Hashing

Tushar B. Kute,
http://tusharkute.com

Hashing

- Hashing is the process of transforming any given key or a string of characters into another value.
- This is usually represented by a shorter, fixed-length value or key that represents and makes it easier to find or employ the original string.

Hashing

- The most popular use for hashing is the implementation of hash tables.
- A hash table stores key and value pairs in a list that is accessible through its index.
- Because key and value pairs are unlimited, the hash function will map the keys to the table size.
- A hash value then becomes the index for a specific element.

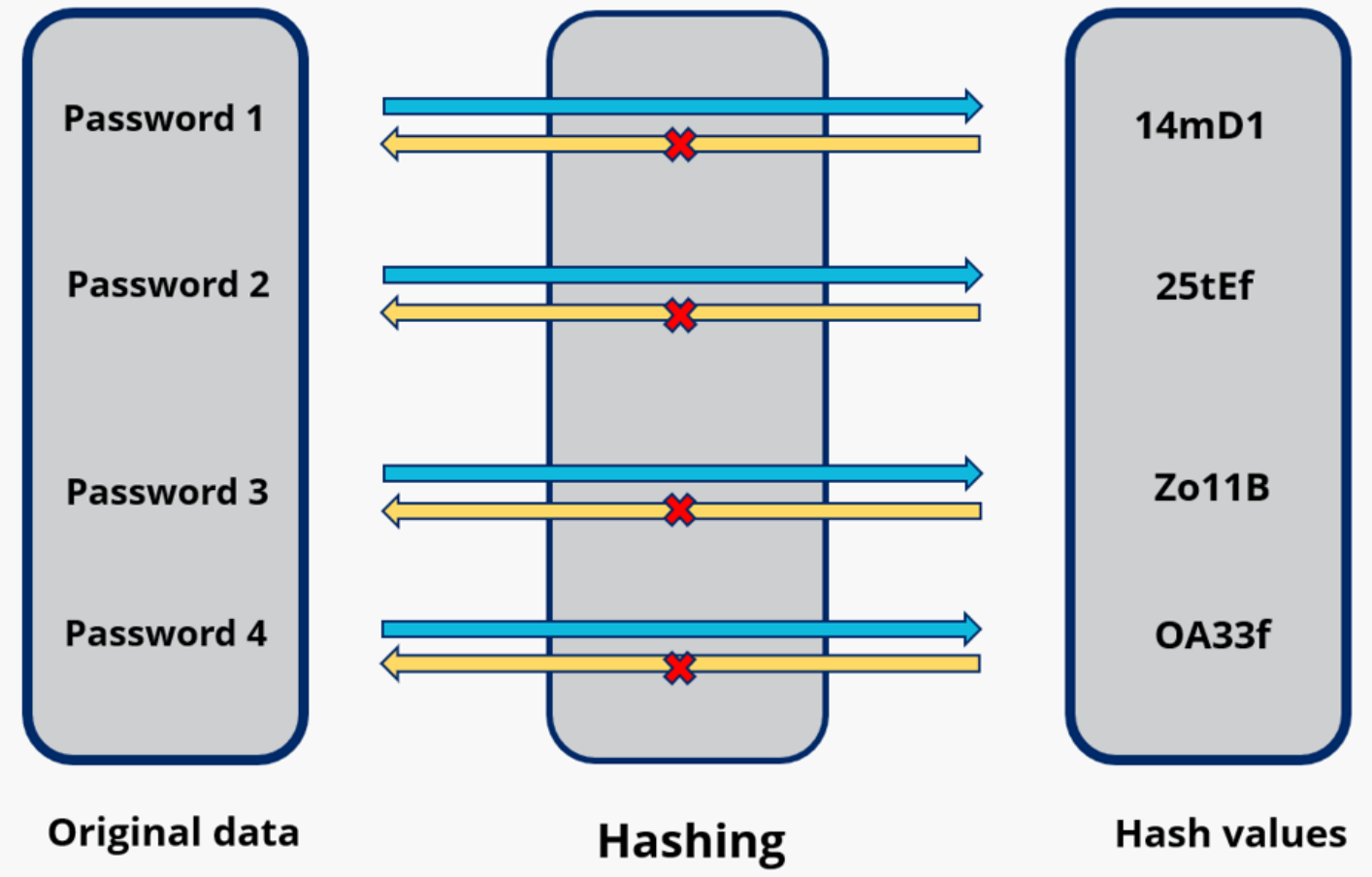
Hashing

- A hash function generates new values according to a mathematical hashing algorithm, known as a hash value or simply a hash.
- To prevent the conversion of hash back into the original key, a good hash always uses a one-way hashing algorithm.
- Hashing is relevant to -- but not limited to -- data indexing and retrieval, digital signatures, cybersecurity and cryptography.

