Ran tester with arrays of size 10, 100, and 1000. The array was unsorted and sorted for different trials. Trials had varying size of doubles. Data in graph is an average of 12 runs with the lowest and highest being dropped, rounded up.

Size		10	100	1000
BST insert	unsorted	820	1390	6482
	sorted	700	2620	14162
	height(Unsorted\ Sorted)	3\9	12\99	22\999
AVL Insert	unsorted	880	1640	4921
	sorted	780	1753	5543
	height(Unsorted\ Sorted)	3\3	7 \ 6	11\9
BST remove	unsorted	140	220	3320
	sorted	130	1040	12492
AVL remove	unsorted	30	60	672
	sorted	12	168	703

AVL tress are significantly faster when it comes to storing sorted arrays. AVL tree's lazy delete method is also much faster across all data set sizes. AVL tree's ability to rebalance after insert causes its runtime to be drastically reduced on sorted data sets; however, it is higher when it comes to unsorted data until the data sets become very large.

Runtimes are as follows

	Insert	remove
AVL	logN	logN
BST	N	N