Hashing

Authored by AllisonP

Table Of Contents

Hashing

This refers to the process of converting a key into an index in a table by performing some simply computed operation on the key.

Hash Tables

These are data structures that very efficiently map *keys* to *values*. When we use a key to look up $value$ in a hash table, we convert the key into a hash code and then use the hash code to map to an index in the table. In other words, we use hashing go from $string \rightarrow hashcode \rightarrow index$ in a hash table.

Advantages

Hashing is really important because it allows you to map an infinite number of keys to some finite number of values. This means you can store your mapped values in a table that's significantly smaller than the number of potential keys or hash codes.

Disadvantages

Because there are generally an infinite number of keys mapping to a finite number of hash codes, two distinct keys can map to the same table index. We call these keys *synonyms*, and we call the actual phenomenon of having multiple keys whose hash codes map to the same table index a *collision*. One way of dealing with this is something called *chaining* where we store each colliding $(key, value)$ pair in a linked list located at their mapped index (i.e., collision site). It's important to store the key with the $value$, or you won't know which $value$ in the list maps to which key.

Asymptotic Analysis: Inserting, Finding, and Deleting

For a good hash table, this is $\Theta(1)$. For a terrible hashtable with a lot of collisions, this is $\mathcal{O}(n)$.