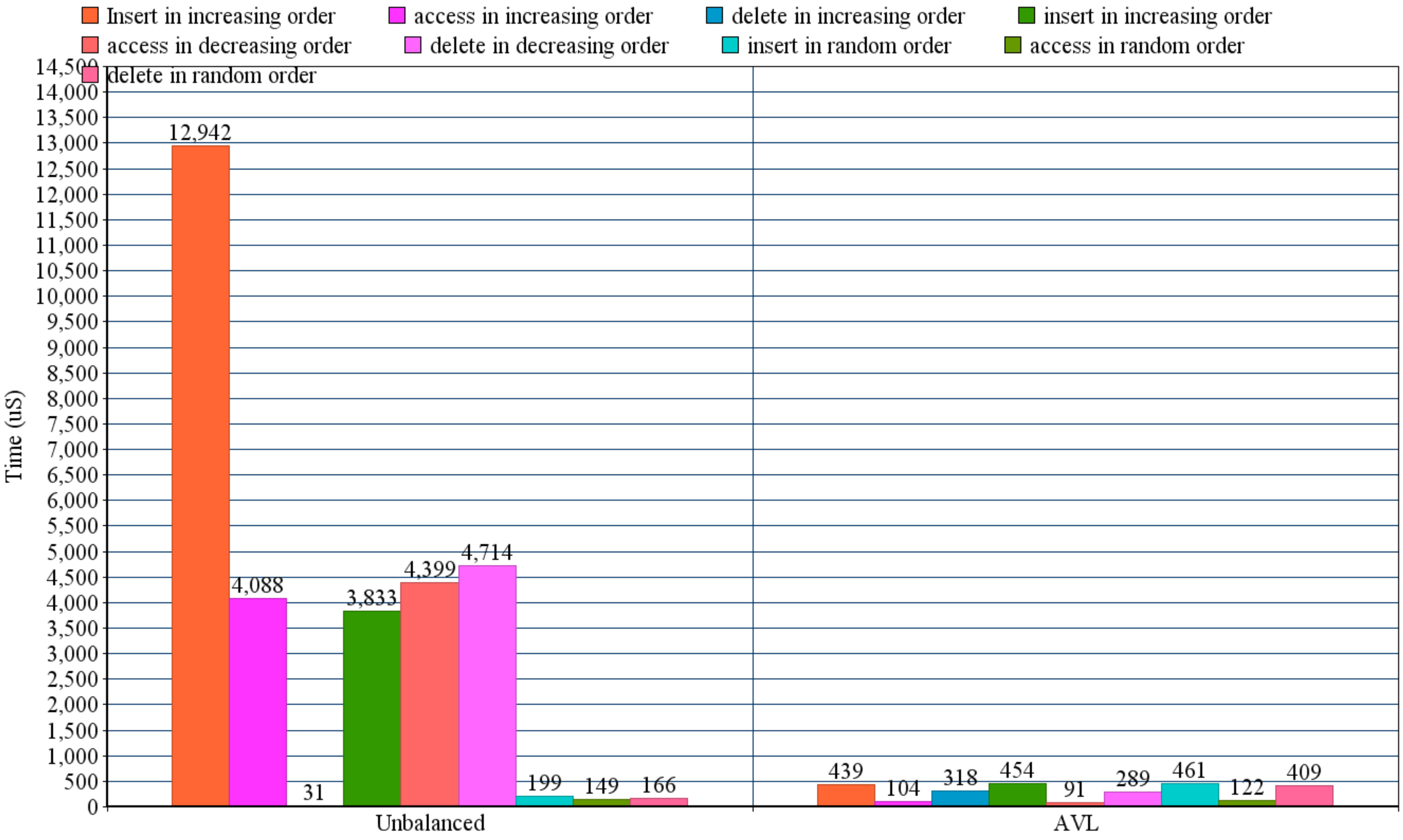
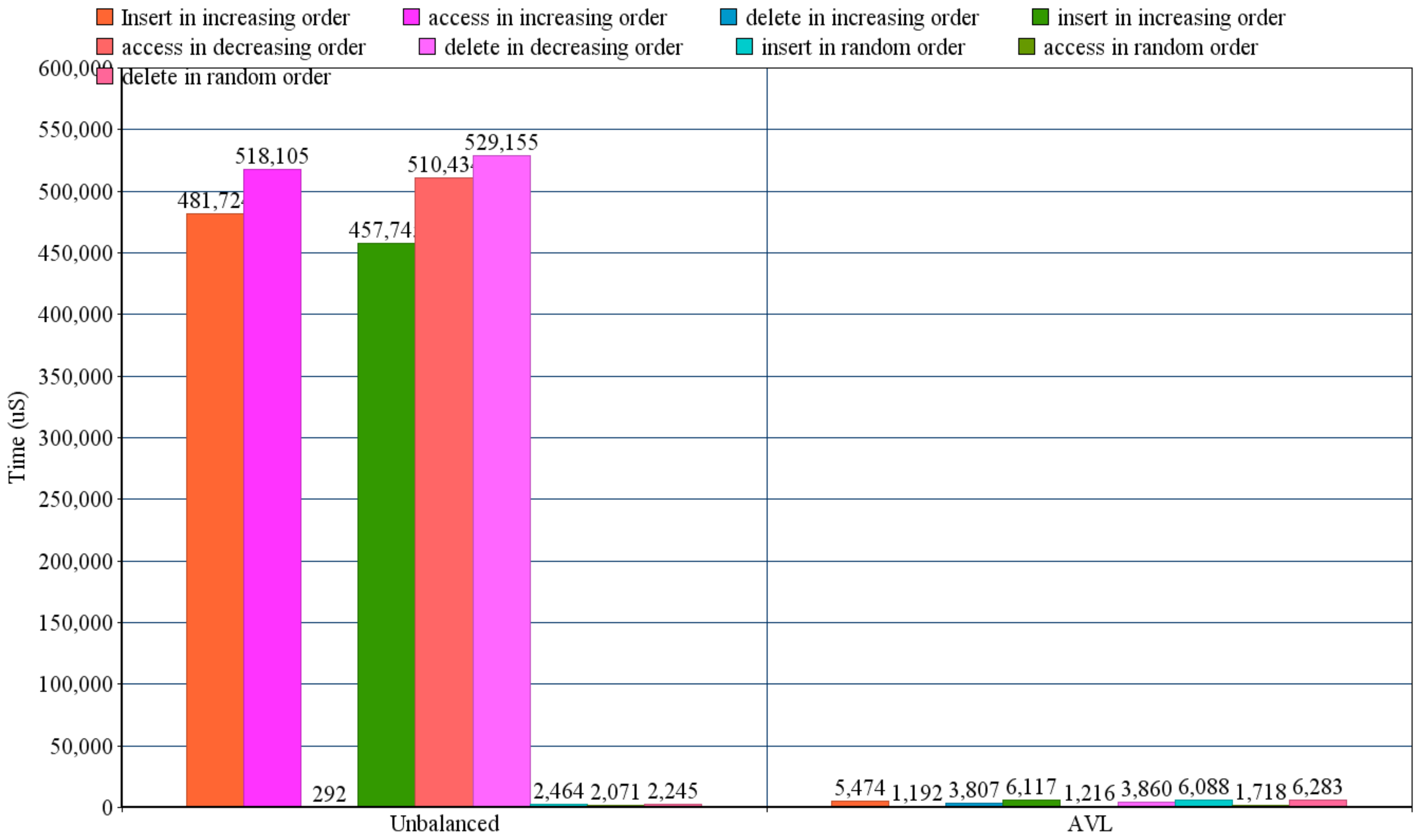


Comparison of AVL and Unbalanced BST: 1,000 operations

Comparison of AVL and Unbalanced BST: 10,000 operations



Analysis

For insertion in increasing order, unbalanced binary search trees are vastly slower than AVL trees are. Without balancing the tree descends in a straight line, essentially making a linked list. For accessing in increasing order, the results show for smaller N items, unbalanced trees perform better, yet still significantly worse than AVL trees. Deletion in increasing order is extremely fast for unbalanced trees, which beat AVL trees by a noticeable amount. The story is much the same for accessing in decreasing order after the N items have been inserted into the trees, with AVL being much quicker. Deletion in decreasing order is a very slow operation for unbalanced trees, it does however become slightly quicker when input sizes are smaller. Inserting in a random order and deleting in random order yields noticeably faster runtimes for unbalanced trees as opposed to AVL trees. The AVL is slower at this because of all of the rebalancing that occurs when we want to insert or delete a node, and unbalanced trees do not have to worry about this balancing. Accessing in a random order is only slightly faster in AVL trees than in unbalanced trees, because of the balancing properties of AVL trees, accessing nodes is slightly quicker.