Vietnam National University,
Ho Chi Minh City

UNIVERSITY OF SCIENCE
FACULTY OF INFORMATION TECHNOLOGY

[CSC10004] Data Structure And Algorithms

EXERCISE 2 REPORT

Implementing Hash Table from scratch

Student Name	Student ID
Bui Minh Duy	23127040

Lecturer In Charge

Bui Duy Dang

Nguyen Thanh Tinh

17/08/2024

Contents

1 Student Information

- **Student ID:** 23127040
- **Full Name:** Bui Minh Duy
- **Class:** 23CLC09

2 How I implemented the requirements

- At first, when implemented these hash tables, I found that in some specific cases, the hash table is not full but the insertion operation is failed. This is because the hash table is not rehashed when the load factor is greater than 0.5. Therefore, I have added the rehashing operation to the insertion operation.
- For the chaining using AVL tree, I have implemented the AVL tree from scratch. The AVL tree is a self-balancing binary search tree, so it is more efficient than the linked list in terms of searching and inserting operations. However, the AVL tree is much more complex than the linked list, so the AVL tree is slower than the linked list in terms of inserting operation.
- On dealing with string keys, I have used a specific hash function based on the hint in the requirements.

3 Detailed Experiments

3.1 Operations

Linear Probing

```
$ g++ main.cpp && ./a
Hash table
+----------------------------------------------------+
|Index  |Key                  |Value                 |
+----------------------------------------------------+
|0      |5                    |Five                  |
|2      |7                    |Seven                 |
|3      |13                   |Thirteen              |
|4      |18                   |Eighteen              |
+----------------------------------------------------+
Add key 2
+----------------------------------------------------+
|Index  |Key                  |Value                 |
+----------------------------------------------------+
|0      |5                    |Five                  |
|1      |2                    |Two                   |
|2      |7                    |Seven                 |
|3      |13                   |Thirteen              |
|4      |18                   |Eighteen              |
+----------------------------------------------------+
Add key 17
+----------------------------------------------------+
|Index  |Key                  |Value                 |
+----------------------------------------------------+
|2      |2                    |Two                   |
|3      |13                   |Thirteen              |
|5      |5                    |Five                  |
|7      |7                    |Seven                 |
|8      |18                   |Eighteen              |
|9      |17                   |Seventeen             |
+----------------------------------------------------+
Remove key 13
+----------------------------------------------------+
|Index  |Key                  |Value                 |
+----------------------------------------------------+
|2      |2                    |Two                   |
|5      |5                    |Five                  |
|7      |7                    |Seven                 |
|8      |18                   |Eighteen              |
|9      |17                   |Seventeen             |
+----------------------------------------------------+
Search for key 13
Key 13 not found
Search for key 5
Value for key 5: Five
```

Figure 1: Linear Probing Operations

The hash table is initialized with size 5. Then, the following operations are performed:

- First, four keys {7, 13, 5, 18} are inserted to the hash table respectively.
- Then, key 2 is inserted (so the hash table becomes full).
- Continue inserting key 17 into the hash table (now the hash table needs to be rehashed to add key 17 successfully, and the indices have been changed).

- After that, key 13 is removed from the hash table.
- Therefore, when performing the search operation for key 13, it is not found in the hash table.
- However, key 5 is found in the hash table.
- Finally, the hash table is released and the program is terminated.

Quadratic Probing

```
$ g++ main.cpp && ./a
Hash table
+--------------------------------------------------+
|Index  |Key                  |Value               |
+--------------------------------------------------+
|0      |5                    |Five                |
|2      |7                    |Seven               |
|3      |13                   |Thirteen            |
|4      |18                   |Eighteen            |
+--------------------------------------------------+
Add key 2
+--------------------------------------------------+
|Index  |Key                  |Value               |
+--------------------------------------------------+
|0      |5                    |Five                |
|1      |2                    |Two                 |
|2      |7                    |Seven               |
|3      |13                   |Thirteen            |
|4      |18                   |Eighteen            |
+--------------------------------------------------+
Add key 17
+--------------------------------------------------+
|Index  |Key                  |Value               |
+--------------------------------------------------+
|1      |17                   |Seventeen           |
|2      |2                    |Two                 |
|3      |13                   |Thirteen            |
|5      |5                    |Five                |
|7      |7                    |Seven               |
|8      |18                   |Eighteen            |
+--------------------------------------------------+
Remove key 13
+--------------------------------------------------+
|Index  |Key                  |Value               |
+--------------------------------------------------+
|1      |17                   |Seventeen           |
|2      |2                    |Two                 |
|5      |5                    |Five                |
|7      |7                    |Seven               |
|8      |18                   |Eighteen            |
+--------------------------------------------------+
Search for key 13
Key 13 not found
Search for key 5
Value for key 5: Five
```

