Agenda →

- ✓ Understanding behind the scenes of hashmap.
- ✓ Closest Duplicates
- ✓ Longest Chain of Consecutive Elements
- ✓ Longest subarray with sum = 0.
- ✓ One more problem (if time permits)

Hashmap / Dictionary / Map.

↳ data structure to store <key, value> pair.

search, add, remove, size, update ⇒ O(1)

Arrays : arr[N]

indices : 0 to N-1 ⇒ act as keys.

Key	value
5	20
16	25

arr[5] = 20

arr[16] = 25

$1 \leq key \leq 10^9$

$1 \leq$ no. of keys $\leq 10^5$

arr[10^9] → X

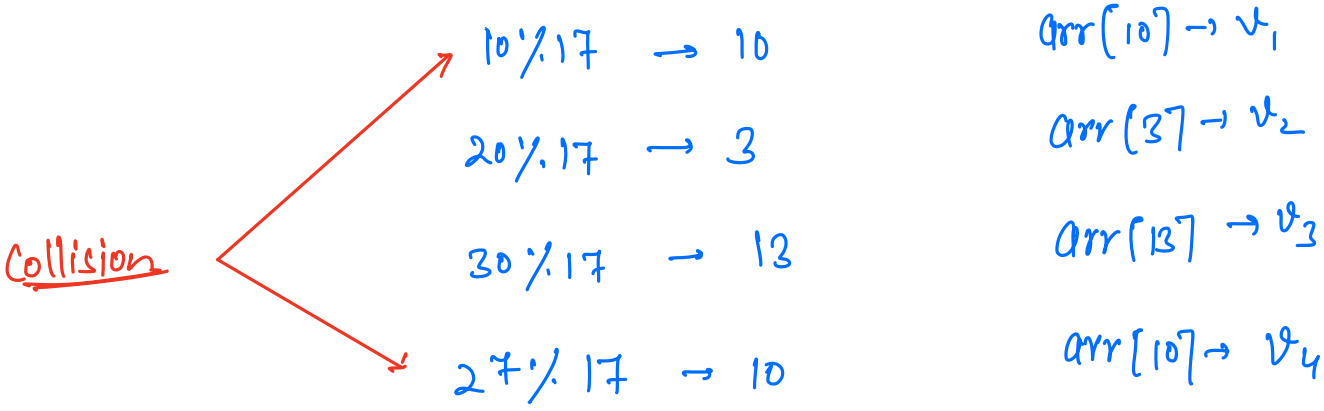
↓

Memory limit Exceded.

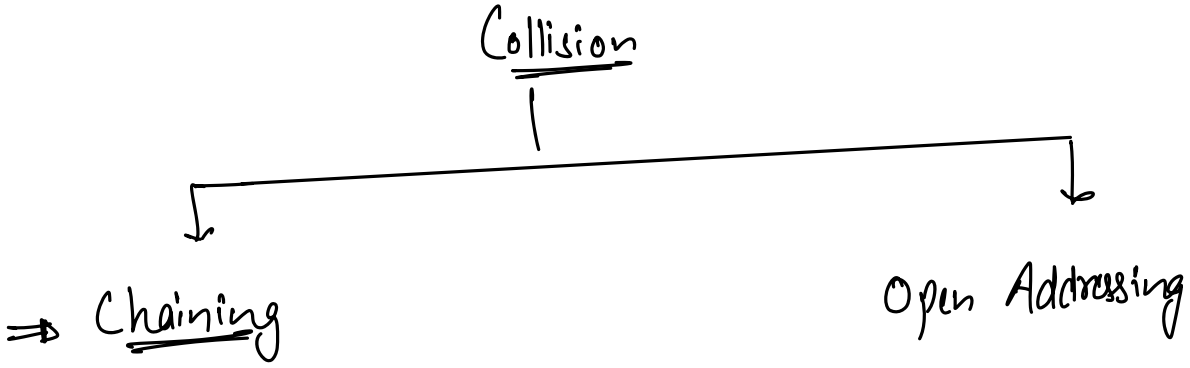
: } %.m ⇒ 0 to m-1.

