Indian Institute of Information Technology Kottayam

Hashing

By

Arun Cyril Jose

Hashing

- Operations in Constant Time:
  - Insert
  - Delete
  - Search

- **Have unique values**.
- Hashing is not useful for:
  - Finding the closest value
  - Sorted data
  - Prefix searching

Applications of Hashing

- Creating Dictionaries.
- Database Indexing.
- Cryptography.
- Working of Cache Memory
- Packet Mapping in Network devices like Routers and Switches.

Hashing

- Direct address Table

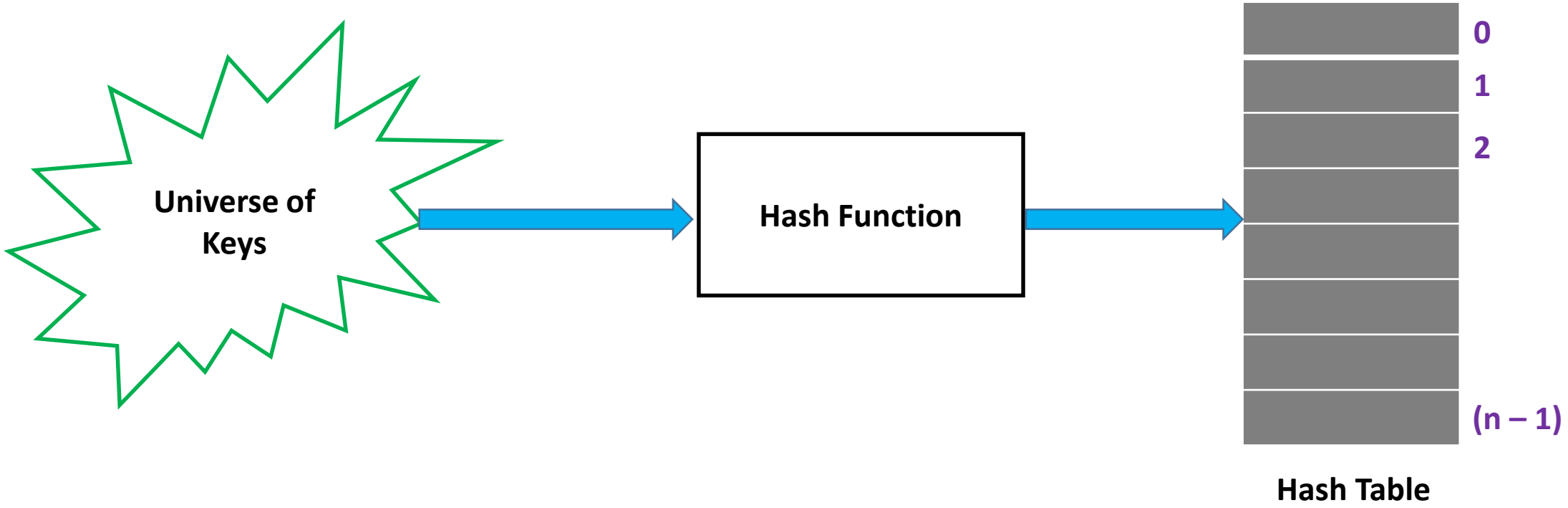
1	2	3	4	5	.....	1000
0	1	2	3	4	.....	999

1	0	1	1	0	.....	1
0	1	2	3	4	.....	999

Boolean Array

- Why Hashing is Relevant or Significant ?

Hashing



Hashing Vs Direct Address Table

- Hash function should map a larger key to a smaller key.

- Direct address Table

1	2	3	4	5	…….	1000
0	1	2	3	4	…….	999

1	0	1	1	0	…….	1
0	1	2	3	4	…….	999

Boolean Array

Hash Functions

- Hash Function

$$h(\text{large\_key}) = \text{large} \% m$$

Choose a Prime number

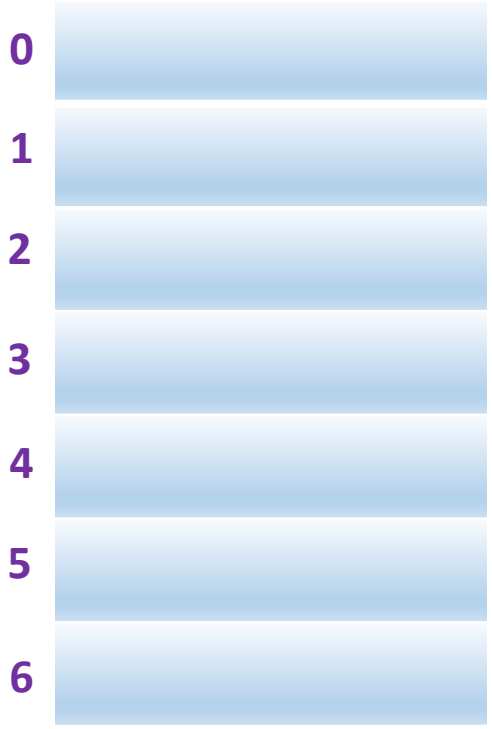
Avoid numbers that are close to power of 10 or 2