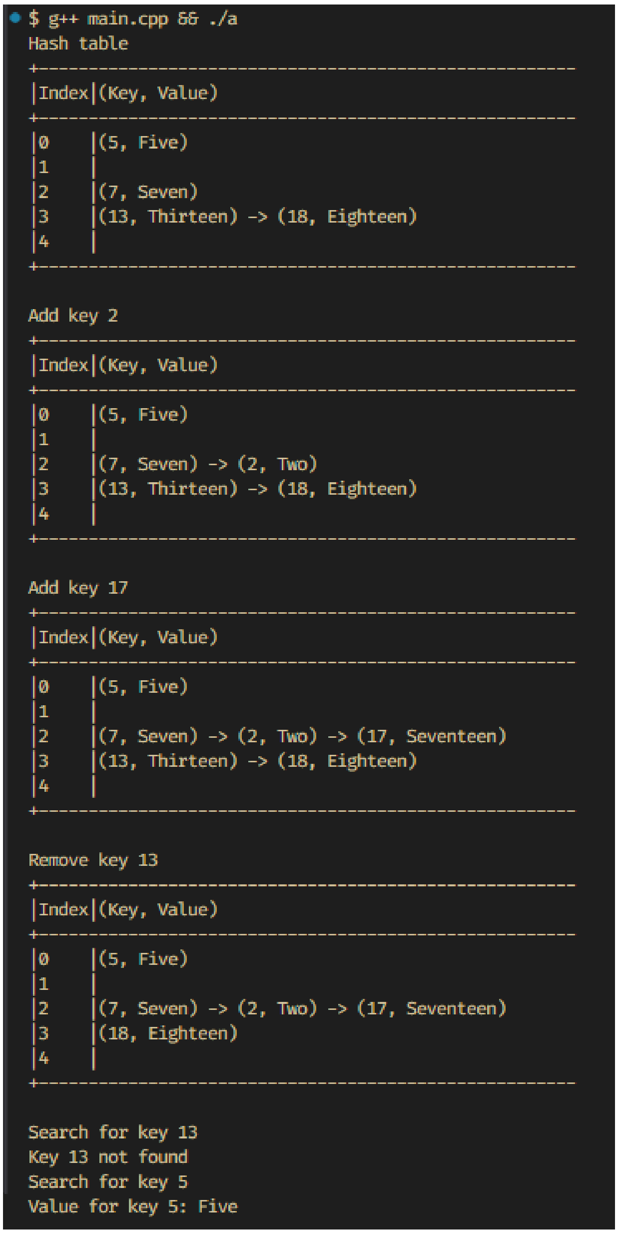
Figure 2: Quadratic Probing Operations

The hash table is initialized with size 5. Then, the following operations are performed:

- First, four keys {7, 13, 5, 18} are inserted to the hash table respectively.

- Then, key 2 is inserted (so the hash table becomes full).
- Continue inserting key 17 into the hash table (now the hash table needs to be rehashed to add key 17 successfully, and the indices have been changed).
- After that, key 13 is removed from the hash table.
- Therefore, when performing the search operation for key 13, it is not found in the hash table.
- However, key 5 is found in the hash table.
- Finally, the hash table is released and the program is terminated.

Chaining using Linked List



```
$ g++ main.cpp && ./a
Hash table
+---------------------------------------------------------
|Index|(Key, Value)
+---------------------------------------------------------
|0    |(5, Five)
|1    |
|2    |(7, Seven)
|3    |(13, Thirteen) -> (18, Eighteen)
|4    |
+---------------------------------------------------------

Add key 2
+---------------------------------------------------------
|Index|(Key, Value)
+---------------------------------------------------------
|0    |(5, Five)
|1    |
|2    |(7, Seven) -> (2, Two)
|3    |(13, Thirteen) -> (18, Eighteen)
|4    |
+---------------------------------------------------------

Add key 17
+---------------------------------------------------------
|Index|(Key, Value)
+---------------------------------------------------------
|0    |(5, Five)
|1    |
|2    |(7, Seven) -> (2, Two) -> (17, Seventeen)
|3    |(13, Thirteen) -> (18, Eighteen)
|4    |
+---------------------------------------------------------

Remove key 13
+---------------------------------------------------------
|Index|(Key, Value)
+---------------------------------------------------------
|0    |(5, Five)
|1    |
|2    |(7, Seven) -> (2, Two) -> (17, Seventeen)
|3    |(18, Eighteen)
|4    |
+---------------------------------------------------------

Search for key 13
Key 13 not found
Search for key 5
Value for key 5: Five
```

