Java : HashMap | HashSet

C++ : unordered_map | unordered_set

Python : dict | set

⇒ insert(—)
update( )
delete()
get()
size()

} O(1) avg case TC.

Q.1 Given an Array of size N, count the no. of duplicate pairs i.e A[i] = A[j] , i != j

Quiz

	0	1	2	3	4	5	6	7	8	9	
A : {	1	2	3	4	1	2	1	4	6	1	}

duplicate pairs

(0,4) (1,5)
(0,6) (3,7)
(0,9)
(4,6)
(4,9)
(6,9)

} 8 pairs.

$$A : \left\{ \begin{array}{cccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 1 & 2 & 3 & 4 & 1 & 2 & 1 & 4 & 6 & 1 \\ & & & & \uparrow & & & & & \\ & & & & i & & & & & \end{array} \right\}$$

Brute force

```
count = 0
for( i= 0; i< N ; i++){
        for( j= i+1 ; j< N; j++){
                if (A[j] == A[i])
                        count++
```

3

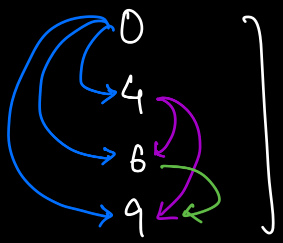
$\underline{\underline{3}}$

TC : $O(N^2)$

SC : $O(1)$

$$A : \left\{ \begin{array}{cccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 1 & 2 & 3 & 4 & 1 & 2 & 1 & 4 & 6 & 1 \end{array} \right\}$$

freq(1) = 4



$n \Rightarrow \boxed{\frac{n(n-1)}{2}} \Rightarrow {}^{n}C_2$

$$A : \left\{ \begin{array}{cccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 1 & 2 & 3 & 4 & 1 & 2 & 1 & 4 & 6 & 1 \\ & & & & & & & & & \uparrow \end{array} \right\}$$

Map

$\langle 1, \cancel{1} \rangle \cancel{2}\, \cancel{3}\, 4$

$\langle 2, \cancel{1} \rangle\, 2$

$\langle 3, 1 \rangle$

$\langle 4, \cancel{1} \rangle\, 2$

$\langle 6, 1 \rangle$

$$\text{Count} = {}^4C_2 + {}^2C_2 + {}^2C_2$$
$$= 6 + 1 + 1$$
$$= \underline{\underline{8}}.$$

Approach:

I Create a frequency map. ⇒ O(N)

II Iterate over the map & calculate the # of pairs $({}^nC_2)$ $n \geq 2$ ⇒ O(N)

$$\boxed{\begin{array}{l} TC : O(N) \\ SC : O(N) \end{array}}$$

Can we do it in 1 iteration?

⇒ YES.

$$A : \left\{ \begin{array}{cccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 1 & 2 & 3 & 4 & 1 & 2 & 1 & 4 & 6 & 1 \end{array} \right\} \uparrow$$

Map

<1, ~~1~~> ~~2~~ ~~3~~ 4

<2, ~~1~~> 2

<3, 1>

<4, ~~1~~> 2

<6, 1>

Count = ~~0~~ ~~1~~ ~~2~~ ~~4~~ ~~5~~ $\boxed{8}$

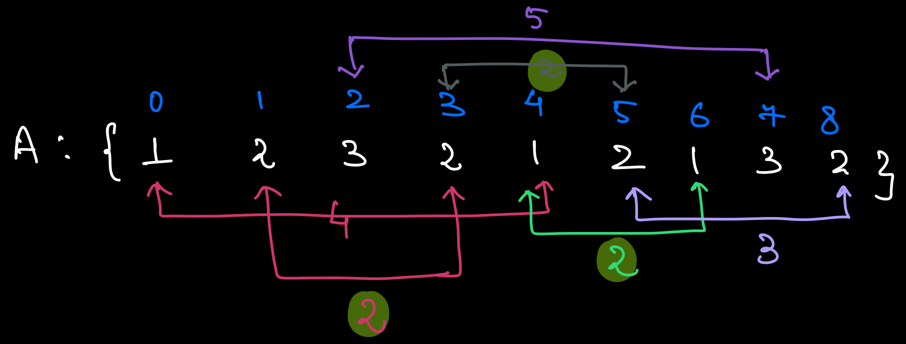
TC : O(N)

