DSA Hash Maps

Hash Maps

A Hash Map is a form of Hash Table data structure that usually holds a large number of entries.

Using a Hash Map we can search, add, modify, and remove entries really fast.

Hash Maps are used to find detailed information about something.

In the simulation below, people are stored in a Hash Map. A person can be looked up using a person's unique social security number (the Hash Map key), and then we can see that person's name (the Hash Map value).

Hash Map

```
0 : 123-4569 Jens
1 :
2 : 123-4570 Peter
3 : 123-4571 Lisa
```

7 :

8 : | 123-4567 | Charlotte | | 123-6574 | Bob |

9 : | 123-4568 | Thomas |

Hash Code

0 % 10 = `0`

Try interacting with the Hash Map

XXX - XXXX Name

put() remove() get() size()

Note: The Hash Map would be more useful if more information about each person was attached to the corresponding social security number, like last name, birth date, and address, and maybe other things as well. But the Hash Map simulation above is made to be as simple as possible.

It is easier to understand how Hash Maps work if you first have a look at the two previous pages about Hash Tables and Hash Sets. It is also important to understand the meaning of the words below.

- **Entry:** Consists of a key and a value, forming a key-value pair.
- **Key:** Unique for each entry in the Hash Map. Used to generate a hash code determining the entry's bucket in the Hash Map. This ensures that every entry can be efficiently located.
- **Hash Code:** A number generated from an entry's key, to determine what bucket that Hash Map entry belongs to.
- **Bucket:** A Hash Map consists of many such buckets, or containers, to store entries.
- **Value:** Can be nearly any kind of information, like name, birth date, and address of a person. The value can be many different kinds of information combined.

The hash function in the simulation above takes the numbers in the social security number (not the dash), add them together, and does a modulo 10 operation (`% 10`) on the sum of characters to get the hash code as a number from 0 to 9.

This means that a person is stored in one of ten possible buckets in the Hash Map, according to the hash code of that person's social security number. The same hash code is generated and used when we want to search for or remove a person from the Hash Map.

The Hash Code gives us instant access as long as there is just one person in the corresponding bucket.

In the simulation above, `Charlotte` has social security number `123-4567` . Adding the numbers together gives us a sum `28` , and modulo 10 of that is `8` . That is why she belongs to bucket `8` .

Modulo: A mathematical operation, written as `%` in most programming languages (or mod in mathematics). A modulo operation divides a number with another number, and gives us the resulting remainder. So for example, `7 % 3` will give us the remainder `1` . (Dividing 7 apples between 3 people, means that each person gets 2 apples, with 1 apple to spare.)

Direct Access in Hash Maps

Searching for `Charlotte` in the Hash Map, we must use the social security number `123-4567` (the Hash Map key), which generates the hash code `8` , as explained above.

This means we can go straight to bucket `8` to get her name (the Hash Map value), without searching through other entries in the Hash Map.

In cases like this we say that the Hash Map has constant time $O(1)$ for searching, adding, and removing entries, which is really fast compared to using an array or a linked list.

But, in a worst case scenario, all the people are stored in the same bucket, and if the person we are trying to find is last person in this bucket, we need to compare with all the other social security numbers in that bucket before we find the person we are looking for.

the entries evenly between the buckets, and to have around as many buckets as Hash Map entries.

Having a lot more buckets than Hash Map entries is a waste of memory, and having a lot less buckets than Hash Map entries is a waste of time.

Note: A social security number can be really long, like 11 digits, which means it is possible to store 100 billion people with unique social security numbers. This is a lot more than in any country's population, and even a lot more than there are people on Earth.

Using an array where each person's social security number is the index in the array where this person is stored is therefore a huge waste of space (mostly empty buckets).

Using a Hash Map (or a database with similar properties) makes more sense as the number of buckets can be adjusted to the number of people.

Hash Map Implementation

Hash Maps in Python are typically done by using Python's own `dictionary` data type, but to get a better understanding of how Hash Maps work we will not use that here.

To implement a Hash Map in Python we create a class `SimpleHashMap`.

Inside the `SimpleHashMap` class we have a method `__init__` to initialize the Hash Map, a method `hash_function` for the hash function, and methods for the basic Hash Map operations: `put`, `get`, and `remove`.

We also create a method `print_map` to better see how the Hash Map looks like.

Example

```
class SimpleHashMap:
    def __init__(self, size=100):
        self.size = size
        self.buckets = [[] for _ in range(size)]  # A list of bucke
        return numeric_sum % 10  # Perform modulo 10 on the sum

    def put(self, key, value):
        # Add or update a key-value pair
        index = self.hash_function(key)
        bucket = self.buckets[index]
        for i, (k, v) in enumerate(bucket):
            if k == key:
                bucket[i] = (key, value)  # Update existing key
                return
        bucket.append((key, value))  # Add new key-value pair if no

    def get(self, key):
        # Retrieve a value by key
        index = self.hash_function(key)
        bucket = self.buckets[index]
        for k, v in bucket:
            if k == key:
                return v
        return None  # Key not found

    def remove(self, key):
        # Remove a key-value pair
        index = self.hash_function(key)
        bucket = self.buckets[index]
        for i, (k, v) in enumerate(bucket):
            if k == key:
                del bucket[i]  # Remove the key-value pair
                return

    def print_map(self):
        # Print all key-value pairs in the hash map
        print("Hash Map Contents:")
        for index, bucket in enumerate(self.buckets):
            print(f"Bucket {index}: {bucket}")
```

Example

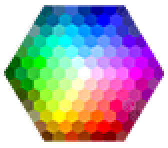
```
            print(f"Bucket {index}: {bucket}")

# Creating the Hash Map from the simulation
hash_map = SimpleHashMap(size=10)

# Adding some entries
hash_map.put("123-4567", "Charlotte")
hash_map.put("123-4568", "Thomas")
hash_map.put("123-4569", "Jens")
hash_map.put("123-4570", "Peter")
hash_map.put("123-4571", "Lisa")
hash_map.put("123-4672", "Adele")
hash_map.put("123-4573", "Michaela")
hash_map.put("123-6574", "Bob")

hash_map.print_map()

# Demonstrating retrieval
print("\nName associated with '123-4570':", hash_map.get("123-457

print("Updating the name for '123-4570' to 'James'")
hash_map.put("123-4570","James")

# Checking if Peter is still there
print("Name associated with '123-4570':", hash_map.get("123-4570"
```

Try it Yourself »

❮ Previous Next ❯

Sign in to track progress

COLOR PICKER



Top Tutorials

HTML Tutorial
CSS Tutorial
JavaScript Tutorial
How To Tutorial
SQL Tutorial
Python Tutorial
W3.CSS Tutorial
Bootstrap Tutorial
PHP Tutorial
Java Tutorial
C++ Tutorial
jQuery Tutorial

Top References

HTML Reference
CSS Reference
JavaScript Reference
SQL Reference
Python Reference
W3.CSS Reference
Bootstrap Reference
PHP Reference
HTML Colors
Java Reference
AngularJS Reference
jQuery Reference

Top Examples

HTML Examples
CSS Examples
JavaScript Examples
How To Examples
SQL Examples
Python Examples
W3.CSS Examples
Bootstrap Examples
PHP Examples
Java Examples
XML Examples
jQuery Examples

Get Certified

HTML Certificate
CSS Certificate
JavaScript Certificate
Front End Certificate
SQL Certificate
Python Certificate
PHP Certificate
jQuery Certificate
Java Certificate
C++ Certificate
C# Certificate
XML Certificate

FORUM ABOUT ACADEMY