Hashing

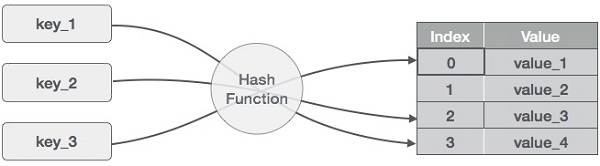
Hash Table is a data structure which stores data in an associative manner. In a hash table, data is stored in an array format, where each data value has its own unique index value. Access of data becomes very fast if we know the index of the desired data.

* It is a data structure in which insertion and search operations are very fast irrespective of the size of the data.
* Hash Table uses an array as a storage medium and uses hash technique to generate an index where an element is to be inserted or is to be located from.

**Hashing**

Hashing is a technique to convert a range of key values into a range of indexes of an array. We're going to use modulo operator to get a range of key values.

Consider an example of hash table of size 20, and the following items are to be stored. Item are in the (key,value) format.



* (1,20)
* (2,70)
* (42,80)
* (4,25)
* (12,44)
* (14,32)
* (17,11)
* (13,78)
* (37,98)

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr.No.** | **Key** | **Hash** | **Array Index** |
| 1 | 1 | 1 % 20 = 1 | 1 |
| 2 | 2 | 2 % 20 = 2 | 2 |
| 3 | 42 | 42 % 20 = 2 | 2 |
| 4 | 4 | 4 % 20 = 4 | 4 |
| 5 | 12 | 12 % 20 = 12 | 12 |
| 6 | 14 | 14 % 20 = 14 | 14 |
| 7 | 17 | 17 % 20 = 17 | 17 |
| 8 | 13 | 13 % 20 = 13 | 13 |
| 9 | 37 | 37 % 20 = 17 | 17 |

**Linear Probing**

As we can see, it may happen that the hashing technique is used to create an already used index of the array. In such a case, we can search the next empty location in the array by looking into the next cell until we find an empty cell. This technique is called linear probing.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr.No.** | **Key** | **Hash** | **Array Index** | **After Linear Probing, Array Index** |
| 1 | 1 | 1 % 20 = 1 | 1 | 1 |
| 2 | 2 | 2 % 20 = 2 | 2 | 2 |
| 3 | 42 | 42 % 20 = 2 | 2 | 3 |
| 4 | 4 | 4 % 20 = 4 | 4 | 4 |
| 5 | 12 | 12 % 20 = 12 | 12 | 12 |
| 6 | 14 | 14 % 20 = 14 | 14 | 14 |
| 7 | 17 | 17 % 20 = 17 | 17 | 17 |
| 8 | 13 | 13 % 20 = 13 | 13 | 13 |
| 9 | 37 | 37 % 20 = 17 | 17 | 18 |

**Basic Operations**

Following are the basic primary operations of a hash table.

* **Search** − Searches an element in a hash table.
* **Insert** − inserts an element in a hash table.
* **delete** − Deletes an element from a hash table.
* Hashing is the process of generating a value from a text or a list of numbers using a mathematical function known as a hash function.There are many hash functions that use numeric numeric or alphanumeric keys. Different hash functions are given below:

## Hash Functions

### Hashing Methods

These are the hashing methods:

1. Direct method
2. Modulo-division
3. Midsquare
4. Digit extraction
5. Rotation
6. Folding

**Direct Method**

In direct hashing the key is the address without any algorithmic manipulation.  
Direct hashing is limited, but it can be very powerful because it guarantees that there are no synonyms and therefore no collision.

**Modulo-division Method**

* This is also known as division remainder method.
* This algorithm works with any list size, but a list size that is a prime number produces fewer collisions than other list sizes.
* The formula to calculate the address is:  
  Address = key MODULO listsize + 1  
  Where listsize is the number of elements in the arrary.