Figure 3: Chaining using Linked List Operations

The hash table is initialized with size 5. Then, the following operations are performed:

- First, four keys {7, 13, 5, 18} are inserted to the hash table respectively.
- Then, key 2 and 7 are inserted.
- After that, key 13 is removed from the hash table.
- Therefore, when performing the search operation for key 13, it is not found in the hash table.
- However, key 5 is found in the hash table.
- Finally, the hash table is released and the program is terminated.

Chaining using AVL Tree

```
$ g++ main.cpp && ./a
Hash table
+---------------------------------------------------
|Index|(Key, Value)
+---------------------------------------------------
|0    |(5, Five)
|1    |
|2    |(7, Seven)
|3    |(13, Thirteen) (18, Eighteen)
|4    |
+---------------------------------------------------

Add key 2
+---------------------------------------------------
|Index|(Key, Value)
+---------------------------------------------------
|0    |(5, Five)
|1    |
|2    |(2, Two) (7, Seven)
|3    |(13, Thirteen) (18, Eighteen)
|4    |
+---------------------------------------------------

Add key 17
+---------------------------------------------------
|Index|(Key, Value)
+---------------------------------------------------
|0    |(5, Five)
|1    |
|2    |(2, Two) (7, Seven) (17, Seventeen)
|3    |(13, Thirteen) (18, Eighteen)
|4    |
+---------------------------------------------------

Remove key 13
+---------------------------------------------------
|Index|(Key, Value)
+---------------------------------------------------
|0    |(5, Five)
|1    |
|2    |(2, Two) (7, Seven) (17, Seventeen)
|3    |(18, Eighteen)
|4    |
+---------------------------------------------------

Search for key 13
Key 13 not found
Search for key 5
Value for key 5: Five
```

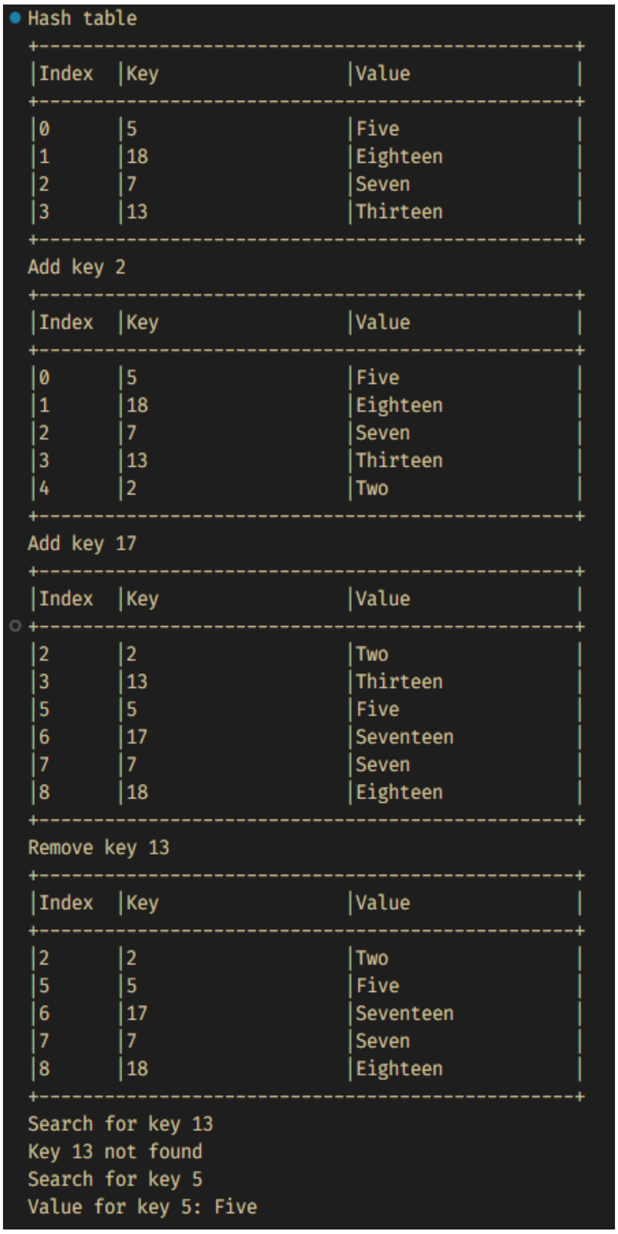
Figure 4: Chaining using AVL Tree Operations

The hash table is initialized with size 5. Then, the following operations are performed:

- First, four keys {7, 13, 5, 18} are inserted to the hash table respectively.
- Then, key 2 and 7 are inserted.
- After that, key 13 is removed from the hash table.
- Therefore, when performing the search operation for key 13, it is not found in the hash table.
- However, key 5 is found in the hash table.

