HASHMAP 1

Class will start at 7:10 AM

Today's Agenda

* Understanding behind the scenes of HashMap

Questions
* Closest Duplicates
* longest chain of consecutive elements
* longest subarray with sum = 0

Hash Map → Array of Linked Lists

↳ Data structure which stores the info in the key-value pair

01. Search for an ele
02 Insert an ele
03. Remove an ele
04. Size
05 Update an ele

} O(1) avg

- key – value

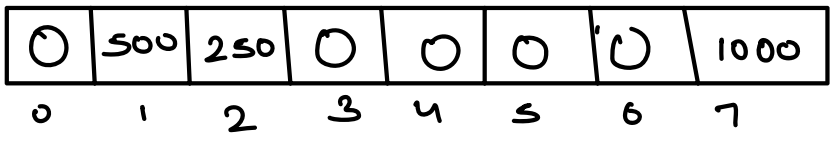
  1 – 500

  2 – 250

  7 → 1000

arr [N]

indices = 0 to n-1



Simplest Hash Map

Q Store frequency of all elements

A = { 3 2 1 2 5 1 5 }
0 1 2 3 4 5 6

freq = max + 1
size

freq = | 0 | 2 | 2 | 1 | 0 | 2 |
0 1 2 3 4 5

```
for (i = 0; i < n; i++) {
    freq[Arr[i]]++;
}
```

TC = O(n)
SC = O(Range of ele)

i → key

freq[i] → frequency of i (value)

If range of A[i] is very large → say A[i] = 10^9

↳ Can we create a freq arr? No

(Memory Limit Exceeded)

(MLE)

Array size = 10^5

Limited memory = M

Can we somehow limit the range of our elements. ? → Modulus operator

Hash function

↓

Provides the key in our memory limit

$h(x) = x \% M$

↓

original key

$h(x) = \underbrace{key}$

New key which is in range of 0 to m-1

Arr = { 10 20 30 27 } M = 17

$h(x) = x \% 17$

$h(x) = 10 \% 17 = 10$ → freq [10]

$h(x) = 20 \% 17 = 3$ → freq [3]

$h(x) = 30 \% 17 = 13$ → freq [13]

$h(x) = 27 \% 17 = 10$ → freq [10]

collision

Is it possible to avoid collision?

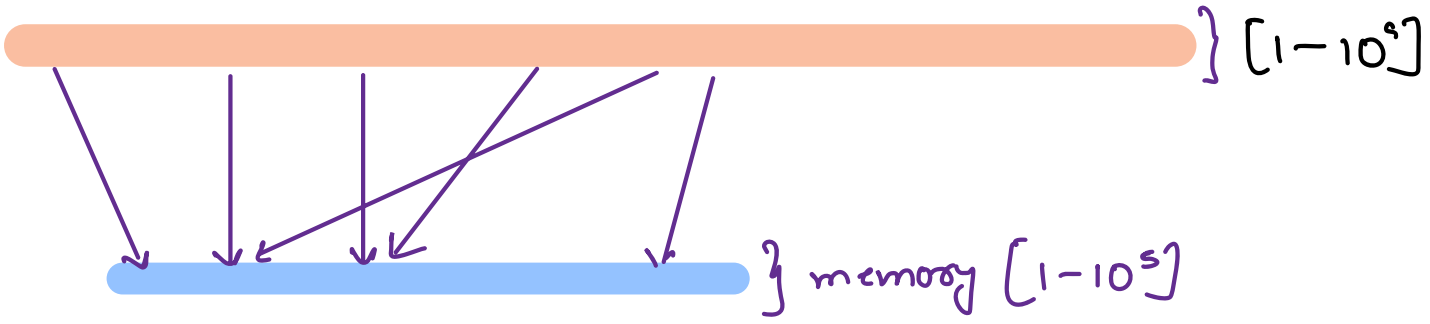
(limited memory) ⟶ No

(pegion hole principle)