Example:  
Given data  
Keys are : 137456 214562 140145  
137456 % 19 +1 = 11  
214562 % 19 + 1 = 15  
140145 % 19 + 1 = 2

**Digit-extraction Method**

* Using digit extraction selected digits are extracted from the key and used as the address.

Example:

* Using six-digit employee number to hash to a three digit address (000-999), we could select the first, third, and fourth digits( from the left) and use them as the address.

The keys are:  
379452 -> 394  
121267 -> 112  
378845 -> 388

**Folding Method**  
Two folding methods are used they are:

1. Fold shift
2. Fold boundary

**1. Fold Shift**

* In fold shift the key value is divided into [parts](http://blog4preps.blogspot.com/2011/07/hashing-methods.html" \t "_top) whose size matches the size of the required address. Then the left and right parts are shifted and added with the middle part.

Key=123 456 789 (9 digit ssc no) 456+123+789=1368(1 is discarded since the resulting sum is greater than 999 the leading digit)

**2. Fold boundary**

* In fold boundary the left and right numbers are folded on a fixed boundary between them. The two outside values are thus reversed.

Key=123456789

456+321+987=1764

After discarding 1 the address is 764.

**Rotation Method**

* Rotation method is generally not used by itself but rather is incorporated in combination with other hashing methods.
* It is most useful when keys are assigned serially

The following are some of the Hash Functions:

### Division Method

This is the easiest method to create a hash function. The hash function can be described as:

h(k) = k mod n

Here, h(k) is the hash value obtained by dividing the key value k by size of hash table n using the remainder. It is best that n is a prime number as that makes sure the keys are distributed with more uniformity.

An example of the Division Method is as follows:

k=1276

n=10

h(1276) = 1276 mod 10

= 6

The hash value obtained is 6

A disadvantage of the division method id that consecutive keys map to consecutive hash values in the hash table. This leads to a poor performance.

### Multiplication Method

The hash function used for the multiplication method is :

h(k) = floor( n( kA mod 1 ) )

Here, k is the key and A can be any constant value between 0 and 1. Both k and A are multiplied and their fractional part is separated. This is then multiplied with n to get the hash value.

An example of the Multiplication Method is as follows:

k=123

n=100

A=0.618033

h(123) = 100 (123 \* 0.618033 mod 1)

= 100 (76.018059 mod 1)

= 100 (0.018059)

= 1

The hash value obtained is 1

An advantage of the multiplication method is that it can work with any value of A, although some values are believed to be better than others.

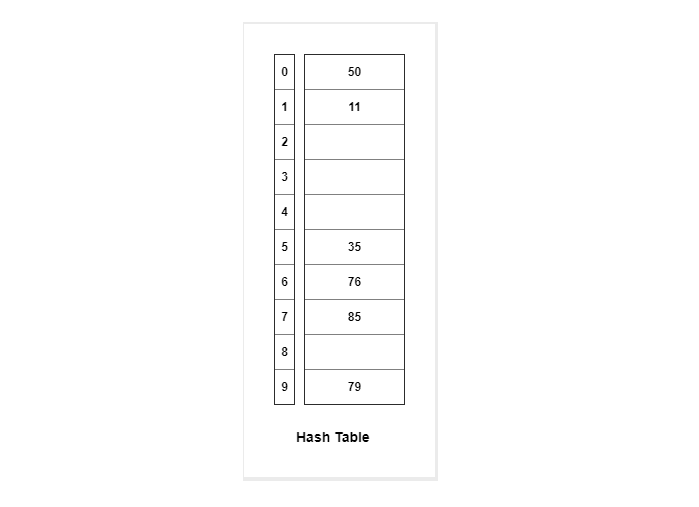
### Mid Square Method

* The mid square method is a very good hash function. It involves squaring the value of the key and then extracting the middle r digits as the hash value. The value of r can be decided according to the size of the hash table.
* An example of the Mid Square Method is as follows:
* Suppose the hash table has 100 memory locations. So **r=2** because two digits are required to map the key to memory location.
* k = 50
* k\*k = 2500
* h(50) = 50
* The hash value obtained is 50

