TASK

Interview Preparation: Hashing

Visit our website

Introduction

WELCOME TO THE INTERVIEW PREPARATION: HASHING TASK!

You have reached your final interview preparation task! Great job! In this task, we will consider one final advanced topic that is often discussed in interviews: Hashing. Hashing is extremely important as it is often used to improve security, ensure version control, and create more efficient data structures. Cryptocurrencies like Bitcoin also rely heavily on hash functions. Use this task to prepare yourself for potential discussions on this important and relevant topic.

Get in touch

Connect for support

Remember that with our courses, you're not alone! You can contact an expert code reviewer to get support on any aspect of your course.

The best way to get help is to login to Discord at **https://discord.com/invite/hyperdev** where our specialist team is ready to support you.

Our team is happy to offer you support that is tailored to your individual career or education needs. Do not hesitate to ask a question or for additional support!

WHAT IS HASHING?

Hashing has to do with taking a certain value (for example your password), applying some kind of mathematical operation to it (called a hashing algorithm or a hashing function) and getting the resulting changed value (known as a hash or hash value or digest message).

WHY DO WE USE HASHING?

There are many important reasons for using hashing. One application of hashing is using it to store passwords in databases. You may need to store your user's password so that you know whether they have entered a valid password or not. However, it would be very insecure to store all passwords as plain text in the database. Rather, it would be more secure if you applied a hashing function to the password and stored the hash/digest in the database instead. See the code below for an example of how a value can be hashed in Node.js using the crypto module.

```
const crypto = require('crypto');   //line 1
const hash = crypto.createHash('sha256');  //line 2

hash.update('some data to hash');  //line 4
console.log(hash.digest('hex'));   //line 5
// Prints:
//   6a2da20943931e9834fc12cfe5bb47bbd9ae43489a30726962b576f4e3993e50
```

Source of code: https://nodejs.org/api/crypto.html#crypto_class_hash

In line 2 of the code above, we specify which hashing algorithm we use to calculate the hash. In this example, we specify that the SHA256 algorithm will be used. Another common hashing algorithm is MD5.

In line 4, we use the `hash.update()` function to pass the value we want to hash to the hashing algorithm. We could pass the user's password as the value here. In line 5, we print the digest to the console using the `hash.digest()` function. The argument (`'hex'`) passed to the `hash.digest()` function specifies what type of encoding will be used. The encoding can be 'hex', 'latin1' or 'base64'.

Hashing can also be used for version control. When given the same input, a specific hashing algorithm will always produce the same digest. Therefore, we can compare the digest of two separate files and if the digests are the same, we will know the files are the same and have not been changed.

Hashing is also often used to create hash tables. Hash tables are data structures that make finding data a lot quicker. We have used data structures like lists and arrays to store collections of data in this Bootcamp. However, as you know, searching for an item in a big array can take time. Hash tables make it much quicker to access data; the order of complexity is O(1) as compared to a binary search's O(log N).

HASH TABLES

As we have seen in the code example in this task, hash values/digests can be string values. However, you can also have hashing functions that receive a string or number value as input and produce an integer as output. These types of hashing algorithms are used to create hash tables. Hash tables store *key-value pairs*. For example, imagine that you would like to store the names and telephone numbers of a group of people. The *key* could be the person's name and the *value* could be the telephone number (see the image below).

To create a hash table to store this information, the key would be passed to the hashing function. The hashing function would produce an integer number as output. This number would be used as the index of where the value would be stored in the hash table. When you want to look up a person's telephone number, you can hash that person's name and know exactly where (at which index) in the hash table the number can be found.

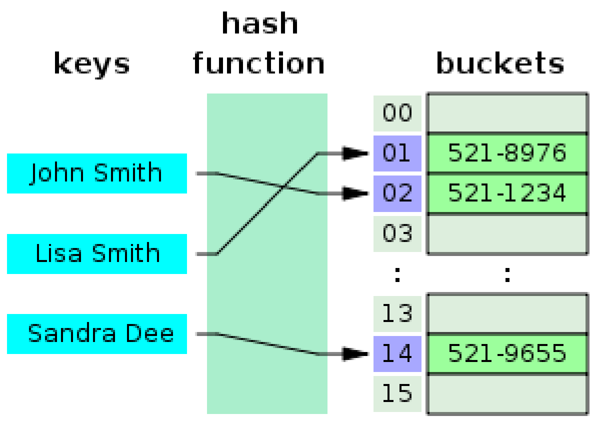

Image source: ***https://users.cs.fiu.edu/~giri/teach/3530/f16/Lectures/LecX-Hashing.pdf***

When filling up the hash table, collisions are possible. A collision occurs when two or more keys have the same hash value and are thus given the same index number in the hash table. Collisions can be handled by using either ***separate chaining*** (if the position is already full, put the value in a list associated with that index) or ***open addressing*** (if the position is already taken, put the value in another open position in the hash table). **This tool** can help visualise how each of the different open addressing methods works.

Compulsory Task 1

Follow these steps:

Create a blog post called "How JavaScript uses hashing" that includes the following:

- Explain what hashing is and what it is used for.
- Explain what a hash table is and what the benefits of this data structure are.
- Explain what the difference is between hashing and encryption. **Recommended reading**.
- In JavaScript, Map objects are often implemented using hash tables. Explain what a Map object is and how it can be used. Compare objects and maps. In your comparison, refer to how hashing is used for each. **Recommended reading 1** and **recommended reading 2**.

- Optionally, also discuss:
  - Explain how collisions are handled when creating a hash table. Explain separate chaining and open addressing. Explain the 3 most common methods for open addressing: linear probing, quadratic probing, and double hashing. **Recommended reading**.
  - Explain how hashing is used in JavaScript to improve the security of your applications. **Recommended reading**.

Well done! You've done it! You have completed all the content for this Bootcamp. Your next step is your mock interview. Congratulations on making it this far!

If you are having any difficulties, please feel free to contact our specialist team **on Discord** for support.

Completed the task(s)?

Ask an expert to review your work!

Review work

Rate us

Share your thoughts

HyperionDev strives to provide internationally-excellent course content that helps you achieve your learning outcomes.

Think that the content of this task, or this course as a whole, can be improved, or think we've done a good job?

Click here to share your thoughts anonymously.