COT5405: Project Description

Due: December 4, 2020, 11:59pm.
Submit through `http://canvas.fsu.edu`
You are not allowed to collaborate on this project.

1 Implementations

All implementations may be done in the language of your choice; but they must run on `linprog.cs.fsu.edu`.

1. Implement the following sorting algorithms.

   - `InsertionSort`
   - `MergeSort`
   - `HeapSort`
   - `QuickSort`

   These algorithms were described in class (pseudocode for each is available in the textbook). Each algorithm should have its own standalone program that accepts, as a command line argument, the name of an input file. For example,

   ```
   HeapSort <input file>
   ```

   where the input file contains one instance per line. Each instance i is a set of n_i distinct integers between 0 and $2^{32} - 1$ represented in text format; each integer is separated by a space. For each instance, your algorithm produces one line of output, consisting of the correctly sorted instance.

2. Implement a standard BinarySearchTree. Also implement a Red-Black tree that supports insertion only. Extra credit: implement red-black tree deletion as well.

2 Evaluation and Report

Write up a final report that includes documentation for all of your implementations, including:

- How the code is organized into different files. Instructions for how to compile your code.

- How data is represented internally. For sorting algorithms, discuss if each implementation sorts in-place or not. For the red-black tree, discuss how nodes in the tree are represented.
- Describe any utility procedures implemented and their parameters.

Perform an empirical evaluation of your implementations as follows:

- Test the sorting algorithms on inputs of size $n \in \{10, 10^2, 10^3, 10^4, 10^5\}$. Each input should consider of n integers $0 \leq i \leq 2^{32} - 1$ randomly permuted. In your report, give the average and sample standard deviation of the wall-clock runtime of each your algorithms on 20 randomly permuted instances of each size n. If an algorithm exceeds 2 hours on an input size n, its results may be omitted on that size. In your report, describe the results qualitatively and compare the different algorithms.
- Evaluate your red-black tree by randomly inserting nodes with key value uniformly randomly chosen in $[1, 1000]$. Plot the height of your tree after each insertion. Compare with the height of your standard BinarySearchTree implementation on the same sequence of insertions.

3 Assessment

Your project submission will be scored as follows.

- (10 points) Your code compiles and correctly runs on a set of test instances.
- (10 points) Your project report contains all of the items discussed above.
- (5 points, extra credit) You correctly implemented red-black tree deletion.
- **You must submit your own code.** This is an individual project; collaborations are not allowed on this project. Submissions will be checked using MOSS, a measurement of software similarity.

4 Submission

Submit one ZIP file, containing all code and the PDF report. This ZIP file must be uploaded to `canvas.fsu.edu` by the project deadline.