Hashing

- Hashing uses functions or algorithms to map object data to a representative integer value.
- A hash can then be used to narrow down searches when locating these items on that object data map.
- For example, in hash tables, developers store data -- perhaps a customer record -- in the form of key and value pairs.
- The key helps identify the data and operates as an input to the hashing function, while the hash code or the integer is then mapped to a fixed size.

Hashing


Password 1
Password 2
Password 3
Password 4
14mD1
25tEf
Zo11B
OA33f
Original data
Hashing
Hash values

Hashing : Applications

- Databases: Hashing can be used to index data in a database, which makes it faster to search for and retrieve data.
- Cryptography: Hashing can be used to create digital signatures and other cryptographic primitives.
- Caching: Hashing can be used to implement caches, which are data structures that store frequently accessed data in memory for faster access.
- Bloom filters: Hashing can be used to implement Bloom filters, which are a space-efficient probabilistic data structure used to test whether an element is a member of a set.

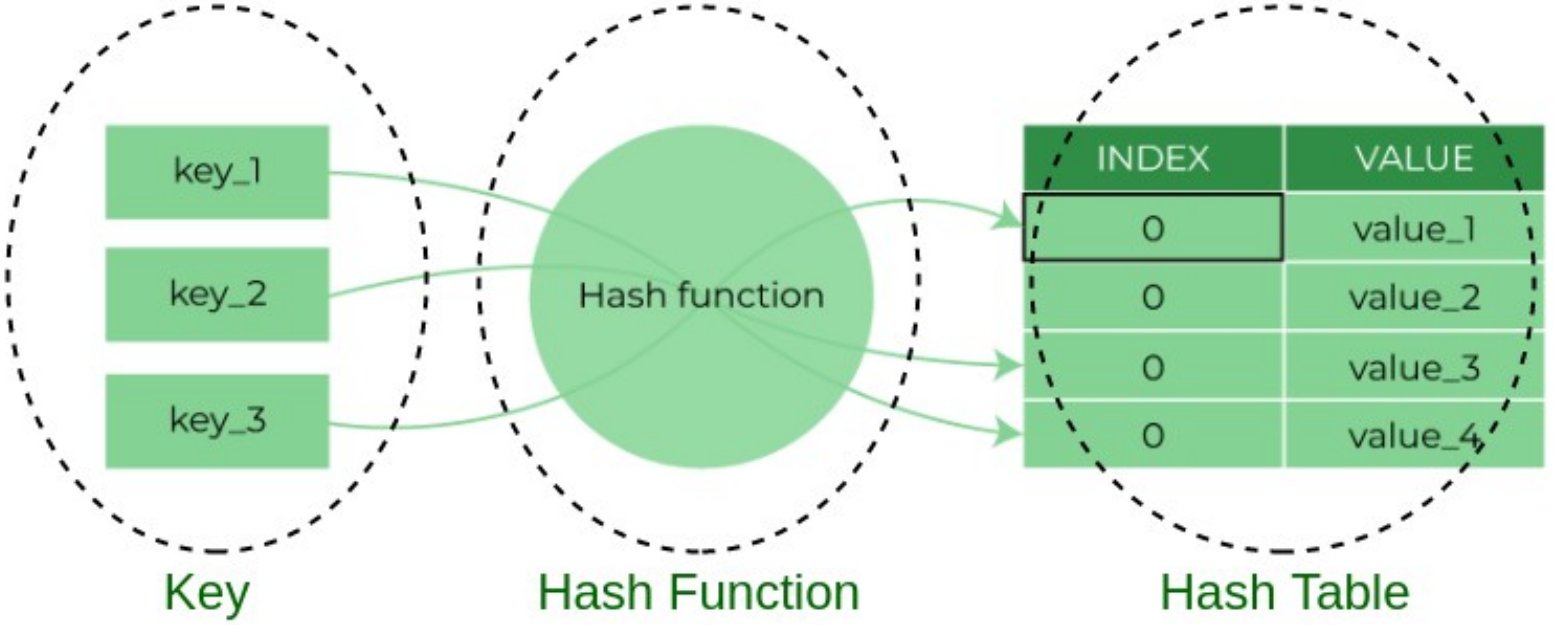
Hash Table

- A hash table is a data structure that maps keys to values. It is implemented using an array, where each element of the array is a linked list of key-value pairs.
- The keys are hashed into a fixed-size bucket index using a hash function.
- To insert a new key-value pair into the hash table, the hash function is used to generate a bucket index.
- The new key-value pair is then added to the linked list at the corresponding bucket index.

Hash Table

- To search for a value in the hash table, the hash function is used to generate a bucket index.
- The linked list at the corresponding bucket index is then searched for the key. If the key is found, the corresponding value is returned. Otherwise, null is returned.
- Hash tables are a very efficient way to store and retrieve data, especially when the keys are evenly distributed.
- However, if the keys are not evenly distributed, the hash table can become inefficient.

