**Project No.7: Hash Application on Student Information System**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Student No. | Email | Responsibility |
| 李志容 | 14346009 | 978282388@qq.com | Parts of codes, parts of the reports |
| 谭笑 | 14346022 | 296309711@qq.com | Parts of codes, parts of the reports |

Data: 2016.6.3

1. **Introduction**

This project is building a student information administration system. Students’ information includes name, age, gender and ID, while the ID is used as a key. To build an efficient and useful system, we use hash tables to store information since hash tables are an efficient implementation of a keyed array data structure, a structure sometimes known as an associative array or map. A hash table is made up of two parts: an array and a mapping function, known as a hash function. In this project, we use mod function as hash function and solve the hash collision by two methods, which are separate chaining and open addressing. In addition, our program has a good user interface.

1. **Analysis and Design**
2. **Store information with hash function**
3. **Data structures**

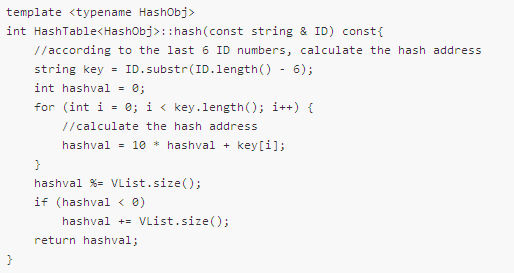
To store the information, we use the data structure of Vector, because it can quickly find an element by index and when the size is fixed, it is very stable. Hash address value is important in storing the information, so if we calculate the hash address, we can directly operate on corresponding position in the vector.

In separate chaining method, we use linked-list to store the element that has the same hash address because it is easy to insert or remove an element on the list.

In open addressing method, we only use vector, but we also set a symbol for each position, to check whether this position is ACTIVE, DELTTION or EMPTY.

1. **Important algorithms**

The most important algorithm is calculating the hash address value for every inserted information, given the ID. Firstly, we get the last 6 numbers of the ID string. Then turn the string of last 6 digits to integer value. At last, take its mod with the whole vector size, which is set in our project is 107 initially. The codes are follow:



1. **Justification**

We only consider the last 6 digits assuming that it is nearly impossible that two different ID that have the same last 6 digits, which is more efficient.

1. **Time complexity**

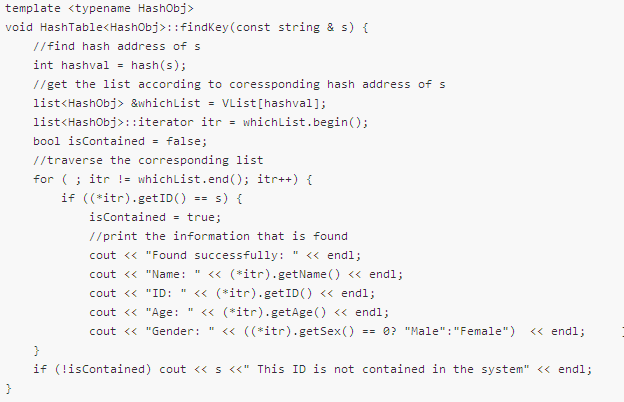
In the process of turning the string to integer, time costing is O(6). Other time cost in calculating. So the overall time complexity is O(constant).

1. **Resolve collision by two methods**
2. **Separate chaining**
3. **Data structures**

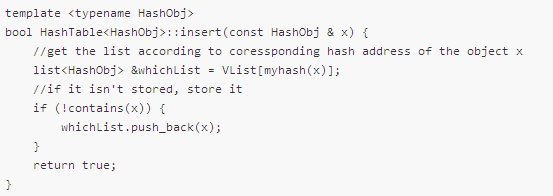
Separate chaining is to keep a list of all elements that hash to the same value. We can use the standard library list implementation. Since a doubled linked list is efficient in deletion and insertion, it is preferable in this project.

