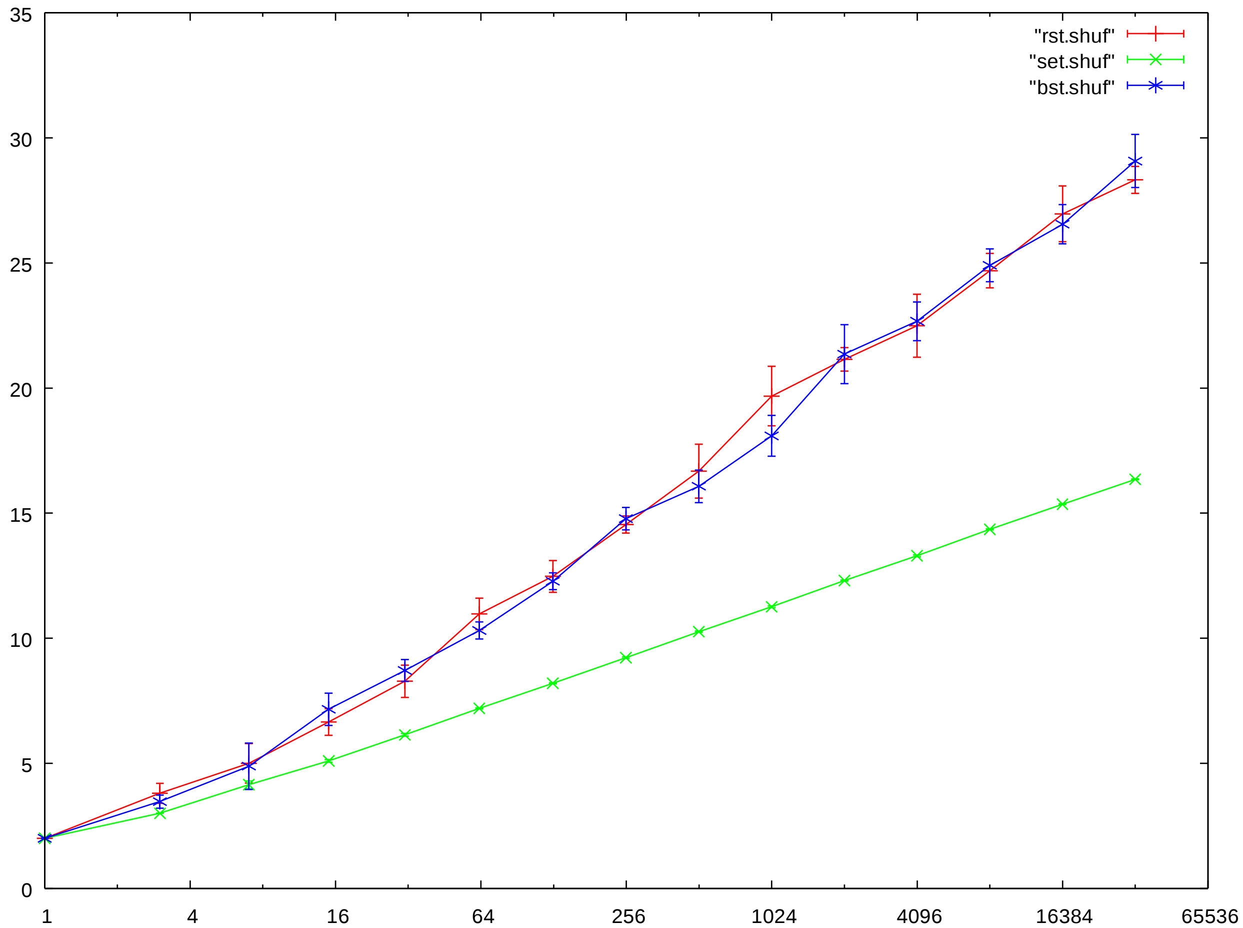
README

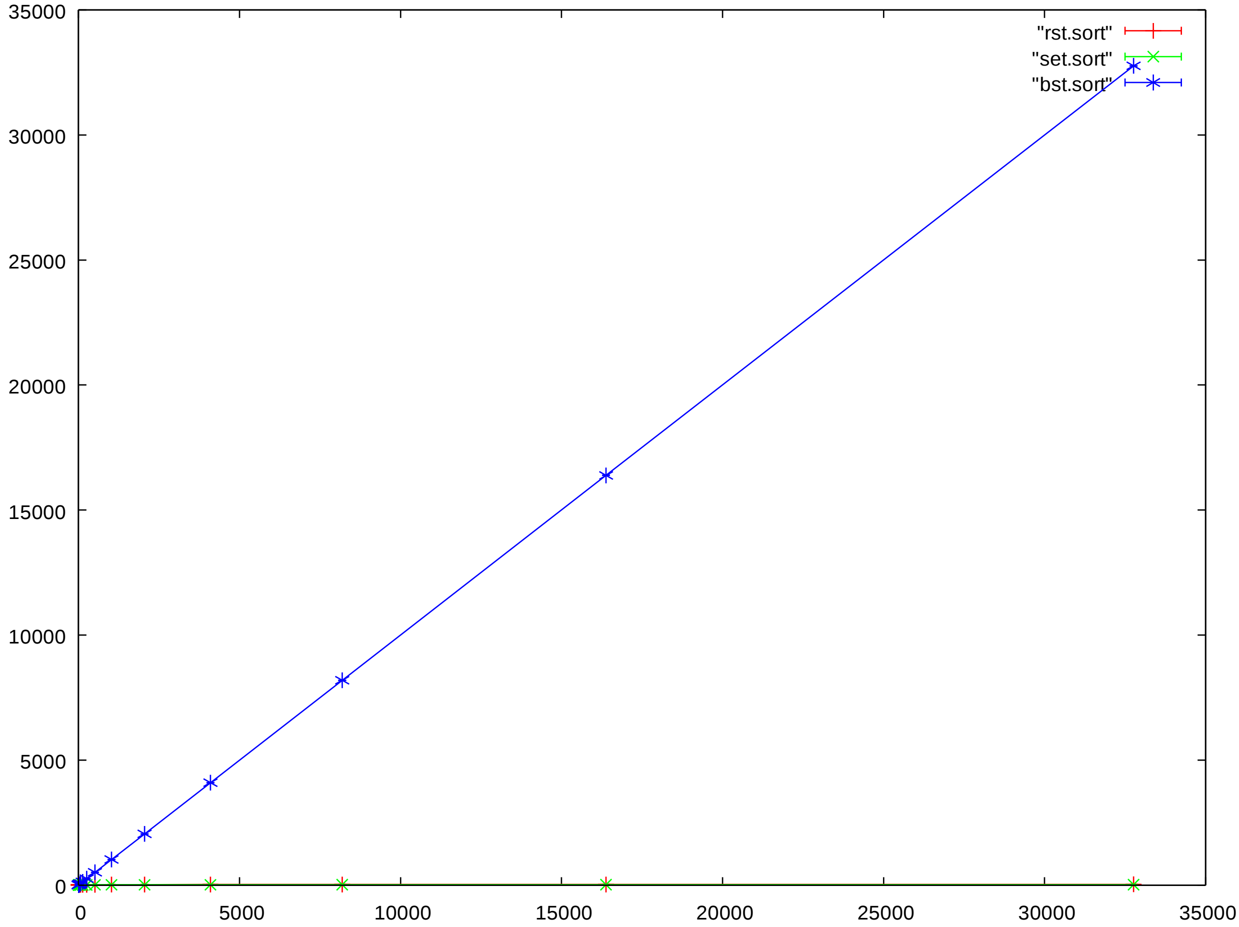
Name: Anish Narsian and Dhanuk Withana



We compared 3 data Structures including Randomized Search Trees(RST), Binary Search Trees(BST) and Red and Black Trees(SET).