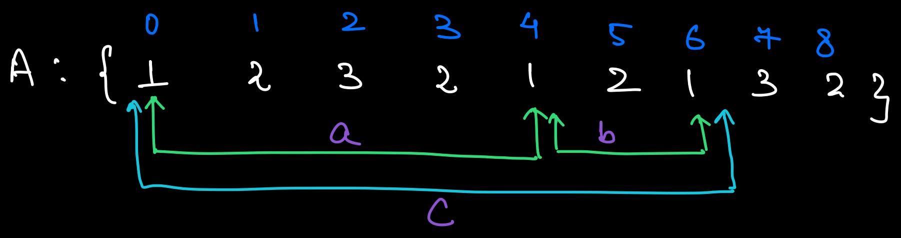
SC : O(N)

Q. Given an Array of size N, return the minimum distance b/w any two duplicate elements.

Google

(i, j) ⇒ A[i] = A[j] & |i-j| is MINIMUM





c > a } min dist will either be a OR b
c > b }

⇒ Minimum distance will always be present b/w adjacent duplicate pairs.

$$A : \left\{ \begin{array}{ccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 2 & 1 & 2 & 1 & 3 & 2 \end{array} \right\} \uparrow$$

$$\begin{tikzcd} \text{map< int} \arrow[d] & \text{, int >} \arrow[d] \\ a[i] & \underline{\underline{\text{index}}} \end{tikzcd}$$

ans = ~~0~~ 2

$\langle 1, \cancel{0} \rangle$ ~~4~~ 6

$\langle 2, \cancel{1} \rangle$ ~~3~~ ~~5~~ 8

$\langle 3, \cancel{2} \rangle$ 7

⇒ Map will contain the latest occurrence of each element.

#

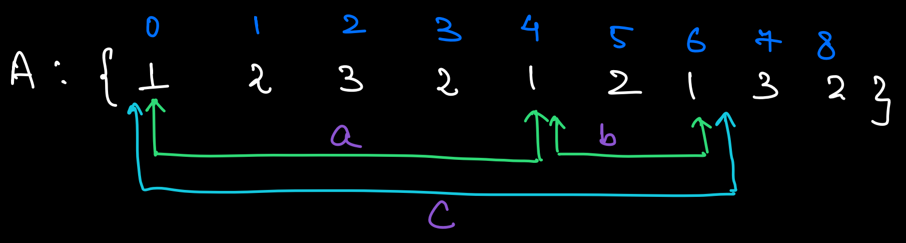
Iterate over the array :

check if A[i] is present in the map
if yes, find the distance & update ans
if distance < ans. Update map with latest
index (i) for A[i].
else : make an entry for (a[i] , i) in
the map.

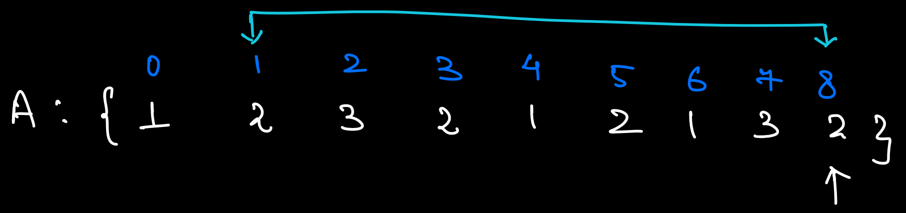
TC : O(N)
SC : O(N)

Q. Given an Array of size N, return the maximum distance b/w any two duplicate elements.

(i, j) ⇒ A[i] = A[j] & |i-j| is MAXIMUM



c > a } max distance = c.
c > b }



⟨1, 0⟩
⟨2, 1⟩
⟨3, 2⟩

ans = -~~0~~ ~~2~~ ~~4~~ ~~6~~ 7

map < int , int >

$$\text{map}\langle \underset{a[i]}{\underset{\downarrow}{\text{int}}}\ ,\ \underset{\text{first occurrence index of } a[i].}{\underset{\downarrow}{\text{int}}} \rangle$$

$$\boxed{\begin{array}{l} TC : O(N) \\ SC : O(N) \end{array}}$$

Q. Given an Array of size N, find the length of largest sequence that can be rearranged to a sequence of consecutive numbers.

