IDSA LP 3: Comparison Study of SkipList, RedBlackTree and Java TreeSet

Team Members:
Adarsh Raghupati NetID: axh190002
Akash Akki NetID: apa190001
Keerti Keerti NetID: kxk190012
Stewart Cannon NetID: sjc160330

TreeSet is one of the most important implementations of the SortedSet interface in Java that uses a Tree for storage. The ordering of the elements is maintained by a set using their natural ordering whether or not an explicit comparator is provided.

Runtime Complexity	Insert	Search	Remove
Average	O(log(n))	O(log(n))	O(log(n))
Worst	O(log(n))	O(log(n))	O(log(n))

SkipList: Skip lists provide an alternative to the BST and related tree structures.

Runtime Complexity	Insert	Search	Remove
Average	O(log(n))	O(log(n))	O(log(n))
Worst	O(n)	O(n)	O(n)

RedBlackTree: A red–black tree is a kind of self-balancing binary search tree with each node having an extra bit to depict color ensuring the tree remains balanced.

Runtime Complexity	Insert	Search	Remove
Average	O(log(n))	O(log(n))	O(log(n))
Worst	O(log(n))	O(log(n))	O(log(n))

Preview: The comparisons are made between the data structures based on each operation. Randomly generated values (4M,16M, 32M..) are stored in the HashSet and the same values are used in the functions for comparisons.

4 Million Data	**Java TreeSet**	**SkipList**	**RedBlackTree**
Add Functionality	Time: 3461 msec. Memory: 438 MB / 2065 MB.	Time: 5624 msec. Memory: 808 MB / 2586 MB.	Time: 4626 msec. Memory: 931 MB / 2586 MB.
Contains Functionality	Time: 579 msec. Memory: 281 MB / 756 MB.	Time: 1606 msec. Memory: 281 MB / 756 MB.	Time: 498 msec. Memory: 281 MB / 756 MB.
Remove Functionality	Time: 235 msec. Memory: 279 MB / 756 MB.	Time: 733 msec. Memory: 279 MB / 756 MB.	Time: 198 msec. Memory: 279 MB / 756 MB.

16 Million Data	Java TreeSet	SkipList	RedBlackTree
Add Functionality	Time: 41173 msec. Memory: 1636 MB / 3217 MB.	Time: 58651 msec. Memory: 2281 MB / 3454 MB.	Time: 50502 msec. Memory: 1466 MB / 4028 MB.
Contains Functionality	Time: 1862 msec. Memory: 1017 MB / 2722 MB.	Time: 6572 msec. Memory: 1018 MB / 2722 MB.	Time: 1484 msec. Memory: 1018 MB / 2722 MB.
Remove Functionality	Time: 2362 msec. Memory: 1015 MB / 2674 MB.	Time: 6800 msec. Memory: 1016 MB / 2674 MB.	Time: 1646 msec. Memory: 1016 MB / 2674 MB.

32 Million Data	Java TreeSet	SkipList	RedBlackTree
Add Functionality	Time: 70480 msec. Memory: 2973 MB / 4028 MB.	Exception in thread "main" java.lang.OutOfMemoryError: Java heap space	Time: 84093 msec. Memory: 2869 MB / 4028 MB.
Contains Functionality	Time: 3743 msec. Memory: 2001 MB / 3552 MB.		Time: 3018 msec. Memory: 2002 MB / 3552 MB.
Remove Functionality	Time: 3706 msec. Memory: 1999 MB / 3312 MB.		Time: 2776 msec. Memory: 1870 MB / 3312 MB.

In this project, three different data structures were tested for runtime and memory used: skip list, red black tree, and Java TreeSet. These were each compared by running one million contains, removes, and between 4 and 32 million add operations.

The skip list structure was the slowest structure across the board, and almost half as fast as the red black tree. The skip list also seemed to scale approximately linearly, doubling in runtime when the number of elements doubled. Memory use for the skip list was also generally between 45% and 65% over the tests.

The red black tree was faster than the skip list but slower than the TreeSet across the board. While the red black tree appeared to be increasing in runtime linearly with the elements, at 32 million elements, the runtime was significantly less than that trend would assume. This could be due to the logarithmic worst case runtime given by a balanced tree structure. The memory for the red black tree also fluctuated between 45% and 78% memory use, with two values being close at 73% and 78%.

The TreeSet structure was the fastest of the 3 data structures across the board. The TreeSet performed close to the red black tree on the 8 million elements, but the time was 20-25% faster than the red black tree on sets of 16 and 32 million. The memory usage of TreeSet showed a higher utilization of 85% and 65% for 4 and 8 million, while the utilization for 16 and 32 million was only 31% and 39%.

In conclusion, the skip list was clearly the slowest, while the red black tree and the TreeSet were more competitive. However, the TreeSet structure was still the fastest of the three. Looking at memory usage, the values seen seemed unstable, however the TreeSet data structure consistently used less amounts of memory than the other structures at higher numbers of elements.