- 7, 10, 12

- “abcd”, “bcda”, “dcba”

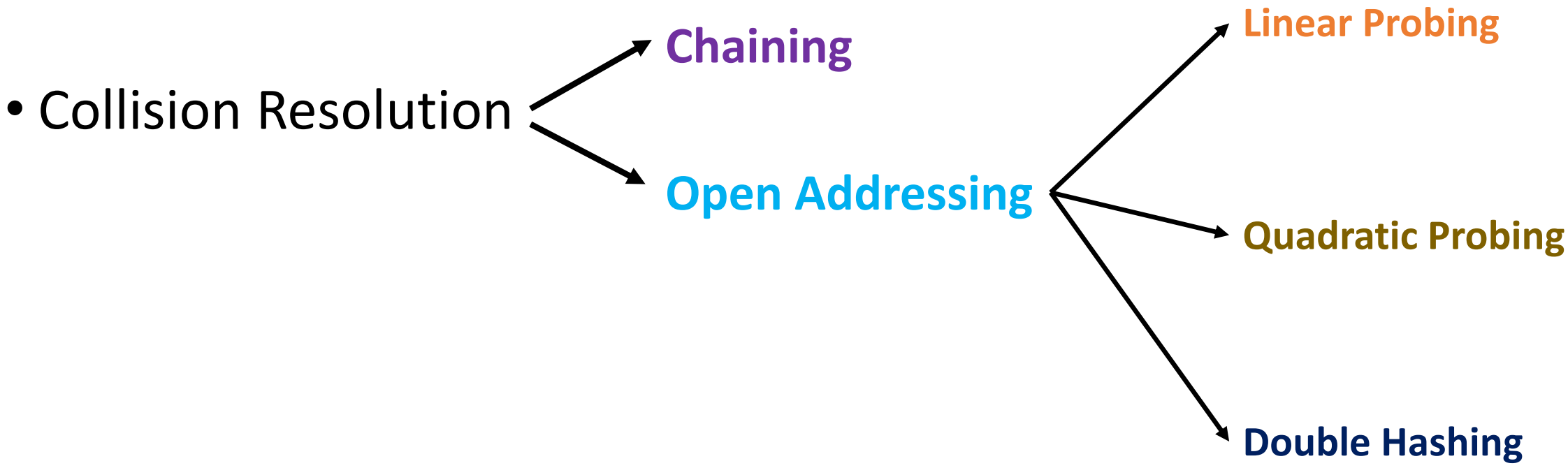
Hash Functions and Collisions

- 23, 47, 49, 21, 6



Hash Table

Collision Resolution



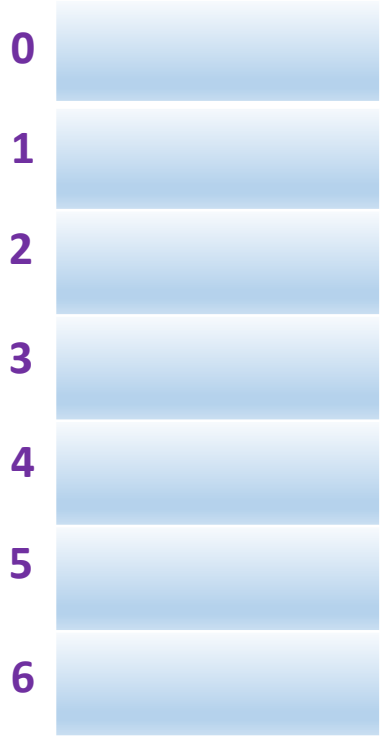
Birthday Paradox

- Probability of two or more people in a group of 23 sharing the same birthday is greater than 50%.

- If the number of people in the group is increased to 70 then the probability of two people sharing the same birthday is increased to 99.9%.

Chaining

- Hash Function: key % 7
- Keys: 50, 21, 58, 17, 15, 49, 56, 22, 23, 25



Hash Table

Expected Time to Search = O (1 + α)

Chaining

- n: number of keys to be inserted.
- m: number of slots in Hash Table.

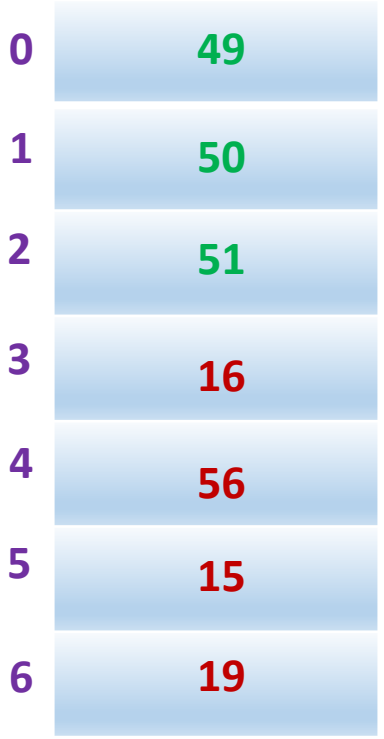
- Load Factor $\boldsymbol{\alpha} = \frac{n}{m}$

- $\boldsymbol{\alpha}$ should be as small as possible.

