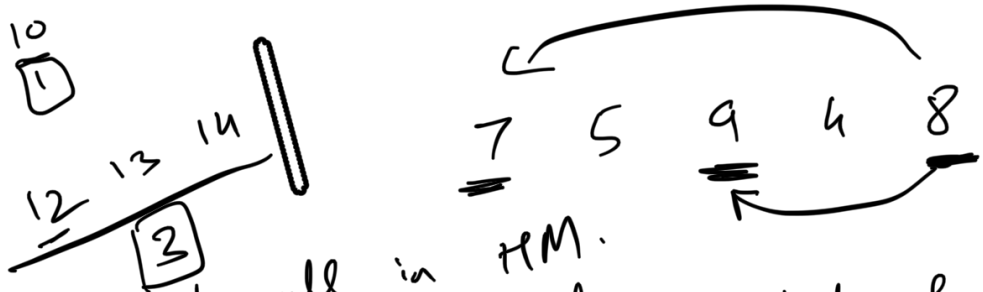
0

→ Every element can be present in exactly 1 consequtive seq.

2 3 4 5 6 7



Starting el → won't have a left consec elem



Put all in HM. ← O(1)

Check: ① If curr el is start of seq (curr-1 is not present in HM)

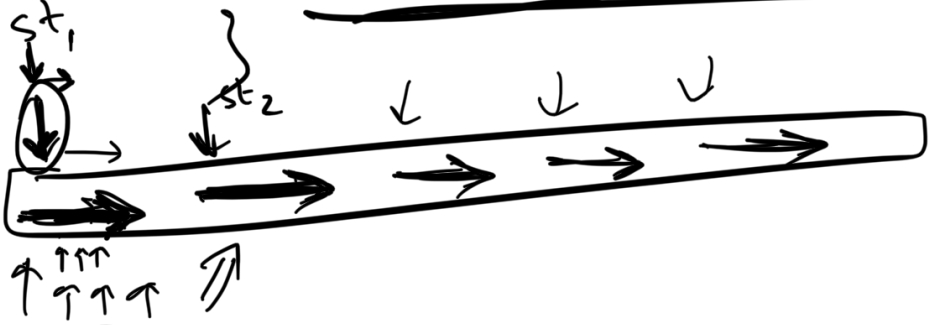
② If next elem is present → O(1)

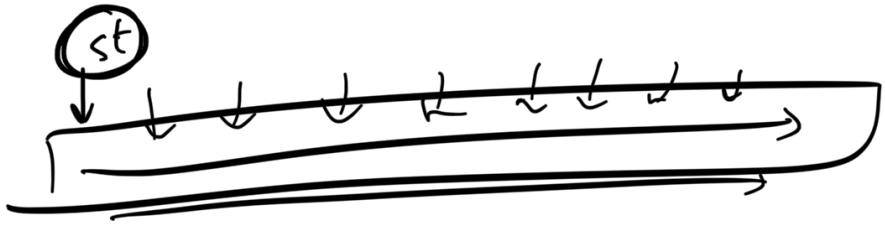
Pseudo:

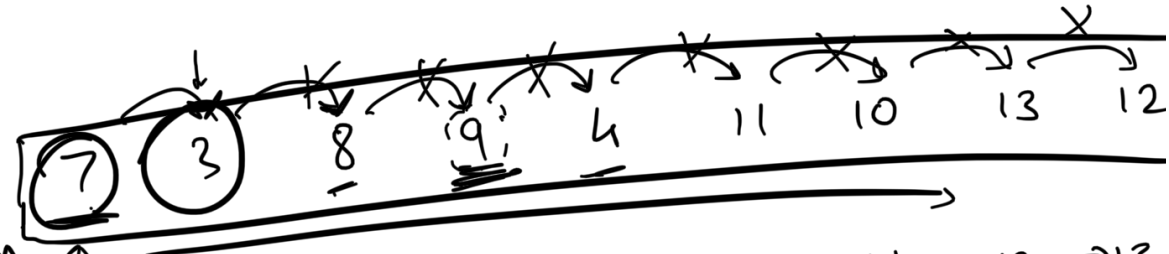
```
for (i=0 → N)                    ⟹ O(N)
    HM.insert(a[i])

                i=0 → H.M. size
→ for (i=0 → N)                  →
    cnt=0
{   // check if A[i] can be start of
    if (!HM.contain(a[i]-1))
```

```
{
    cnt = 1;
    start = A[i]
    while ((start+1) is in HM)
    {
        cnt++;
        start++;
    }
}
ans = max (ans, cnt)
```







H.M

7
3
8
9
4
11
10
13
12

7 -> 8 -> 9 -> 10 -> 11 -> 12 -> 13

curr_ans = $\boxed{7}$

3 -> 4

2.

T.C => O(N).

S.C => O(N).