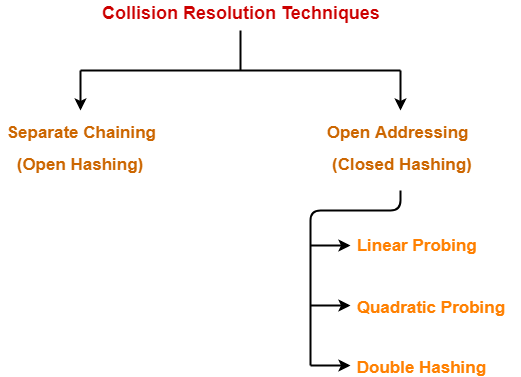
Digit extraction method:

## Hash Tables

* A hash table is a data structure that maps keys to values. It uses a hash function to calculate the index for the data key and the key is stored in the index.
* An example of a hash table is as follows:
* The key sequence that needs to be stored in the hash table is:
* 35 50 11 79 76 85
* The hash function h(k) used is:
* h(k) = k mod 10
* Using linear probing, the values are stored in the hash table as:



**Collision Resolution Techniques-**



1. Separate Chaining
2. Open Addressing

**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**.

### ****For Searching-****

* In worst case, all the keys might map to the same bucket of the hash table.
* In such a case, all the keys will be present in a single linked list.
* Sequential search will have to be performed on the linked list to perform the search.
* So, time taken for searching in worst case is O(n).

### ****For Deletion-****

* In worst case, the key might have to be searched first and then deleted.
* In worst case, time taken for searching is O(n).
* So, time taken for deletion in worst case is O(n).



If Load factor (α) = constant, then time complexity of Insert, Search, Delete = Θ(1)

**Problem-**

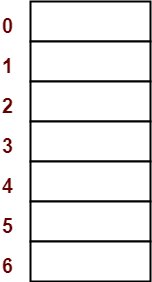
Using the hash function ‘key mod 7’, insert the following sequence of keys in the hash table-

50, 700, 76, 85, 92, 73 and 101

Use separate chaining technique for collision resolution.

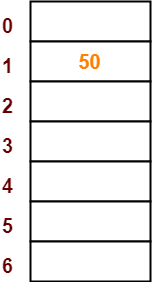
### ****Step-01:****

* Draw an empty hash table.
* For the given hash function, the possible range of hash values is [0, 6].
* So, draw an empty hash table consisting of 7 buckets as-



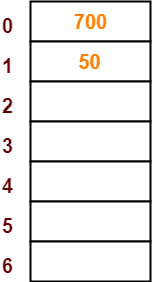
### ****Step-02:****

* Insert the given keys in the hash table one by one.
* The first key to be inserted in the hash table = 50.
* Bucket of the hash table to which key 50 maps = 50 mod 7 = 1.
* So, key 50 will be inserted in bucket-1 of the hash table as-



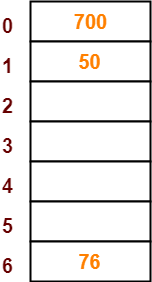
### ****Step-03:****

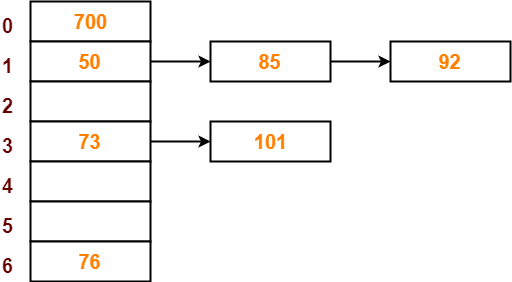
* The next key to be inserted in the hash table = 700.
* Bucket of the hash table to which key 700 maps = 700 mod 7 = 0.
* So, key 700 will be inserted in bucket-0 of the hash table as-



### ****Step-04:****

* The next key to be inserted in the hash table = 76.
* Bucket of the hash table to which key 76 maps = 76 mod 7 = 6.
* So, key 76 will be inserted in bucket-6 of the hash table as-





**Load Factor (α)-**

Load factor (α) is defined as



If Load factor (α) = constant, then time complexity of Insert, Search, Delete = Θ(1)

**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

Let us discuss how operations are performed in open addressing-

### ****Insert Operation-****

* Hash function is used to compute the hash value for a key to be inserted.
* Hash value is then used as an index to store the key in the hash table.

