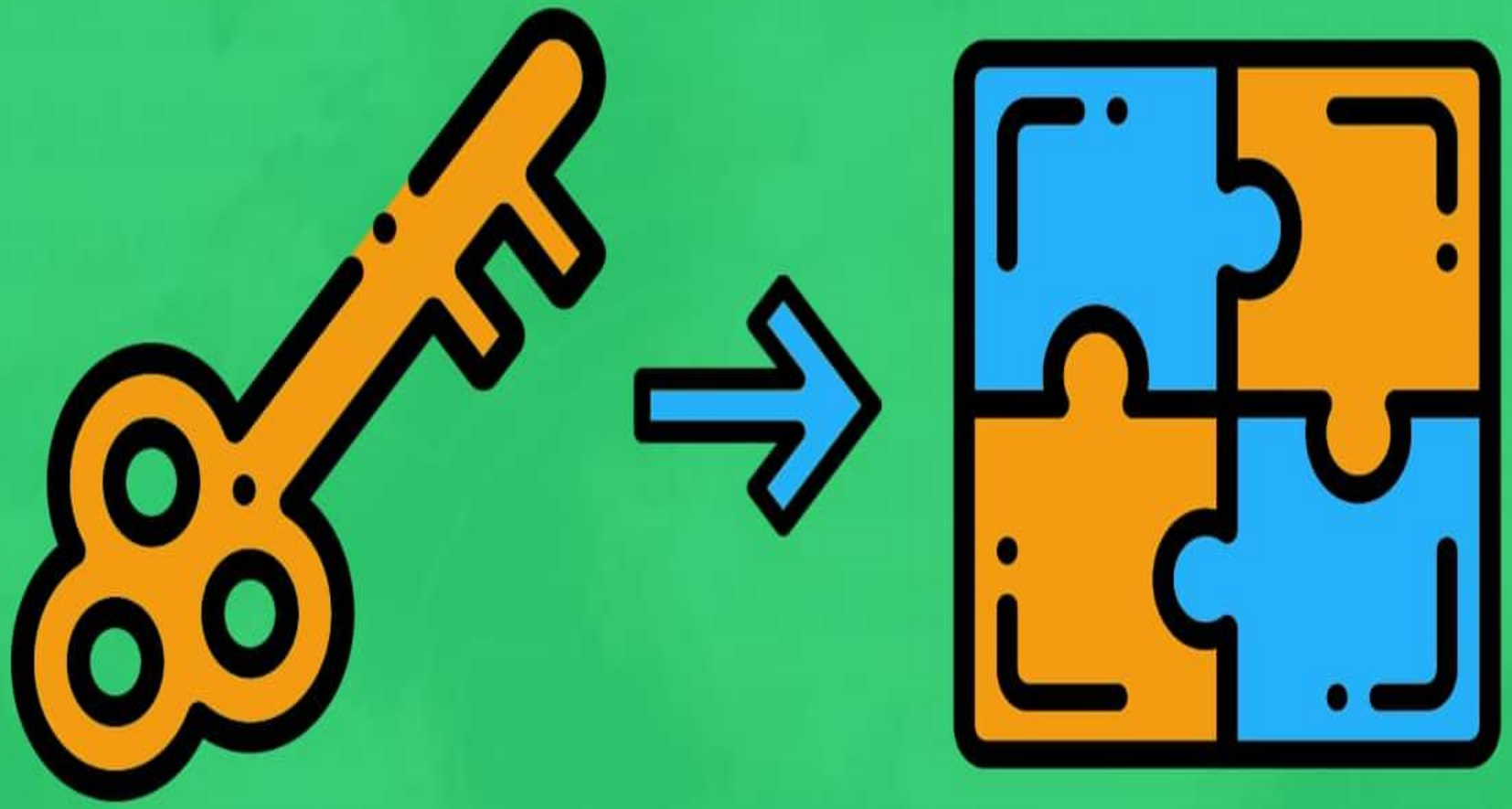
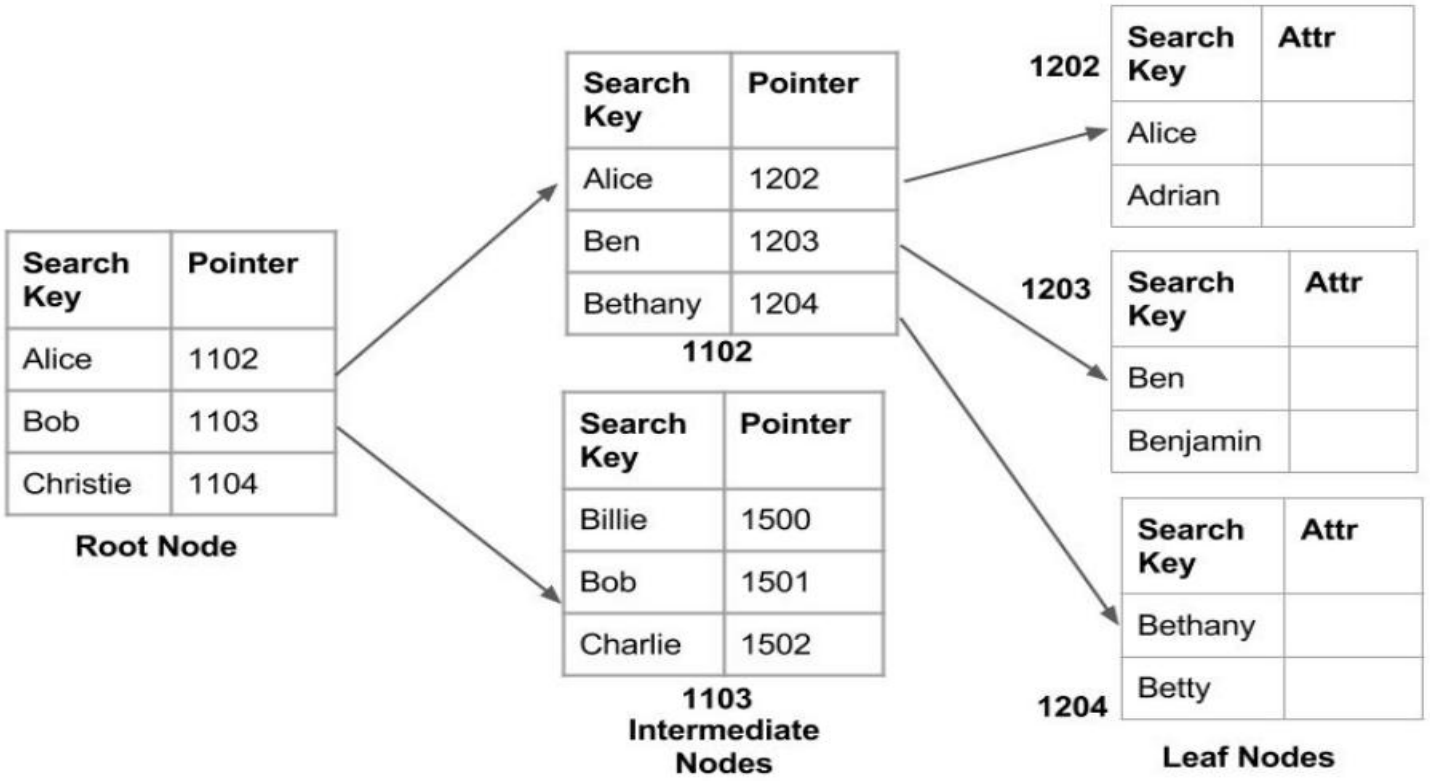


