CompSci 261P Project #2: Binary trees and their variants

In this project you are to implement several different algorithms for storing, querying, and modifying ordered data, and perform a empirical comparative analysis of their running times. The data structures to be implemented are:

1. A standard (unoptimized) binary tree
2. At least one form of balanced binary tree (e.g., AVL tree, red-black tree, weak AVL tree)
3. At least one alternative to a balanced binary tree (e.g., B-tree, skip-list, treap, splay tree)
4. At least one additional algorithm. It could be a second algorithm from either #2 or #3.

For each data structure you implement, you must at minimum implement the following operations:

- Create (i.e., create a data structure with no elements).
- Search
- Insert
- Delete

Feel free to implement other operations (for example, merge) on data structures that support them.

For the empirical analysis, for each implemented data structure on sequences of Insert, Delete, and Search operations. Run the algorithms on a variety of test cases to determine the relative efficiencies of the algorithms as functions of various parameters, such as the number of elements, or the relative frequence of insert, search and delete operations. How do your results compare with theoretical predictions of worst-case and amortized-time? You are encouraged to consider other factors as well. For example, if you generate your test cases randomly, you how do your choice of probability distribution and the distribution parameters affect your results?

Deliverables

1. A ZIP file that includes the following:
    - Source code (C, C++, Java, or Python) of your program that implements and experimentally tests the running times of the data structures and their operations. For implementing the algorithms, the only libraries you should use are dynamic arrays and linked lists.
    - The data used for any test cases or experiments
2. A Report (in PDF) that includes the following:
    - A pseudo-code/English description of each of the algorithms that you implemented and a theoretical analysis of the running times of the algorithms. This portion of the report should include pseudo-code summaries of the algorithms. It should also include reasons why you chose the particular additional algorithm(s) that you chose.
    - A comparative experimental analysis of the algorithms and conclusions drawn from the experiments. This portion of the report should include visual plots showing comparitive empirical performance of the algorithms for various types of inputs.

Project Submission

Submit your project using the EEE Dropbox system. Your submission should go into the following two drop boxes:

- CS 261P Lab 2 Report: Submit your Report (a single PDF file) here
- CS 261P Lab 2 Pgms: Submit a single zip file containing all your code and your data here.

Grading

- Code and documentation: 40%
- Experimental analysis: 60%

Final Note

- **Test your code throughly**. If we find bugs in your code, then not only will you lose points for the code and documentation, it calls into question your experimental analysis.