# Hashing

A technique or process of mapping keys to values into the hash table by using a hash function. It is done for faster access of elements. The efficiency of hashing depends upon the hash function used.

## Hash Function

The hash function converts a large number or other key to a smaller number and uses the smaller number as a index in a table called hash table. **Hash table is an array that stores pointer to records.** A good hash function must have following properties :

- Efficiently computable.
- Uniformly Distribute Key (Each table position equally likely for each key)

## Collision Handling

A hash function gives us a small number for a big key, there is a possibility that two keys result in a same value. The situation where a newly inserted key maps to an preoccupied slot in hash table is called collision.
We use collision handling technique to handle these situation :-

- **Chaining**
- **Open Addressing**

### Chaining

Here we make each cell of hash table point to a linked list. And perform addition of node when collision occurs. It's simple but requires additional memory outside hash table. Chaining works better when key is equally likely to be hashed to any slot of table(Uniform Hashing).

| Advantage | Disadvantage |
| --- | --- |
| Simple to implement. | Wastage of space. |
| Hash table never fills up. | If chain become long, search time can be O(N). |
| Less sensitive to the load factors. | Uses extra space for links. |

**Data Structures For Storing Chains:**

- Linked lists
    - Search: O(l) where l = length of linked list
    - Delete: O(l)
    - Insert: O(l)
    - Not cache friendly(Accessing)
- Dynamic Sized Arrays ( Vectors in C++, ArrayList in Java, list in Python)
    - Search: O(l) where l = length of array
    - Delete: O(l)
    - Insert: O(l)
    - Cache friendly

- Self Balancing BST ( AVL Trees, Red Black Trees)
    - Search: O(log(l))
    - Delete: O(log(l))
    - Insert: O(l)
    - Not cache friendly
    - Java 8 onwards use this for HashMap

## Open Addressing

Here no additional memory outside hash table is used. All elements are stored in hash table itself. So at any point, the size of the table must be greater than or equal to number of keys.

- Insert (k) : Keep probing until an empty slot is found. Once and empty slot
- Search (k) : Keep probing until slot's key doesn't become equal to k or an empty slot is reached.
- Delete (k) : We can't simply delete a key nor the search will fail. So slots of deleted keys are marked as "deleted". The insertion can happen at "deleted" slot but search doesn't stop at "deleted".

**Techniques for Open Addressing :**

1. Linear Probing : In linear probing we linearly probe for next slot.

    ```
    If slot hash(x)%s is full then (hash(x)+1)%s
    If slot (hash(x)+1)%s is full then (hash(x)+2)%s
    ```

    Challenges is Linear Probing :

    a ) Primary Clustering :- Many consecutive elements form groups and it starts taking time to find free slot or to search an element.

    b) Secondary Clustering :- Less severe , two records have same probe sequence if their initial position is same.

2. Quadratic Probing : We look for i^2th slot in ith iteration.

    ```
    If slot hash(x)%s is full then (hash(x)+1*1)%s
    If slot (hash(x)+1*1)%s is full then (hash(x)+2*2)%s
    ```

3. Double Hashing : We use another hash function and look for i*hash2(x) in ith rotation.

    ```
    If slot hash1(x)%s is full then we try (hash1(x)+i*hash2(x))%s
    ```

## Difference in Chaining & Open Addressing

| Chaining | Open Addressing |
|---|---|
| Simpler to implement. | Requires more computation. |
| Hash table never fills up. We can always add more element to linked list. | Table may become full. |
| Less sensitive to load factors. | Sensitive to load factors requires expansion when LF>1 |
| Mostly used when count of keys inserted or deleted are unknown. | Mostly used when frequency of keys is known. |
| Cache performance isn't good as linked list is used for storing chain. | Provides better cache performance as everything is stored in the same table. |
| Wastage of space.(Some parts of hash table are never used.) | A slot can be used even if an input doesn't map to it. (During Probing) |
| Extra space for links. | No links |