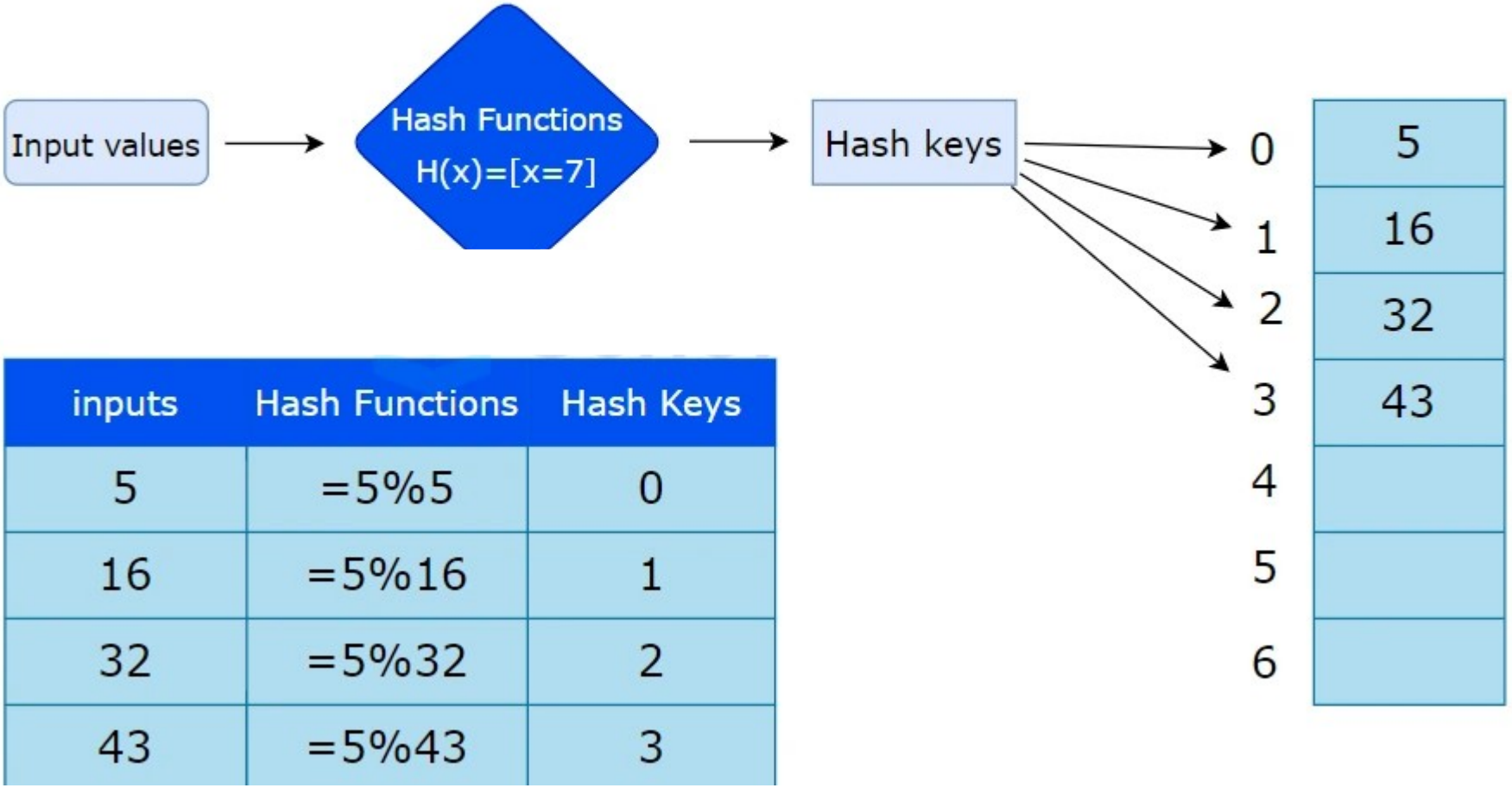
Hash Table


key_1
key_2
key_3
Hash function
INDEX
VALUE
0
value_1
0
value_2
0
value_3
0
value_4
Key
Hash Function
Hash Table

Hash Table

- Here are some examples of how hash tables are used:
  - In a database, a hash table can be used to index the data, which makes it faster to search for and retrieve data.
  - In a web browser, a hash table can be used to cache frequently accessed web pages.
  - In a compiler, a hash table can be used to store the symbols in the program.
  - In a programming language, a hash table can be used to implement a dictionary or associative array.

Hashing



inputs	Hash Functions	Hash Keys
5	=5%5	0
16	=5%16	1
32	=5%32	2
43	=5%43	3

Types of Hash Functions

- The primary types of hash functions are:
  - Division Method.
  - Mid Square Method.
  - Folding Method.
  - Multiplication Method.

Types of Hash Functions

- Division Method
  - The easiest and quickest way to create a hash value is through division. The k-value is divided by M in this hash function, and the result is used.
- Formula:

$$h(K) = k \bmod M$$

- (where k = key value and M = the size of the hash table)

Types of Hash Functions

- Advantages:
  - This method is effective for all values of M.
  - The division strategy only requires one operation, thus it is quite quick.
- Disadvantages:
  - Since the hash table maps consecutive keys to successive hash values, this could result in poor performance.
  - There are times when exercising extra caution while selecting M's value is necessary.

Types of Hash Functions

- Example:

  k = 1987

  M = 13

  h(1987) = 1987 mod 13

  h(1987) = 4

Types of Hash Functions

- Mid Square Method
