ITU Computer Engineering Department
BLG 223E Data Structures 23-24 Fall
Homework 5 : Compherensive Comparison

January 8, 2024

1 Introduction

In the final assignment, you will conduct experiments on your in-term works, do a comparison using the results that you get, and prepare a report including these experiment results and your comments on the results.

2 Assignment

2.1 Data Structures

Firstly, let's remember the data structures that you have implemented and were responsible in the previous assignments.

- File I/O (Homework 1)
- Array (Homework 1)
- Linked List (Homework 2)
- STL Vector (Homework 2)
- STL List (Homework 2)
- Binary Search Tree (Homework 3)
- STL Map (Homework 3)
- Skip List (Homework 4)

Using these data structures, you must **measure the execution times** of the basic operations. Instructions on the manner of time measuring were given in the previous assignments, also mentioned in Section 5.

2.2 Operations

Here is the list of basic operations of the structures.

- **insert** : Create a node randomly and insert it into the structure
- **search** : Select a search key randomly and find it in the structure. The key that you are looking for might not be in the structure.
- **remove** : Select a random key, find it in the structure and delete the node. Again, the key that you are looking for might not be in the structure.

You will have three groups where you will compare the execution times of the given operations per each data structure for 10 datasets, which differ in size. After you measure the times, draw a plot showing the results. Remember, since you will be conducting experiments on 10 different datasets, **there will be 10 points in the x-axis of the plot.**

Prepare your plot in a way that all the structures can be compared in terms of an operation for every group, meaning that there will be a graph for every operation for each group. **In total, there will be 3 (# of operations) x 3 (# of groups) = 9 different graphs** for the operations *insert*, *search*, *remove*

2.3 Groups

- Group 1
  - File I/O (Homework 1)
  - Array (Homework 1)
  - Linked List (Homework 2)
  - Binary Search Tree (Homework 3)
  - Skip List (Homework 4)
- Group 2
  - Array (Homework 1)
  - Linked List (Homework 2)
  - STL Vector (Homework 2)
  - STL List (Homework 2)
- Group 3
  - Binary Search Tree (Homework 3)
  - STL Map (Homework 3)
  - Skip List (Homework 4)

2.4 Example

Figure 1 shows how your graphs should look like. You will have size of the datasets on x-axis and measured time on y-axis. Note that the numbers may differ for your measurements.

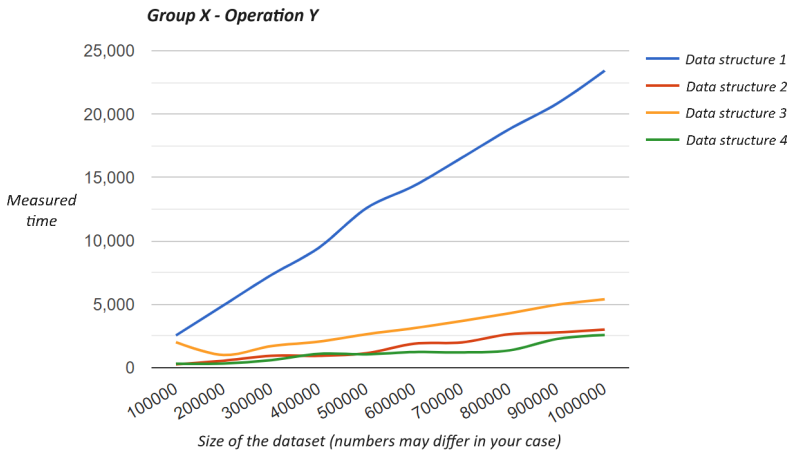

Figure 1: Group X - Operation Y

Name your graphs in the form of "Group X - Operation Y" like **"Group 1 - ADD"**, **"Group 2 - DELETE"** etc.

After you obtained the results, **comment on the outcomes of the experiments** with your own words. If you do not have a functioning code in your hand for a specific data structure, mention those in the related parts and include your prediction on how that data structure would behave under those circumstances. However do not include your 'predicted behaviour' in your plots.

If you did not implement the desired code before but you want to get full points from this homework, then you are free to complete your missing parts for the previous homeworks and include the results in the plot as well.

Be aware that in case you implement a structure that you have not submitted in the previous assignments, it will not affect your previous grades.

3 Datasets

You are responsible from generating your own datasets, you can do this in any language you are comfortable with. You can take the provided datasets for your homeworks as a base if you want. The important thing is, deciding on

the dataset sizes. Determine which dataset size is needed to show comparison between data structures. For example, you can try with the values in range [1k, 5k, 10k, ..., 100k, 250k]. Or, just go with the values that provides meaningful results for you.

4 Things To Pay Attention

- It is always better to perform a test multiple times, so you will be able to see the average performance of your structure. For example, you should perform *LinkedList.insert(...)* five times and calculate the average of these results.
- There will be no operations file, so you can either manually test the specific operation or write a simple main program, in which you call the corresponding operation of the structure. Note that we do not want you to make measurements on different set of operations, we want you to measure operations that are mentioned in the Section 2, for the data structures that are also mentioned in the Section 2.
- [1] performs similar approach to what we want from you for this homework so you can take a look at that website.

5 Execution Time Measuring

Measure the execution times to the terminal to compare each solution. You can use the following code and time.h library:

```
#include <time.h>   // library
clock_t start = clock();   // start to measure
//.... code for measuring
clock_t end = clock();   // finish measure

//measurement time in milliseconds
(double)(end - start) * 1000 / CLOCKS_PER_SEC;
```

6 Submission Rules

- Make sure you write your name and student number in the report.
- You will submit **a single PDF file** and **all the source code of your implementations**.
- DO NOT SHARE ANY CODE OR TEXT that can be used as a part of an assignment.

- Only electronic submissions through Ninova will be accepted no later than deadline.
- You may discuss the problems at an abstract level with your classmates, but you should not share or copy code from your classmates or from the Internet. You should submit your **own, individual homework.**
- Academic dishonesty, including cheating, plagiarism, and direct copying, is unacceptable.
- If you have any question about the homework, you can send e-mail to Batuhan Can (canb18@itu.edu.tr).
- Note that **YOUR CODES AND REPORTS WILL BE CHECKED WITH THE PLAGIARISM TOOLS!**

References

[1] B. Wicht, "C++ benchmark – std::vector vs std::list vs std::deque," Dec 2012. [Online]. Available: https://baptiste-wicht.com/posts/2012/12/cpp-benchmark-vector-list-deque.html