Interview Cake

← course home (/table-of-contents)

Hashing and Hash Functions

A **hash function** takes data (like a string, or a file's contents) and outputs a *hash*, a fixed-size string or number.

For example, here's the MD5 hash (MD5 is a common hash function) for a file simply containing "cake":

```
DF7CE038E2FA96EDF39206F898DF134D
```

And here's the hash for the same file after it was edited to be "cakes":

```
0E9091167610558FDAE6F69BD6716771
```

Notice the hash is *completely* different, even though the files were similar. Here's the hash for a long film I have on my hard drive:

```
664f67364296d08f31aec6fea4e9b83f
```

The hash is the same length as my other hashes, but this time it represents a much bigger file—461Mb.

We can think of a hash as a "fingerprint." We can trust that a given file will always have the same hash, but we can't go from the hash back to the original file. Sometimes we have to worry about multiple files having the same hash value, which is called a **hash collision**.

Some uses for hashing:

1. **Dictionaries**. Suppose we want a list-like data structure with constant-time lookups, but we want to look up values based on arbitrary "keys," not just sequential "indices." We could allocate a list, and use a hash function to translate keys into list indices. That's the basic idea behind a dictionary!
2. **Preventing man-in-the-middle attacks**. Ever notice those things that say "hash" or "md5" or "sha1" on download sites? The site is telling you, "We hashed this file on our end and got this result. When you finish the download, try hashing the file and confirming you get the same result. If not, your internet service provider or someone else might have injected malware or tracking software into your download!"

← course home (/table-of-contents)

Next up: Hash Table → (/concept/hash-map?course=fc1&section=hashing-and-hash-tables)