**Searching**

Searching is a process of finding a value in a list of values. In other words, searching is the process of locating given value position in a list of values.

**Types of searching techniques:**

1. **Linear Search Algorithm (Sequential Search Algorithm)**

Linear search algorithm finds a given element in a list of elements with **O(n)** time complexity where **n** is total number of elements in the list. This search process starts comparing search element with the first element in the list. If both are matched then result is element found otherwise search element is compared with next element in the list. Repeat the same until search element is compared with last element in the list, if that last element also doesn't match, then the result is "Element not found in the list". That means, the search element is compared with element by element in the list.

**Example of linear search**



# Binary Search Algorithm

Binary search algorithm finds a given element in a list of elements with **O(log n)** time complexity where **n** is total number of elements in the list. The binary search algorithm can be used with only sorted list of elements. That means, binary search is used only with list of elements that are already arraged in an order. The binary search can not be used for list of elements arranged in random order. This search process starts comparing the search element with the middle element in the list. If both are matched, then the result is "element found". Otherwise, we check whether the search element is smaller or larger than the middle element in the list. If the search element is smaller, then we repeat the same process for left sublist of the middle element. If the search element is larger, then we repeat the same process for right sublist of the middle element. We repeat this process until we find the search element in the list or until we left with a sublist of only one element. And if that element also doesn't match with the search element, then the result is "Element not found in the list".

**Example of binary search**

# Binary Search Algorithm

1. **Hashing**

Hashing is an another approach in which time required to search an element doesn't depends on the total number of elements. Using hashing data structure, a given element is searched with **constant time complexity**. Hashing is an effective way to reduce the number of comparisions to seach an element in a data structure.  
  
**Hashing is defined as :**

**Hashing is the process of indexing and retrieving element (data) in a data structure to provide faster way of finding the element using hash key.**

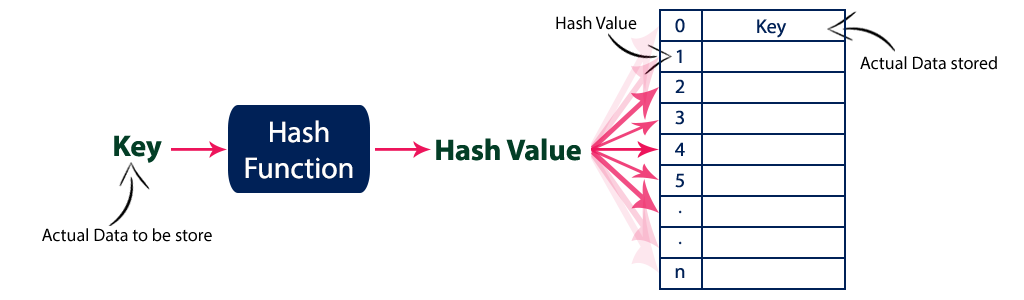
Here, hash key is a value which provides the index value where the actual data is likely to be stored in the data structure.  
  
In this data structure, we use a concept called **Hash table** to store data. All the data values are inserted into the hash table based on the hash key value. Hash key value is used to map the data with index in the hash table. And the hash key is generated for every data using a **hash function**. That means every entry in the hash table is based on the hash key value generated using hash function.  
  
**Hash Table is defined as**

**Hash table is just an array which maps a key (data) into the data structure with the help of hash function such that insertion, deletion and search operations are performed with constant time complexity (i.e. O(1)).**

Hash tables are used to perform insertion, deletion and search operations very quickly in a datastructure. Using hash table concept, insertion, deletion and search operations are accomplished in constant time complexity. Generally, every hash table makes use of a function called **hash function** to map the data into the hash table.  
  
