Compression

6 min

The current hashing function will return a wide range of integers — some of which are not indices of the hash map array. To fix this, we need to use *compression*.

Compression means taking some input and returning an output only within a specific range.

In our hash map implementation, we're going to have our hashing function handle compression in addition to hashing. This means we'll add an additional line of code to compress the hashCode before we return it.

The updated .hash() should follow these steps:

initialize hashCode variable to 0

for each character in the key

  add the character code and hashCode to hashCode

return compressed hashCode

Instructions

1. Checkpoint 1 Passed

1.

Currently, our .hash() method is generating an integer representing an index but it's not guaranteed that this index will be within the bounds of the hash map's array.

To do this, we'll use *modular arithmetic*. Because modular arithmetic prevents a value from growing larger than some limit, it's a common solution when we want a value to "wrap around".

After the for loop in the .hash() of HashMap, compress the value stored in hashCode by using modular arithmetic to return the remainder of dividing hashCode by the length of the hash map.

Hint

The modulo operator, %, in JavaScript returns the remainder when dividing two numbers.

For example:

```
11 % 2 // => 1
```

2. Checkpoint 2 Passed

2.

Check your work. Save a new HashMap instance with a size of 3, in a constant myHashMap and use the new .hash() to log the result of hashing the key 'id'. Hash and log 'id' again. Are the logged values the same or are they different?

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 % this.hashmap.length;
  }
}

module.exports = HashMap;

const myHashMap = new HashMap(3);
console.log(myHashMap.hash('id'));
console.log(myHashMap.hash('id'));
```