Binary Tree
Unbalanced Binary tree
Size of problem: **n** (number of kvps)

Set(insert) operation:
Kvps: 199 total compares 1447
Average compares 1447/199 = 7.27 compares

Get(search) operation:
Existing-Max Compares = 16, average Compares = 8.27
Miss-Max compares = 11, average compares = 8.3

In the unbalanced binary tree, the worst-cast time for both the get and set is O(n) where n is number of nodes. The data support this analysis, as the max compares for the search operation (existing and miss) and the average compares for insert are proportional to total kvps

Balanced Binary Tree
Size of the problem: **n** (number of kvps)

Set (insert) operation:
Total kvps = 199. Total compares = 1253
Average compares = 1253/199 = 6.30

Get (Search) operation:
Existing - Max Compares = 8, Average Compares = 6.76
Miss - Max Compares = 8, Average Compares = 7.65

In the balanced binary tree, the worst case time complexity for both operations is O(log n) where **n** is number of nodes. The data supports this analysis as the max compares for search operation and average compares for insert are proportional to the logarithm of the total kvps

Hash Table

Hash Table Analysis
Size of the problem: n (number of key-value pairs), M (number of buckets)

Set(insert) operation
A hash table with a good hash function, the average time complexity for the set operation is O(1) if the load factor is kept low. The data shows that as the number of buckets increases, the total compares for insert operations decreases, approaching the O(1) time complexity.

Get(search) Operation
The data shows that as the number of buckets increases, the average compares for the search operations decrease approaching O(1) complexity

Big O Analysis Summary

Binary Tree:

Unbalanced: both the get and set have a worst-case time complexity of O(n)
Balanced: Both get and set have a worse-case time complexity of O(log n)

Hash Table:
Both get and set operations, the time complexity is O(1) if the load is kept low. As buckets increase, time complexity approaches the optimal O(1) complexity.