- The following steps are required to calculate this hash method:
  - k*k, or square the value of k
  - Using the middle r digits, calculate the hash value.
- Formula:

  h(K) = h(k x k)

  (where k = key value)

Types of Hash Functions

- Advantages:
  - This technique works well because most or all of the digits in the key value affect the result. All of the necessary digits participate in a process that results in the middle digits of the squared result.
  - The result is not dominated by the top or bottom digits of the initial key value.
- Disadvantages:
  - The size of the key is one of the limitations of this system; if the key is large, its square will contain twice as many digits.
  - Probability of collisions occurring repeatedly.

Types of Hash Functions

- Example:

  k = 60

  Therefore,

  k = k x k

  k = 60 x 60

  k = 3600

  Thus,

  h(60) = 60

Types of Hash Functions

- Folding Method
- The process involves two steps:
  - With the exception of the last component, which may have fewer digits than the other parts, the key-value k should be divided into a predetermined number of pieces, such as k1, k2, k3,..., kn, each having the exact same amount of digits.
  - Add each element individually. The hash value is calculated without taking into account the final carry, if any.

Types of Hash Functions

- Formula:

  k = k1, k2, k3, k4, ….., kn

  s = k1+ k2 + k3 + k4 +….+ kn

  h(K)= s

  (Where, s = addition of the parts of key k)

Types of Hash Functions

- Advantages:
  - Creates a simple hash value by precisely splitting the key value into equal-sized segments.
  - Without regard to distribution in a hash table.
- Disadvantages:
  - When there are too many collisions, efficiency can occasionally suffer.