- Finally, the hash table is released and the program is terminated.

Double Hashing



```
Hash table
+--------------------------------------------------+
|Index  |Key                  |Value               |
+--------------------------------------------------+
|0      |5                    |Five                |
|1      |18                   |Eighteen            |
|2      |7                    |Seven               |
|3      |13                   |Thirteen            |
+--------------------------------------------------+
Add key 2
+--------------------------------------------------+
|Index  |Key                  |Value               |
+--------------------------------------------------+
|0      |5                    |Five                |
|1      |18                   |Eighteen            |
|2      |7                    |Seven               |
|3      |13                   |Thirteen            |
|4      |2                    |Two                 |
+--------------------------------------------------+
Add key 17
+--------------------------------------------------+
|Index  |Key                  |Value               |
+--------------------------------------------------+
|2      |2                    |Two                 |
|3      |13                   |Thirteen            |
|5      |5                    |Five                |
|6      |17                   |Seventeen           |
|7      |7                    |Seven               |
|8      |18                   |Eighteen            |
+--------------------------------------------------+
Remove key 13
+--------------------------------------------------+
|Index  |Key                  |Value               |
+--------------------------------------------------+
|2      |2                    |Two                 |
|5      |5                    |Five                |
|6      |17                   |Seventeen           |
|7      |7                    |Seven               |
|8      |18                   |Eighteen            |
+--------------------------------------------------+
Search for key 13
Key 13 not found
Search for key 5
Value for key 5: Five
```

Figure 5: Double Hashing Operations

The hash table is initialized with size 5. Then, the following operations are performed:

- First, four keys {7, 13, 5, 18} are inserted to the hash table respectively.
- Then, key 2 is inserted (so the hash table becomes full).
- Continue inserting key 17 into the hash table (now the hash table needs to be rehashed to add key 17 successfully, and the indices have been changed).
- After that, key 13 is removed from the hash table.

- Therefore, when performing the search operation for key 13, it is not found in the hash table.
- However, key 5 is found in the hash table.
- Finally, the hash table is released and the program is terminated.

3.2 Performance

Linear Probing

```
$ g++ experiment.cpp && ./a
Searching for author name of the first book:
Hash Table: Mark P. O. Morford
Time taken: 2 microseconds
Vector: Mark P. O. Morford
Time taken: 12 microseconds
---------------------------------------------
Searching for author name of the middle book:
Hash Table: BEVERLY CLEARY
Time taken: 0 microseconds
Vector: BEVERLY CLEARY
Time taken: 9329 microseconds
---------------------------------------------
Searching for author name of the last book:
Hash Table: Christopher Biffle
Time taken: 2 microseconds
Vector: Christopher Biffle
Time taken: 18268 microseconds
---------------------------------------------
Searching for key not exist:
Hash Table: Not found
Time taken: 2 microseconds
Vector: Not found
Time taken: 44878 microseconds
```

Figure 6: Linear Probing Performance

- Time Complexity of Linear Probing Search: $O(n)$
- Time Complexity of Linear Seach Algorithm: $O(n)$

Though having the same theorectical time complexity, the actual time execution of Linear Probing Search is much faster than Linear Search Algorithm, especially in the worst case.

Quadratic Probing

```
$ g++ experiment.cpp && ./a
Searching for author name of the first book:
Hash Table: Mark P. O. Morford
Time taken: 0 microseconds
Vector: Mark P. O. Morford
Time taken: 1 microseconds
---------------------------------------------
Searching for author name of the middle book:
Hash Table: BEVERLY CLEARY
Time taken: 0 microseconds
Vector: BEVERLY CLEARY
Time taken: 9975 microseconds
---------------------------------------------
Searching for author name of the last book:
Hash Table: Christopher Biffle
Time taken: 4 microseconds
Vector: Christopher Biffle
Time taken: 35996 microseconds
---------------------------------------------
Searching for key not exist:
Hash Table: Not found
Time taken: 5 microseconds
Vector: Not found
Time taken: 43514 microseconds
```

Figure 7: Quadratic Probing Performance

- Time Complexity of Quadratic Probing Search: $O(n)$
- Time Complexity of Linear Seach Algorithm: $O(n)$

Though having the same theorectical time complexity, the actual time execution of Quadratic Probing Search is much faster than Linear Search Algorithm, especially in the worst case.

