Agenda →

- Understanding behind the scenes of hashmap.
- Closest Duplicates
- Longest Chain of Consecutive Elements
- Longest subarray with sum = 0.

HashMap. → data structure.

Key, value.

search → O(1)
add → O(1)
remove → O(1)
size → O(1)
update → O(1)

$\left.\begin{matrix}\text{search} \to O(1)\\ \text{add} \to O(1)\\ \text{remove} \to O(1)\\ \text{size} \to O(1)\\ \text{update} \to O(1)\end{matrix}\right] \to O(1)$ Avg.

Array. arr[N]

indices → [0 to N-1] ⇒ act as Keys.

Key	value
5	20
16	25

arr[5] = 20
arr[16] = 25.

$1 \le$ Key $\le 10^9$

arr[10^9] = ____ ✗

M.L.E. Error

$1 \le$ total number of keys $\le 10^5$

$\left.\begin{matrix}\_\\ \_\\ \_\\ \_\\ \_\end{matrix}\right\} \% m \Rightarrow 0$ to $m-1$.

Assumption → let's say we can have maximum size of array = 17
⇓ indices
[0, 16]

Keys → {10, 20, 30, 27}

Hashing function → Key % 17

10 % 17 → 10
20 % 17 → 3
30 % 17 → 13
27 % 17 → 10

Collision

arr[10] → v_1
arr[3] → v_2
arr[13] → v_3
arr[10] → v_4

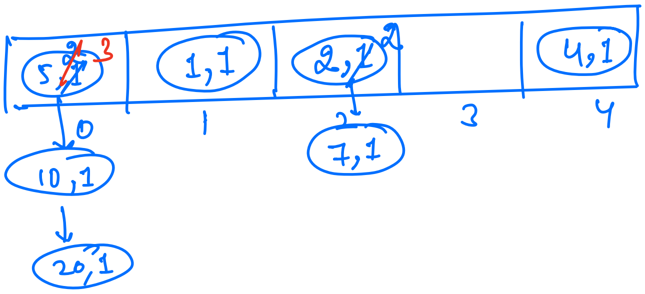
Collision.

- Chaining
- Open Addressing.
  - Linear Probing
  - Quadratic Probing

elements: [2 4 5 7 10 2 5 20 5 1]

[store freq of all the elements.]

Size allowed → 5   element % 5



int [] arr = new int [N];

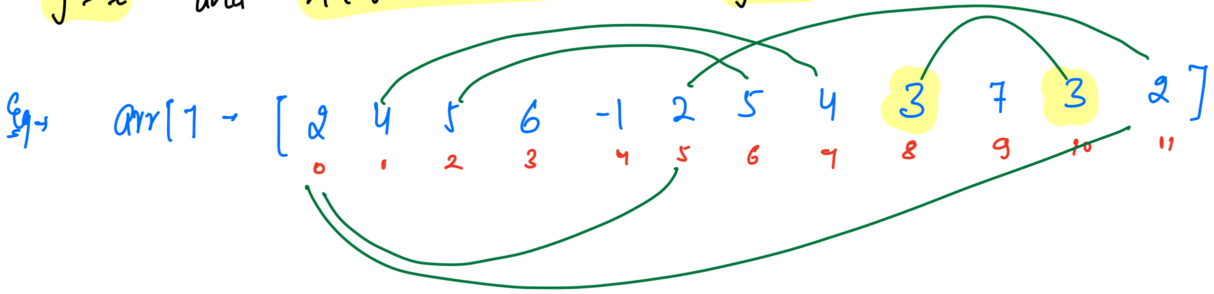
LinkedList < Node > [ ] hashmap = new LinkedList [N];

Node
int val
Node next

Node
int key
int value
Node next

Closest Duplicates

Q) Given an integer array of size N. Find pair(i, j) such that $j > i$ and $A[i] == A[j]$ and $j - i$ is minimum.



Eg → arr[] → [2 4 5 6 -1 2 5 4 3 7 3 2]

idea-1 Consider all the pairs T.C → $O(N^2)$

↙ NlogN → sorting ✗ → BS ✗

↘ N

X ← 7 → X ← 5 → X ← 10 → X ← 3 → X

observation → For every element, keep a track of last occurence of that element.

element → last occurence.

arr → [1 2 3 6 1 2 3 2 1]

0 1 2 3 4 5 6 7 8

i ↓ (at index 8)

element → last occu.