**A hash function is defined as** :

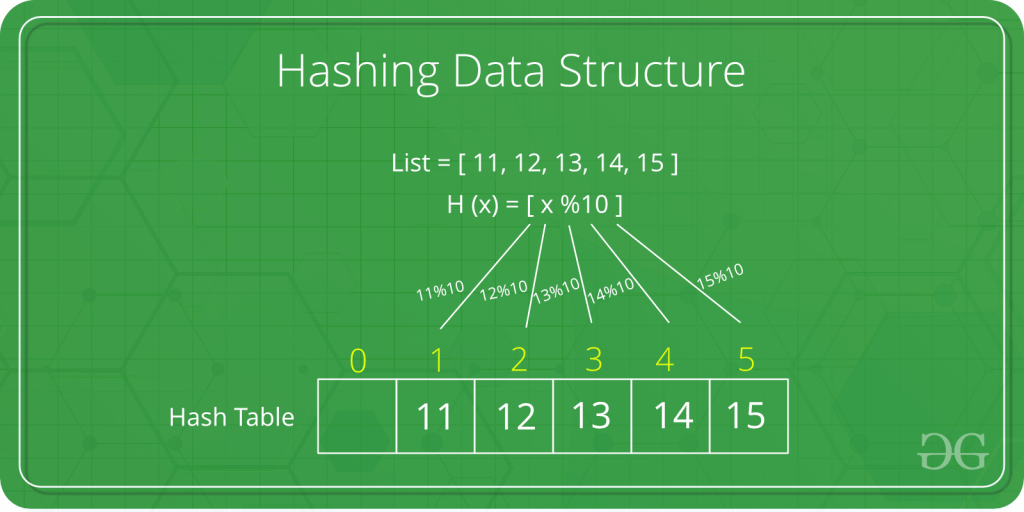
**Hash function is a function which takes a piece of data (i.e. key) as input and produces an integer (i.e. hash value) as output which maps the data to a particular index in the hash table.**

**Basic concept of hashing and hash table is shown in this diagram**



**Example**

Assume a hash function H(x) maps the value  at the index **x%10** in an Array. For example if the list of values is [11,12,13,14,15] it will be stored at positions {1,2,3,4,5} in the array or Hash table respectively.



* **Collision in Hashing-**

**In hashing-**

* Hash function is used to compute the hash value for a key.
* Hash value is then used as an index to store the key in the hash table.
* Hash function may return the same hash value for two or more keys.

**When the hash value of a key maps to an already occupied bucket of the hash table, it is called as a Collision.**

**Example of collision in hashing –**

Suppose we have elements 10,20,30,40 and we will insert these elements in a hash table using hash function h(x) %10. Where x is the element. So,

10%10 is equal to 0, and we place element 10 at index 0

Similarly 20% 10 is 0, and we place element 10 at index 0

30%10 and 40 %10 all are 0, so we have to place all the elements at index 0. This is called collision in hashing.

**0**

10,20,30,40

**1**

**2**

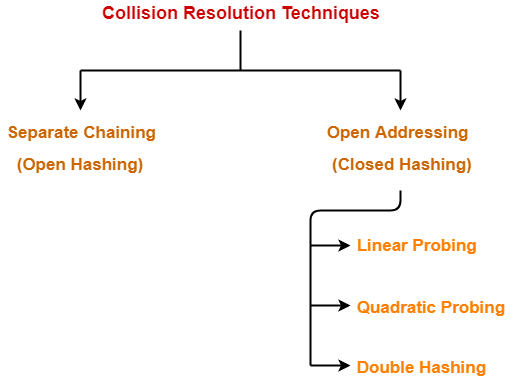
**3**

**4**

**5**

**Collision Resolution Techniques-**

Collision resolution techniques are classified as-



**Separate Chaining-**

To handle the collision,

* This technique creates a linked list to the slot for which collision occurs.
* The new key is then inserted in the linked list.
* These linked lists to the slots appear like chains.
* That is why, this technique is called as **separate chaining**.

**Open Addressing-**

In open addressing,

* Unlike separate chaining, all the keys are stored inside the hash table.
* No key is stored outside the hash table.

**Techniques used for open addressing are-**

* Linear Probing
* Quadratic Probing
* Double Hashing

**1. Linear Probing-**

In linear probing,

* When collision occurs, we linearly probe for the next bucket.
* We keep probing until an empty bucket is found.

**2. Quadratic Probing-**

In quadratic probing,

* When collision occurs, we probe for i2‘th bucket in ith iteration.
* We keep probing until an empty bucket is found.

**3. Double Hashing-**

In double hashing,

* We use another hash function hash2(x) and look for i \* hash2(x) bucket in ith iteration.
* It requires more computation time as two hash functions need to be computed.

**Note : For linear search and binary search programs refer txt files.**