Hashing

Concept of hashing / Basic mechanism of hashing:

In data structure, the process of mapping data elements or records (known as **hash key**) to appropriate indices or **buckets** of an array (known as **hash table**) using a function (known as **hash function**) is called hashing. Using hashing data structure, a given element is searched with **constant time complexity i.e. O(1).** Hashing is an effective way to reduce the number of comparisons to search or insert an element in the hash table.

Advantages:

Unlike other searching techniques,

- Hashing is extremely efficient in terms of number of comparisons. It tends to minimize number of comparisons as much as possible while searching.
- The time taken by it to perform the search does not depend upon the total number of elements.
- In ideal case, it completes the search with constant time complexity O(1) which is better than other searching techniques like linear search and binary search.

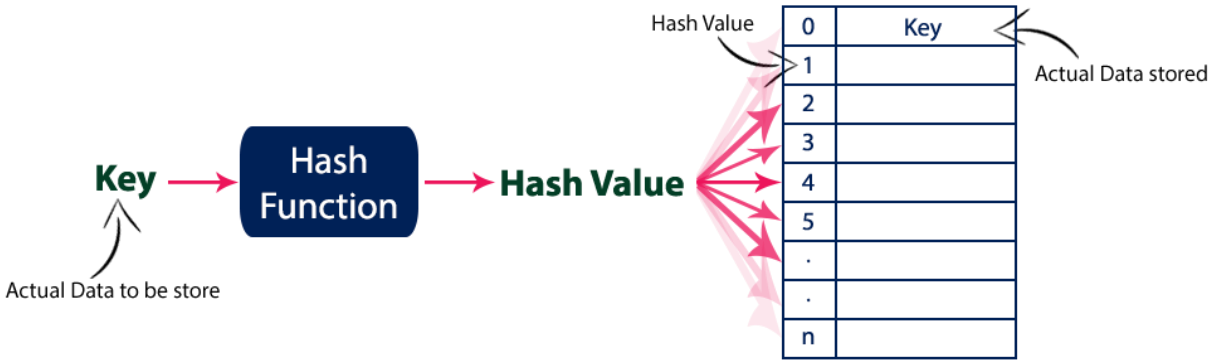
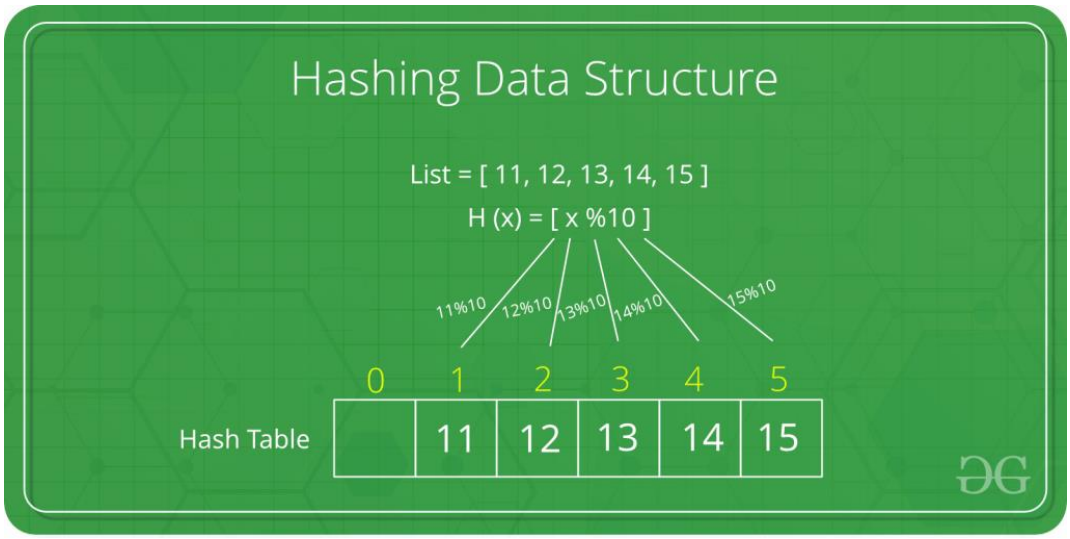


Figure: Concept of hashing

Abhishek Dey
Assistant Professor
Department of Computer Science
Bethune College, Kolkata

Example of ideal Hashing:



Here, List is the set of hash keys which are mapped by hash function H to indices of a hash table of size 6. In this example, H is taken as, H(x) = x % 10. This is a case of ideal hashing where no collision occurs i.e. more than one key value is not mapped into same index.

Properties of a good hash function:

To achieve a good hashing mechanism, it is important to have a good hash function with the following basic requirements:

(1) It should be easy to compute and must not become a complex algorithm in itself.

(2) It should provide a uniform distribution across the hash table and should not result in clustering of key values.

(3) A good hash function should generate addresses or indices with minimum number of collisions.

(4) Range of generated indices by hash functions should be within maximum index of hash table.

(5) Hash function should use hash key values to generate addresses.

Abhishek Dey
Assistant Professor
Department of Computer Science
Bethune College, Kolkata

Types of Hash Functions:

(1) Division / Modulo-Division:

H(k) = k mod m

Generally, m should be the size of hash table and also m should be a prime number.

