Welcome ☺

Agenda : Hashing
Collisions
Types of collisions.

Q Given an array of size N and Q queries. In each query an element is given. We have to check whether the element exist or not in the given array.

eg: [2, 4, 11, 14]

K = 4 => True
K = 7 => False.

Bruteforce

∀query iterate the array.

T.C = O(N * Q)

DAT → Direct Access Table

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
x	x	✓	x	✓	x	x	x	x	x	x	✓	x	x	✓

Advantages

→ Insertion O(1)
Deletion O(1)
Searching O(1)

Potential Issues

1) Wastage of space.

$A = \{ 23, 70, 90 \}$

2) Inability to create big arrays

10^6 or 10^7 array.

3) Handling negative values.

Overcoming issues.

→ array size of 10

$A : \{ 21, 42, 37, 45, 99, 30 \}$

$21 \% 10 \Rightarrow 1$ → Hash Key

$42 \% 10 \Rightarrow 2$

$37 \% 10 \Rightarrow 7$

$45 \% 10 \Rightarrow 5$

$99 \% 10 \Rightarrow 9$

$30 \% 10 \Rightarrow 0$

Hash Table

0	1	2	3	4	5	6	7	8	9
30	21	42			45		37		99

Issues in hashing.

eg: [21, 42, 37, 45, 77, 99, 31 }

21 % 10 ⇒ 1

42 % 10 ⇒ 2

37 % 10 ⇒ 7

45 % 10 ⇒ 5

77 % 10 ⇒ 7

99 % 10 ⇒ 9

31 % 10 ⇒ 1

colliding

0	1	2	3	4	5	6	7	8	9
	21	42			45		37		99
	31						77		

⇒ Can we completly avoid collisions ?

NO

Pigeon Hole Principle

11 Pigenons ⇒ 8 holes

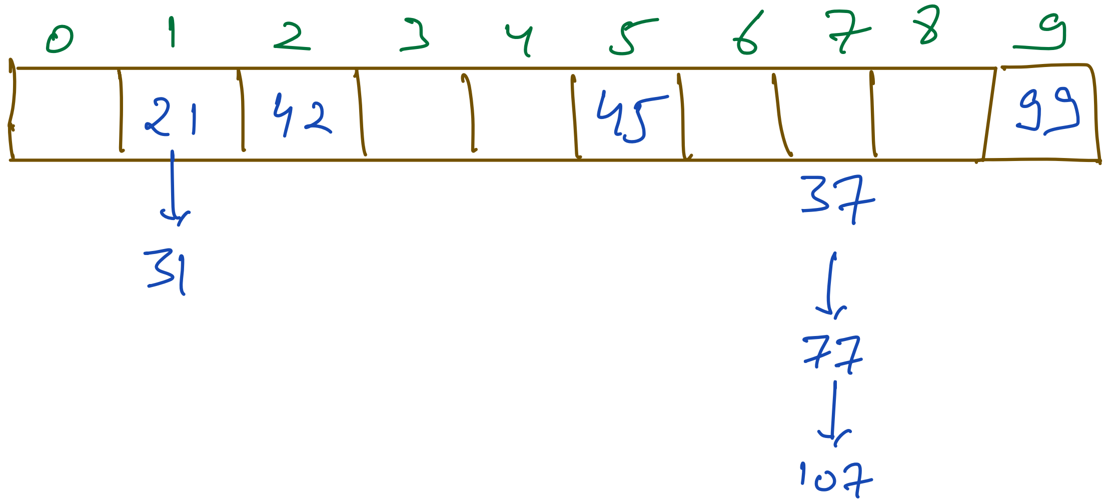
Collission Resolution Techniques

1) Open Hashing
   - ↓ Chaining ✓

2) Closed Hashing
   - Linear Probing
   - Quadratic Probing
   - Double Hashing

Chaining

⇒ Array of linked lists.



Time complexity.

1) Insertion.

head → O(1)

tail → O(N) time

2) Deletion / Searching.

T.C ⇒ O(N) worst case

λ function

→ ratio of (# elements inserted)/(size of array)

→ retriggers hashing to avoid/reduce collisions.

Key ~~Value~~