Console Output:

run:
Please give number of elements to be inserted : 5000000
Checking...
Binary search tree skipped due to large size.
AVL search tree: 1006 milliseconds.
Red-Black search tree: 1040 milliseconds.
Splay search tree: 828 milliseconds.
Sorting: 129 milliseconds.
AVL search tree: 1264 milliseconds.
RBT search tree: 2814 milliseconds.
BUILD SUCCESSFUL (total time: 9 seconds)

Theoretical Points:

AVL and Red-Black Trees are very similar, although share some key differences. The height of an AVL Tree is max around 1.4*log*(n), while a Red-Black Tree has a max height of 2*log*(n). On insertion, an AVL Tree may need O(*log*(n)) operations, as it may need to rebalance the Tree. Counter to this, a Red-Black Tree will give O(1). This relates that the AVL Tree is usually faster on searching, but slower on insertion.

For a Splay Tree, the search may be O(n) if the elements are inserted in order. This case causes a Splay tree to be slower than most AVL and Red-Black Trees, as the inputs will usually be random. The benefits come from the self-adjusting aspect of the Splay Tree.

Experimental Points:

Although most results are very close, there is still some divergence. The Red-Black Tree has a worse run time in general compared to the AVL tree in both insertion and searching. Although the Red-Black tree has better theoretical insertion, when given a large and random set of data, it must do a lot of rebalancing that the AVL tree may not need to do. The Splay tree was the fastest as the inOrder function helps it insert everything in order.