Pigeon hole principle → N holes & (N+1) pegion

& you want every pigeon to live in hole

→ atleast one hole having more than 1 pigeon



Chaining → Handle the collision

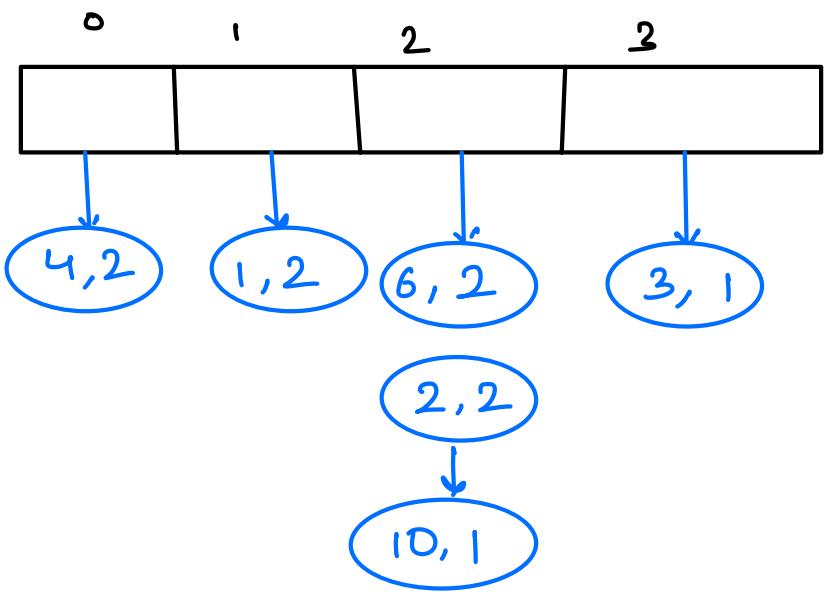
Instead of storing the freq of A[i],

store a list containing (A[i], freq[A[i]])

