How to Write a Hash Function

3 min

You might be thinking at this point that we've glossed over a very important aspect of a hash table here. We've mentioned that a hash function is necessary, and described some features of what a hash function does, but never really given an implementation of a hash function that does not feel like a toy example.

Part of this is because a hash function needs to be simple by design. Performing complex mathematical calculations that our hash table needs to compute every time it wants to assign or retrieve a value for a key will significantly damage a hash table's performance for two things that it should be able to do quickly.

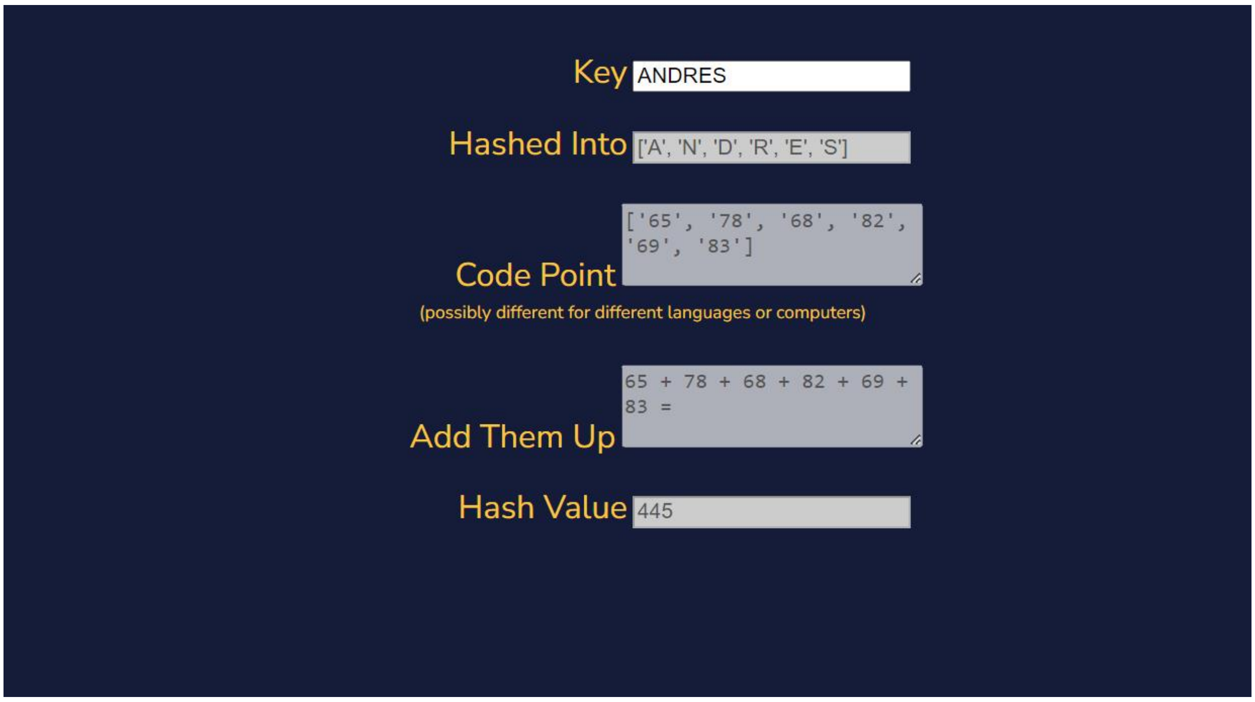
Hash functions also need to be able to take whatever types of data we want to use as a key. We only discussed strings, a very common use case, but it's possible to use numbers as hash table keys as well.

A very common hash function for integers, for example, is to perform the modular operation on it to make sure it's less than the size of the underlying array. If the integer is already small enough to be an index into the array, there's nothing to be done.

Many hash functions implementations for strings take advantage of the fact that strings are represented internally as numerical data. Frequently a hash function will perform a shift of the data bitwise, which is computationally simple for a computer to do but also can predictably assign numbers to strings.

Instructions

In the application change the Key and watch how a hashing function might create a hash value for that key.



Key	ANDRES
Hashed Into	['A', 'N', 'D', 'R', 'E', 'S']
Code Point (possibly different for different languages or computers)	['65', '78', '68', '82', '69', '83']
Add Them Up	65 + 78 + 68 + 82 + 69 + 83 =
Hash Value	445