HPC Techniques Assignment 7

Michael Tierney 11348436

April 2015

Performance

Binary trees if perfectly balanced have a depth of $ceil(log_2(N+1))$ where N is the number of data points. This balance is dependent on the insertion regime. Two regimes, random and ordered are compared in figure 1. It's clear from this plot that the random regime results in a much more balanced tree. The tree ,when inserted into in an ordered fashion, is just a linked list with depth equal to N while the random insert should result in a depth closer to the theoretical. This sensitivity of the tree depth to insertions is made even more apparent if a tree has been inserted into and has had data removed many times. The tree depth degrades as the tree progressively becomes more unbalanced. There exists two common solutions to this problem. The first is to use a data object called a red and black tree. This colours nodes red and black and removes and inserts based on these colours. This results in much more balanced trees as balance is enforced using the colouring mechanism upon insertion and removal. Another solution is to use a splay tree. In a splay tree recently accessed nodes move closer to the root of the tree. This leads to the tree becoming self optimising and as a result should perform better for many search sequences routines. The splay tree also suffers from poor balance in a strictly increasing insertion regime. The tree should become more balanced after a series of searches however as the structure will change when moving data to the root node. In summary binary search trees are very susceptible to becoming poorly balanced and slightly altered data structures have to be used to enforce balance.

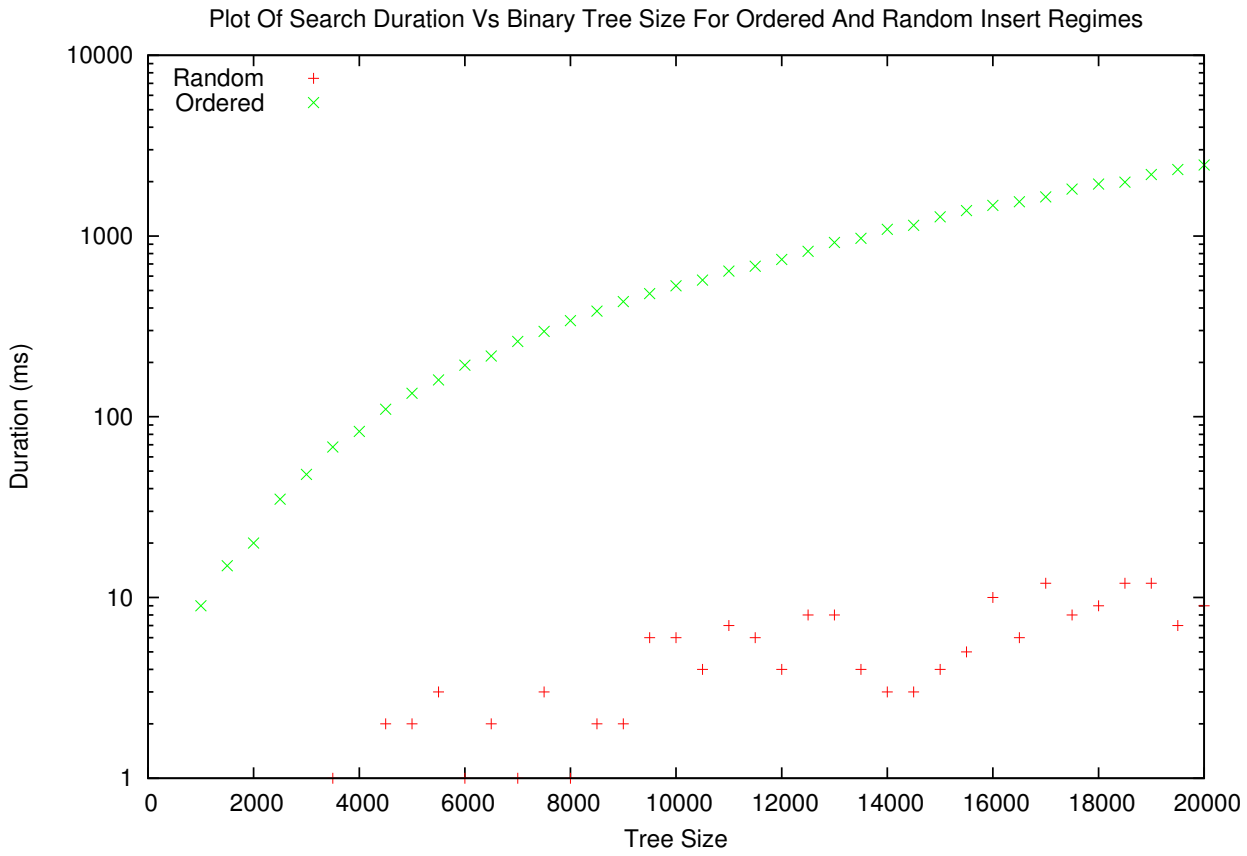

Figure 1: Plot of serach duration Vs tree size for different insertion regimes. It's clear from this plot that a random insert regime results in much faster searches. This is due to a more balanced tree which results in a shorter path to the data.