Open Addressing

- Number of slots in the Hash Table **>=** Number of Keys to be inserted.

- Keys: 50, 51, 49, 16, 56, 15, 19
- Hash (key) = key % 7

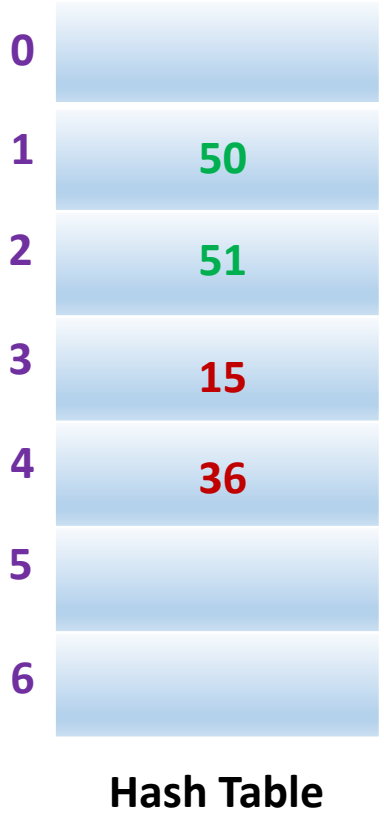


Hash Table

Open Addressing

- insert (50), insert (51), insert (15), search (15), delete (15), search (15)
- Hash (key) = key % 7

- For Search,
- Empty Slot
- Key is Found
- Traversed through the whole Hash Table.
- Imagine before delete (15) we have insert (36).
- delete (15), search (36).



Linear and Quadratic Probing

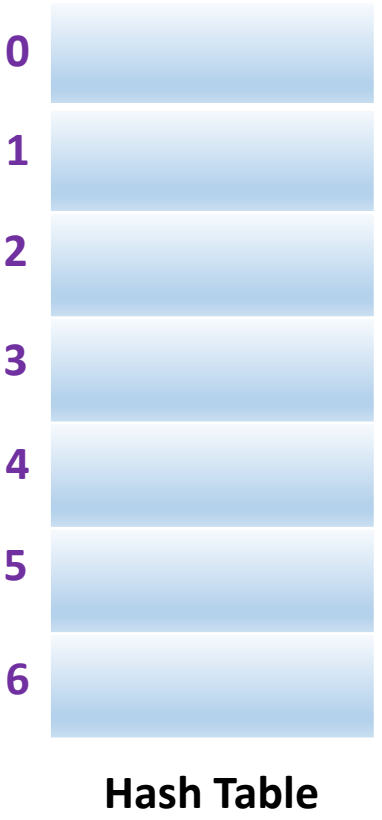
- Clustering Problem.

Linear Probing: (h(key) + i) % m

- **Quadratic Probing**

Quadratic Probing: (h(key) + i^2) % m

- Secondary Clustering Problem.



Linear and Quadratic Probing

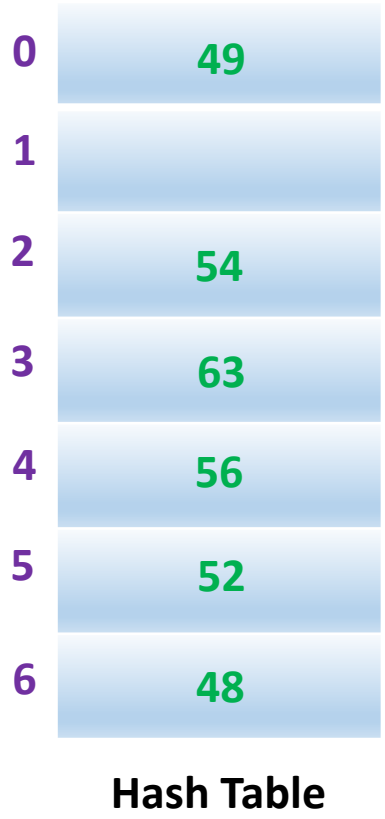
- In Linear and Quadratic Probing,
  1. **$\alpha < 0.5$** i.e. number of slots should be more than double the number of keys.
  2. 'm' should always be prime.

Double Hashing

- Two Hash functions are used.

$$hash(key, i) = (h_1(key) + i\, h_2(key)) \% m$$

- 49, 63, 56, 52, 54, 48



Chaining Vs Open Addressing

- **Chaining**
  - Hash Table never Fills [Chain size gets bigger].
  - Less sensitive to hash function.
  - Poor Cache Performance.
  - Extra space for Links or chains.

- **Open Addressing**
  - Table may become full, may have to resize the table.
  - Extra care required to avoid clustering.
  - Cache Friendly.
  - No extra space for Links.