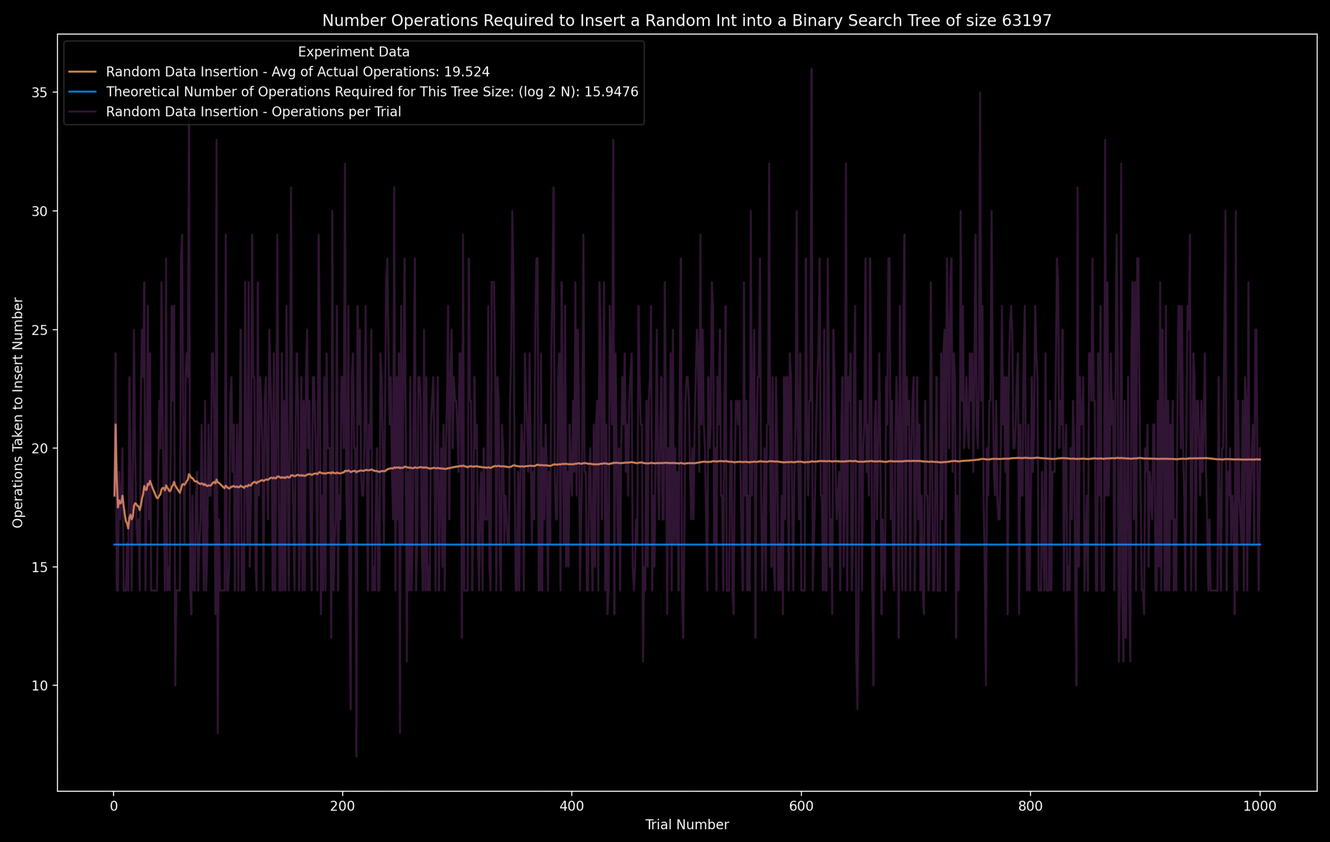