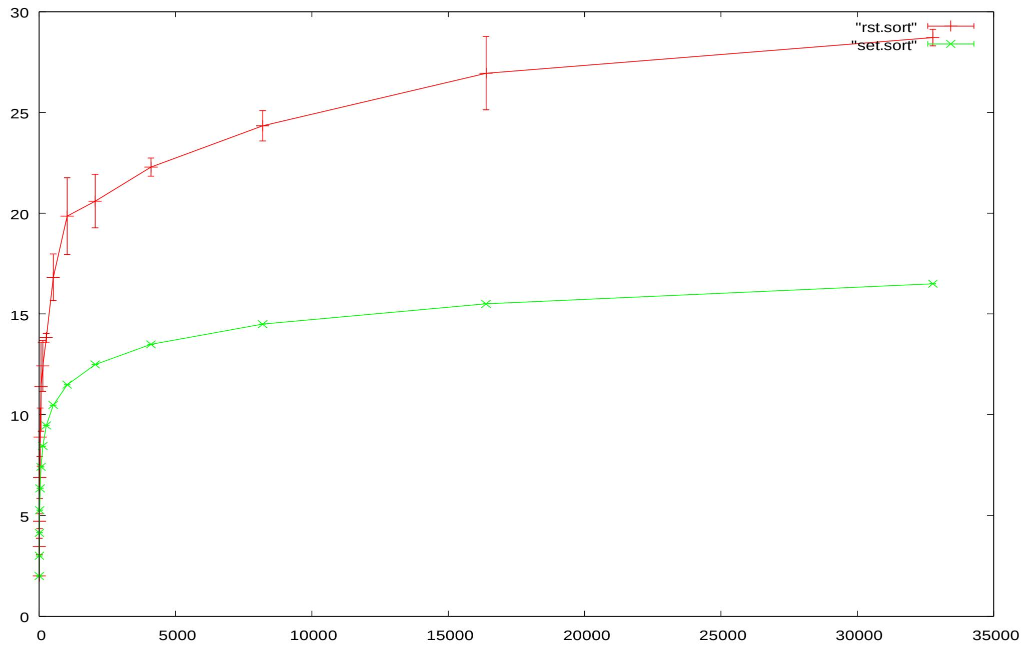
The methodology: We input k elements as on the x axis and run this N times. The average is got by dividing the amount of time taken to input k elements and dividing it by N.

