CS300 – Spring 2022-2023 - Sabancı University

Homework #4 – Phonebook V2

Due: 15/05/2023, Monday, 22:00

Brief description

In this homework, you are going to implement a phonebook which will help the user to make several operations faster such as searching, adding, deleting contacts. Additionally, you will compare the speed of two different data structures for your phonebook architecture. For this purpose, you are required to store the contact information in the **Binary Search Tree (BST)** and **HashTable** and experience the implementation and performance differences between both data structures. For each unique contact, you will insert a node into both of your data structures, i.e. Binary Search Tree (BST) and Hash Table. Your tree implementations **MUST** be **template-based**.

The Contacts Input File

We will provide you with input files (phonebooks) which are lists of contacts of different sizes where each line is composed of:

firstName lastName phoneNumber city

There is only ONE space between each field. And, each person has only ONE first and last name. Therefore, in each line, there are exactly 4 strings. An example input file can be as follows:

```
Harold Stenseth +905547710771 Izmir
John Watson +905550292913 Izmir
Sharon Persaud +905593325178 Istanbul
Roger Jochim +905568022432 Ankara
Frank Thompson +905515328944 Istanbul
Norman Ho +905574240361 Ankara
Pamela Alexander +905597007856 Izmir
Rhonda Cashion +905531542095 Mugla
Ashley Tirado +905555055270 Ankara
Patricia Goode +905514483268 Izmir
Charles Smith +905538018232 Mugla
Charity Ruggiero +905567253292 Bursa
Elbert Womack +905551674361 Istanbul
Ashley Jepsen +905514534601 Izmir
Carolyn Johnson +905593164303 Mugla
Shirley Mickelson +905543846013 Bursa
```

You can assume that the given input file is valid.

BST (Binary Search Tree)

You may use the BST class that you have implemented in Homework3.

HashTable

You are going to implement a hash table data structure adopting any of the collision handling strategies, such as separate chaining, linear or quadratic probing, etc. You may use any strategy which may accelerate the HashTable operations.

You need to rehash every time when the load of the table exceeds the load factor that you have determined for your HashTable. For each rehash operation you need to print the internal information of the HashTable such as the previous capacity, current capacity, previous load, and the current load. The load will be calculated as (item count / capacity).

Additionally, you need to develop a hash function aiming to have collision free hash values for different WordItem structs.

Program Flow

Once you run your program, it should read the input file (***One of the samples given to you, ex:*** *PhoneBook-sample2.txt*) and load all its contacts into a **BST** and a **HashTable**. For each contact with a unique full name appearing in the file, you need to insert a new contact into the BST tree and the hashtable.

After you insert all the contacts, you need to get a query from the console which consists of a contact's full name consisting of two strings (HINT: You may use `getline(cin, line)`). This line might consist of more than one word. Then, **your program will display the contact's first name, last name, phone number and city information that matches the search full name.**

For the comparison of the speed of using different data structures, you need to perform the same operation, processing the query and building the result, without printing the result 500 times and taking the average timing. (The sample runs will explain it thoroughly).

For timing:

```
int k = 500;
auto start = std::chrono::high_resolution_clock::now();
for (int i = 0; i < k; i++)
{
      // FindContact(with BST);
}
auto BSTTime = std::chrono::duration_cast<std::chrono::nanoseconds>
                  (std::chrono::high_resolution_clock::now() - start);
cout << "\nTime: " << BSTTime.count() / k << "\n";
start = std::chrono::high_resolution_clock::now();
for (int i = 0; i < k; i++)
{
      // FindContact(with hashtable);
}
```

```
auto HTTime = std::chrono::duration_cast<std::chrono::nanoseconds>
                  (std::chrono::high_resolution_clock::now() - start);
cout << "\nTime: " << HTTime.count() / k << "\n";
```

Rules

- HashTable implementation **must be faster** than the BST implementation especially **in the sorted large phonebook files**.
- The resulting HashTable load factor (λ) **must be larger than 0.25** and **lower than 0.9** (0.25 < λ < 0.9) whatever the input size is. Otherwise your grade will be 0. Pick the upper bound of load value that gives you best results in terms of speed.

Output

While processing the input file, you need to print the information about the HashTable every time when the rehashing occurs, i.e. display the previous capacity, the previous load, current capacity, current load, and the number of items in the HashTable. After the processing is finished, you need to print the final item count in the HashTable and the current load factor λ.

After the user enters the contact's full name to be searched, you need to display the contact's first name, last name, phone number and city information.

Lastly you compute the time it takes to do the search with each data structure you've implemented (BST, Hash Table) and display them with the speedup ratio of these timing values compared to each other.

Sample runs

Sample Run 1

```
Enter the file name: PhoneBook-sample2.txt
rehashed...
previous table size:53, new table size: 107, current unique word count 49,
current load factor: 0.448598
rehashed...
previous table size:107, new table size: 223, current unique word count 98,
current load factor: 0.434978

After preprocessing, the unique word count is 175. Current load ratio is
0.784753
Enter name to search for: Helen Esparza

Searching for an item in the phonebook (BST) . . .
Phonebook: Searching for: (Helen Esparza)
====================================
Helen Esparza +905586926452 Ankara

Searching for an item in the phonebook (HashTable) . . .
Phonebook: Searching for: (Helen Esparza)
====================================
Helen Esparza +905586926452 Ankara

BST Search Time: 0.032
Hash Table Search Time: 0.032
Speed Up: 1
```