A[ ] = {3 6 4 1 2 10 2 4 6 1}

M = 4

freq arr =

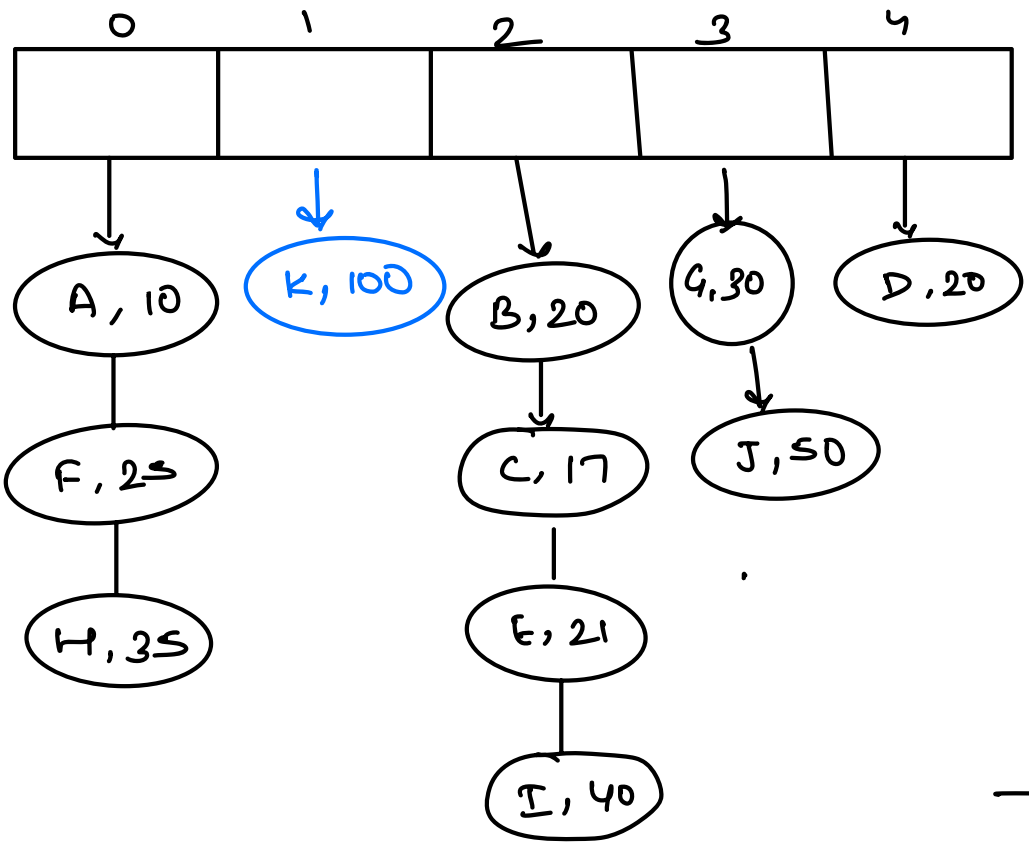


Cons of chaining → TC → O(1) → 90%

Worst → O(n) → 10%

Example → Population Map

Maintain the size of Linked List = 2 → loading factor



Given ↓

A - 10

B - 20

C - 17

D = 20

E = 21

F = 25

G = 30

H = 35

→ I = 40

J = 50

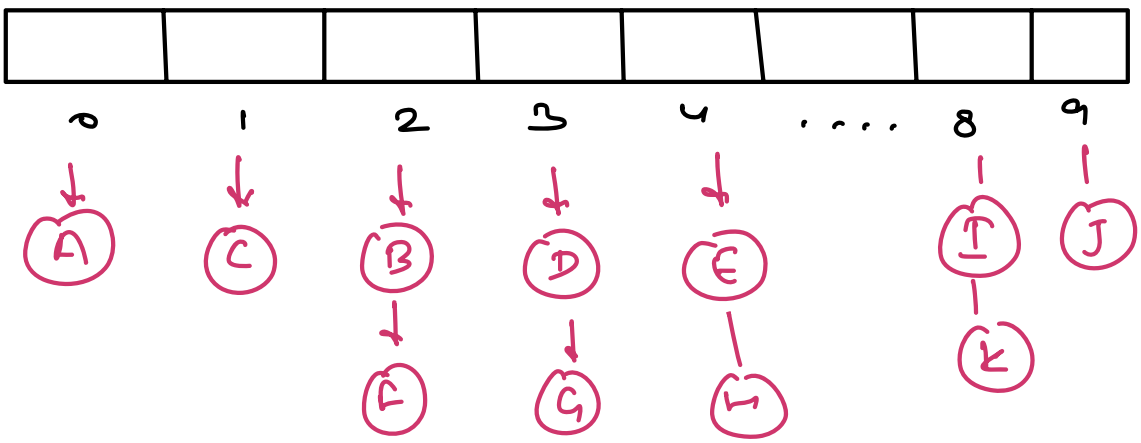
K = 100

Average size of a LL = $\frac{\text{No. of nodes}}{\text{No. of LL (size of array)}}$

$$= \frac{10}{5} = 2$$

Avg size of a LL = $\frac{11}{5} > 2$

Rehashing →



Avg size of LL = $\frac{11}{10} = 1.1 < 2$

TC = O(1)

8:13 → 8:23 AM

Closest duplicates

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

ar[ ] =

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

Brute force → Consider all pairs

→ Find the duplicate pairs in all pairs, find the distance & update your ans accordingly

TC = $O(n^2)$

↙ ↘

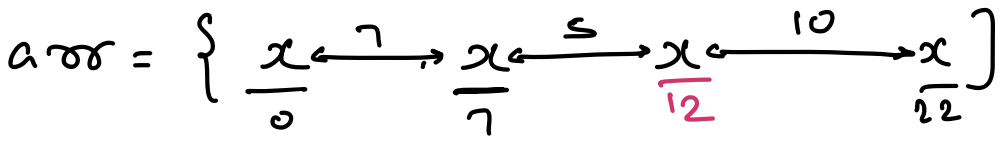
Nlogn N

X → Sorting

X → Binary search

arr = { x .... x ... x ..... x }

Observation → We only need the closest value of x to update your ans



$x = \emptyset \neq 12$

ans = ∅ 7

ans = min(∞, 7−0) = 7

ans = min(7, 12−7) = 5

ans = Math.min(5, 22−12) = 5

Conclusion → We need to keep track of last occurrence of the values

arr = { 1 2 3 6 2 1 }
0 1 2 3 4 5

Hashmap
key - value

1	—	0
2	—	~~1~~ 4
3	—	2
6	—	3

key = arr[i]

value = index

ans = ∞

ans = min(∞, 4−1) = 3

ans = min(3, 5−0) = 3

Pseudo code →

```
HM <I, I> map = new HM<>();
int ans = Integer.MAX_VALUE;
for (i=0; i<n; i++){
    int k = arr[i]
    if (map.containskey(k) == false){
        map.put(k, i);
    }
    else {
        int lo = map.get(k);
        ans = Math.min(ans, i-lo);
        map.put(k, i);
    }
}
```

