HashMate – Visualization of hashing algorithms

Kirill Milintsevich and Salijona Dyrmishi
Institute of Computer Science, University of Tartu, Estonia
milintsevich@gmail.com, dyrmishisalijona@gmail.com

1. Introduction

This project helps to get the visual intuition on how the hashing works. With HashMate, you can learn about hashing using visual interactive examples. Moreover, the website provides useful information about different hashing techniques. We believe, that this project will help new students to grasp the concept more easily.

2. About Hashing

Hashing is the process of converting an input of any size and length, to either a number or a fix-sized string of text using a mathematical function. In this process, there are several elements: an input to be hashed and a **hash function** that produces a certain **hash value**.

There are many ways to hash an input. This makes choosing a good hash function difficult. A good hash function should:

- be stable;
- be fast;
- produce a uniformly distributed hash values

One of the main challenges in hashing is avoiding **collisions**. Sometimes, a hash function produces the same hash value for different outputs. It's near impossible to completely avoid this when the number of inputs is high but there exist various methods to minimize the collisions.

Sometimes, a handful of hash functions is not enough. In this case, so called **families** of hash functions are used. A family of hash functions can generate different hash functions dynamically.

Hashing has different applications: from cryptography to creating data structures with fast access to the elements or compact size.

3. Supported methods

This project supports these types of hashing:

1. Simple Hashing
2. Closed addressing - Linear probing
3. Closed addressing - Quadratic probing
4. Open addressing - Chaining
5. Universal Hashing
6. Bloom filters

Users can choose the size of the table from 1 to 20. After an element is inserted calculations are shown and the location in canvas. Users can search for elements in the table.

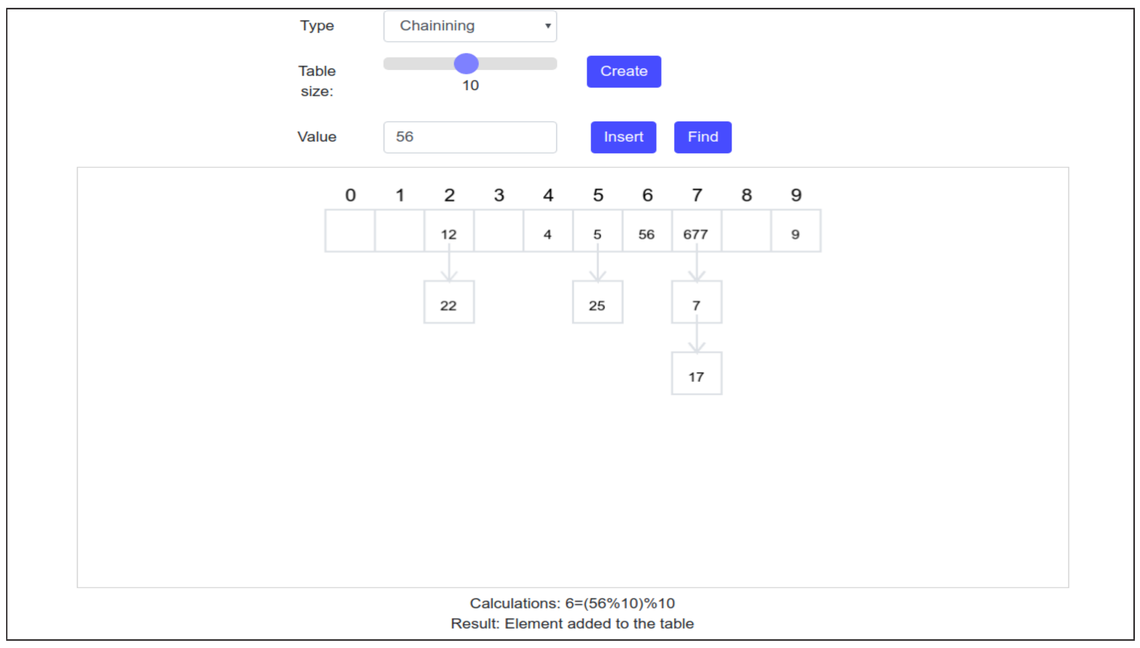
4. Link

HashMate visualizations were created using HTML canvas element to draw graphics, JavaScript and JQuery to process the hashing and finally, the bootstrap library to make the interface. The project can be found here **https://github.com/501Good/HashMate**

5. Demonstration

Bloom filters are an efficient data structure used to test if an element is a member of the set. They guarantee 0 false negatives rate but can contain false positives. Murmur and FNV hashing algorithms are used to insert the values on bloom filter. Users can see a historic of values hashed and their positions on the table.

Chaining comes as solution to solve collisions. Whenever a collisions happens the element is added in a linked list. Users can view the chaining process step by step.



We used **Universal Hashing** to show the concept of collisions. The user can pick the table size and a number of random elements to be inserted to the hash table using the universal hash function. The user is later provided with an interactive graph that shows the distribution of collisions.