Algorithms → Hashing → Hash table

Theory: Hash table

🕓 11 minutes 7 / 7 problems solved

Start practicing

3514 users solved this topic. Latest completion was about 7 hours ago.

§1. Introduction

Hashing is a technique in computing that constitutes the representation of an object, called a **hash value**, by a unique sequence, called a **hash key**, in a data structure, called a **hash table**. Hashing involves the process of converting an object of arbitrary size to a key of definite size using a function known as the **hash function**.

Hashing is an important concept in programming because it solves the problem of searching for an object (and associated data) in a table in $O(1)$ time. This concept is used in encryption and file verification as well.

§2. Hash Table

A hashtable is an associative array data structure that maps keys to values. A hashtable consists of **buckets** or **slots** which are indexed by unique keys, generated using a hashing function.

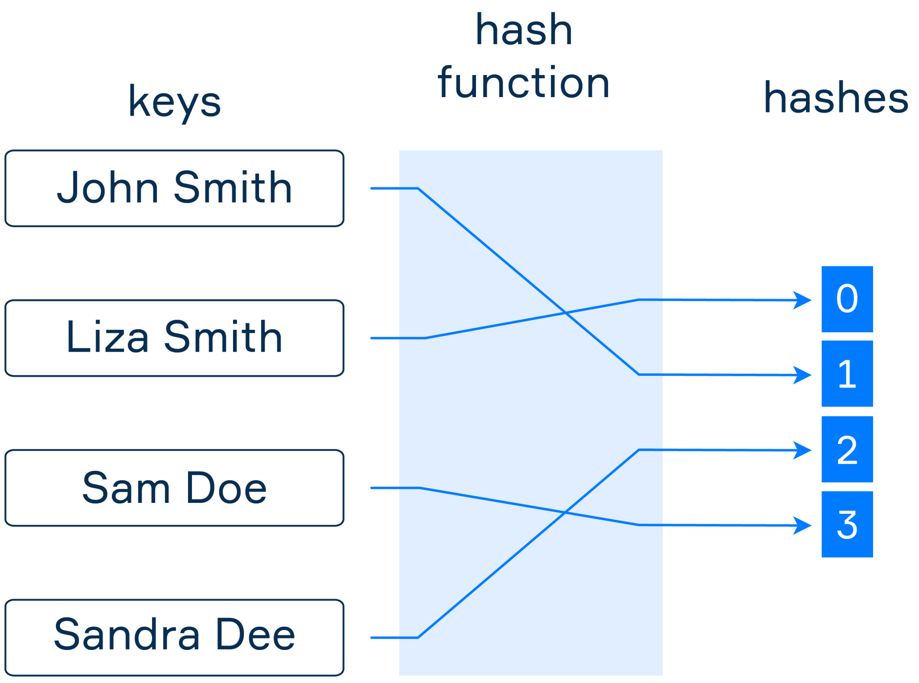
A simple example of a hash table would be a table consisting of columns for student names and unique admission codes for the associated students.

Student Admission Code	Student Name
589337	Bob Johnson
957874	Alice Anders
880384	Carol Smith
997201	Dave Jones

Here, the admission codes are the keys that are used as a reference or index to the student name value.

§3. Hashing Functions

The hashing function is used to convert an arbitrary sized input to a unique value. If we have an array that can hold M key-value pairs, then the hashing function maps any given key to a small integer in the range [0, M-1] that can be used as an index in the hash table. The table below represents how a hashing function can be used to index name keys by numerical hashes.



Current topic:

✓ Hash table ···

Topic depends on:

✓ Data structures ···

✓ Hashing: overview ···

Topic is required for:

Map ···

Hash table in Java ···

hashCode() and equals() ···

Collisions$^{\beta}$ ···

✓ Hashable ···

Table of contents:

Source link

A good hashing function is:

- **Valid:** computes a key for all input sizes in $O(1)$ time;
- **Deterministic:** gives unique key output for any input every time it is computed;
- **Uniform:** makes sure the keys are uniformly distributed.

§4. Hashing Function Example

A simple method for hashing integers is called modular hashing or remainder hashing. For our hashing function, we choose a prime number M and for any positive integer key k, compute the remainder when dividing k by M. This function is easy to compute and is effective in dispersing the keys evenly between 0 and M-1. The following table gives an example hash table using modulo 11 as the hashing function.

Item Number	Hash Value
54	10
15	4
739	2
81	4

Modular hashing works for strings as well too. We simply treat them as a long sequence of integers by converting each character to its ASCII equivalent.

There are other examples of hashing functions for strings. Imagine that we develop a hashing function that concatenates the ASCII values of the string characters to create the hash value.

For example, for the input of "abc", the hash value would be 979899.

Report a typo

266 users liked this theory. 29 didn't like it. **What about you?**

😍 🙂 😐 🙁 😡

Start practicing

Comments (18) Hints (0) Useful links (1) Show discussion