TC = O(n)
SC = O(n)

02. 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 }

Sequence = 100, 101 → 2
3, 4, 5, 6 → 4
10, 11 → 2
20 → 1

arr = { -1 8 2 3 7 1 4 9 }
0 1 2 3 4 5 6 7

Sequence of consecutive ele
= 1 2 3 4 = 4
= 7 8 9 = 3
= -1 = 1

Idea 1 → Sort the array

arr = { -1 | 1 2 3 4 | 7 8 9 }
1 4 3

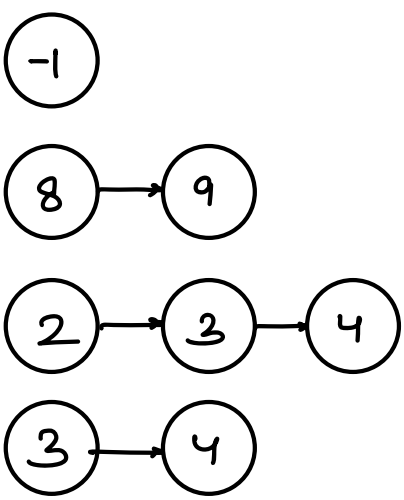
Ans = 4

TC = O(nlogn)
SC = O(1)

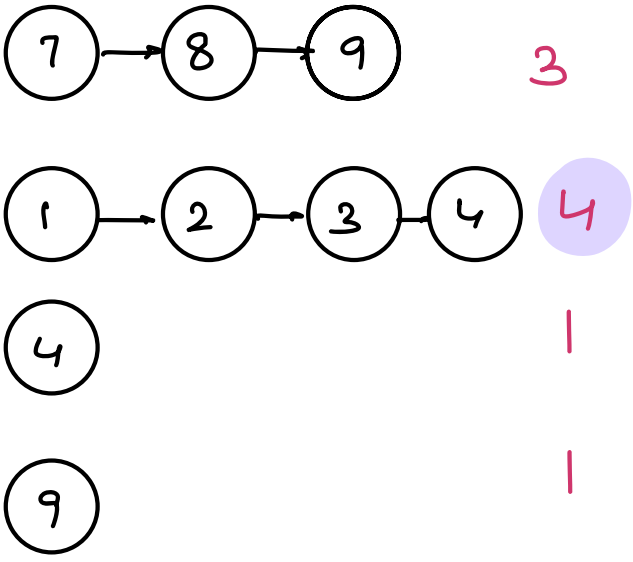
Idea 2 →

arr = { -1 8 2 3 7 1 4 9 }
0 1 2 3 4 5 6 7

Hashset = { -1 , 8, 2, 3, 7, 1, 4, 9 }



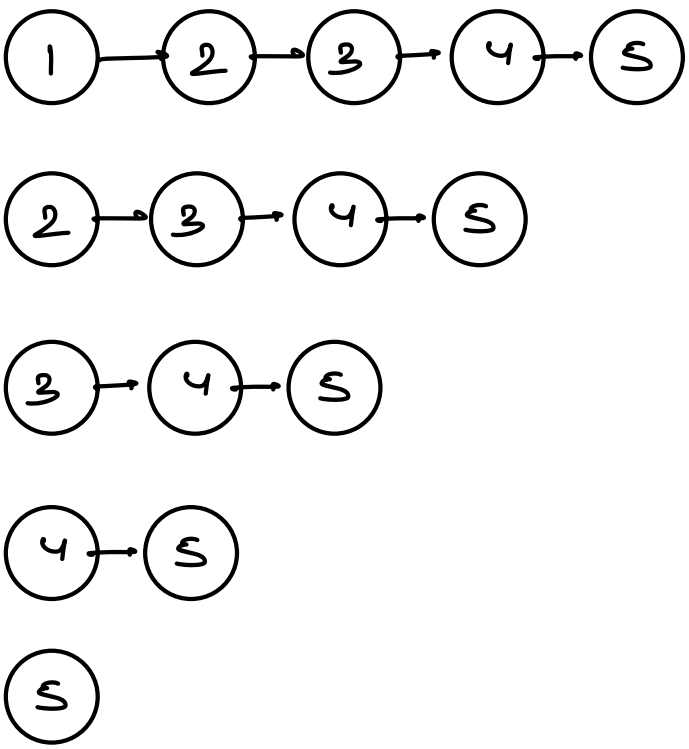
TC = O(n²)
SC = O(n)



