## Today's Content

* Length of longest sequence inc. by 1
* No. of distinct points

Q1: Given a[N]. Find length of longest sequence which can be rearranged in a strictly increasing order by 1.

Note: Index element don't have to be continuous.

```
              0   1   2  3  4   5  6  7
Eg 1:  A[] = {-1, 8,  5, 3, 10, 2, 4, 9}
       seq = {2, 3, 4, 5}   → 4
       seq = {8, 9, 10}     → 3
```

```
a[] = {3, 8, 2, 1, 9, 6, 5, 6, 7, 2}
  seq = {1, 2, 3}          → 3
  seq = {5, 6, 7, 8, 9}    → 5
```

idea 1: Sort the array

```
          a[] = {-1, 8, 5, 3, 10, 2, 4, 9}
Sort:      -1   2  3  4  5   8  9  10          ans = 4
            1   1  2  3  4   1  2  3
```

```
a[] = {3, 8, 2, 1, 9, 6, 5, 6, 7, 2}
Sort:   1  2  2  3  5  6  6  7  8  9           ans = 5
      1  2  2  3  1  2  2  3  4  5
```

TC: $O(n\log n + n) = O(n\log n)$

SC: $O(1)$

## idea 2: Hashset idea :

a[] = {-1, 8, 2, 3, 7, 1, 4, 9}

{1, 2, 3, 4} → len = 4

{7, 8, 9} → len = 3

{8, 9} → len = 2

{1, 2, 3} → len = 3 {1, 2} → len = 2

⊛ Obs: if for any element $x$, $x-1$ exists

$x-1, x \ldots$ will form a longer seq.

∴ $x-1$ will start the sequence.

### Check who can start the sequence :

| | | |
|---|---|---|
| -1 | ✓ | len = 1 |
| 8 | ✗ | |
| 2 | ✗ | |
| 3 | ✗ | |
| 7 | ✓ | {7, 8, 9} len = 3 |
| 1 | ✓ | {1, 2, 3, 4} len = 4 |
| 4 | ✗ | |
| 9 | ✗ | |

Eg: a[] = {9, 7, 8, 6, 10}

### Check who can start the sequence :

| | | |
|---|---|---|
| 9 | ✗ | |
| 7 | ✗ | |
| 8 | ✗ | |
| 6 | ✓ | {6, 7, 8, 9, 10} len = 5 |
| 10 | ✗ | |

∴ idea 2: for every a[i], check if it can be a starting point or not.
if it can be starting point, get length of seq. & calc. max.

Pseudocode:

```
int longest Seq (int a[]) {
    // insert all a[i] into hs.
    ans = 0
    for(i=0; i<n; i++) {          for (int x : hs) {      ↳ check syntax in your lang of choice
        x = a[i]
        if (hs.find(x-1) == false) { // x is the starting point
            len=0 , y=x
            while(hs.find(y) == true) {
                len = len+1 , y=y+1
            }
            ans = max(ans, len)
        }
    }
    return ans
}
```

TC: $O(n+n)$
SC: $O(n)$

Edge Case:

a[] = {6, 6, 6, 6, 8, 9, 7, 10}        hs: {6, 7, 8, 9, 10}

Who Can Start?

| | | | |
|---|---|---|---|
| 6 | ✓ | y = ~~6~~ ~~7~~ ~~8~~ ~~9~~ 10<br>len = 1 2 3 4 5 | 6 → 7 → 8 → 9 → 10   5 |
| 6 | ✓ | 6 → 7 → 8 → 9 → 10 | len = 5 |
| 6 | ✓ | 6 → 7 → 8 → 9 → 10 | len = 5 |
| 6 | ✓ | 6 → 7 → 8 → 9 → 10 | len = 5 |

repetitions

∴ Instead, iterate over Hashset.

