

Hash Functions

4 min

A hash function takes a string (or some other type of data) as input and returns an array index as output. In order for it to return an array index, our hash map implementation needs to know the size of our array. If the array we are saving values into only has 4 slots, our hash map's hashing method should not return an index bigger than that.

In order for our hash map implementation to guarantee that it returns an index that fits into the underlying array, the hash function will first compute a value using some scoring metric: this is the hash value, hash code, or just the *hash*. Our hash map implementation then takes that hash value [mod](#) the size of the array. This guarantees that the value returned by the hash function can be used as an index into the array we're using.

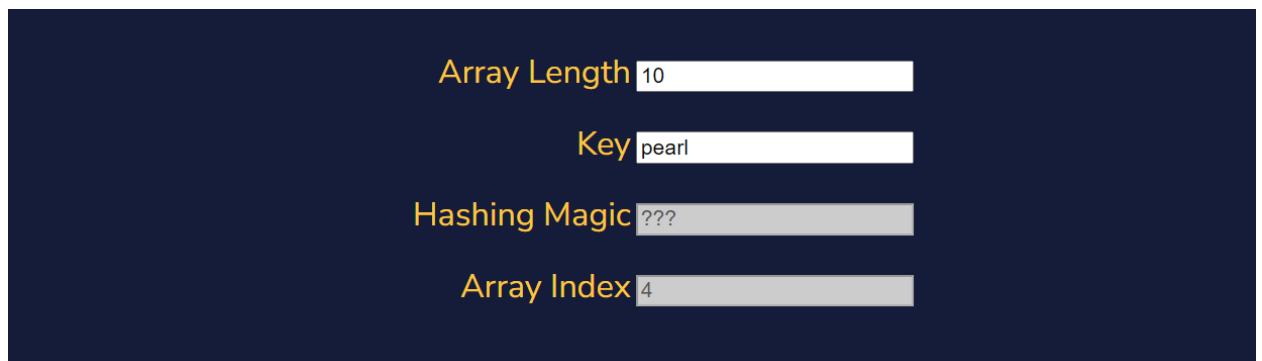
It is actually a defining feature of all hash [functions](#) that they greatly reduce any possible inputs (any string you can imagine) into a much smaller range of potential outputs (an integer smaller than the size of our array). For this reason, hash functions are also known as *compression functions*.

Much like an image that has been shrunk to a lower resolution, the output of a hash function contains less data than the input. Because of this, hashing is not a reversible process. With just a hash value it is impossible to know for sure the key that was plugged into the hashing function.

Instructions

Why is it necessary for a hash function to not be reversible? Does this “only works in one direction” criterion sound familiar?

Change the key and array size in the application. See how different keys give different indices, but sometimes different keys give the same index. Notice that the array index is always smaller than the length of the array.



Array Length	10
Key	pearl
Hashing Magic	???
Array Index	4