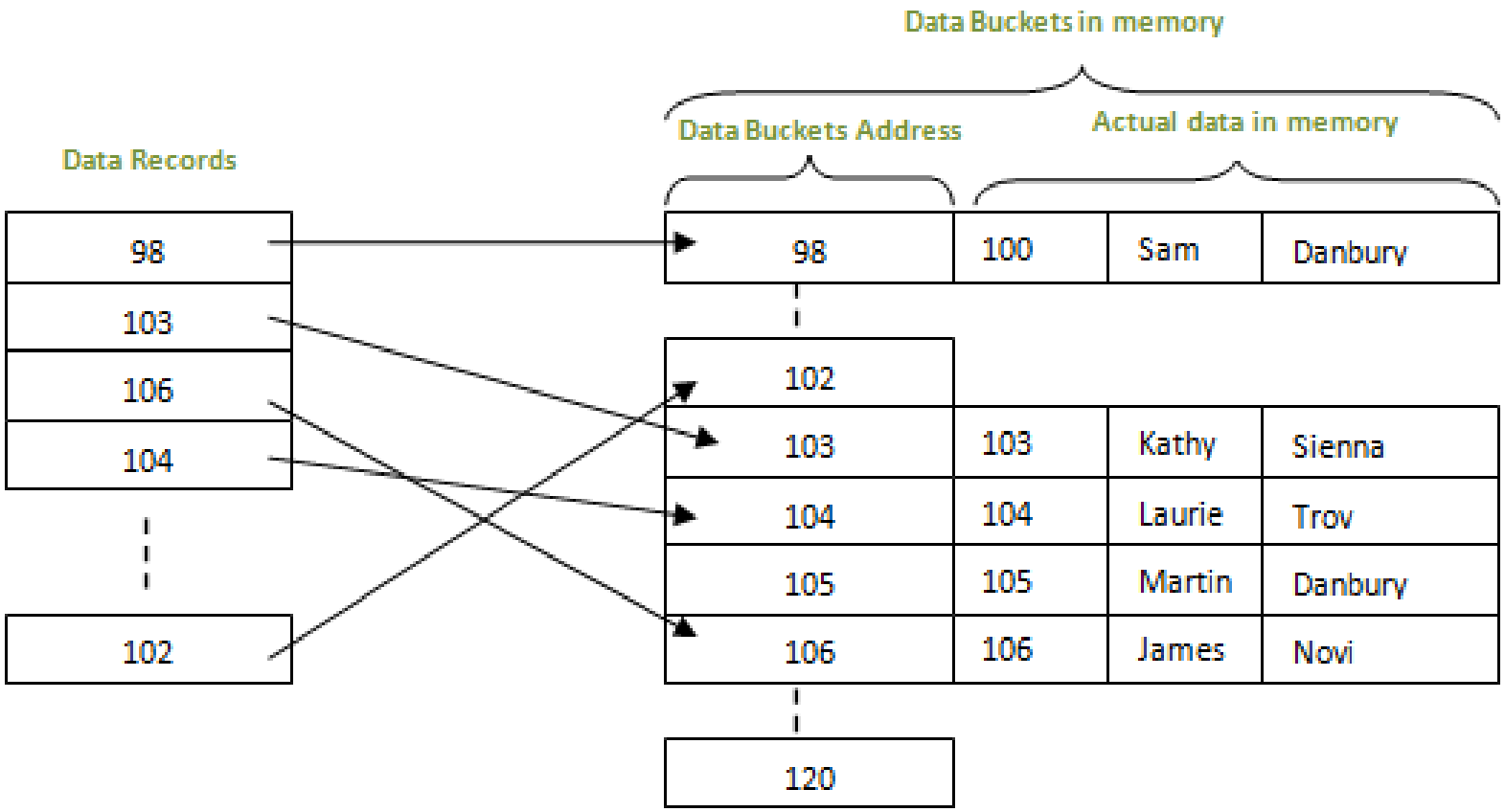
What is Hashing?

