CS300 – Spring 2018-2019 - Sabancı University

Homework #3 – Dictionary v2

Due April 20 Saturday at 22:00

Brief Description

In this homework, you will write an English-to-Turkish dictionary as in the second homework, this time by using hash tables instead of binary search trees. Additionally, you will compare the speed of different data structures for your dictionary architecture. The dictionary will have hundreds of words and their corresponding meanings. In this case, you will keep a hash table of word entries to make translations faster than the other data structures.

First, you are going to preprocess the dictionary file provided to you. The dictionary file will be in the same format as in HW2. For each word in the dictionary file and its corresponding meanings will be inserted as a node into both of your data structures, i.e. Binary Search Tree (BST) and Hash Table. Hence, you need to implement a templated BST and Hash Table. Using both of the data structures, you need to print out the result of cumulative queries into an output file. In addition, you will compare the speed of them by comparing total elapsed query time. After comparing the speed of them, you will show how much speed up you obtained by using the hash table. You may use the same data structures and classes you used in HW2.

BST (Binary Search Tree)**:**
This will be the class that you have implemented in HW2.

HashTable:

You are going to implement a hash table data structure adopting **separate chaining** as the collision handling strategy. You are allowed to use only vector, linked list or binary search tree (BST) data structure to establish the separate chaining mechanism.

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 and the current load. The load will be calculated as (item count / capacity). Note that in the sample run given below, the <*your current load factor*> needs to be filled with the load factor you have during the execution of your program.

Additionally, you need to develop a hash function aiming to have collision-free (as much as possible) hash values for different words.

Program Flow

Program starts with reading the “dict.txt” file. Each line in the file provides an English word with its Turkish translation as in Homework 2. Note that there might be empty lines in this file, so you need to skip them. The words in each line are separated by spaces. There is a possibility of having more than one space between each word, and also there may be some words without any translations, and also the program is not case sensitive. Note that, this “dict.txt” file is given to you ONLY for debugging purposes. We might use different files to grade your submissions.

For each unique word appearing in the dictionary file, you need to insert a new node into the tree and the hash table. The initial size of hash table will be 53. Each rehash operation is performed in a way that the next table size of hash table will be the next prime number that comes right after the current table size is doubled. For example, if initial table size is 53, the next table size become 107 since 107 is the first prime number after 106=53x2. While building your hash table from the given dictionary file, you need to print the information about the HashTable every time when the rehashing occurs, i.e. display the **previous capacity**, **the current capacity**, **the current load**, and **the number of items in the HashTable**. Once the preprocessing is finished, you need to print the final item count in the HashTable and the current load factor λ.

After building the BST and the Hash Table, your program preprocesses the query file and stores all queries in a linear data structure (vector/array, etc.). The query file is composed of words on each line. This line might consist of extra whitespace other than the word itself. You need to query each word in the query file. Then, you need to output the corresponding query results for that query file into an output file. Assume that all queries in the query file are valid queries and all search operations will be successful. Each line in the output file will be composed of the queried word itself and its corresponding translation in a tab separated format. Output file names for Hash Table and BST will be ht_results.txt and bst_results.txt respectively.

Then, the program computes the time it takes to perform all queries in the file with each data structure you’ve implemented so far (BST, Hash Table) and displays them with elapsed time of all queries, the average query time, the speedup ratio of these timing values compared to each other and total elapsed time for k query where k = {1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096}.

Lastly, you will prepare a report includi***ng the console output of program and a graph (elapsed time vs k graph)*** comparing elapsed time of both data structures for N to 4096N queries where N is your total number of queries. So you will basically run queries 1, 2, 4, ..., 4096 times for the whole query file and output the elapsed time to the console.

It's important to show that your HashTable implementation **must be faster** than the BST implementation. Otherwise, your grade will be 0.