1. **Important algorithms**

The first important algorithm is the **FIND** operation. To perform a find operation, we use the hash function to determine which list to traverse. Then search the appropriate list.



The second important algorithm is the **INSERT** operation. To perform an insert, we check the appropriate list to see whether the element is already in place. If the element turns out to be new, it can be inserted at the front of the list, since it is convenient and also because frequently it happens that recently inserted elements are the most likely to be accessed in the near future.



1. **Time complexity**

Suppose that the average length for each list is β, then the FIND operation need to use O(constant), which is used to calculate the hash value and O(β), the average time use to find the correct position. So the total time is O(constant).

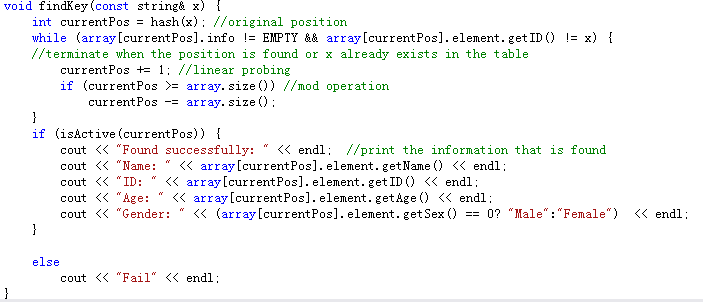
The INSERT operation is the same.

1. **Open addressing**
2. **Data structures**

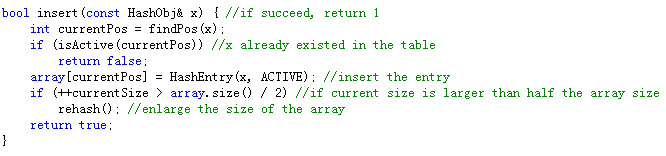
Open addressing is to resolve collisions by the method of probing, which means searching for alternative positions in the array to place the key. There are three main types of open addressing——linear probing, quadratic probing and double probing. In this project, we choose to use linear probing which searches the table for the nearest following free position to insert the new key.

1. **Important algorithms**

The first important algorithm is the **FIND** operation. This function finds the hash address given a key string and output the corresponding information of that position. First, find the original position by hash function. Then resolve the collisions if any. If the key exists in the table, output the information corresponding to the key. Otherwise output the error information.



The second important algorithm is the **INSERT** operation. This function inserts a key into the hash table. If the insertion succeeds, return true. If it fails which means the key already exists in the table, return false. First, find the position to insert the key. If that position is marked as active, it means the insertion fails. If not, insert the key to that position and mark it as active. Then if the current size is larger than half the array size, double the array size because we need the load factor to be less than 0.5.



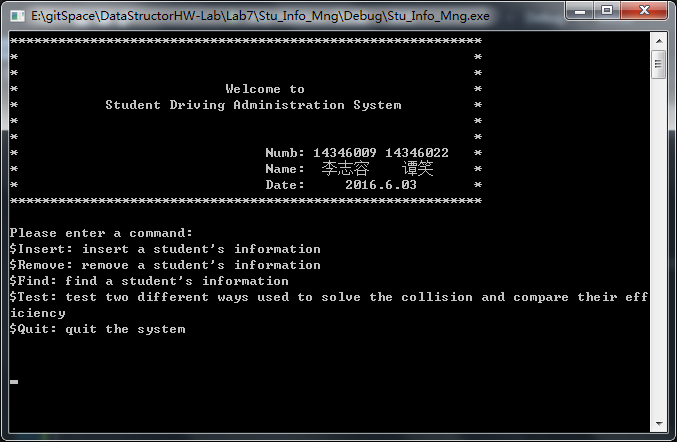
1. **Time complexity**

The time complexities for both **FIND** and **INSERT** operations are O(constant).