Difference between Indexing and Hashing

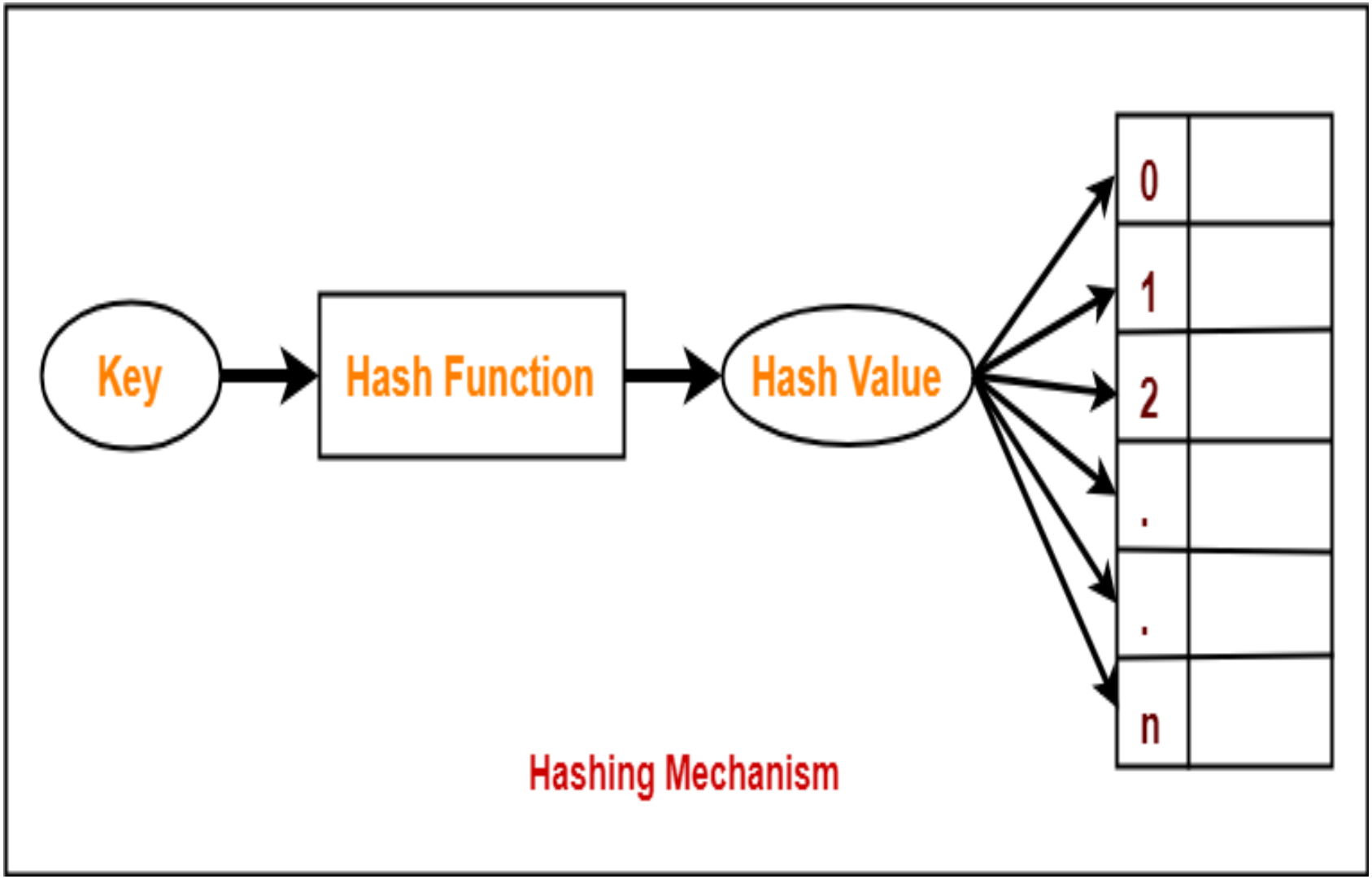


Non clustered index

