Todays Content

1. longest subset inc order
2. Closest pair

10:35

28 Given ar[N] elements, calculate length of longest subset which can be re-arranged in strictly increasing order by 1.

Note: Pick any no: of ar[] elements in any order.

Constraints

$1 <= N <= 10^6$

$-10^9 <= ar[i] <= 10^9$

	0	1	2	3	4	5	6	7
Ex1: ar[] = {	-1	8	5	3	10	2	4	9 }

ans1: { 8 10 9 } = { 8 9 10 } l=3

ans2: { 5 3 2 4 } = { 2 3 4 5 } l=4

return 4.

	0	1	2	3	4	5	6	7	8	9
Ex2: ar[] = {	3	8	2	1	9	6	5	6	7	2 }

ans1: { 8 9 5 6 7 } = { 5 6 7 8 9 } l=5

ans2: { 3 2 1 2 } = { 1 2 2 3 } * Not inc by 1.

ans3: { 3 2 1 } = { 1 2 3 } l=3

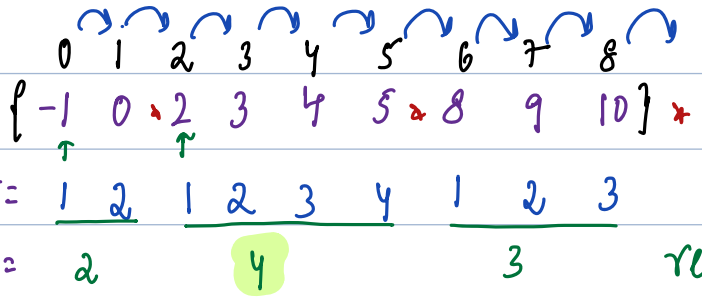
return 5.

Idea2:

Dry Run: Sort arr[] & cnt adj elements to get longest inc sequence

0	1	2	3	4	5	6	7	8
-1	8	5	3	10	2	4	9	0

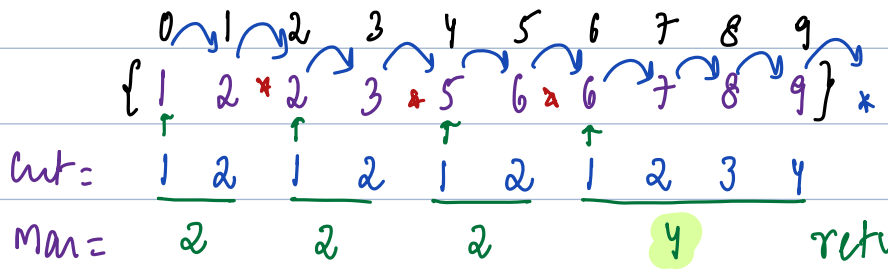
arr[] = {-1 8 5 3 10 2 4 9 0}



0	1	2	3	4	5	6	7	8	9
3	8	2	1	9	6	5	6	7	2

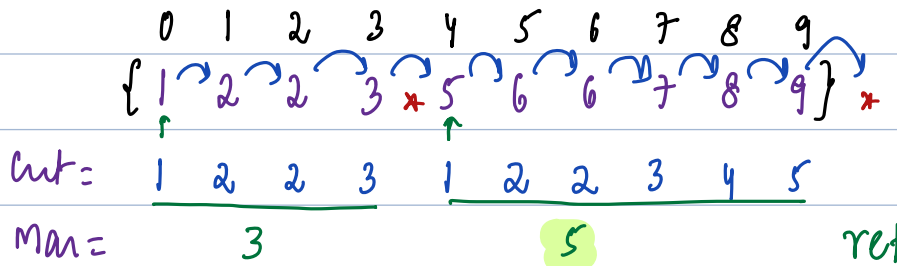
arr[] = {3 8 2 1 9 6 5 6 7 2}

Dry Run1:



- Reason: We stop seqnce when elment repeats, this is a mistake we continue when elment repeats, we don't inc count.

Dry Run2:



```
int manlen(vector<int>& ar){ TC: O(NlogN + N) = O(NlogN)
    sort(ar.begin(), ar.end());
    int ans = 0, N = ar.size();
    int c = 1;                    → i == N-1 : stop
    for(int i = 0; i < N-1; i++){ # i = N-1, ar[i+1] - ar[i] # Error.
        # cmp ar[i] with ar[i+1]                ar[N] - ar[N-1]
        if( ar[i+1] - ar[i] == 1){
            c++;
        }
        else if( ar[i+1] - ar[i] == 0){
            continue;
        }
        else{
            # ar[i] * ar[i+1]
            ans = max(ans, c);
            c = 1;
        }
    }
    return max(ans, c); # last sequence will not be be compared to
}                         ans in loop, hence we compare it outside
                          loop & return biggest
```

Dry Run 3:

	0	1	2	3	4	5	6	7	8	9
	{ 1	2	2	3	5	6	6	7	8	9 }
cut =	1	2	2	3	* 1	2	2	3	4	(5)

ans = 0 ans = max(c, ans)

ans = 3

Idea2: For every ar[i]
Calculate length of longest sequence starting from ar[i].