The above graph compares the result for the find function on a shuffled set of data. The above plot is done on a log scale. With the straight line nature of above graph we see that these have a time complexity of O(logn) for the find for each.

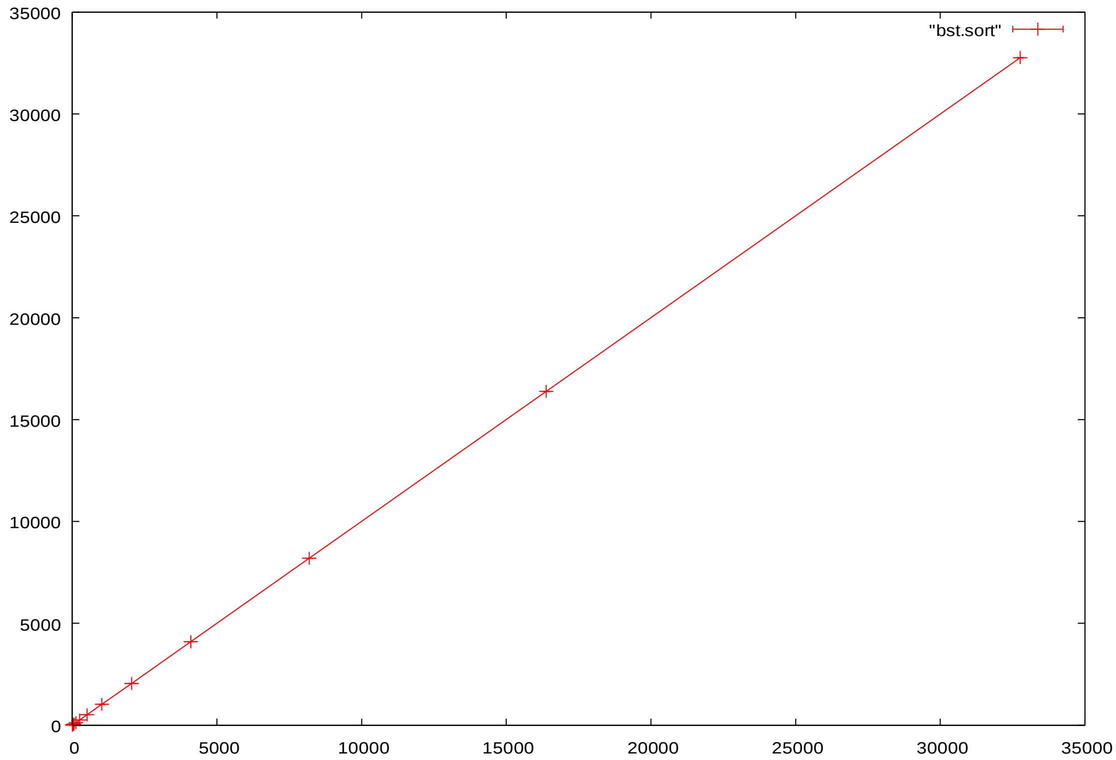


This is the graph for inserting sorted data and then finding it. This is based on a normal, non-log scale. The graphs below shows separate graphs which show the complexity for each of the RST, BST and SET. The fact that BST is O(N) makes showing all of them on the same graph difficult.

Entering a sorted list means that a one sided tree(like a linked list will be created. Thus O(n) complexity makes sense. On the other hand RST and Red and Black trees have keys and coloring respectively that ensures some amount of sorting and thus a better complexity. Red and Black trees are self balancing hence they must create a tree and ensure O(logn).



This is the RST and set for sorted. This is because O(n) complexity on previous graph prevents correct visualization for these two.



This is the BST for sorted. It has O(n).