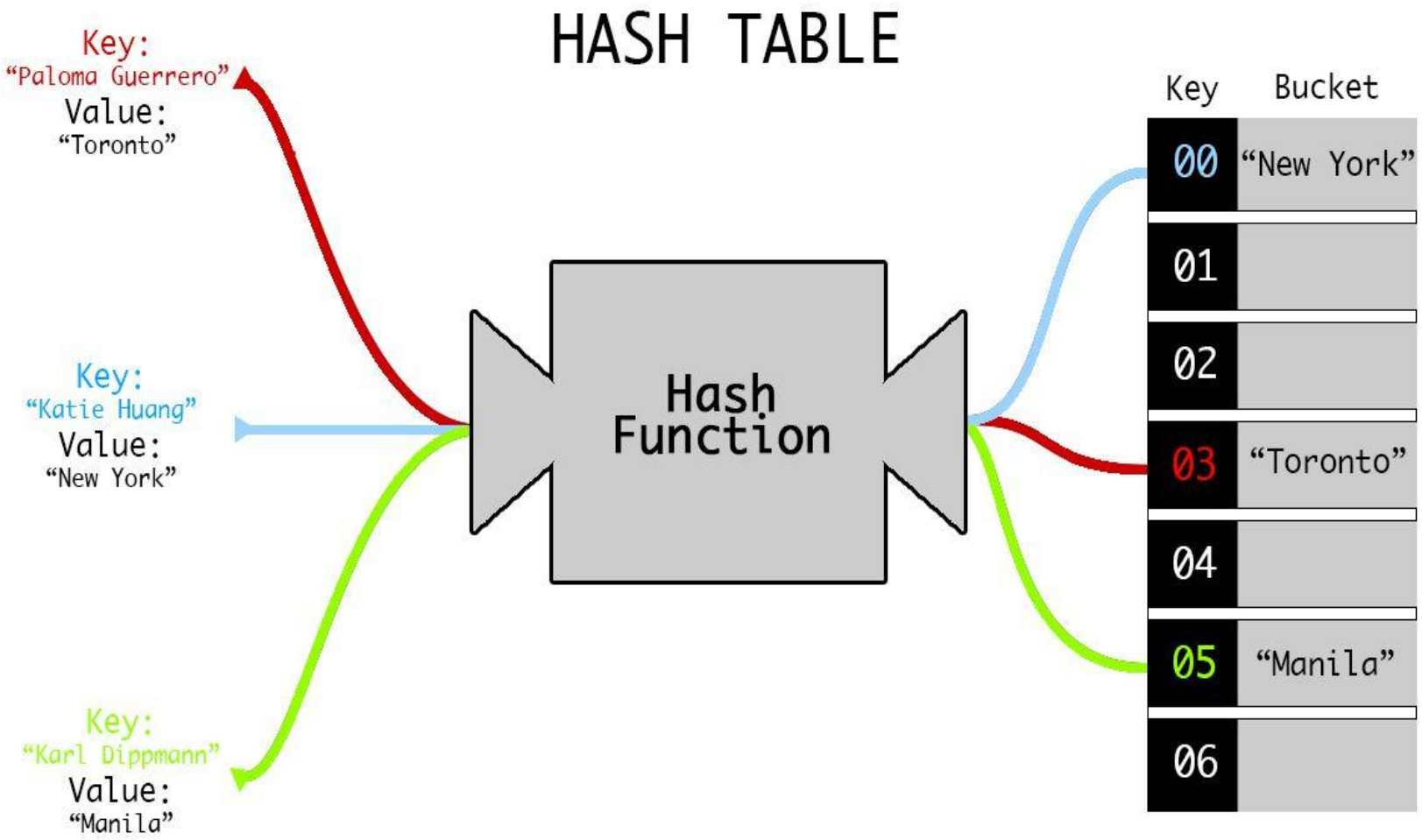
Difference between Indexing and Hashing



What is Hashing ?