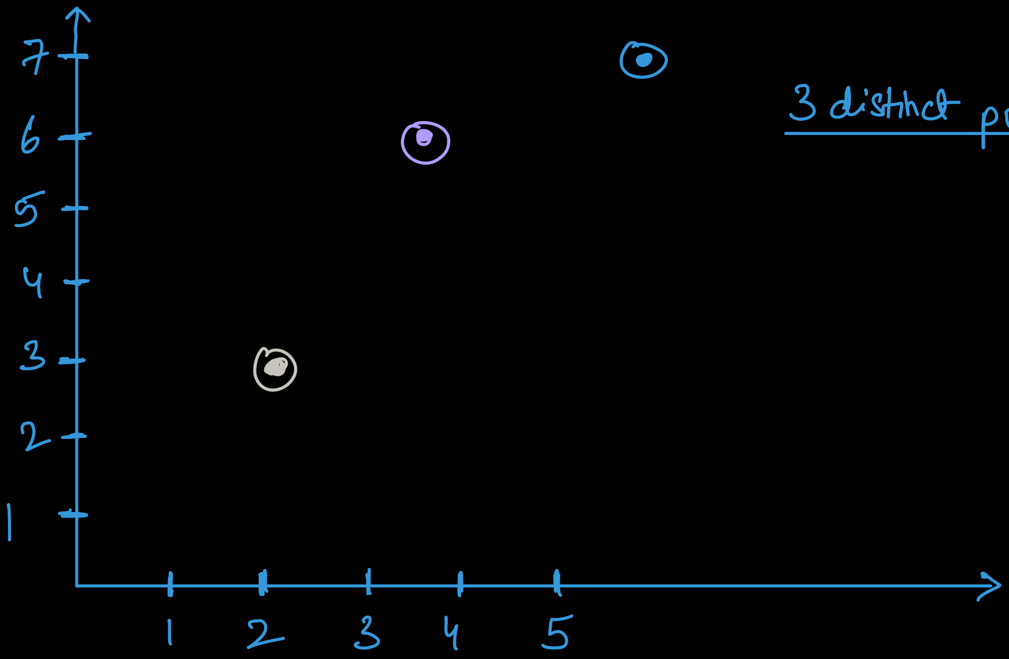
Break: 8:30 am

Q2: Given N 2D points {x,y}. Calc no. of distinct points.

Eg:

$$\begin{array}{ccccccc} & 0 & 1 & 2 & 3 & 4 \\ x[\,] = \{ & 2, & 3, & 2, & 5, & 3 & \} \\ y[\,] = \{ & 3, & 6, & 3, & 7, & 6 & \} \end{array}$$

ith point is (x[i], y[i])



3 distinct points

idea 1: Insert points in a Hashmap<int, int> X

{2, 4}   {2, 6} → overwritten

idea 2: Store all points in a Hashset<pair<int, int>> hs

{2,4}   {2,6}   {2,4}

HS: $\begin{bmatrix} \{2,4\} \\ \{2,6\} \end{bmatrix}$

return hs.size()

To do this: Overriding the Hash fn

## What is a hashCode() ?

It is an integer value that is associated with each object in Java. It's main purpose is to help with hashing in Hash Tables.

This method is implemented by default in Object class. ∴ User-defined classes also inherit it.

* It returns same integer value.

∴ To have your own implementation of hashCode => Override it.

* Whenever you override hashCode(), you MUST override equals() also. & vice-versa.

```
class Pair {
        int first
        int second
        public Pair(int a, int b) {
                first = a
                second = b
        }
        @Override
        public boolean equals(Object o) {
                Pair p = (Pair) o;
                if(p.first == first && p.second == second) {
                        return true
                }
                return false
        }
        @Override
        public int hashCode() {
                return 31* first + second
        }
}
```

```
public int solve (int [][] A) {
        n = A.length
        List <Pair> l = new ArrayList <>();
        for(i=0; i<n; i++) {
            l.add(new Pair(A[i][0], A[i][1]))
        }
        Set <Pair> distinctPoints = new HashSet <>(l)
        return distinctPoints.size()
}
```

idea 3: Finding an easier soln

"Smart idea": Store points as string.

$(x,y)$ = "x"+"y"

(2,4) = "24"        (2,6) = "26"

(12,3) = "123"        (1,23) = "123"

diff b/w them

Introduce a special char:    $(x,y)$ = "x" + "@" + "y"

(12,3) = "12@3"        (1,23) = "1@23"

```
int 2Dpoints (int x[], int y[]) {
        Hashset <string> hs
        for (i=0; i<n; i++) {
            // ith point is (x[i], y[i])
            String p = String.valOf(x[i]) + "@" + String.valOf(y[i])
            hs.add(p)
        }
        return hs.size()
}
```

TC: O(n)

SC: O(n)