Today's Agenda

Hashing - Internal Implementation.

will start in next 3-4 minutes.

Q: Given an array of size N & Q queries. In each query, an element is given.

We need to check if that element exists or not in the given array.

arr[7] → [2, 4, 11, 15, 6, 8, 14, 9]

Q=4

K=4 → true

K=10 → false

K=17 → false

K=14 → true.

B.f idea. →

for every query, iterate on array elements & check if that element is present or not.

[T.C → O(Q*N)
S.C → O(1)]

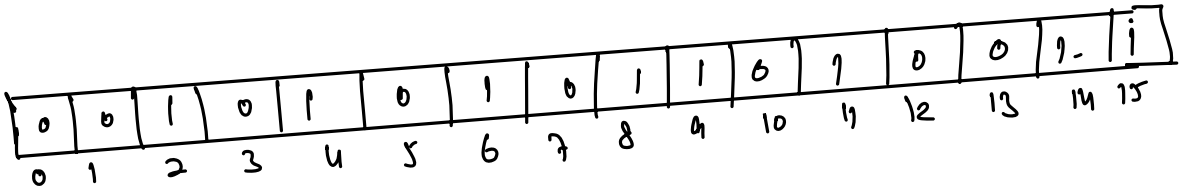
idea-2. → Sort the array & apply B.S for every query.

[T.C → O((Q+N)·logN)
S.C → depends on inbuilt sorting algo used]

Idea-3.

Create an array where i^{th} index denotes the presence of i.

arr[] → [2, 4, 11, 15, 6, 8, 14, 9]



Q=4

K=4

K=10

K=17

K=14

insert → O(1)

search → O(1)

delete → O(1)

Issues with this representation →

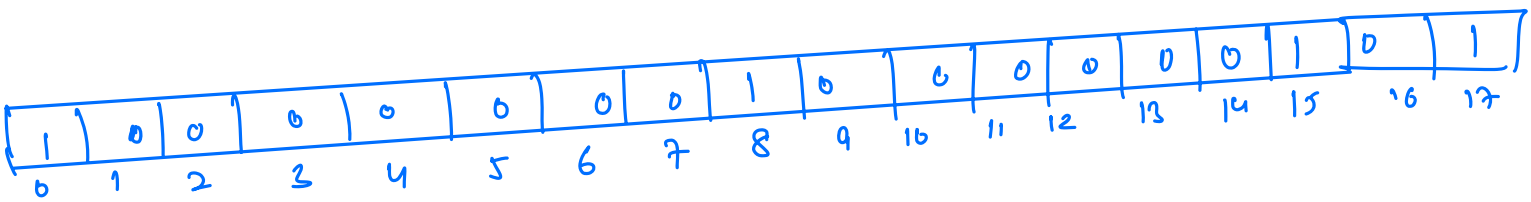
① Wastage of space → [2, 10^5, 15]

② Inability to create big arrays

arr → [100, 3, 10^9, 16, 1]

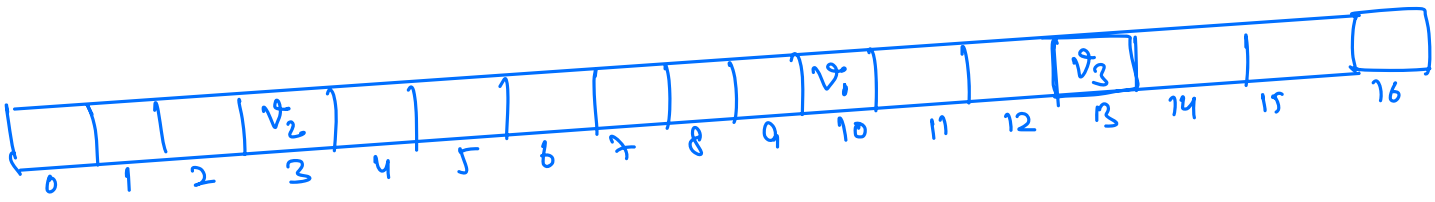
③ Adjustments for -ve values →

arr[] → [12, 3, -5, 10]



Assumption - Max size of array possible → 17
[0-16]

arr[] → [10, 20, 30, 27]