1 → ~~0~~ 4
2 → ~~1~~ ~~5~~ 7
3 → ~~2~~ 6
6 → 3

ans = INT_MAX

ans = min(INT_MAX, 4-0) = 4

ans = min(4, 5-1) = 4

ans = min(4, 6-2) = 4

ans = min(4, 7-5) = 2

ans = min(2, $\underline{8}$ - $\underline{4}$)

↑ curr idx ↑ last occurence

of current element

pseudo-code

```
hm < int , int > ;   ans = INT_MAX
for( int i=0 ; i<N ; i++){
        if ( arr[i] is not present in hm){
                hm.insert(arr[i], i) ;
        }
        else{
                lo = hm[arr[i]] ;
                ans = min( ans , i - lo ) ;
                hm.update( arr[i], i);
        }
}
return ans;
```

T.C → O(N)
S.C → O(N)

actual syntax.

```
int lo = hm.get(arr[i]);
```

Q. Given an array of size N. Find the length of longest sequence of consecutive elements.

Eg. [100 4 3 6 10 20 11 5 101]

100 → 101 ⇒ 2

3 → 4 → 5 → 6 ⇒ 4 (ans.)

10 → 11 ⇒ 2

arr → [-1 8 2 3 7 1 4 9]

1, 2, 3, 4 ⇒ 4.

idea.1. Sorting → [-1 | 1 2 3 4 | 7 8 9]

① ④ ③

T.C → O(N logN)

pseudo-code → (#todo)

idea.2. Let's try to treat every element as the starting point of the consecutive sequence.

arr → [-1 8 2 3 7 1 4 9]

	length.
-1 → 0	1
8 → 9 → 10	2
2 → 3 → 4 → 5	3
3 → 4 → 5	2
7 → 8 → 9 ← 10	3
1 → 2 → 3 → 4 → 5	4
4 → 5	1
9 → 10	1

Hashset.

-1, 8,
2, 3, 7
1, 4, 9

Analyse T.C.

[2 5 3 1 6 4]

	iteration.
1 → 2 → 3 → 4 → 5 → 6	6 +
2 → 3 → 4 → 5 → 6	5 +
3 → 4 → 5 → 6	4 +
4 → 5 → 6	3 +
5 → 6	2 +
6	1

⇒ $\frac{N(N+1)}{2}$ iterations

T.C → $O(N^2)$

$x-1$ x

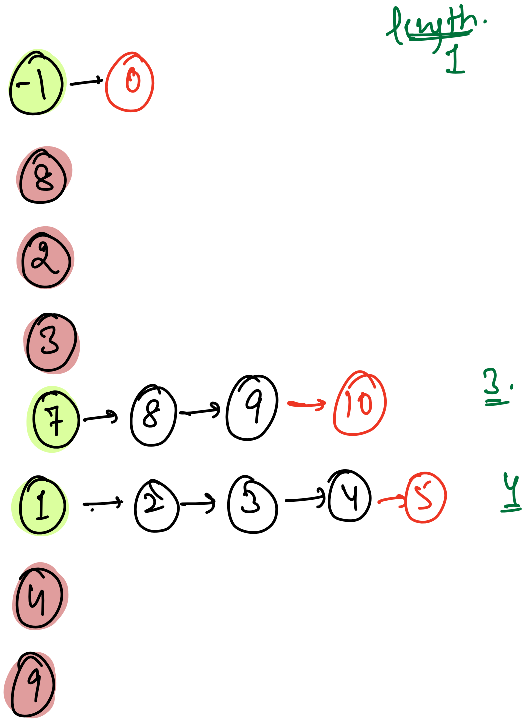
observation.→ If $x-1$ is present in the array, should we start our sequence from x ?

⇒ x can't be a starting point of sequence.

arr[] = [-1 8 2 3 7 1 4 9]

Hashset

-1, 8,
2, 3, 7
1, 4, 9



pseudo code.

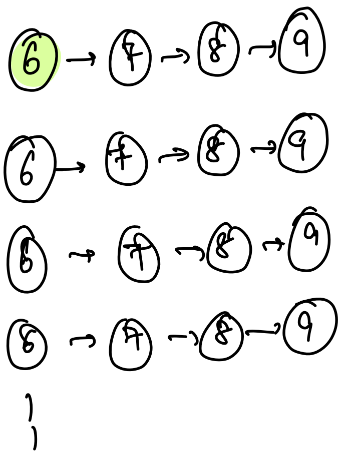
① Insert all the elements in hashset, ans = 0

```
for ( iterate on hashset ) {          → Google search
        ~~x = arr[i]~~  x → element from hashset.
        if ( x-1 is not present in hs ){
                chain = 1
                y = x+1
                while ( y is present in hs){
                        chain ++
                        y ++
                }
                ans = max (ans, chain);
        }
}
return ans
```

T.C → O(N)
S.C → O(N)

arr → [6 6 6 6 6 6 6 6 6 7 8 9 9 9]

6, 7, 8, 9



Q) Given an array of integers. Find the length of longest subarray with sum = 0.

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

* → if elements are repeating in psum[], then we have a subarray with sum = 0.

* → to find the longest length ⇒ farthest duplicates in pSum[].

* edge case ⇒ pSum[i] == 0 ⇒ length of subarray → $\boxed{i+1}$