Hash Table Data Structure

By Dr Shantanu Pathak

Hash Table Introduction

- Stores data in <Key,Value> format
- For efficient search on any size of data
- Insertion, deletion and Searching is very fast compared to array / other data structures

- How searching is made faster here ?
- By using Hashing function while inserting the elements
- Hashing function directly gives the location based on key
- So in single step one can find the element <key,value>

Hash Table Example

- There are 7 keys to be stored
- [23,56,21,45,78,99,77]
- Assume hash function **f = (key % 10)**
- So, hash value of every key is
- [3,6,1,5,8,9,7]

- **Find 77** key in Hash Table
- Calculate hash function f for 77 -> 7
- So, 77 will be found at index 7

- **Find 60** key in Hash Table
- Calculate hash function f for 60 -> 0 (Not found)

Index	Key
0	
1	21
2	
3	23
4	
5	45
6	56
7	77
8	78
9	99

Hash Table Performance

- Search -> O(1)
- Insert -> O(1)
- Delete -> O(1)

Performance depends on GOOD hashing Function

What is Good hashing Function?

fast

avoids same index for different keys (collision)

evenly spreads data across the whole table

Types of Hashing Functions

- 1. Division method
  - Mod with value n
- 2. Multiplication Method
  - Multiply with a constant
- 3. Universal hashing
  - Select a random hashing function from set of hashing functions every time

Collision and Its Resolution Techniques

- What happens if hash function returns same index for multiple keys?
- Its called as **collision**

- Ex. F= (key % 10)
- then key 34 and 24 will have same index 4

- F = (key % 19)
- Will there be collision in this hash function

Collision Resolution Techniques

- Separate Chaining (Most popular and used)
- Linear Probing
- Quadratic Probing
- Double Hasing