Assignment 1 (Due October 20 2019)

You are encouraged to work in groups (but not more than 2 persons in a group).Show all your computation to get full credit. Please put comments in your code. I will deduct points for uncommented codes. You can write the programs in any language you choose, so long as we can run it. If you use code, other than your own, cite the source.

Part 1 Sorting Algorithms (50)

Create sets of 1 Million integers with the following characteristics;

- Sets where no numbers repeat
- Sets where the range of numbers is 1% of the array size
- Sets where no numbers repeat and each integer is of size 20

For each of these arrays compare the performance of the following algorithms;

- Quicksort
- Quicksort, where you change to Insertion sort when the partition size is small.
- Radixsort
- Timsort (https://en.wikipedia.org/wiki/Timsort)

To compare the performance, these are the steps to do;

- Create the array
- Run the algorithm
- Tabulate the time. Only count the time for sorting, not for creating the array.

Repeat this step 10 times. Each time re-create the array, so that you are running the algorithm on a possibly new array. Create a table with the rows being the algorithms (4 rows) and the columns representing the runs of each different inputs (10*3=30 columns) (20)

Write a short report of your observations about the performance of the algorithm along with graphs to highlight your results (10)

Explain with figures how the Timsort algorithm works (20)

Part 2 Red Black Tree (50)

Create sets of 1 Million integer, where no integer is repeated. Insert these numbers to an (i) AVL tree and (ii) R-B tree. Compute the time taken to insert all the numbers. Repeat the experiment 10 times, each time regenerating the set. In a table report (a) the time taken to complete the insertion, (b) the height of the tree, (c) the black height of the R-B Tree (15).

Using a random number generator, select 10% of the numbers in the trees and delete them. Repeat the experiment 10 times. Report your answers in a tables (15).

Consider the case where in an R-B tree, a red node can have a red child, if its parent is black in color. All other constraints of the R-B tree are maintained. Discuss whether (or whether not) the tree will still have $O(logn)$ height. Discuss a insertion scheme that will maintain the height and the constraints (20).