Collision

10 % 17 → 10

20 % 17 → 3

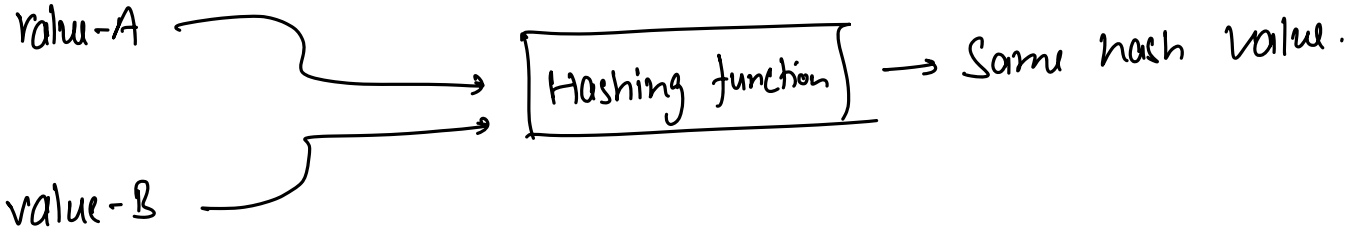
30 % 17 → 13

27 % 17 → 10

[Key % 17] → Hashing function.

Hashing → Hashing is a process where we pass our data through the hash function.

↳ give us the index to which data must be mapped.

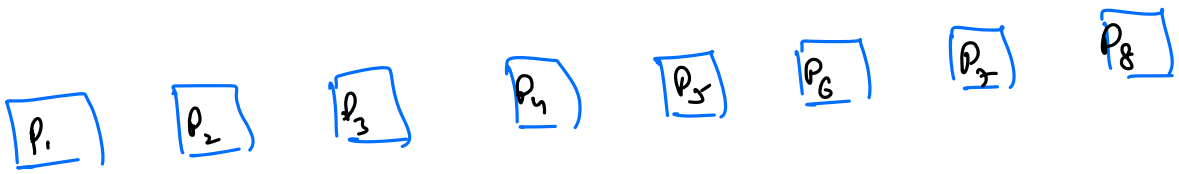


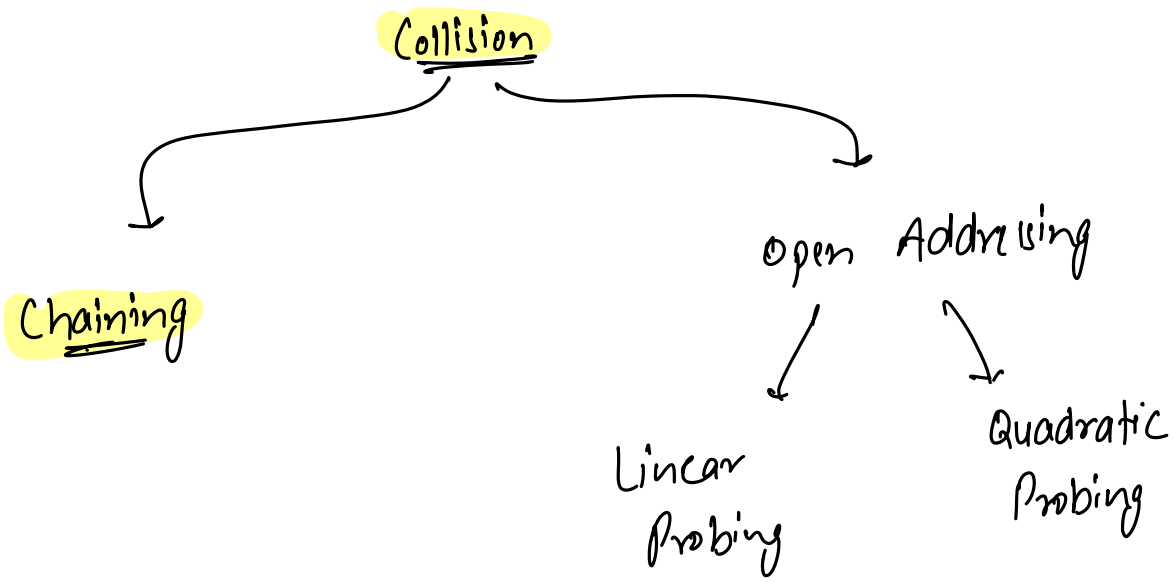
Q → Can we completely avoid collision?