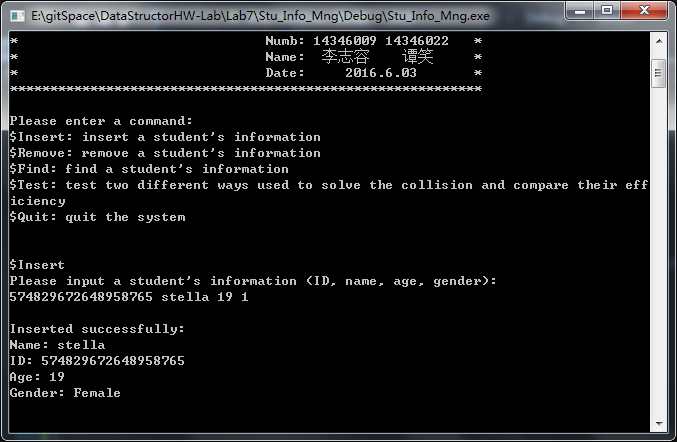
1. **Comparison**

These two methods differ mainly in data structures. Separate chaining uses linked list. The table size can be smaller than the number of keys. Deletion is convenient. Open addressing finds alternative position. It’s more efficient for a small number of keys. But it can easily cause clustering.

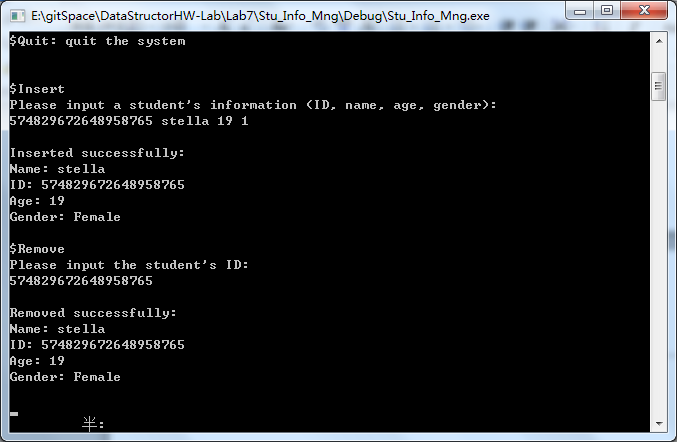
1. **Test**
2. **Initial interface:**



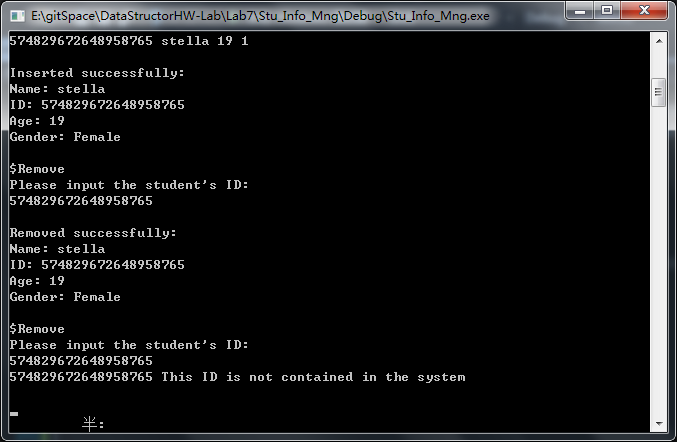