Ex1:

	0	1	2	3	4	5	6	7	8
ar[] = {	-1	8	5	3	10	2	4	9	0 }

ans = 4

Start:

-1 : 0 1 * 2
8 : 9 10 11 * 3
5 : 6 * 1
3 : 4 5 6 * 3
10 : 11 * 1
2 : 3 4 5 6 * 4
4 : 5 6 * 2

Hint: Store all ar[] in hashset, it can search if an element exists or not in O(1)

9 : 10 11 * 2
0 1 * 1

Ex2:

ar[] = { 9 7 6 8 10 } Insert in ts : { 6 8 9 10 7 }

Start:

9 : 10 11 * l = 2
7 8 9 10 11 * l = 4
6 : 7 8 9 10 11 * l = 5
8 : 9 10 11 * l = 3
10 : 11 * l = 1

Iterating : 2 + 4 + 5 + 3 + 1
: 1 + 2 + 3 + 4 + 5 S[5] = 15
For ar[N]
TC : 1 + 2 + 3 + .. N = $\frac{(N)(N+1)}{2}$ = O(N²)

Issue: We iterate on same sequence multiple times hence it's going till O(N²)

Hint: We start sequence from ar[i] when ar[i]-1 doesn't exist

ar[] = { 9 7 6 8 10 }

✓8 9 :
✓6 7 :
*5 6 : 7 8 9 10 11 * l = 5
✓7 8 :
✓9 10 ;

Ex2:

0	1	2	3	4	5	6	7
6	6	6	6	8	9	7	10

arr[] = {6 6 6 6 8 9 7 10}

* 5 → 6 → 7 → 8 → 9 → 10 → 11 * l = 5
* 5 → 6 → 7 → 8 → 9 → 10 → 11 * l = 5
* 5 → 6 → 7 → 8 → 9 → 10 → 11 * l = 5
* 5 → 6 → 7 → 8 → 9 → 10 → 11 * l = 5

Issue: Because arr[] elements are repeating, same sequence will repeat multiple times

Hint: To avoid above we iterate on hashset

Final Idea:

Iterate on hashset:

for element x: We start sequence if x-1 doesn't exist

Dry Run:

0	1	2	3	4	5	6	7	8	9
-1	8	5	3	10	2	4	9	2	5

arr[] = {-1 8 5 3 10 2 4 9 2 5}

↓

hs = {-1 8 5 3 10 2 4 9}

-2 ← -1 → 0 * l = 1

7 ← 8 → 9 → 10 → 11 * l = 3

4 ← 5:

2 ← 3:

9 ← 10

1 ← 2 → 3 → 4 → 5 → 6 * l = 4 → 4

3 ← 4

6 ← 9

Total iterating = O(N+N) = O(N) SC: O(N)

Outer loop: Iterating on hashset = O(N)

Inner loop: Sum of all sequence length: O(N)

Instructor: TC need more detailed explanation

```
int longestSequence(vector<int> &ar) {   TC: O(N+N) = O(N)
                                         SC: O(N)
    unordered_set<int> hs;
    for(int i=0; i<ar.size(); i++){
        hs.insert(ar[i]);
    }

    int ans=0;
    for(auto x: hs){
        # we start sequence from x, if x-1 doesn't exist
        if(hs.find(x-1)==hs.end()){
            # start seq of x :
            int s=x, c=0
            while(hs.find(s)!=hs.end()){
                c++; # one seq length
                s++; # go to next ele eq search
            }

            ans=max(ans, c);
        }

    }

    return ans;
}
```

28 Given ar[N]

Find min abs|i-j| such that ar[i]==ar[j] & i ≠ j

	0	1	2	3	4	5	6
Ex1 : ar[ ] = {	3	4	7	6	3	7	3 }

Pairs:

Ideal:

	0	1	2	3	4	5	6	7	8	9
Ex2: ar[ ] = {	3	4	6	5	3	4	5	3	4	3 }

Hint: Highlight 3's

Show for every 3 we compare with latest occur.

For every elemt compare with its latest occurn.

Dry Run:

Hashmap:

```
int minclosest(vector<int> row){
```

3