Time complexity Analysis

arr = { 1 2 3 4 5 }

set = { 1 2 3 4 5 }



Len	Total Iteration
5	5 (N)
4	4 (N-1)
3	3 (N-2)
2	2
1	1

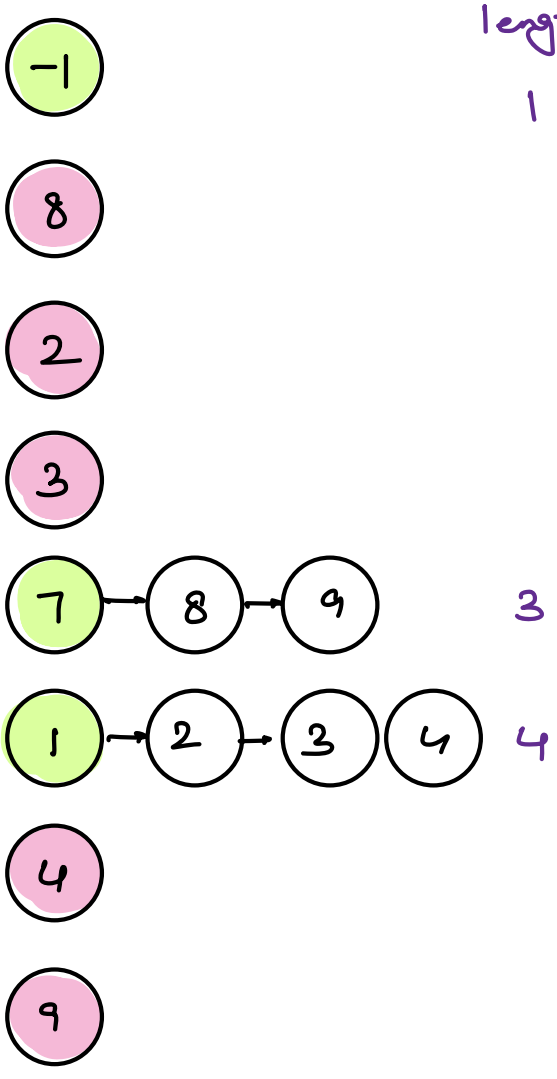
$$TC = \frac{n(n+1)}{2} \simeq O(n^2)$$

[x] [x-1]

Obs → If x-1 is present, then x can't be the starting point

arr = {-1 8 2 3 7 1 4 9}
0 1 2 3 4 5 6 7

set = {-1 8 2 3 7 1 4 9}



TC = O(2n)
= O(n)

Pseudocode

```
HashSet<Integer> set = new HS<>();

for ( i=0; i<n; i++) { set.add (arr[i]);
```

```
for (i=0 ; i<n; i++) {
    int x= arr[i]
    if ( set. contains (x-1) == false )
        chain= 1
        y = x+1
        while (set.contains(y)){
            chain++ :
        }   y++ ;
        ans= Math.max (ans , chain) ;
    }
}
return ans ;
```

Iterate on hashset

Google check

Value from hashset

arr = { 6 , 6 , 6, 6, 6, 6, 7, 8, 9 }

set = { 6, 7, 8, 9 }

↓

keys

ans

6 → 7 → 8 → 9 = 4

6 → 7 → 8 → 9 = 4

6 → 7 → 8 → 9 = 4

6 → 7 → 8 → 9 = 4

6 → 7 → 8 → 9 = 4

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

Prefix sum → { 2 4 5 **2** 6 9 10 **2** 8 6 7 }

Elements are repeating in psum

↳ we have the subarray sum = 0

Insert the subarray sum (psum) in hm

& if it comes again → update the ans

but don't update the value of last occurrence

Code → {TODO}