Hashing: overview

Theory | Practice | 100% completed, 0 problems solved

Theory

6 minutes reading

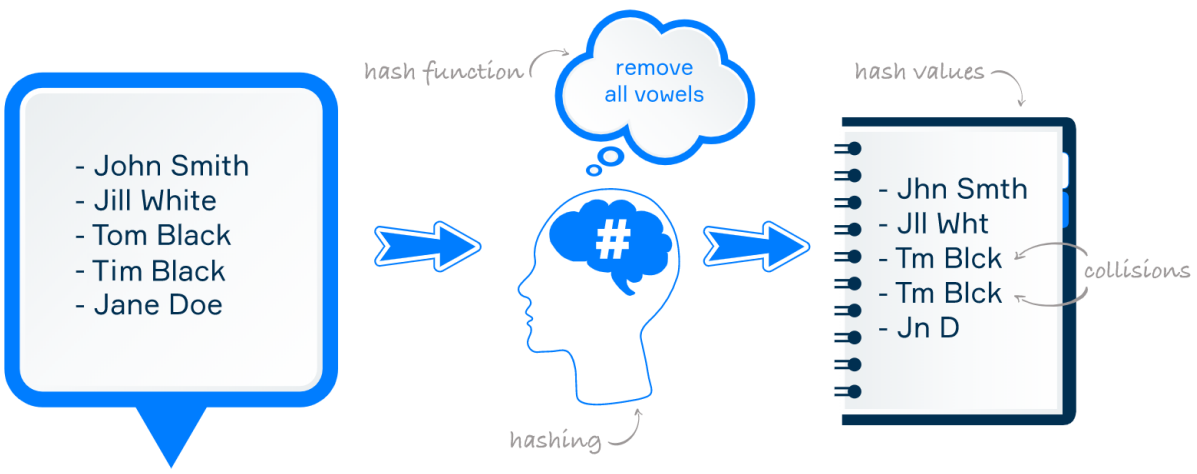
Unskip this topic | Start practicing

Hashing is a technique widely used in programming. Whether you send a message over the internet, log in to a website, or search for a file on your computer, you are using **hash functions**! But what are they and what do they do?

§1. What is hashing?

Formally, hash functions are functions that we can use to map data of arbitrary size to fixed-size values. That's quite vague, so let's look at a real-life comparison to understand them better:

Imagine you have a friend Paul who keeps a phone book. Your friend is lazy and doesn't want to spend a lot of time writing the full names of his contacts. So, instead, Paul writes only the consonants in their names. For example, instead of *John Smith,* Paul writes *Jhn Smth.* In quite a few ways, this process is similar to **hashing**. We can consider that "remove all vowels" is the **hash function**. We call the result of applying a hash function to some input a **hash value**, or simply its hash. In our case, *Jhn Smth* is the hash value of *John Smith*. Now, not all hash values will be distinct: think about two people with names *Tim Black* and *Tom Black*. The hash value for both of those names will be *Tm Blck.* When this happens, we call it a **collision**.



The main difference between our example and a real hash function is that, in our case, the hash values do not have a fixed size. A more accurate example would be if Paul wrote the first 5 letters in their names. The hash value for *John Smith* would be *John S*, for *Tim Black* it would be *Tim Bl*, and for *Tim Blacksmith,* it would be *Tim Bl* too. There are more names than possible ways to write 5 letters, so we are guaranteed to have collisions. The same thing is also true for hash functions. They take input that can be really big and return something of a fixed size, so there is no way to completely avoid collisions!

§2. Applications of hash functions

As we mentioned in the beginning, hash functions have many applications. Let's look at a few of the most important ones:

- Message digests

1 required topic

Computer algorithms

4 dependent topics

Hash function

Index() and in under the hood

HashMap

String hashing

Say you have a message that you want to send to a friend over the internet but are afraid that someone might change its contents before it reaches your friend. One of the things you can do is to compute a hash value for your message before sending it. When your friends receive it, they compute the hash value of the message using the same hash function as you. You can then compare the two hashes and check that they are equal. Hash functions used for this have the property that it is hard to find collisions. Such hash functions are called **cryptographic** hash functions. They can have other properties too, like the one in the following example.

- Password Storage

Ever wondered why websites don't have an option to send you your password in case you forget it, and they make you reset it? To be able to send it back, they have to store it in plain text. By doing so, if someone gets access to the password database, they can easily steal all the accounts! What websites do is store a hash of your password. When you send the password to log in, they compute its hash and check if it is equal to what they have stored. In this case, finding collisions is not such a big problem. Here, you need to know that if someone finds the hash of your password, they should not be able to find out your password from it.

- Hash tables

A more common use in day-to-day programming is hash tables. They are fast and convenient data structures that use hashing. With them, you can search, insert, or remove elements from a table. The main idea behind them is that you want to use hash values to index data in an array. For example, if you want to use a hash table to store a phone book, you can save the pair *(Tim Black, 0123456789)* at index *Tm Blck.* Then, to find Tim's phone number, you only have to search at index *Tm Blck.* After that, you can also save *(Tom Black, 9876543210)* at the same index and, whenever you need to find Tim's or Tom's phone number, you only have to search between two pairs, rather than the whole phone book. Hash functions used in hash tables are less restrictive than the ones used for message digests and password storage, and their hash values are, generally, numbers. We will look closer at hash tables in the following topics.

§3. Conclusion

You now have an idea about hashing and hash functions, what they are and where you can use them. This should be more than enough for now to emphasize their importance. In the next topics, we will study in more detail hash functions, hash tables, and collisions.

Table of contents:

Report a typo

705 users liked this piece of theory. 4 didn't like it. **What about you?**

😍 🙂 😐 🙁 😡

Start practicing

Unskip this topic

Comments (17) Useful links (4) Show discussion