1. **Insert a student’s information:**



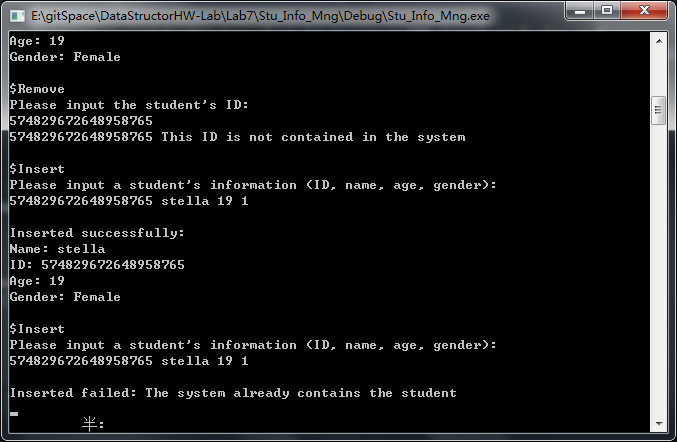
1. **Remove a student’s information:**



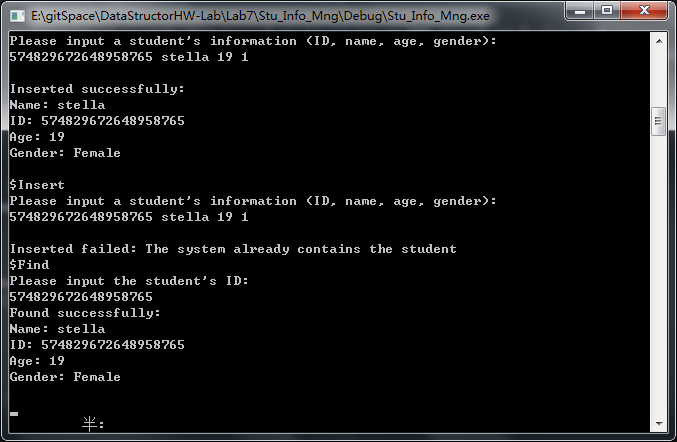
1. **Remove a non-existed information:**



1. **Insert an existed information:**

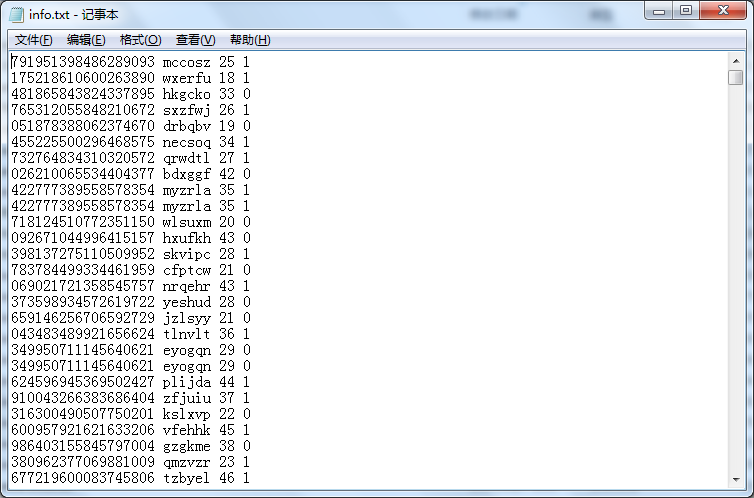


1. **Find the information given an ID:**



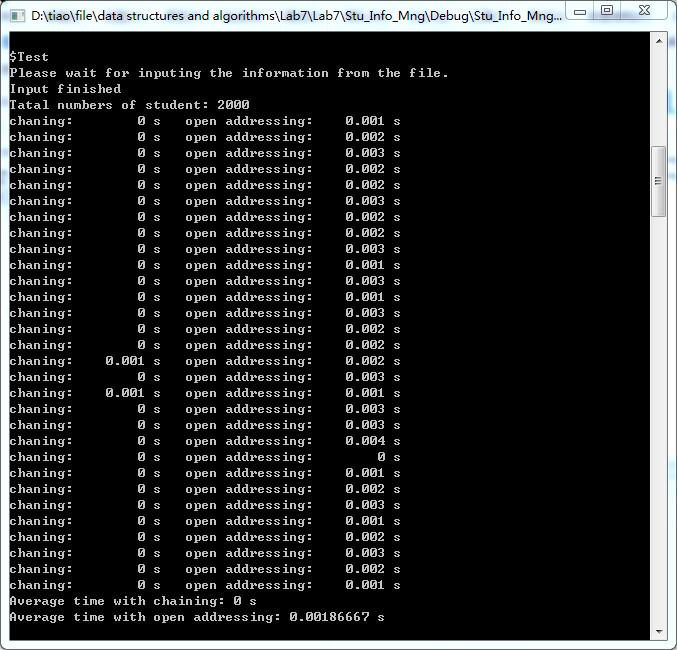
1. **Test the finding time that using chaining and open addressing:**