Not possible.

Pigeon - Hole principle.

11 - pigeons & 8 - holes





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

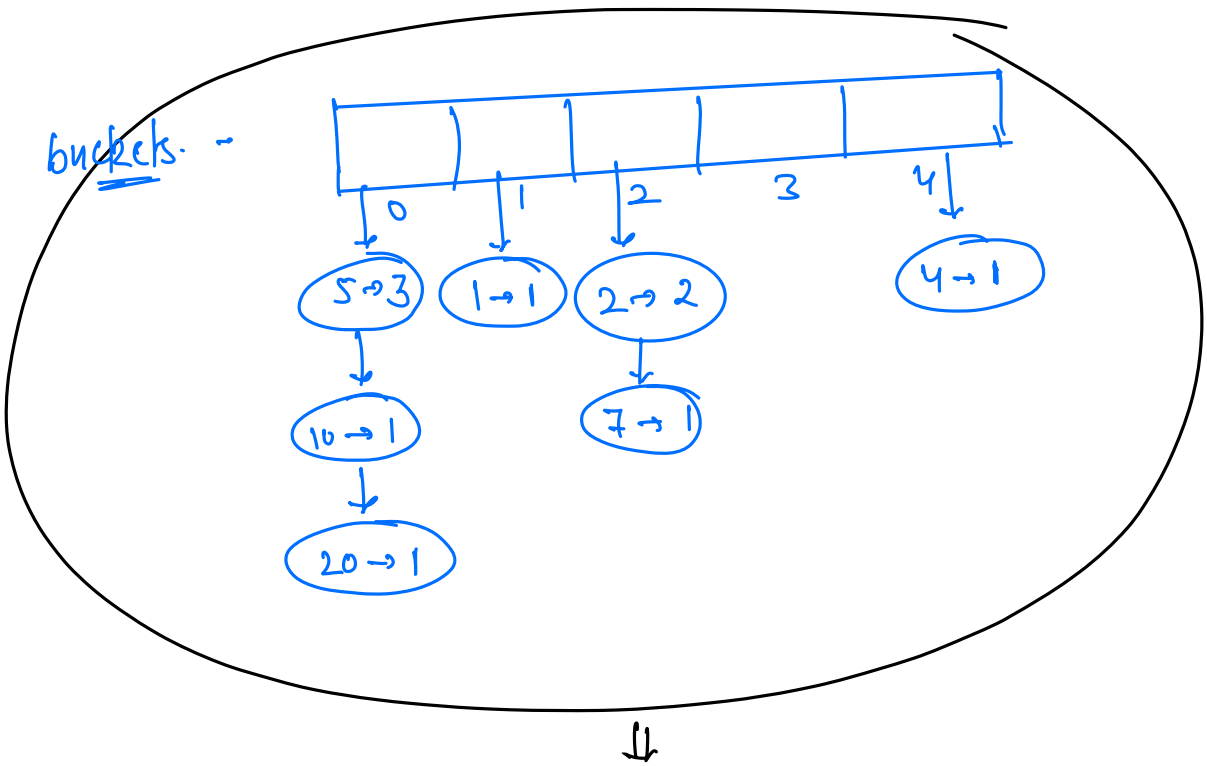
0 1 2 3 4 5 6 7 8 9

→ store frequency of all the elements.

Assm :- max size of array allowed → 5

Hashing function → element % 5

buckets :-



⇓

Array of linked-lists ⇒ Hashmap.

Average size of linked-list = $\frac{3+1+2+0+1}{5} = \frac{7}{5} = 1.4$

[Loading factor (λ)]

λ = total no. of nodes / size of array

Insertion →

① Calculate hash value or bucket index.

② Check if key is already present in bucket

→ if it is already present (update value)

→ otherwise, add a new Node.