Assumption → Maximum possible size of array → 17.

Keys → {10, 20, 30, 27}



[Key % 17] → Hashing function

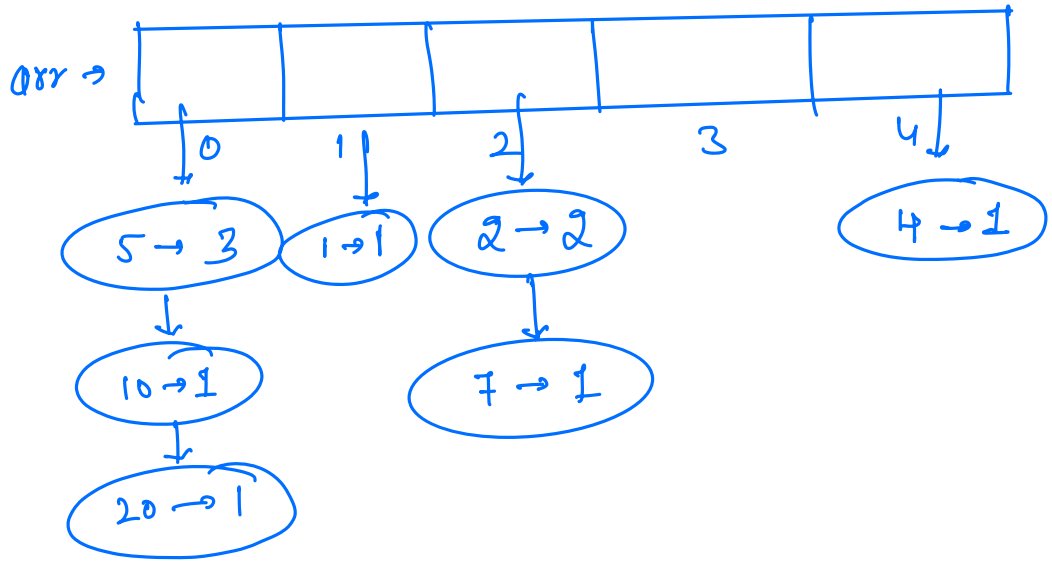


elements → [2 4 5 7 10 2 5 20 5 1]

size allowed → 5

Store frequency of all the elements.

O(1) time and least collision { Hashing function → element % 5




```
Node {
    int key;
    int value;
    Node next
}
```

Average size of linked list = $\frac{10^6}{10^6}$

= 1. (λ)

add →
remove →
update →
size →
search →

$O(\lambda)$

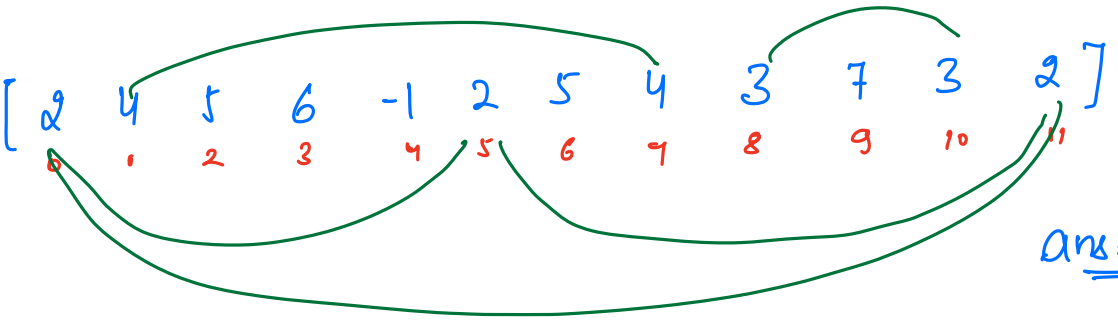
$\Rightarrow O(1)$ in average case.