Google
MS
Amazon
- - -

A : {100 , 4 , 200 , 1 , 3 , 2}

⇓

{4 , 1 , 3 , 2} ⇒ ④

A : { 2 4 6 8 }

⇒ ①

Quiz

$$A : \begin{array}{cccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ \hline -1 & 8 & 2 & 3 & 7 & 1 & 4 & 9 \end{array}$$

↓

{2 3 1 4} ⇒ ④

Quiz

[5, 9, 100, 1, -1, 2, 3, 99, 98, 11, 101, 15, 102]

{100, 99, 98, 101, 102} ⇒ 5

Ex {9, 8, 1, 100, 2, 99, 100, 98, 4}

{100, 99, 98} ⇒ 3

SORT

$$A: \left[\begin{array}{cccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ -1 & 8 & 2 & 3 & 7 & 1 & 4 & 9 \end{array}\right]$$

↓ sort

{-1 1 2 3 4 7 8 9}

l = 1

l = ~~1~~ ~~2~~ ~~3~~ 4

l = ~~1~~ ~~2~~ 3

ans = ~~1~~ 4

TC: O(NlogN)

SC: O(N) ⇒ Merge

O(logN) ⇒ Quick

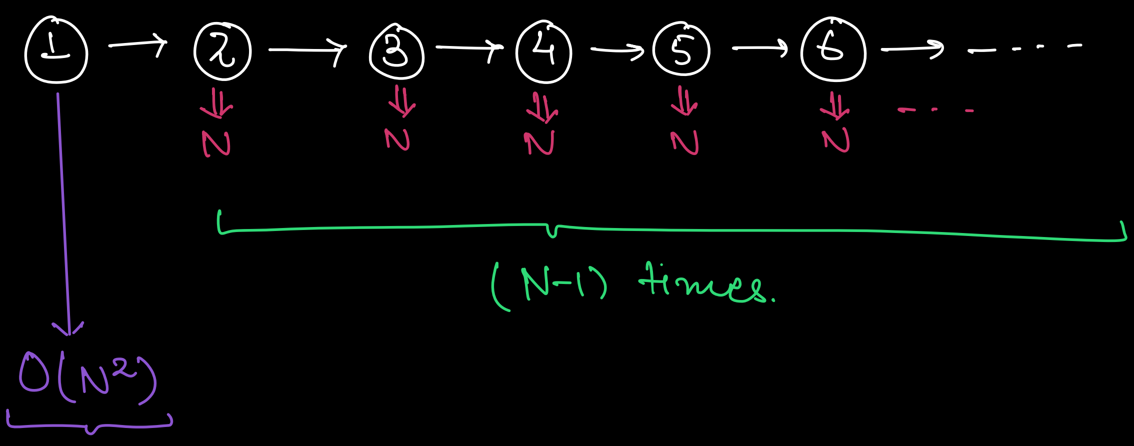
Brute force

for every element in the Array, try to find out the length of longest consecutive sequence starting with this element.

$$A : \begin{array}{cccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ [-1 & 8 & 2 & 3 & 7 & 1 & 4 & 9] \end{array}$$

	length
Consecutive seq. starting at -1 ⇒ -1, ~~0~~	1
Consecutive seq. starting at 8 ⇒ 8, 9, ~~10~~	2
Consecutive seq. starting at 2 ⇒ 2, 3, 4, ~~5~~	3
Consecutive seq. starting at 3 ⇒ 3, 4, ~~5~~	2
Consecutive seq. starting at 7 ⇒ 7, 8, 9, ~~10~~	3
Consecutive seq. starting at 1 ⇒ 1, 2, 3, 4, ~~5~~	4
Consecutive seq. starting at 4 ⇒ 4, ~~5~~	1
Consecutive seq. starting at 9 ⇒ 9, ~~10~~	1

Quiz TC ?

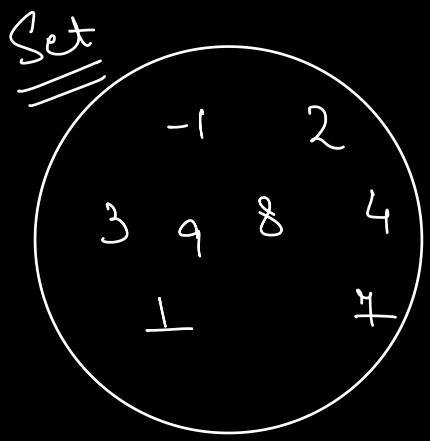


$\boxed{TC : O(N^3)}$

1 search ⇒ O(N)

HashSet ⇒ Search : O(1)