H.M

7
3

6 -> 7 -> 8 -> 9 -> X

0
6
3
9
4

3 → 4

8 7 6 5 4 3 2 1

8
7
6
5
4
3
2
1

1 → 2 → 3 → 4 → 5 → 6 → 7 → 8

2N.

O(N).

Intro to Sorting

(specific sort to be taken in adv. but

will be seeing usage of sort() fn.

Q.iz T.C. for sort funct^n → O(NlogN)

C	C++	Java	Pu
qsort(a-size)	sort(begin, end)	Collection.sort()	l.s

(tim sort → hybrid sort

Q.1 Noble integers

Given array, check if there exists an integer P in the array such that (no. of element greater than P] = P

eg. [3, 2, 1, 10] Ans ⇒ True

0 ele.

1 ele 2 ele

eg. [1, 1, 3, 1] ⟹ True.

[1, 3, 3, 3] ⟹ False.

eg. [2, 8, 4, 6, 5, 1, 10]

p = count nos greater than p.

⇓ sort

[1 2 4 5 6 8 10] ⟹ Tr

4 elements.

N-1

Sort( ) . i

for (i = 0 → N)

if (N-1-i = a[i])

return true

wrong for duplicates.

return fal

[1 3 4 4 5 8 9]

[1 4 5 8 9 9 10 12 1

. (A[i] == A[i+1])

if (A[i] - ...)

i + T i

Sort (A.) → O (N log N)

for (i = 0; i < N-1; i++)

{ if (A[i] == A[i+1])

continue.

if (A[i] == N - i - 1)

return true;

T.C ⇒ O(N log

}

return false;

0 1 2 3 4 5 6

1 2 4 5 6 8 10

↑ i

i = 2

N = 7

7 - 2 - 1

N - 1 - i

T.C ⇒

[1 4 4 5 7 8] O(N × log N + N

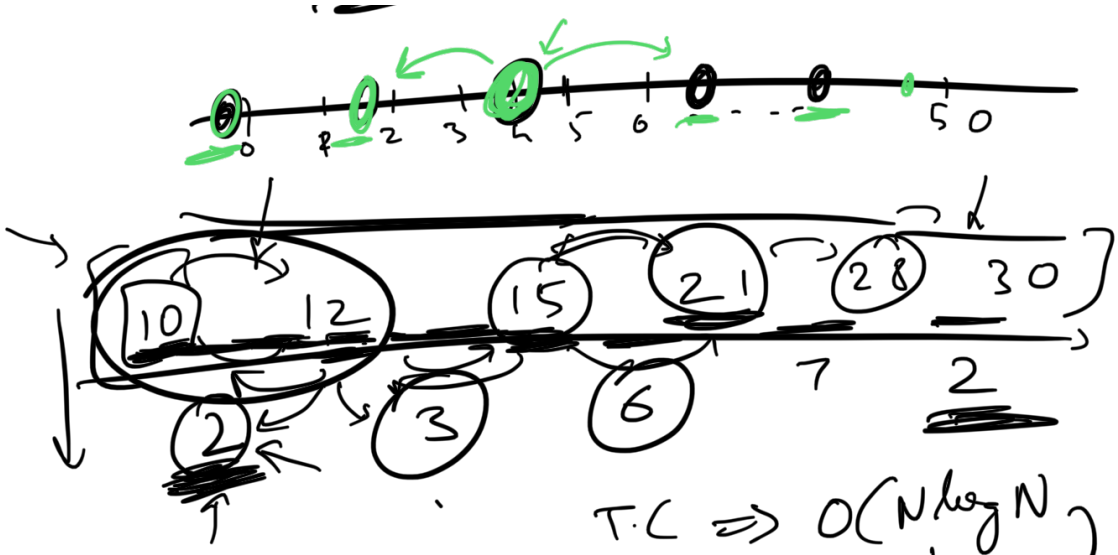
⇓

O (N log N).

Q.2 Given array, find pair of numbe with minimum difference.

eg. [10, 20, 21, 28, 12, 15]

Ans → (10, 12) → 2



T.C ⇒ $O(N \log N + N)$

T.C ⇒ $O(N \log N)$

Q.3 Microsoft

Given array, check if we can form an A.P using all values of arr

A.P ⇒ difference b/w neighbour is same

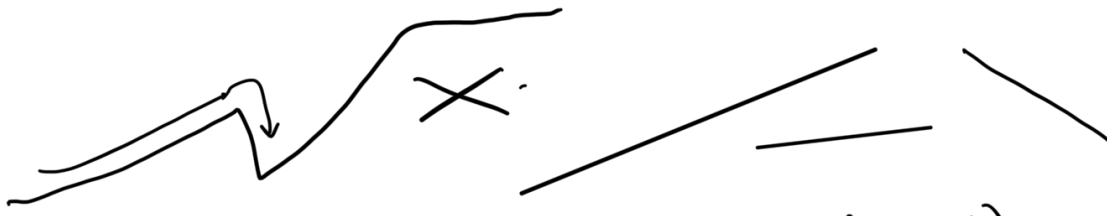
(2, 4, 6, 8, ...)

a, a+d, a+2d, ...