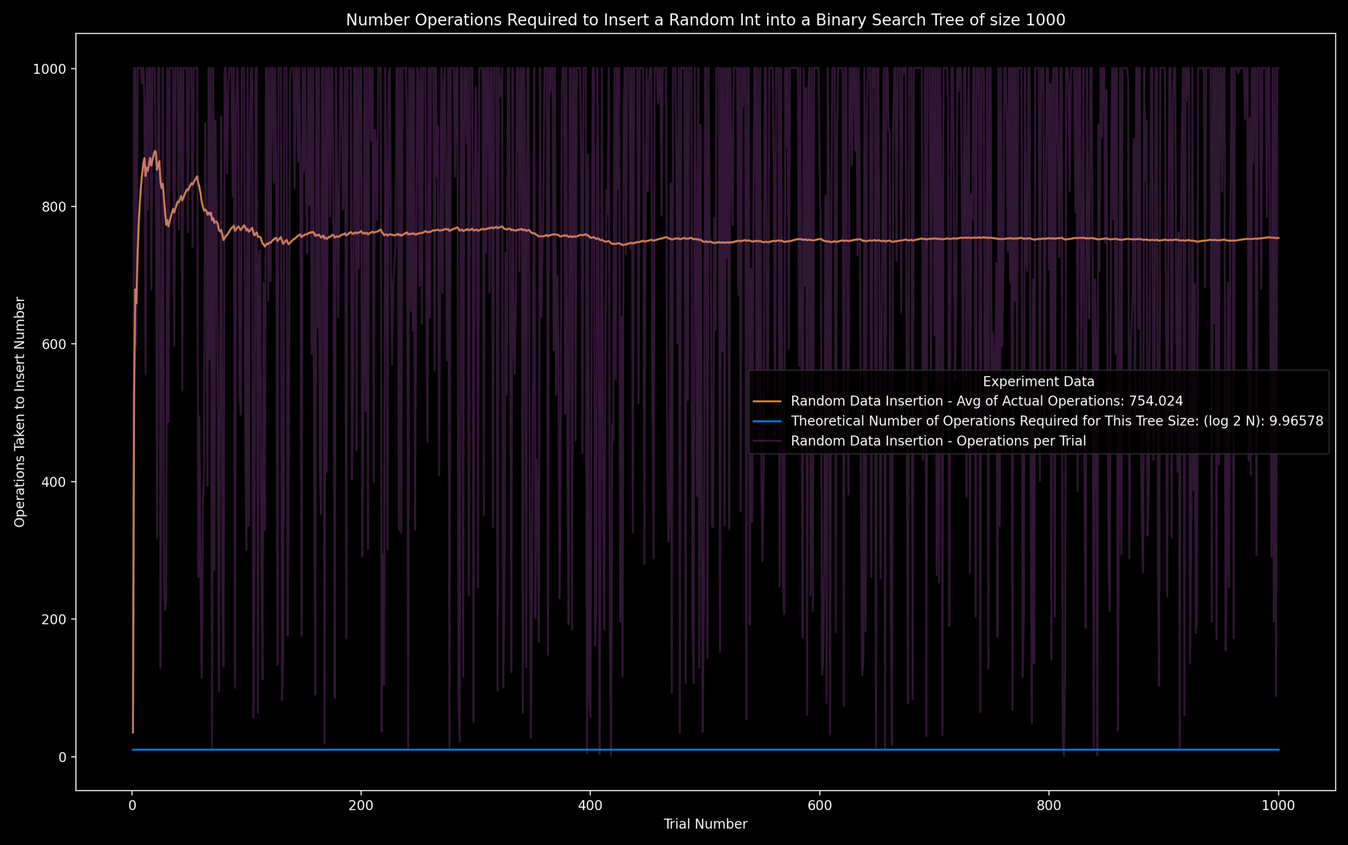