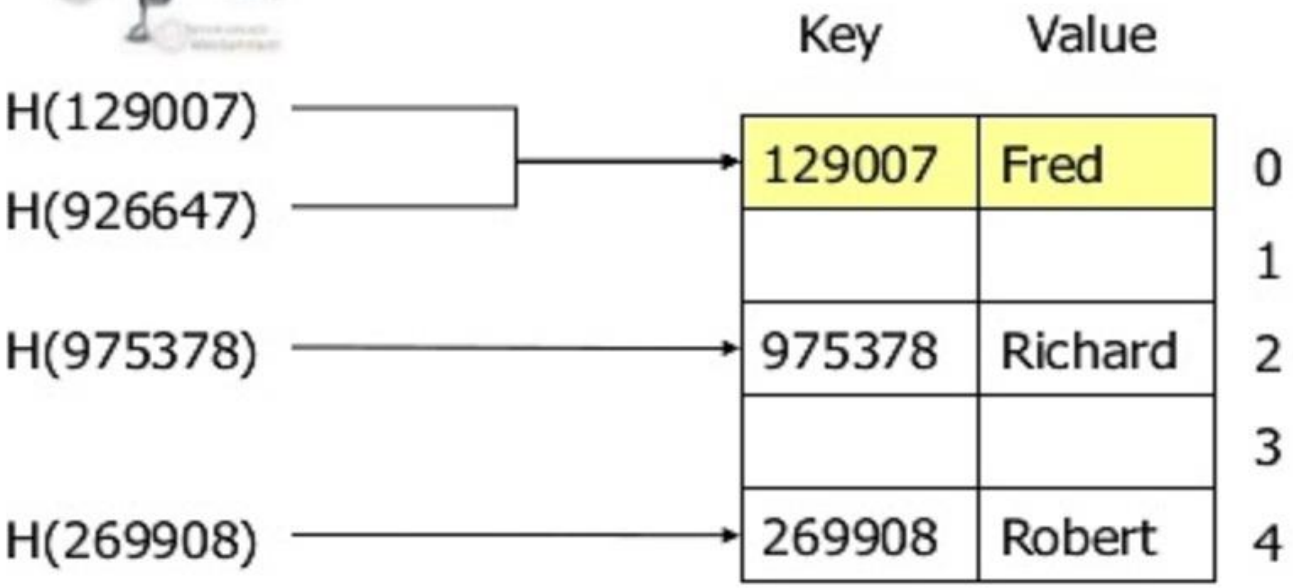
Hash Table

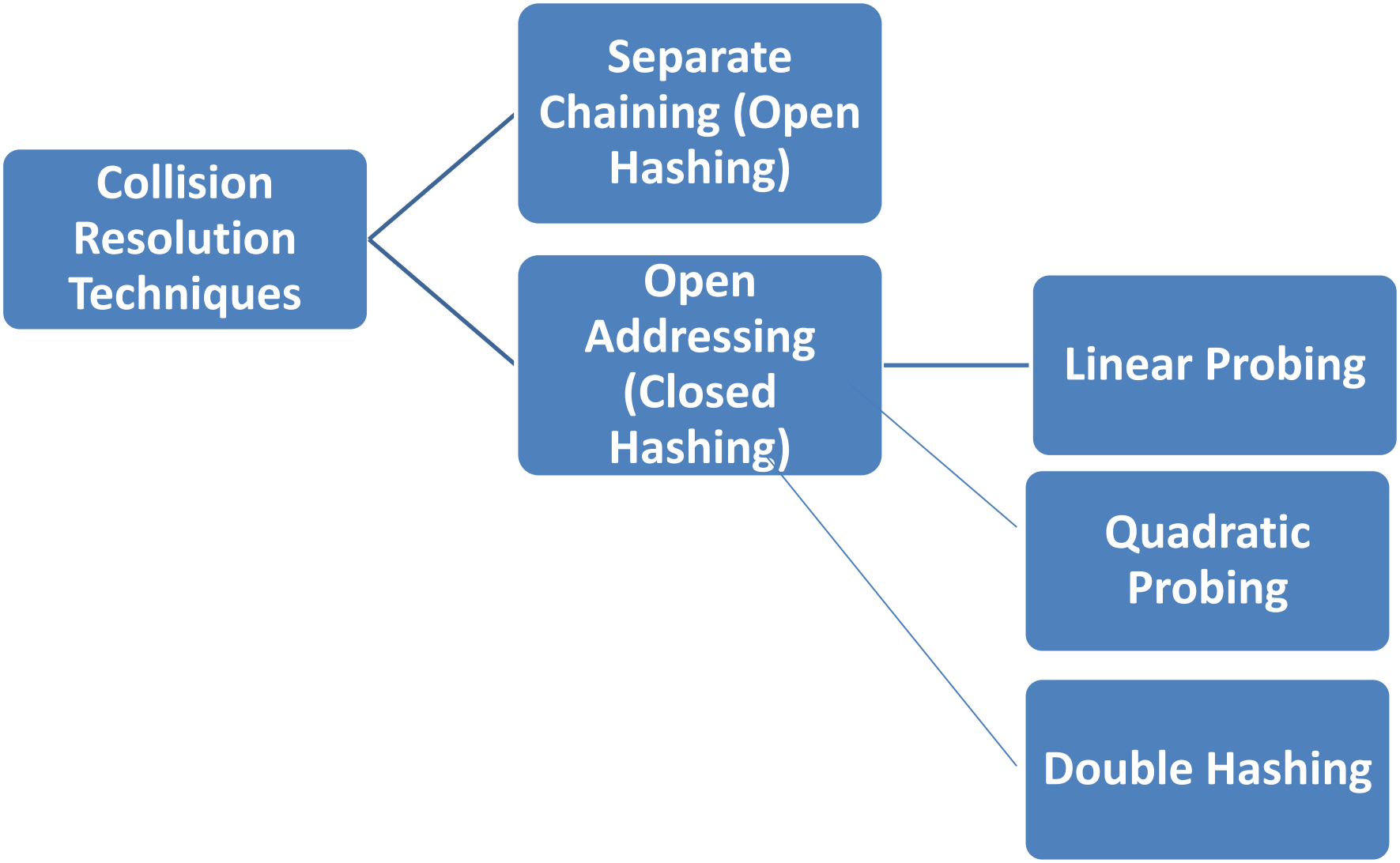

HASH TABLE
Key:
"Paloma Guerrero"
Value:
"Toronto"
Key:
"Katie Huang"
Value:
"New York"
Key:
"Karl Dippmann"
Value:
"Manila"
Hash Function
Key
Bucket
00
"New York"
01
02
03
"Toronto"
04
05
"Manila"
06


COLLISION

Collision

- If the hash function returns a slot that is already occupied then there is a collision.



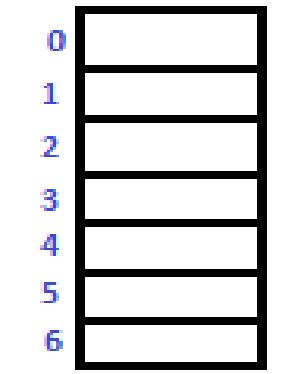

Collision Resolution Techniques
Separate Chaining (Open Hashing)
Open Addressing (Closed Hashing)
Linear Probing
Quadratic Probing
Double Hashing

Separate Chaining

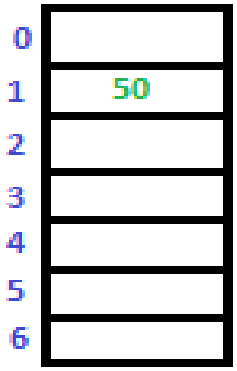
The idea is to make each cell of hash table point to a linked list of records that have same hash function value.

Let us consider a simple hash function as “**key mod 7**” and sequence of keys as 50, 700, 76, 85, 92, 73, 101.

