Assignment 04
Sorting numbers using Binary Search tree and Quick sort

DS286.Aug16 Data Structures and Programming

October 13, 2016

Submission is due on or before **Wednesday, October 26, 2016, 11:59pm IST**.
The assignment carries **75 points**, which is 7.5% of the course weightage.

1 Question

This assignment requires you to perform sorting of a large list of numbers using an inorder traversal of a *Binary Search Tree (BST)* and using *Quick sort*, and compare their performances for input lists of different sizes. You will also learn to use *C++ Standard Template Library (STL)* using `vector`, that is like a `List` abstract data type.

You are given the code for the `main.cpp` method and header files for `BST.h` and `Quick.h`. Your first task will be to complete the implementation of the various methods in these header files in `BST.cpp` and `Quick.cpp` files, respectively. Note that *all the methods* of BST and QuickSort header files **must** be implemented even if only some of them are used in the sorting program. We may test those other methods separately. You can add your own *private* methods in the `BST.cpp` and `Quick.cpp` files but **must not change** the signature of the *public* methods in the header files. You **must not change** `main.cpp` either.

2 Program Outline

You are given two header files `BST.h` and `Quick.h`, and the program file `main.cpp` with the testing harness. You are also given three sample input files, `input1.txt`, `input2.txt` and `input3.txt`. Your first task is to implement a Binary Search tree data structure in the file `BST.cpp` with all methods present in `BST.h`. Also you are required to implement Quick sort in `Quick.cpp` with all methods present in `Quick.h`.

1. The given main file `main.cpp` accepts a file name as the command line argument. The file will have some distinct random numbers to be sorted.

2. The `main()` function reads the file line by line and populates a `vector` with the numbers.
3. Then the methods `BSTsort()` and `QUICKsort()` are called to sort the numbers stored in the `vector` and populates another `vector` with the sorted numbers.
4. At the end, the `main()` reports the time taken for sorting by BST and Quick sort.
5. It also performs sanity checks on whether the sorted lists returned by BST and Quick sort are indeed sorted or not.
6. Assume that the input file will only have distinct numbers. There will be no duplicates.

You are also required to submit a *short report* ($\leq$ 3 pages) that discusses the computational and storage complexities of these two sorting approaches. Further, you should also validate if these complexities match the actual sorting time taken by the two approaches. For this, you will run experiments using different appropriate input sizes (including the three input files already given). You should submit a table with three columns: the input size, the time taken by BSTsort and time taken by QUICKsort. These are generated by the given `main()` method. You should also plot these in a graph where the X Axis is the input size, the Y axis is the observed and expected times for the two approaches. Describe the system used to generate these results (CPU, memory). You should also analyze the results, and also discuss the pros and cons of these implementation approaches in the report.

You have to write the `Makefile` for this assignment yourself. It should produce an executable named `Sorting.out` that corresponds to the `main.cpp` that is provided. Note the camel casing.

3 Sample inputs and results

```
./Sorting.out input1.txt
bst,10000,12.345
quick,10000,1.234
Sorting is successful :-)
```

4 Submission Instruction

Please follow these instructions carefully. We use automated scripts for evaluation. So a failure to follow these instructions will mean that your submission will not be evaluated.

- Only write your code in the two files: `BST.cpp` and `Quick.cpp`. These program files should implement the methods from the corresponding header files.
- Do not modify the other files that are provided.
- Place all your files including source file, executable file and `Makefile`, in a single folder whose name is determined as follows. My full name is "Prateeksha Varshney" where "Prateeksha" is my first name, so the folder name should be `04Prateeksha` for my submission. Please note the capitalization of first letter of the first name. The final contents of the folder would be as follows:

```
04Prateeksha
|- main.cpp
|- BST.h
|- BST.cpp
|- Quick.h
|- Quick.cpp
|- Sorting.out
|- Makefile
|- report.pdf
```

- This folder should be compressed using the `tar` program as follows:
  `tar -cvf 04Prateeksha.tar 04Prateeksha/`
  Note: Any other compression format *will not be accepted* and will be treated as no submission. Its your responsibility to check if the file can be properly uncompressed and all files inside are intact.
- Send a separate mail to the TA's email address `prateeksha@grads.cds.iisc.ac.in` with the subject line `DS286.Aug16.A04`. Do not write anything more or less in the subject line. Do add any text in the body. Do not send the assignment as a reply-email to any other mail.
- **Only one submission will be accepted**. If multiple emails or files are received, **only the first one** will be taken as the submission. Only the submission received **before the deadline** will be accepted.
- Use only the C++ language for completing this assignment. Make sure the code compiles and executes correctly on the head node of the `turing.cds.iisc.ac.in` server using `g++` command. You will need to pass the `-std=c++11` flag to the compiler to use STL's `vector`. The code will be compiled and tested on this machine during evaluation.
- Indent/format the code and add inline comments describing that the code is supposed to do. This will help you debug better, and give us an insight on the logic you are using.

- It is your responsibility to remove all debug statements you may have added during development and testing your code. The evaluation of your submission is done using automated scripts, and your console output will be tested against a predetermined correct output. If the outputs do not match exactly, it will be taken as wrong output.

5 Ethics

You should not get assistance from other students or external sources in directly solving the assignment. Getting help on generic C++ and data structures concepts, e.g., on using lists, strings, libraries, compilation, etc. is accepted. You are encouraged to post questions to the course mailing list so that the TA, instructors or other students can respond. This also ensures you do not have an unfair advantage/disadvantage over other students. If you have received assistance from other sources, send a *separate email to the Instructor and the TA* disclosing the external sources and type of support received.

By making a submission, you are asserting that all code that you submit was designed and developed by you. Do NOT copy and paste code from anyone else! All code will be verified using a plagiarism checker, and *penalties will be imposed* if plagiarism is found from unattributed sources.