**🔷 THEORY**

**📌 Hash Table**

A **hash table** is a data structure that stores data in key-value pairs using a hash function to calculate the index into an array.

**📌 Hash Function**

A hash function converts a given key (telephone number) into an index in the hash table.  
Basic form used:

ini

CopyEdit

index = key % size

**📌 Collision Handling**

When two keys hash to the same index, a **collision** occurs.  
To resolve collisions, we use:

* **Linear Probing**: Check next slots sequentially (index + i) % size.
* **Quadratic Probing**: Check slots using squares of i  
  (index + i^2) % size.

**📌 Use Case**

Telephone numbers are stored in a hash table and searched efficiently using the above collision strategies. The number of comparisons is counted to evaluate performance.

**🔷 ALGORITHM**

**🔸 INSERT (Linear Probing)**

1. Compute index: index = key % size
2. If slot is empty, insert.
3. If occupied, check next slot (index + i) % size until empty.
4. Insert and print number of comparisons.

**🔸 INSERT (Quadratic Probing)**

1. Compute index: index = (key + i²) % size
2. If slot is empty, insert.
3. If occupied, increase i and repeat.
4. Insert and print number of comparisons.

**🔸 SEARCH (Linear Probing)**

1. Compute index: index = key % size
2. Check if slot contains key.
3. If not, probe (index + i) % size.
4. If found, report success and comparisons; else, not found.

**🔸 SEARCH (Quadratic Probing)**

1. Compute index: index = (key + i²) % size
2. Check if slot contains key.
3. If not, increase i and repeat.
4. If found, report success and comparisons; else, not found.