T.C → O(λ)

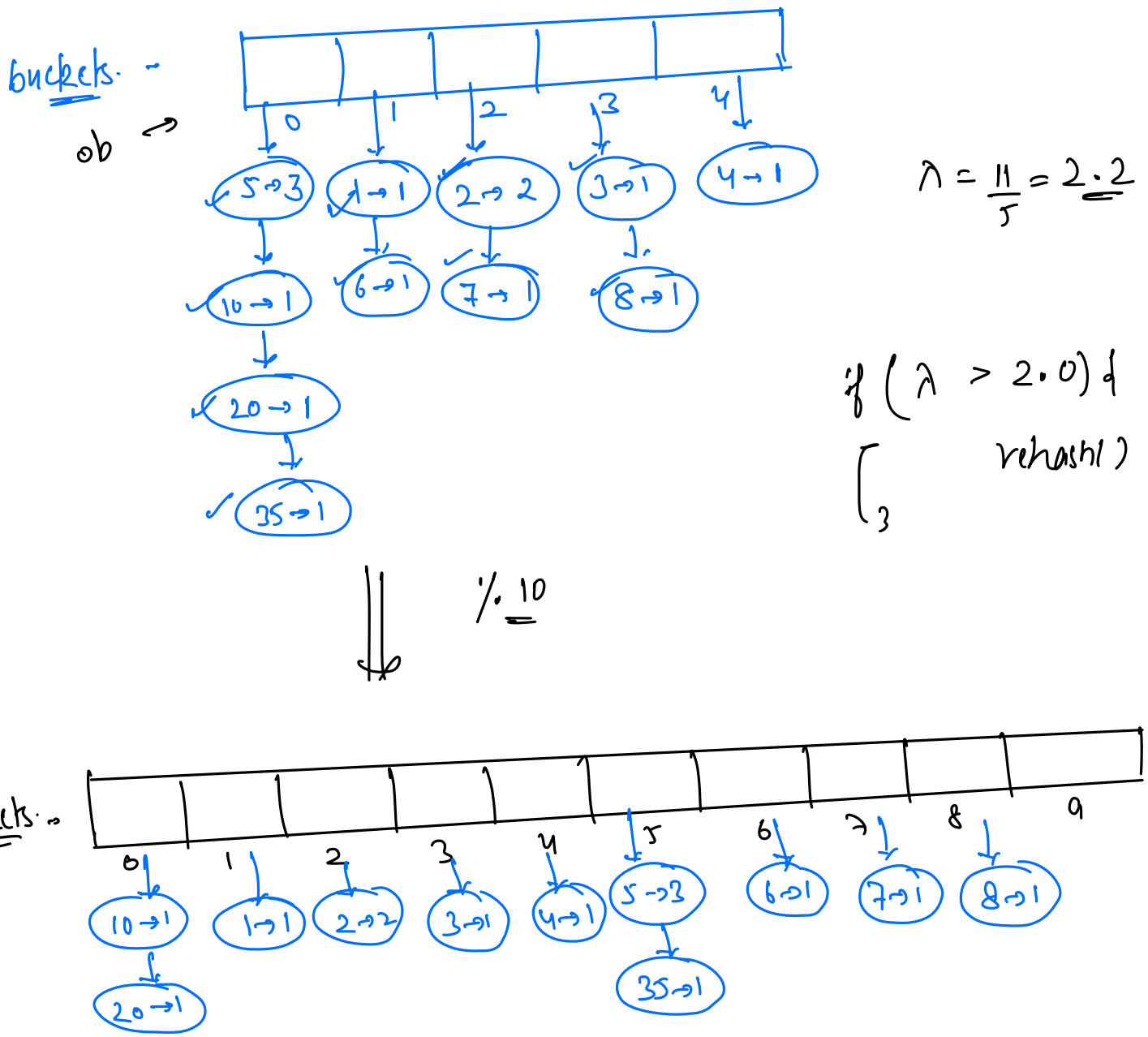
If we can somehow maintain, the average size of linked list as constant (λ)

T.C for insertion → Averagely constant.

Implementation of Hashmap →

https://www.scaler.com/topics/java/online-java-compiler/?snippet_slug=746dda157372a0ba6f98

T.C for insertion, deletion, searching → O(1)



$\lambda = \frac{11}{10} = 1.1 < 2.$