**Firstly, input the information from the file:**

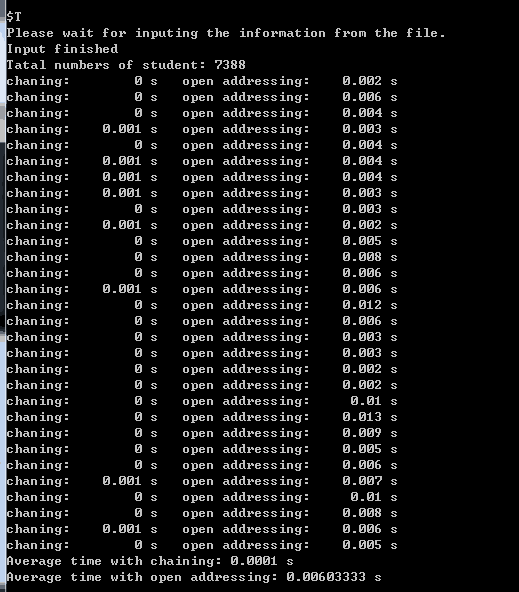


**Then randomly find the information given a random ID. Test for 30 times and calculate average time:**

**When the number of data in the input file (info.txt) is 2000:**



**When the number of data in the input file (info.txt) is more than 7000:**



1. **Conclusion and Discussion**

**Conclusion:**

From the result above, we find that the more information that stored in the file, the more time it needs to find specific information. In addition, whether the capacity of information that stored in the file is large or not, the cost time using separating chaining is much less than using open addressing.

**Achieved:**

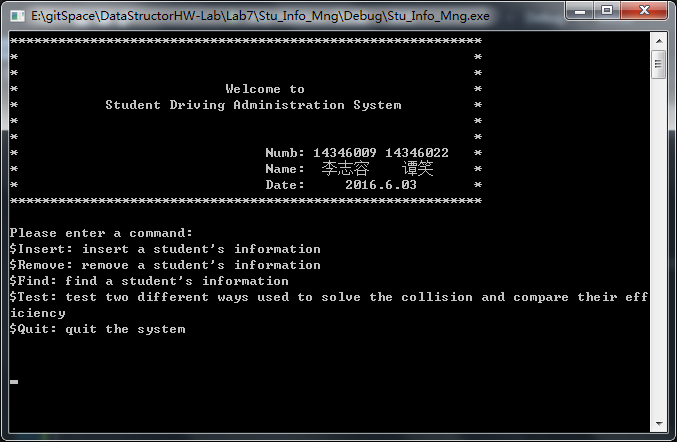
We have achieved to manage the examinees’ information by using hash table and resolve the collisions by two methods——separate chaining and open addressing.

**Not achieved:**

We didn’t use two methods to insert and remove information in the main function. We only use separate chaining.

**Highlights:**

1. We have designed a user-friendly interface which allows the users to choose any operations they want.



1. We have randomly generated a database of more than 7000 examinees. We read it when comparing the efficiencies of the two methods. Only if we have a large amount of data can we compare them more clearly.
2. **Appendices**

Refer to “codes-hash.pdf”.

1. **References**
2. 《数据结构与算法实验实践教程》——乔海燕、蒋爱军、高集荣、刘晓铭
3. 《Data Structures and Algorithm Analysis in C++》——Mark Allen Weiss
4. 《Introduction to Programming with C++》——Y. Daniel Liang
5. <https://en.wikipedia.org/wiki/Open_addressing>
6. <https://en.wikipedia.org/wiki/Linear_probing>
7. <http://soj.sysu.edu.cn/show_problem.php?pid=1007&cid=569>