Types of Hash Functions

- Example:

  k = 12345

  k1 = 67; k2 = 89; k3 = 12

  Therefore,

  s = k1 + k2 + k3

  s = 67 + 89 + 12

  s = 168

Types of Hash Functions

- Multiplication Method
  - Determine a constant value. A, where (0, A, 1)
  - Add A to the key value and multiply.
  - Consider kA's fractional portion.
  - Multiply the outcome of the preceding step by M, the hash table's size.
- Formula:

  h(K) = floor (M (kA mod 1))

  (Where, M = size of the hash table, k = key value and A = constant value)

Types of Hash Functions

- Advantages:
  - Any number between 0 and 1 can be applied to it, however, some values seem to yield better outcomes than others.
- Disadvantages:
  - The multiplication method is often appropriate when the table size is a power of two since multiplication hashing makes it possible to quickly compute the index by key.

Types of Hash Functions

- Example:

  k = 5678

  A = 0.6829

  M = 200

  Now, calculating the new value of h(5678):

  h(5678) = floor[200(5678 x 0.6829 mod 1)]

  h(5678) = floor[200(3881.5702 mod 1)]

  h(5678) = floor[200(0.5702)]

  h(5678) = floor[114.04]

  h(5678) = 114

  So, with the updated values, h(5678) is 114.

Hash Collision

- When a hash algorithm generates the same hash value for two separate input values, this is known as a hash collision.
- However, it's crucial to note that collisions are not an issue; rather, they constitute a key component of hashing algorithms.
- Because various hashing methods used in data structures convert each input into a fixed-length code regardless of its length, collisions happen.
- The hashing algorithms will eventually yield repeating hashes since there are an infinite number of inputs and a finite number of outputs.

Types of Hashing in Data Structures

- The two main hashing methods used in a data structure are:
  - Open hashing/separate chaining/closed addressing
  - Open addressing/closedhashing

Types of Hashing in Data Structures

- Open hashing/separate chaining/closed addressing
  - A typical collision handling technique called "separate chaining" links components with the same hash using linked lists.
  - It is also known as closed addressing and employs arrays of linked lists to successfully prevent hash collisions.

Types of Hashing in Data Structures

- Closed hashing (Open addressing)
  - Instead of using linked lists, open addressing stores each entry in the array itself. The hash value is not used to locate objects.
  - In order to insert, it first verifies the array beginning from the hashed index and then searches for an empty slot using probing sequences.
  - The probe sequence, with changing gaps between subsequent probes, is the process of progressing through entries.

Types of Hashing in Data Structures

- There are three methods for dealing with collisions in closed hashing:
  - Linear Probing
  - Quadratic Probing
  - Double-Hashing

Linear Probing

- Linear probing includes inspecting the hash table sequentially from the very beginning. If the site requested is already occupied, a different one is searched. The distance between probes in linear probing is typically fixed (often set to a value of 1).
- Formula:

  index = key % hashTableSize

- Sequence

  index = (hash(n) % T)

  (hash(n) + 1) % T

  (hash(n) + 2) % T

  (hash(n) + 3) % T … and so on.

Quadratic Probing

- The distance between subsequent probes or entry slots is the only difference between linear and quadratic probing. You must begin traversing until you find an available hashed index slot for an entry record if the slot is already taken. By adding each succeeding value of any arbitrary polynomial in the original hashed index, the distance between slots is determined.
- Formula:

  index = index % hashTableSize

- Sequence:

  index = (hash(n) % T)

  (hash(n) + 1 x 1) % T

  (hash(n) + 2 x 2) % T

  (hash(n) + 3 x 3) % T … and so on

Double Probing

- The time between probes is determined by yet another hash function. Double hashing is an optimized technique for decreasing clustering. The increments for the probing sequence are computed using an extra hash function.
- Formula

  (first hash(key) + i * secondHash(key)) % size of the table
- Sequence

  index = hash(x) % S

  (hash(x) + 1*hash2(x)) % S

  (hash(x) + 2*hash2(x)) % S

  (hash(x) + 3*hash2(x)) % S … and so on

Thank you

@mitu_skillologies

@mITuSkillologies

@mitu_group

@mitu-skillologies

@MITUSkillologies

kaggle

@mituskillologies

Web Resources
https://mitu.co.in
http://tusharkute.com

@mituskillologies

contact@mitu.co.in

tushar@tusharkute.com