A Red-Black Tree should operate in O(log n). Do to inefficiencies in the code, the runtime appears to be O(n2) or perhaps even O(n).

There are (at least) two inefficiencies in the implementation. The inefficiencies affect the expected O( ) runtime of the Red-Black tree by adding O(1) operations whose constant times are so great that the constants overwhelm the expected O( ) runtime of a Red-Black tree.

One of the inconsistencies comes from the insert method. Before being called, the code tests to see if the node already exists. If the node does not exist, the insert function is called. Then the nodes existence is called into question again before the actual insertion happens. When the function has run its course and we exit back to the main code, the insertion is called again. One insertion for the price of three.

There is also a small problem with malloc being called for each new node. Instead of allocating memory for the entire block of nodes and partitioning it out or using “new” for each new node, the code calls malloc at every new node creation and allocates a small portion of memory each time. This causes additional drag on the run time, forcing the O() further away from its intended O(log n).

A prime example of why code from the internet is bad and why an algorithm with a known runtime when implemented rashly can run amuck.