Chaining using Linked List

```
$ g++ experiment.cpp && ./a
Searching for author name of the first book:
Hash Table: Mark P. O. Morford
Time taken: 2 microseconds
Vector: Mark P. O. Morford
Time taken: 2 microseconds
---------------------------------------------
Searching for author name of the middle book:
Hash Table: BEVERLY CLEARY
Time taken: 0 microseconds
Vector: BEVERLY CLEARY
Time taken: 9888 microseconds
---------------------------------------------
Searching for author name of the last book:
Hash Table: Christopher Biffle
Time taken: 2 microseconds
Vector: Christopher Biffle
Time taken: 22707 microseconds
---------------------------------------------
Searching for key not exist:
Hash Table: Not found
Time taken: 16 microseconds
Vector: Not found
Time taken: 26897 microseconds
```

Figure 8: Chaining using Linked List Performance

- Time Complexity of Chaining Search using Linked List: $O(n)$
- Time Complexity of Linear Seach Algorithm: $O(n)$

Though having the same theorectical time complexity, the actual time execution of Chaining Search using Linked List is much faster than Linear Search Algorithm, especially in the worst case.

Chaining using AVL Tree

```
$ g++ experiment.cpp && ./a
Searching for author name of the first book:
Hash Table: Mark P. O. Morford
Time taken: 2 microseconds
Vector: Mark P. O. Morford
Time taken: 1 microseconds
--------------------------------------------
Searching for author name of the middle book:
Hash Table: BEVERLY CLEARY
Time taken: 1 microseconds
Vector: BEVERLY CLEARY
Time taken: 9460 microseconds
--------------------------------------------
Searching for author name of the last book:
Hash Table: Christopher Biffle
Time taken: 1 microseconds
Vector: Christopher Biffle
Time taken: 18354 microseconds
--------------------------------------------
Searching for key not exist:
Hash Table: Not found
Time taken: 2 microseconds
Vector: Not found
Time taken: 43215 microseconds
```

Figure 9: Chaining using AVL Tree Performance

- Time Complexity of Chaining Search using AVL Tree: $O(\log n)$
- Time Complexity of Linear Seach Algorithm: $O(n)$

Chaining Search using AVL Tree has a better time complexity than Linear Search Algorithm in theorectical, and the actual time also proves that.

Double Hashing

```
$ g++ experiment.cpp && ./a
Searching for author name of the first book:
Hash Table: Mark P. O. Morford
Time taken: 1 microseconds
Vector: Mark P. O. Morford
Time taken: 1 microseconds
---------------------------------------------
Searching for author name of the middle book:
Hash Table: BEVERLY CLEARY
Time taken: 0 microseconds
Vector: BEVERLY CLEARY
Time taken: 10373 microseconds
---------------------------------------------
Searching for author name of the last book:
Hash Table: Christopher Biffle
Time taken: 4 microseconds
Vector: Christopher Biffle
Time taken: 19550 microseconds
---------------------------------------------
Searching for key not exist:
Hash Table: Not found
Time taken: 3 microseconds
Vector: Not found
Time taken: 25592 microseconds
```

Figure 10: Double Hashing Performance

- Time Complexity of Double Hashing Search: $O(n)$
- Time Complexity of Linear Seach Algorithm: $O(n)$

Though having the same theorectical time complexity, the actual time execution of Double Hashing Search is much faster than Linear Search Algorithm, especially in the worst case.

4 Self-Evaluation

No.	Details	Score
1	Linear Probing	100%
2	Quadratic Probing	100%
3	Chaining using Linked List	100%
4	Chaining using AVL Tree	100%
5	Double Hashing	100%
6	Experiments	100%
7	Report	100%
	Total	**100%**

5 Exercise Feedback

What I learned

- An thorough understanding of the concept of Hashing and its applications.
- Different types of Hashing techniques and their implementations.
- The application of AVL Tree in Hashing.

What I found challenging

- At first, the implementation of AVL Tree in Hashing was quite difficult and complex. However, after many attempts, I was able to understand the concept and method clearly and implement it successfully.

6 References

- AVL Tree Data Structure by GeeksforGeeks.
- A Hybrid Chaining Model with AVL and Binary Search Tree to Enhance Search Speed in Hashing by International Journal of Hybrid Information Technology.
- GitHub Copilot.