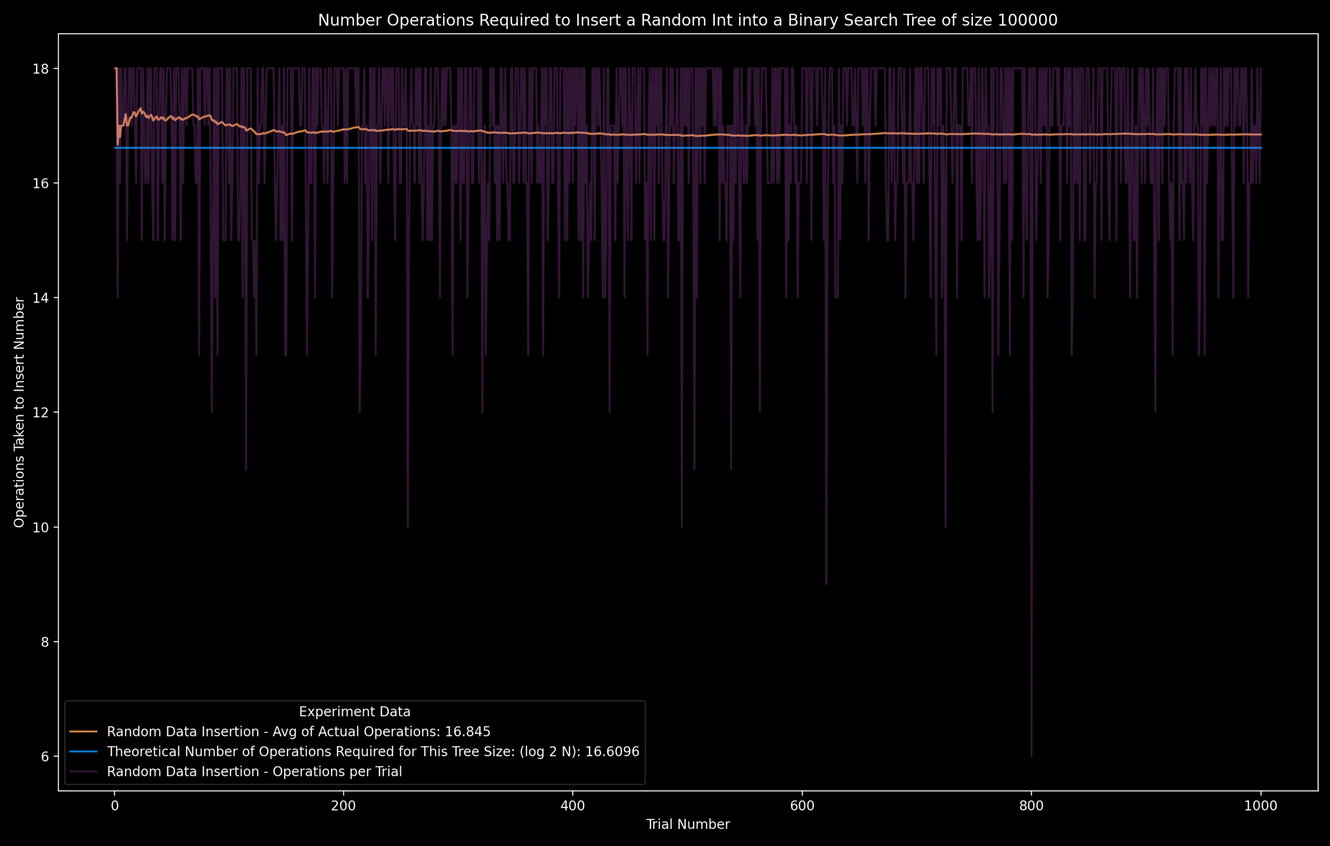
Average Case Scenario

Binary Search Tree built from a vector of randomized integers. Randomization of the integers allows the insertion process to approach the ideal (in blue) O(log2N). After the tree is built via the random integer process, data shows how many insertion operations it takes to place a randomaly chosen integer that falls within the values already in the tree (values range from 0–2N tree size).



Worst Case Scenario

Binary Search Tree built from a vector of in-order integers. Building a tree via ascending data results in a linked list structure. Inserting random data after a tree is built in this way results in the least efficient process by which to insert data into a BST: O(N) time complexity due to the linked list structure of the tree.



Optimized Scenario

Building a tree via the divide and conquer method results a nearly perfect insertion time of O(log2N). Divide and conquer takes an ordered array and recursively splits the array in the middle, using that middle value as the root for each node. This ensures that each value inserted falls to its ideal spot in the tree (smaller values to left of parent node and larger values to the right of parent node).