Hashing

10 min

The *hashing function* is the secret to efficiently storing and retrieving values in a hash map. A hashing function takes a key as input and returns an index within the hash map's underlying array.

This function is said to be *deterministic*. That means the hashing function must always return the same index when given the same key. This is important because we will need to hash the key again later to retrieve the stored value. We won't be able to do this unless our hashing function behaves in a predictable and consistent way.

Getting an integer representing an index can be done by summing up each character code of the key (as a numeric value) with the running total of the previously summed character codes.

The hashing function should follow this logic:

```
declare hashCode variable with value of 0


for each character in the key
  add the sum of the current character code value and hashCode to hashCode


return hashCode
```

Adding the sum of hashCode and the character code to the hashCode again creates a deterministic and also non-reversible implementation of a hashing function. This avoids generating a duplicate index if keys have the same characters in different orders, such as bat and tab.

Instructions

1. Checkpoint 1 Passed

1.

Create a HashMap method, .hash(), with key as a parameter. This method will take a string and use it to generate an index in the hash map's internal array.

2. Checkpoint 2 Passed

2.

To generate an index for each key-value pair, we'll calculate a number based on the characters in the input string. Declare a variable whose value can be changed within .hash() called hashCode. Assign it an initial value of 0.

This variable will keep a running total of character codes.

Hint

The let keyword creates a variable whose value can be reassigned later.

3. Checkpoint 3 Passed

3.

After declaring hashCode, create a for loop that loops through each index of key.

Hint

This is an example of a for loop in JavaScript:

```
const recipes = ['jollof', 'adobo', 'frybread'];

for(let i = 0; i < recipes.length; i++) {
  console.log(`Serving up ${recipes[i]}!`);
}
```

4. Checkpoint 4 Passed

4.

Inside of the for loop convert each character in key to an integer using the JavaScript string method .charCodeAt().

This method only works on strings and converts a character at a specific index into an integer between 0 and 65535. This integer represents the equivalent Unicode value of that character.

To use .charCodeAt() call it on a string with the index of the character you want the character code of:

```
// The code below will return the character code of 'H'
'Hello world!'.charCodeAt(0) // => 72
```

Add the result of calling .charCodeAt() on the current character of key and hashCode to the hashCode variable.

Outside of the for loop, return the finished hashCode.

Hint

Using the string method .charCodeAt(), your code should add and assign the character code of i to hashCode. Your solution should look like this:

```
hashCode += hashCode + key.charCodeAt(i);
```

The hashCode += allows the hashing function to avoid generating duplicate hashCodes if keys have the same characters in different orders, such as bat and tab.

HashMap.js

```
class HashMap {

  constructor(size = 0) {

    this.hashmap = new Array(size)

      .fill(null);

  }


  hash(key) {
    let hashCode = 0;

    for (let i = 0; i < key.length; i++) {

      hashCode += hashCode + key.charCodeAt(i);

    }

    return hashCode;

  }


}


module.exports = HashMap;
```