19.6 Summary

In this chapter, we learned about hashing, a powerful technique for turning a more complex object like a `String` into a numerically representable value like an `int`.

The *hash table* is a data structure that combines the *hash function* with the fact that arrays can be indexed in constant time. Using the hash table and the map abstract data type, we can build a `HashMap` which allows for amortized constant time access to any key-value pair so long as we know which bucket the key falls into.

However, we quickly demonstrated that this naive implementation has several drawbacks: the ability to represent all different kinds of objects, memory efficiency, and collisions.

We investigated the importance of the `hashCode()` function to gain an understanding of how it affects the runtime of the hash table. To allow for smaller `size()` in the array, we used the modulo operator to shrink hash values down to a specified range of numbers. We then added external chaining to solve collisions by allowing multiple entries to live in a single bucket in the form of a LinkedList, and explored resizing functionality based on the load factor!

Previous
19.5 Resizing & Hash Table Performance

Next
19.7 Exercises

Last updated 1 year ago