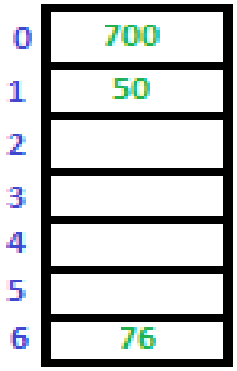
Separate Chaining



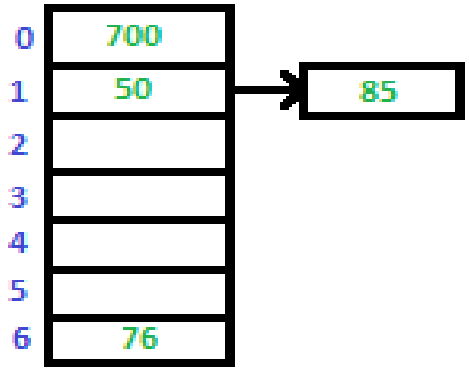
Initial Empty Table



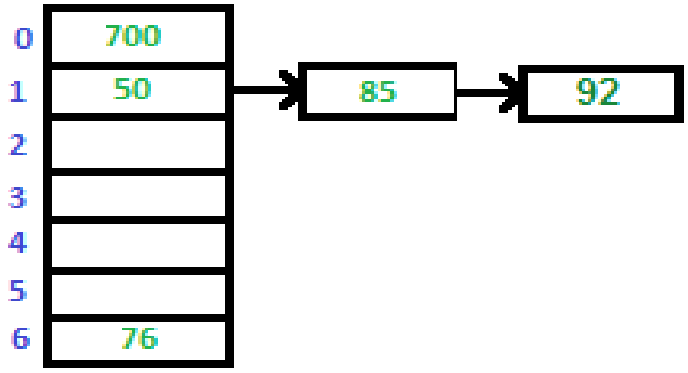
Insert 50



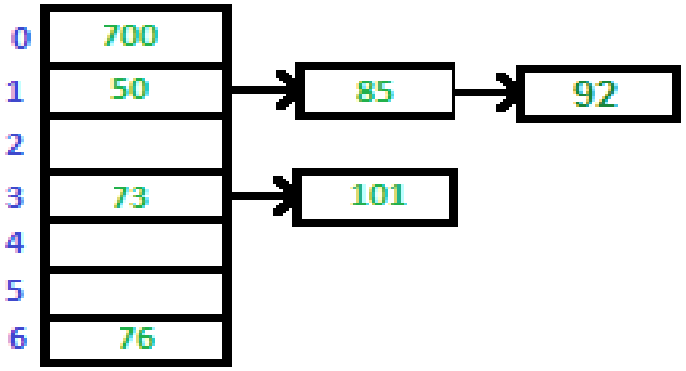
Insert 700 and 76



Insert 85: Collision
Occurs, add to chain



Inser 92 Collision
Occurs, add to chain



Insert 73 and 101

Performance of Chaining

m = Number of slots in hash table

n = Number of keys to be inserted in hash table

Load factor $\alpha = n/m$

Expected time to search = $O(1 + \alpha)$

Expected time to insert/delete = $O(1 + \alpha)$

Time complexity of search insert and delete is $O(1)$ if α is $O(1)$

Open Addressing

All elements are stored in the hash table itself. So at any point, size of the table must be greater than or equal to the total number of keys.

Open Addressing is done following ways:

- Linear Probing
- Quadratic Probing
- Double Hashing

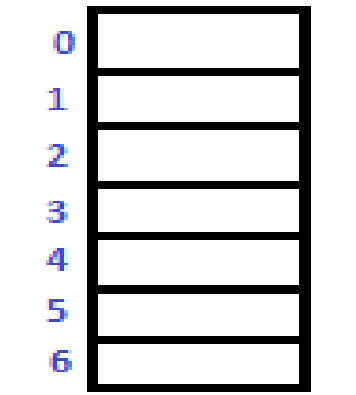
Linear Probing

- Linearly probe for next slot.
- Let **hash(x)** be the slot index computed using hash function and **S** be the table size

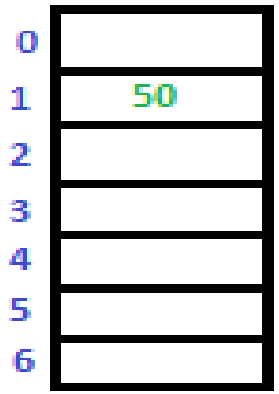
  If slot hash(x) % S is full, then we try (hash(x) +1) % S

Let us consider a simple hash function as “key mod 7” and sequence of keys as 50, 700, 76, 85, 92, 73, 101.