In case of collision,

* Probing is performed until an empty bucket is found.
* Once an empty bucket is found, the key is inserted.
* Probing is performed in accordance with the technique used for open addressing.

### ****Search Operation-****

To search any particular key,

* Its hash value is obtained using the hash function used.
* Using the hash value, that bucket of the hash table is checked.
* If the required key is found, the key is searched.
* Otherwise, the subsequent buckets are checked until the required key or an empty bucket is found.
* The empty bucket indicates that the key is not present in the hash table.

### ****Delete Operation-****

* The key is first searched and then deleted.
* After deleting the key, that particular bucket is marked as “deleted”.

## ****NOTE-****

* During insertion, the buckets marked as “deleted” are treated like any other empty bucket.
* During searching, the search is not terminated on encountering the bucket marked as “deleted”.
* The search terminates only after the required key or an empty bucket is foun

Let us discuss how operations are performed in open addressing-

### ****Insert Operation-****

* Hash function is used to compute the hash value for a key to be inserted.
* Hash value is then used as an index to store the key in the hash table.

In case of collision,

* Probing is performed until an empty bucket is found.
* Once an empty bucket is found, the key is inserted.
* Probing is performed in accordance with the technique used for open addressing.

### ****Search Operation-****

To search any particular key,

* Its hash value is obtained using the hash function used.
* Using the hash value, that bucket of the hash table is checked.
* If the required key is found, the key is searched.
* Otherwise, the subsequent buckets are checked until the required key or an empty bucket is found.
* The empty bucket indicates that the key is not present in the hash table.

### ****Delete Operation-****

* The key is first searched and then deleted.
* After deleting the key, that particular bucket is marked as “deleted”.

## ****NOTE-****

* During insertion, the buckets marked as “deleted” are treated like any other empty bucket.
* During searching, the search is not terminated on encountering the bucket marked as “deleted”.
* The search terminates only after the required key or an empty bucket is foun

**Open Addressing Techniques-**

Techniques used for open addressing are

## ****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.

### ****Advantage-****

* It is easy to compute.

### ****Disadvantage-****

* The main problem with linear probing is clustering.
* Many consecutive elements form groups.
* Then, it takes time to search an element or to find an empty bucket.

### ****Time Complexity-****

|  |
| --- |
| Worst time to search an element in linear probing is O (table size). |

This is because-

* Even if there is only one element present and all other elements are deleted.
* Then, “deleted” markers present in the hash table makes search the entire table.

## ****2. Quadratic Probing-****

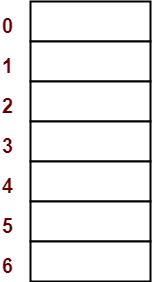
In quadratic probing,

* When collision occurs, we probe for i2‘th bucket in ith iteration.
* We keep probing until an empty bucket is found.
* **PRACTICE PROBLEM BASED ON OPEN ADDRESSING-**
* **Problem-**
* Using the hash function ‘key mod 7’, insert the following sequence of keys in the hash table-
* 50, 700, 76, 85, 92, 73 and 101
* Use linear probing technique for collision resolution.
* **Solution-**

The given sequence of keys will be inserted in the hash table as-

### ****Step-01:****

* Draw an empty hash table.
* For the given hash function, the possible range of hash values is [0, 6].
* So, draw an empty hash table consisting of 7 buckets as-

