Benchmarking Results

Resulting times for 'Find Last', 'Delete Last' and 'Insert' for each table are the average results of 100 tests performed for each function, and for each data structure

Vector and Binary Search Tree performed 100 tests for each function per table

N = 1,000	Find Last (average of 100)	Delete Last (average of 100)	Insert (average of 100)
Vector	7.28 seconds	7.14 seconds	12.86 seconds
Binary Search Tree	0.000212 seconds	0.011 seconds	0.077 seconds

N = 10,000	Find Last (average of 100)	Delete Last (average of 100)	Insert (average of 100)
Vector	72.73 seconds	72.32 seconds	129.64 seconds
Binary Search Tree	0.0003 seconds	0.00035	0.143 seconds

N = 100,000	Find Last (average of 100)	Delete Last (average of 100)	Insert (average of 100)
Vector	710.77 seconds	705.66 seconds	1,269 seconds
Binary Search Tree	0.00035 seconds	0.00036 seconds	0.123 seconds

Discussion, Reflection and Analysis

All three benchmarks have shown that our Binary Tree structure is overall significantly more efficient than the Vector. We have outlined some of our observations below

For the benchmark where N = 1000, for some reason the delete for the Binary Tree was a lot slower than the search. Although the difference between the search (0.000212 seconds) and the delete (0.011 seconds) is negligible, we

assumed that those two operations would take almost the exact same amount of time. This is because delete also performs a search but with the additional step of deleting the node that it finds. It would have made more sense if that node had children, but we know that this node did not have any children so it only had 1 node to delete.

This oddity did not persist in our benchmarks where N = 10,000 and N = 100,000 as the search and delete for the Binary Tree in both benchmarks were almost exactly the same.

One observation that we made was that the amount of time it took for the vector to perform an operation is directly proportional to the number of elements in the dataset. For example, for the benchmark where N = 1000, the search operation for a vector took 7.28 seconds. For N = 10,000 which is 10 times larger than the previous benchmark, the time it took for a vector to perform a search was also 10 times longer (72.73 seconds). This is true for all three operations for the vector. For N = 100,000 each operation once again was 10 times longer than the previous benchmark for the vector. This makes the time complexity for vectors O(N). Meanwhile, the time it took for the Binary Tree to perform its operations remained almost exactly the same between each benchmark, with increases in time being mere fractions of a millisecond.