↳ (Non-decreas / non-increasing)

eg. [3, 5, 1]

→ 1, 3, 5 ⇒ $\boxed{\text{True}}$



sort & check diff → $O(N \log N + N)$

(2 7 4 4 3 6 5)

all ↑ sort

[2 3 4 5 6 7]

[4 4 4]

Using hashmap.

1 min 2nd min

AP ⇒ [a, a+d, a+2d, a+3d, ... a+(n-1)

[2 8 4 6] → H.M.

2 → 4 → 6 → 8

d = 2nd min − min

N

T.C ⇒ O(N).

S.C ⇒ O(N).

[7 5 10 9 3] N = 5

min ⇒ 3
2nd min ⇒ 5 } ⇒ O(N).

H.M

[7 5 11 9 3]

5 − 3 = 2

3 → 5 → 7 → 9 → 11 → True

False.

4

[7 5 11 9 3 100] N = 6

False.

3 d = 2
5

[3 → 5 → 7 → 9 → 11]

5

2 2 4 6 8 10 N = 6.

d = 0 [2 → 2]

d = 1 ⇐ [2 1 3 5] ⇒ 10.

$\frac{n}{2}\left[2a + (n-1)d\right]$

d = 1 ⇐ [2 1 3 5] ⇒ 10

Q.4. Given array of colors → 0 1 2

sort the array.

A ⇒ [1, 0, 2, 2, 1, 1, 0, 0, 1, 2] ⇒ Sort

eg.

Brute → sort → O(N log N). X

0 < 1 < 2

out Sort

[0, 0, 0, 1, 1, 1, 1, 2, 2, 2]

Count Sort

[0] → 3
[1] ⇒ 4
[2] ⇒ 3

Fill gives ans

[0, 0, 0 | 1 1 1 | 2 2 2]

⇓

T.C ⇒ O(N)

~~(using 2-pointers also)~~

HM[0] ++

A[i] → O(1)

HM[i]

Q.5. Given (distinct elements) array, return K^{th} minimum element.

eg. [2, 0, 5, 7, 3, 8, 4, 1] K = 3

Ans = 2.

Sort → return A[K-1] ⇒ T.C ⇒ O(N log n

⇓ Condition

Total no. of swaps ≤ K.

[0, 1, 2, 3, 4, 5, 7, 8]

↳ 1 swap with min always.

⇓ T.C ⇒ O(KN)

if done $\Rightarrow$ Bubble/select sort

N times. $\Rightarrow$ Sorted array $\Rightarrow$ $O(N^2)$

Find k^{th} min using $\leq k$ swaps $\Rightarrow$ [2 5 1 12 3 8]

Find min $\Rightarrow$ 1 $\Rightarrow O(N)$

$k=3$ [1 5 2 12 3 8]

find min $\Rightarrow$ 2 $\Rightarrow O(N)$

[1 2 5 12 3 8]

Find min $\rightarrow$ 3

[1 2 | 3 12 5 8]

k — N times $\rightarrow$ sorted array N times

$O(kN)$ $\qquad$ $O(N^2)$.

Q.6. Given array, remove all elemets.
Cost of removing an element
= sum of array elem before remov
Minize the total cost.

eg. [2, 3, 1, 4]

remove 2 $\Rightarrow$ Total cost $= 2+3+1+4$
$= 10$

[3, 1, 4]

remove 3 $\Rightarrow$ $3+1+4 \Rightarrow 8$

[1, 4]

remove 4 ⇒ 1+4 = 5

[1]

remov 1 ⇒ 1

Total cost = 10+8+5 = $\boxed{24}$

minimize th

Obs: [2, 3, 1, 4]

remove 4 Total cost = 10.

[2 3 1] ← minimix.

Greedily choose max

rem 3. Total cost ⇒ 6.

[2, 1] ⇒ cost ⇒ 3

rem 2

[1]

rem 1 ⇒ 1

$\boxed{20}$

[7, 1, 5, 2]

sum

[1, 2, 5, 7]

Algo: Sort array & start removing largest

T.C ⇒ O(N log N)

N

sort(); → O(N log N)

sum = 0, ans = 0;

for (i=0 → N) sum += a[i] → O(N)

for (i = n-1; i ≥ 0; i--)

```
for ( i = ... )
{
    ans += sum
    sum -= a[i]
}
return ans ;
```

$\rightarrow O(n)$

Q.1 Given array, print sort/arrange array in sinosidal form

[4, 10, 2, 3, 0, 8]

[2, 3, 9, 1, 5, 6]

2 3 9 1 5 6

① → sort the array.

[1 2 3 5 6 9]

1 9 2 6 3 5

for (i = n-1 ; i < i ; i++, i--)

```
for ( i=0, j=n-1 ) ; ; ) ; )
{   print (a[i], a[j]);

}
```