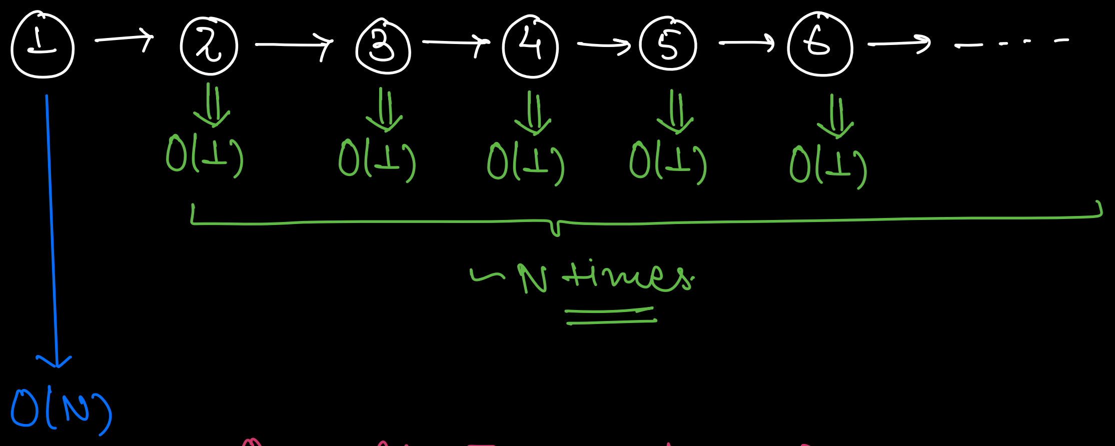
$$A : \begin{array}{cccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ -1 & 8 & 2 & 3 & 7 & 1 & 4 & 9 \\ & \uparrow & & & & & & \end{array}$$



-1 ⇒ -1, ~~0~~

8 ⇒ 8, 9, ~~10~~

⋮



Overall TC : O(N×N)

SC : O(N)
↓
Set.

```
//Build Set
HashSet<int> set;
for(i=0; i<N; i++)  set.insert(a[i]);
ans = -∞
for(i=0; i<N; i++) {
        l = 0,  x = a[i];
        while( set.contains(x) ) {
                l++
                x+1;
        }
        ans = max(ans, l);
}
return ans;
```

[1 2 3 4 5 6 7 8 9 10]

1 → 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, ~~11~~

2 → 2, 3, 4, 5, 6, 7, 8, 9, 10, ~~11~~

3 → 3, 4, 5, 6, 7, 8, 9, 10, ~~11~~

4 → 4, 5, 6, 7, 8, 9, 10, ~~11~~

5 → 5, 6, 7, 8, 9, 10, ~~11~~

6 → 6, 7, 8, 9, 10, ~~11~~

7 → 7, 8, 9, 10, ~~11~~

[1 2 3 4 5 6 7 8 9 10]

↑

1 → 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 ⟹ l = 10

2 → X

3 → X

4 → X

5 → X

6 → X

7 → X

8 → X

9 → X

10 → X

TC : O(N)

SC : O(N)

```
//Build Set
HashSet<int> set;
for(i=0; i<N; i++)  set.insert(a[i]);
ans = -∞
for(i=0; i<N; i++){
        if(!set.contains(a[i]-1)){
                l=0,  n=a[i];
                while(set.contains(n)){
                        l++
                        n+1;
                }
                ans = max(ans, l);
        }
}
return ans;
```

[5 1 3 4 2 6 7 8 9 10]

5 → x

1 →

3 → x

4 → x

⋮

{9, 8, 1, 100, 2, 99, 100, 98, 4}
↑

9 → X

8 → 8, 9, ⇒ l = 2

1 → 1, 2 ⇒ l = 2

100 → X

2 → X

99 → X

100 → X

98 → 98, 99, 100 ⇒ l = 3

4 → 4, ⇒ l = 1

TC : O(N)

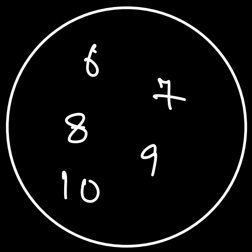
SC : O(N)

Eg

A : {6 6 6 6 6 6 6 6 6 7 8 9 10}
↑

6 → 6, 7, 8, 9, 10
6 → 6, 7, 8, 9, 10
6 → 6, 7, 8, 9, 10
6 → 6, 7, 8, 9, 10
6 → 6, 7, 8, 9, 10
6 → 6, 7, 8, 9, 10

9 times.



7 → X
8 → X
9 → X

⇒ **Optimisation**

Instead of iterating over an Array, iterate over Set.

——— * ———