(2) Midsquare:

H(k) = x, where x is obtained by selecting some digits from middle of k^2 (k is the key)

(3) Folding:

Partition the key k into a number of parts $k_1, k_2, \ldots, k_n$, where each part, except possibly the last, has the same number of bits or digits as the required address width. Then the parts are added together, ignoring the last carry, if any. Alternatively,

$$H(k) = k_1 + k_2 + \cdots + k_n$$

where the last carry, if any, is ignored. If the keys are in binary form, the exclusive-OR operation may be substituted for addition. There are many variations known in this method. One is called the *fold shifting method,* where the even number parts, $k2$, $k4$, ... are each reversed before the addition. Another variation is called the *fold boundary method.* Here, two boundary parts, namely, k_1 and k_n, each are reversed and then added to all other parts. As an example, let us take the size of each part to be 2; the following calculations are performed on the given key values (integers) as shown below:

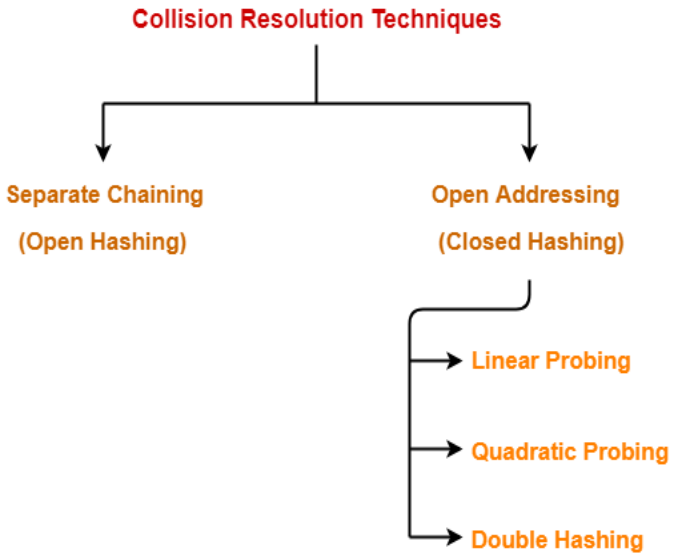
k:	1522756	5499025	11943936
Chopping:	01 52 27 56	05 49 90 25	11 94 39 36
Pure folding:	01 + 52 + 27 + 56 = 136	05 + 49 + 90 + 25 = 169	11 + 94 + 39 + 36 = 180
Fold shifting:	10 + 52 + 72 + 56 = 190	50 + 49 + 09 + 25 = 133	11 + 94 + 93 + 36 = 234
Fold boundary:	10 + 52 + 27 + 65 = 154	50 + 49 + 90 + 52 = 241	11 + 94 + 39 + 63 = 207

Collision Resolution Techniques:

When a hash function maps the hash key to an already occupied bucket of the hash table, it is called as a Collision. Irrespective of how good a hash function is, collisions are bound to occur. Therefore, to maintain the optimal performance of hashing, it is important to manage or handle collisions in some efficient way. Collision Resolution Techniques are the techniques used for resolving or handling the collision.

Abhishek Dey
Assistant Professor
Department of Computer Science
Bethune College, Kolkata

Collision resolution techniques are classified as-



Open Addressing(Closed Hashing):

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

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.

Abhishek Dey
Assistant Professor
Department of Computer Science
Bethune College, Kolkata

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.

Open Addressing Techniques-

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 **primary clustering**. In primary clustering, many consecutive elements form groups or clusters. Then, it takes a considerable amount of time to search an element or to find an empty bucket. In worst case, time to search an element in linear probing is O(n) [n is the number of buckets in the hash table].

2.Quadratic Probing- In quadratic probing, when collision occurs, we probe for i^2 th bucket in ith iteration. We keep probing until an empty bucket is found.

Advantage- It avoids primary clustering.

Disadvantage- (a) **secondary clustering**, (b) inability to access all positions in the hash table.

3.Double Hashing- In double hashing, we use another hash function H2(x) and look for i*H2(x) bucket in ith iteration.

Advantage- Avoids primary and secondary clustering.

Disadvantage- It requires more computation time as two hash functions need to be computed.

Load factor (α) is defined as:

$$\text{Load Factor } (\alpha) = \frac{\text{Number of elements present in the hash table}}{\text{Total size of the hash table}}$$

In open addressing, the value of load factor always lies between 0 and 1. This is because, in open addressing, all the keys are stored inside the hash table. So, size of the table is always greater or at least equal to the number of keys stored in the table.

Abhishek Dey
Assistant Professor
Department of Computer Science
Bethune College, Kolkata

Separate Chaining(Open Hashing):

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**.

Advantage- Overflow situation never arises. Hash table maintains linked lists which can contain any number of key values. Collision resolution can be achieved very efficiently. Insertion, searching, deletion operations are quick and easy. Open hashing is best suitable in applications where number of key values vary drastically as it uses dynamic storage management policy.

Disadvantage- Space complexity is more than closed hashing; extra storage space is needed for maintaining link fields (pointers). In worst case, all the keys (say, n number of 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).

Load factor for separate chaining can be greater than 1.

Reference books and website links:

1. Data Structures Through C In Depth by S.K.Srivastava and Deepali Srivastava
2. Data Structures Using C by Reema Thareja
3. https://www.hackerearth.com/practice/data-structures/hash-tables/basics-of-hash-tables/tutorial/
4. https://www.gatevidyalay.com/hashing/

Abhishek Dey
Assistant Professor
Department of Computer Science
Bethune College, Kolkata