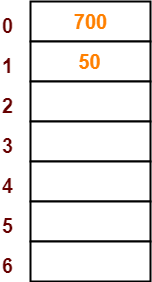


### ****Step-02:****

* Insert the given keys in the hash table one by one.
* The first key to be inserted in the hash table = 50.
* Bucket of the hash table to which key 50 maps = 50 mod 7 = 1.
* So, key 50 will be inserted in bucket-1 of the hash table as-

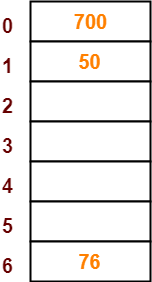
### ****Step-03:****

* The next key to be inserted in the hash table = 700.
* Bucket of the hash table to which key 700 maps = 700 mod 7 = 0.
* So, key 700 will be inserted in bucket-0 of the hash table as-



### ****Step-04:****

* The next key to be inserted in the hash table = 76.
* Bucket of the hash table to which key 76 maps = 76 mod 7 = 6.
* So, key 76 will be inserted in bucket-6 of the hash table as-

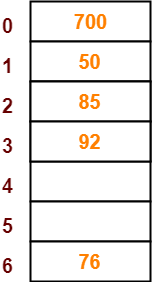


### ****Step-05:****

* The next key to be inserted in the hash table = 85.
* Bucket of the hash table to which key 85 maps = 85 mod 7 = 1.
* Since bucket-1 is already occupied, so collision occurs.
* To handle the collision, linear probing technique keeps probing linearly until an empty bucket is found.
* The first empty bucket is bucket-2.
* So, key 85 will be inserted in bucket-2 of the hash table as-

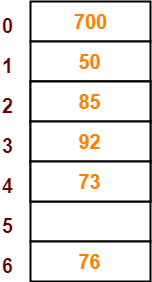
### ****Step-06:****

* The next key to be inserted in the hash table = 92.
* Bucket of the hash table to which key 92 maps = 92 mod 7 = 1.
* Since bucket-1 is already occupied, so collision occurs.
* To handle the collision, linear probing technique keeps probing linearly until an empty bucket is found.
* The first empty bucket is bucket-3.
* So, key 92 will be inserted in bucket-3 of the hash table as-



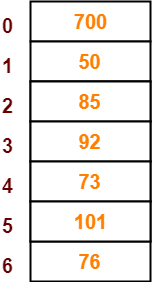
### ****Step-07:****

* The next key to be inserted in the hash table = 73.
* Bucket of the hash table to which key 73 maps = 73 mod 7 = 3.
* Since bucket-3 is already occupied, so collision occurs.
* To handle the collision, linear probing technique keeps probing linearly until an empty bucket is found.
* The first empty bucket is bucket-4.
* So, key 73 will be inserted in bucket-4 of the hash table as-



### ****Step-08:****

* The next key to be inserted in the hash table = 101.
* Bucket of the hash table to which key 101 maps = 101 mod 7 = 3.
* Since bucket-3 is already occupied, so collision occurs.
* To handle the collision, linear probing technique keeps probing linearly until an empty bucket is found.
* The first empty bucket is bucket-5.
* So, key 101 will be inserted in bucket-5 of the hash table as-



### Pseudorandom Method

In the **pseudorandom method,** the key is used as the seed in a pseudorandom number generator and the resulting random number is then scaled into the possible address range using modulo division. A common random number generator is shown below.

y = ax + c

This method is also known as **MAD** which stands for **multiply, add and divide**. To use the pseudorandom number generator as a hashing method, we set x to the key, multiply it by the coefficient a, and then add the constant c. The result is then divided by the list size with the remainder (see “Modulo-Division Method,”) being the hashed address. Let’s demonstrate the concept with an example from Figure 6. To keep the calculation reasonable, we use 17 and 7 for factors *a* and c, respectively. Also, the list size in the example is the prime number 30

y = ((17 \* 121267) + 7)modulo 307

y = (2061539 + 7) modulo 307

y = 2061546 modulo 307 y = 4