Closest Duplicates -

Q) Given an integer array of size N. Find the minimum distance b/w two same elements.

$-10^9 \le arr[i] \le 10^9$
$2 \le N \le 10^5$

Eg → arr[] = [2 4 5 6 -1 2 5 4 3 7 3 2]
0 1 2 3 4 5 6 7 8 9 10 11



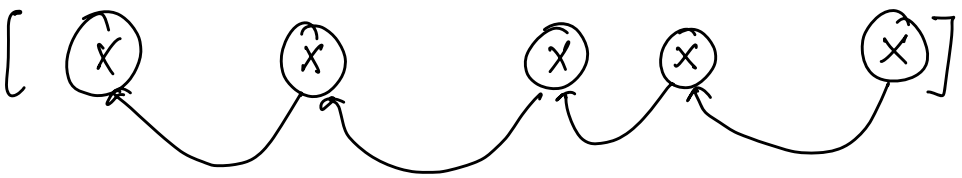
ans = 2.

idea-1 → Consider all the pairs.

T.C → $O(N^2)$

$N \log N$ → Sorting ✗

N → B.S. ✗



idea → for every element, keep a track of last occurence of that element.

Eg → [1 2 3 6 1 2 3 2 1]
0 1 2 3 4 5 6 7 8

element	l.o
1 →	~~0~~ ~~4~~ 8
2 →	~~1~~ ~~5~~ 7
3 →	~~2~~ 6
6 →	3

ans → ~~0~~ ~~4~~ 2

pseudo-code

```
ans → ∞
Hashmap <integer, integer> map;

for( i=0; i<N; i++) {
    if( arr[i] is not present in the map){
        map.insert( arr[i], i);
    }
    else{
        lo = map[arr[i]];
        ans = Min( ans, i - lo);
        map.update( arr[i], i);
    }
}
return ans;
```

T.C → O(N)
S.C → O(N)

Q) Given an array of size N. Find the length of longest sequence which can be arranged to get consecutive elements.

A: [-1, 8, 5, 2, 3, 7, 1, 4, 9]

-1 | 1, 2, 3, 4, 5 | 7, 8, 9

ans = 5.

idea-1. Sorting.

[-1 | 1, 2, 3, 4, 5 | 7, 8, 9]

① | ⑤ | ③

ans = 5.

T.C → O(NlogN)

idea - consider all the elements as the starting point of the consecutive sequence.

	len
-1 x	1
8 → 9	2
5	1
2 → 3 → 4 → 5	4
3 → 4 → 5	3
7 → 8 → 9	3
1 → 2 → 3 → 4 → 5	5
4 → 5	2
9	1

set

-1, 8, 5

2, 3, 7

1, 4, 9

ans = 5.

Analyse T.C.

arr → [2 5 3 1 6 4]

(6)

(5) → (6)

(4) → (5) → (6)

(3) → (4) → (5) → (6)

(2) → (3) → (4) → (5) → (6)

(1) → (2) → (3) → (4) → (5) → (6)

→ 1
→ 2
→ 3
→ 4
→ 5
:
→ N

$\frac{N(N+1)}{2} \approx N^2$

observation - If x-1 is present in the array, then we shouldn't consider x as the starting point.

arr → [2 5 3 1 6 4]
x x x [6] x x

(1) → (2) → (3) → (4) → (5) → (6)

set
2, 5
3, 1
6, 4

touching an element 3 times ⇒ 3N

pseudo-code.

① HashSet<integer> hs;

Insert all the elements in the hashset.

② for(iterate on set){ → Google!!

~~x = arr[i];~~ → x → get the element on the set.

if(x-1 is not present in hs){

chain = 1

y = x+1;

while(y is present in the 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 7 8 9 9 8 8 9]

0 1 2 3 4 5 6 7 8 9 10 11 12 13

6 → 7 → 8 → 9

6 → 7 → 8 → 9

6 → 7 → 8 → 9

6 → 7 → 8 → 9

6, 7

8, 9

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

arr[] → [4, -3, -1, 2, -2] ans = 5.