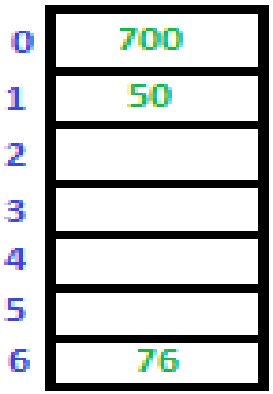
Linear Probing



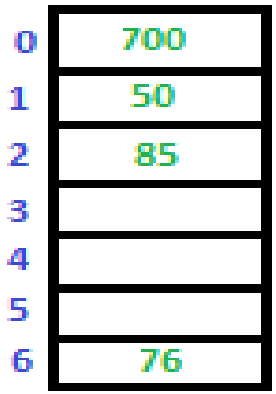
Initial Empty Table



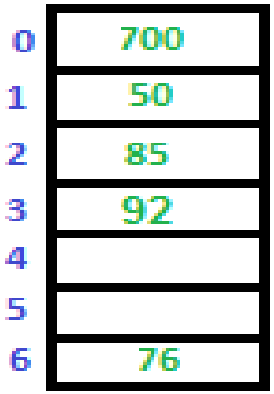
Insert 50



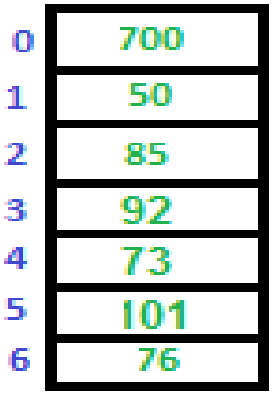
Insert 700 and 76



Insert 85: Collision Occurs, insert 85 at next free slot.



Insert 92, collision occurs as 50 is there at index 1. Insert at next free slot



Insert 73 and 101

Quadratic Probing

- We look for i^2'th slot in i'th iteration.
- Let hash(x) be the slot index computed using hash function.

If slot hash(x) % S is full, then we try (hash(x) + 1*1) % S

Double Hashing

- We use another hash function hash2(x) and look for i*hash2(x) slot in i'th rotation.
- Let hash(x) be the slot index computed using hash function.

If slot hash(x) % S is full, then we try (hash(x) + 1*hash2(x)) % S

Comparison

Linear Probing	• Best cache performance but suffers from clustering. • Easy to compute.
Quadratic Probing	• Lies between the two in terms of cache performance and clustering.
Double Hashing	• Poor cache performance but no clustering. • More computation time as two hash functions need to be computed.

Performance of Open Addressing

m = Number of slots in the hash table

n = Number of keys to be inserted in the hash table

Load factor $\alpha = n/m$ (< 1)

Expected time to search/insert/delete $< 1/(1 - \alpha)$

So Search, Insert and Delete take $(1/(1 - \alpha))$ time

Rehashing

- Hash Table may get too full.
- Solution: Rehash

  Build another table that is twice as big and has a new hash function.

  Move all the element from smaller to bigger table.
- Cost of Rehashing = O(N)

Rehashing Example

Original Hash Table

0	6
1	15
2	
3	24
4	
5	
6	13

After Inserting 23

0	6
1	15
2	23
3	24
4	
5	
6	13

After Rehashing

0	
1	
2	
3	
4	
5	
6	6
7	23
8	24
9	
10	
11	
12	
13	13
14	
15	15
16	

QUESTIONS

Given the following input (4322, 1334, 1471, 9679, 1989, 6171, 6173, 4199) and the hash function x mod 10, which of the following statements are true?
i. 9679, 1989, 4199 hash to the same value
ii. 1471, 6171 has to the same value
iii. All elements hash to the same value
iv. Each element hashes to a different value
(A) i only
(B) ii only
(C) i and ii only
(D) iii or iv

QUESTIONS

Consider a hash table with 100 slots. Collisions are resolved using chaining. Assuming simple uniform hashing, what is the probability that the first 3 slots are unfilled after the first 3 insertions?

(A) (97 × 97 × 97)/100^3

(B) (99 × 98 × 97)/100^3

(C) (97 × 96 × 95)/100^3

(D) (97 × 96 × 95)/(3! × 100^3)

QUESTIONS

Which one of the following hash functions on integers will distribute keys most uniformly over 10 buckets numbered 0 to 9 for i ranging from 0 to 2020?

(A) h(i) =i^2 mod 10

(B) h(i) =i^3 mod 10

(C) h(i) = (11 $*$ i^2) mod 10

(D) h(i) = (12 $*$ i) mod 10

QUESTIONS

How many different insertion sequences of the key values using the same hash function and linear probing will result in the hash table given in Question 3 above?

(A) 10
(B) 20
(C) 30
(D) 40

QUESTIONS

A hash table of length 10 uses open addressing with hash function h(k)=k mod 10, and linear probing. After inserting 6 values into an empty hash table, the table is as shown below.

0	
1	
2	42
3	23
4	34
5	52
6	46
7	33
8	
9	

Which one of the following choices gives a possible order in which the key values could have been inserted in the table?

(A) 46, 42, 34, 52, 23, 33
(B) 34, 42, 23, 52, 33, 46
(C) 46, 34, 42, 23, 52, 33
(D) 42, 46, 33, 23, 34, 52