For timing (in nanoseconds)**:**

```
auto start = std::chrono::high_resolution_clock::now();
// QueryDocument;
auto time = (std::chrono::high_resolution_clock::now() - start).count();
cout << "Elapsed time: " << time.count()  << "\n";
```

Sample Run

Console output:

```
Building a binary tree for dict.txt...
Building a hash table for dict.txt...
rehashed...
previous table size:53, new table size: 107, current unique word count 27,
current load factor: 0.252336
rehashed...
previous table size:107, new table size: 223, current unique word count 54,
current load factor: 0.242152
rehashed...
previous table size:223, new table size: 449, current unique word count 112,
current load factor: 0.249443
rehashed...
previous table size:449, new table size: 907, current unique word count 225,
current load factor: 0.248071
rehashed...
previous table size:907, new table size: 1823, current unique word count 454,
current load factor: 0.24904
rehashed...
previous table size:1823, new table size: 3659, current unique word count 912,
current load factor: 0.249248
After preprocessing, the unique word count is 995. Current load ratio is
0.271932
Running queries in query1.txt...

***********************************************
Benchmark results for Binary Search Tree (BST):
***********************************************
+ Elapsed time: 475745 ns
+ Average query time: 18297 ns

***********************************************
Benchmark results for Hash Table:
***********************************************
+ Elapsed time: 303001 ns
+ Average query time: 11653 ns
```

+ Speed up: 1.57011x

Time measurements in ns (N, 4096N):
****
bst

N	time
1	1249290
2	11735683
4	60159879
8	251834657
16	963508445
32	3560277059
64	13898084110
128	52331268292
256	208132422653
512	809732504757
1024	3594116584677
2048	16463567626447
4096	62667030494729

ht

N	time
1	650836
2	4996776
4	27548487
8	119427742
16	468319728
32	1815657336
64	6929926633
128	27282594450
256	110339318856
512	477585853385
1024	2096513776691
2048	8581880366057
4096	35488726870271

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) of CS300. Office hours of TAs are at the syllabus. Recitations will partially be dedicated to clarifying 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 and Objections

Careful about the semi-automatic grading: Your programs will be graded using a semi-automated system. Therefore, you should follow the guidelines about input and output order; moreover, you should also use same prompts as given in the Sample Runs. Otherwise semi-automated 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 program (which is bad) and unnecessary code duplications will also affect your grade.**
- Please submit your own work only (even if it is not working). It is really easy to find out "similar" programs!
- For detailed rules and course policy on plagiarism, please check out http://myweb.sabanciuniv.edu/gulsend/courses/cs201/plagiarism/

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: It is your right to object to your grade if you think there is a problem, but before making an objection please try the steps below and if you still think there is a problem, contact the TA that graded your homework from the email address provided in the comment section of your announced homework grade or attend the specified objection hour in your grade announcement.

- Check the comment section in the homework tab to see the problem with your homework.
- Download the .zip file you submitted to SUCourse and try to compile it.
- Check the test cases in the announcement and try them with your code.

- Compare your results with the given results in the announcement.

What and where to submit (IMPORTANT)

Submissions 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 user name 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_hw3.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 in addition to your main.cpp in your VS solution. Compress all your necessary files using WINZIP or WINRAR programs. Please use "**zip**" compression. "rar" or another compression mechanism is NOT allowed.
- Check that your compressed file opens up correctly and it contains your **cpp** file. You will receive no credits if your zip file does not expand or it does not contain the correct file.
- The naming convention of the zip file is the same as the cpp file (except the extension of the file of course). The name of the zip file should be as follows.

  "**SUCourseUserName_yourLastname_yourName_HWnumber.zip**"

  For example; zubzipler_Zipleroglu_Zubeyir_hw3.zip is a valid name, but

  hw1_hoz_HasanOz.zip, HasanOzHoz.zip are NOT valid names.

Submission:

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

1. Click on "Assignments" at CS300 SUCourse.
2. Click Homework 3 in the assignments list.
3. Click on "Add Attachments" button.
4. Click on "Browse" button and select the zip file that you generated.
5. Now, you have to see your zip file in the "Items to attach" list.
6. Click on "Continue" button.

7. Click on "Submit" button. We cannot see your homework if you do not perform this step even if you upload your file.

Resubmission:

- After submission, you will be able to take your homework back and resubmit. In order to resubmit, follow the following steps.
    1. Click on "Assignments" at CS300 SUCourse.
    2. Click Homework 3 in the assignments list.
    3. Click on "Re-submit" button.
    4. Click on "Add/remove Attachments" button
    5. Remove the existing zip file by clicking on "remove" link. This step is very important. If you don't delete the old zip file, we get both files and the old one may be graded.
    6. Click on "Browse" button and select the new zip file that you want to resubmit.
    7. Now, you have to see your new zip file in the "Items to attach" list.
    8. Click on "Continue" button.
    9. Click on "Submit" button. We cannot see your homework if you do not perform this step even if you upload your file.

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!
Oğuz Özsaygın and Gülşen Demiröz