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

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

⇒ farthest duplicates in psum[].

for closest duplicates → last occurence

for farthest duplicates → first occurence

#code →

```
ans → -∞ , Map<int, int> map;
map.insert ( 0, -1): // Create psum[N]
for(i:0; i<N; i++){
    if (psum(i) is present in map){
        ans: max( ans, i - map[psum(i)]);
    }
    else{
        map.insert(psum(i), i);
    }
}
return ans;
```

T.C → O(N)
S.C → O(N)

Flip and Find Nearest

Given a binary string of length N and Q no. of queries.

Every query has 2 integers →

First integer denotes the type of query → 1, 2

Second integer denotes index x.

if type = 1, flip the value at index x.

if type = 2, find the index of nearest '1' from x.

↓

if there are multiple indices, return the one with lower index.

if there are no such index, return -1.

str → [0 1 0 0 1 0 0 1 0 0 0 1 1 0]

indices: 0 1 2 3 4 5 6 7 8 9 10 11 12 13

Q = 5.

q1 → 7

q3 → 4

q5 → 11

2, 9

1, 7

2, 6

1, 12

2, 9

Treeset / ordered-set:

hr → [0 1 0 0 1 0 0 ~~1~~ 0 0 0 1 ~~1~~ 0]
(0 1 2 3 4 5 6 7 8 9 10 11 12 13)

Q=5.

✓ 2, 9
✓ 1, 7
✓ 2, 6
✓ 1, 12
✓ 2, 9

1
4
~~7~~
11
~~12~~

9-7=2 11-9=2
6-4=2 11-6=5
9-4=5 11-9=2.

[7, 4, 11]

if query type = 2.

→ just smaller element in set [log N]
→ just greater element in set [log N]

insertion → logN

→ Search the syntax of treeset, ordered set in your language & try to code it yourself.

T.C → O(N·logN + Q*logN)
S.C → O(N)

psum[] = [3 1 0 9 6 0 3 1 2 9]

{ 1st Re-attempt }

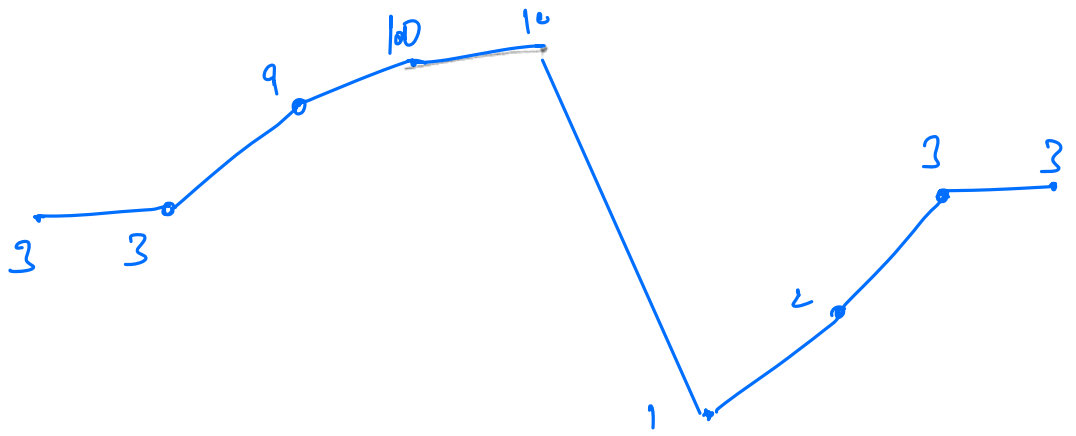
{ 3 qustion }

arr[] = [1 2 3 3 3 3 9 10 15]

⇓

[3 3 9 10 10 1 2 3 3]

0 1 2 3 4 5 6 7 8



→ Count sort ✓

→ Radix sort

e, u, u

Java → stable.

C# → inbuilt sort is not stable.