Sample Run 2

```
Enter the file name: PhoneBook-Sample4.txt
rehashed...
previous table size:53, new table size: 107, current unique word count 49,
current load factor: 0.448598
rehashed...
previous table size:107, new table size: 223, current unique word count 98,
current load factor: 0.434978
rehashed...
previous table size:223, new table size: 449, current unique word count 202,
current load factor: 0.447661
rehashed...
previous table size:449, new table size: 907, current unique word count 406,
current load factor: 0.446527
rehashed...
previous table size:907, new table size: 1823, current unique word count 818,
current load factor: 0.448162
rehashed...
previous table size:1823, new table size: 3659, current unique word count
1642, current load factor: 0.448483

After preprocessing, the unique word count is 2671. Current load ratio is
0.729981
Enter name to search for: Victoria Swafford

Searching for an item in the phonebook (BST) . . .
Phonebook: Searching for: (Victoria Swafford)
====================================
Victoria Swafford +905564660215 Istanbul

Searching for an item in the phonebook (HashTable) . . .
Phonebook: Searching for: (Victoria Swafford)
====================================
Victoria Swafford +905564660215 Istanbul

BST Search Time: 0.906
Hash Table Search Time: 0.062
Speed Up: 14.6129
```

Sample Run 3

```
Enter the file name: phonebook.txt

rehashed...

previous table size:53, new table size: 107, current unique word count 49,
current load factor: 0.448598

rehashed...

previous table size:107, new table size: 223, current unique word count 98,
current load factor: 0.434978

rehashed...

previous table size:223, new table size: 449, current unique word count 202,
current load factor: 0.447661

rehashed...

previous table size:449, new table size: 907, current unique word count 406,
current load factor: 0.446527

rehashed...

previous table size:907, new table size: 1823, current unique word count 818,
current load factor: 0.448162

rehashed...

previous table size:1823, new table size: 3659, current unique word count
1642, current load factor: 0.448483

rehashed...

previous table size:3659, new table size: 7321, current unique word count
3295, current load factor: 0.449939


After preprocessing, the contact count is 5829. Current load ratio is 0.796203
Enter name to search for: William Mitchell

Searching for an item in the phonebook (BST) . . .
Phonebook: Searching for: (William Mitchell)
====================================
William Mitchell +905597139795 Ankara

Searching for an item in the phonebook (HashTable) . . .
Phonebook: Searching for: (William Mitchell)
====================================
William Mitchell +905597139795 Ankara

BST Search Time: 0.064
Hash Table Search Time: 0.062
Speed Up: 1.03226
```

General Rules and Guidelines about Homeworks

The following rules and guidelines will be applicable to all homeworks, unless otherwise noted.

How to get help?

You may ask questions to TAs (Teaching Assistants) and LAs (Learning Assistants) of CS300. Office hours of TAs can be found at SUCourse. Recitations will partially be dedicated to clarify the issues related to homework, so it is to your benefit to attend recitations.

What and Where to Submit

Please see the detailed instructions below/in the next page. The submission steps will get natural/easy for later homeworks.

Grading

Your programs should follow the guidelines about input and output order; moreover, you should also use the exact same prompts as given in the Sample Runs. Otherwise the grading process will fail for your homework, and you may get a zero, or in the best scenario you will lose points.

Grading:

- ❑ Late penalty is 10% off the full grade and only one late day is allowed.
- ❑ **Having a correct program is necessary, but not sufficient to get the full grade. Comments, indentation, meaningful and understandable identifier names, informative introduction and prompts, and especially proper use of required functions, unnecessarily long programs (which is bad) and unnecessary code duplications will also affect your grade.**
- ❑ Please submit your own work only (even if not working). It is easy to find "similar" programs!
- ❑ For detailed rules and course policy on plagiarism, please check out http://myweb.sabanciuniv.edu/gulsend/courses/cs201/plagiarism/

Plagiarism will not be tolerated!

Grade announcements: Grades will be posted in SUCourse, and you will get an Announcement at the same time. You will find the grading policy and test cases in that announcement.

Grade objections: Since we will grade your homeworks with a demo session, there will be very likely no further objection to your grade once determined during the demo.

What and where to submit (IMPORTANT)

Submission guidelines are below. Most parts of the grading process are automatic. Students are expected to strictly follow these guidelines in order to have a smooth grading process. If you do not follow these guidelines, depending on the severity of the problem created during the grading process, 5 or more penalty points are to be deducted from the grade.

Add your name to the program: It is a good practice to write your name and last name somewhere in the beginning program (as a comment line of course).

Name your submission file:

- ❑ Use only English alphabet letters, digits or underscore in the file names. Do not use blank, Turkish characters or any other special symbols or characters.
- ❑ Name your cpp file that contains your program as follows.
  "**SUCourseUserName_yourLastname_yourName_HWnumber.cpp**"
- ❑ Your SUCourse user name is actually your SUNet username which is used for checking sabanciuniv e-mails. Do NOT use any spaces, non-ASCII and Turkish characters in the file name. For example, if your SUCourse user name is cago, name is Çağlayan, and last name is Özbugsızkodyazaroğlu, then the file name must be:
  cago_ozbugsizkodyazaroglu_caglayan_hw4.cpp
- ❑ Do not add any other character or phrase to the file name.
- ❑ Make sure that this file is the latest version of your homework program.
- ❑ You need to submit ALL .cpp and .h files including the data structure files in addition to your main.cpp in your VS solution.
- ❑ The name of the main cpp file should be as follows.
  "**SUCourseUserName_yourLastname_yourName_HWnumber.cpp**"
  For example zubosman_Osmanoglu_Zubeyir_hw3.cpp is a valid name, but hw2_hoz_HasanOz.cpp, HasanOzHoz.cpp are NOT valid names.

Submission:

Submit via SUCourse ONLY! You will receive no credits if you submit by other means (e-mail, paper, etc.).

Successful submission is one of the requirements of the homework. If, for some reason, you cannot successfully submit your homework and we cannot grade it, your grade will be 0.

Good Luck!
Gülşen Demiröz & Arca Özkan