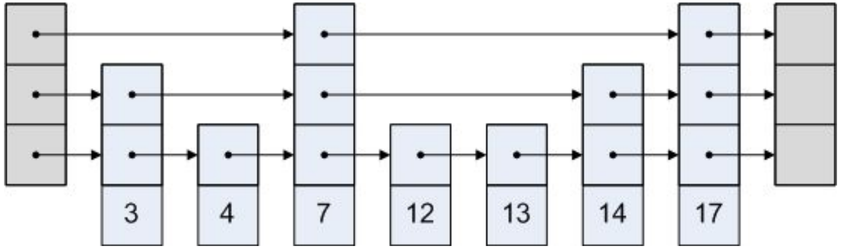
SKIP LIST vs RED BLACK TREES

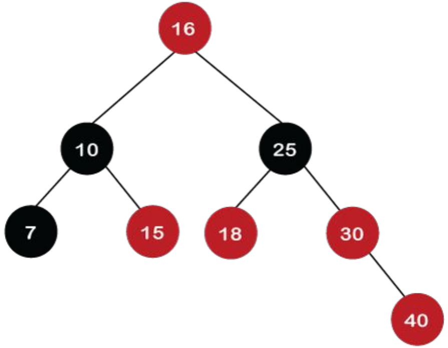
Group Members:
Nabila Zahra
Iqra Azfar
Muhammad Youshay
Rabia Shahab

Skip List vs Red Black Tree

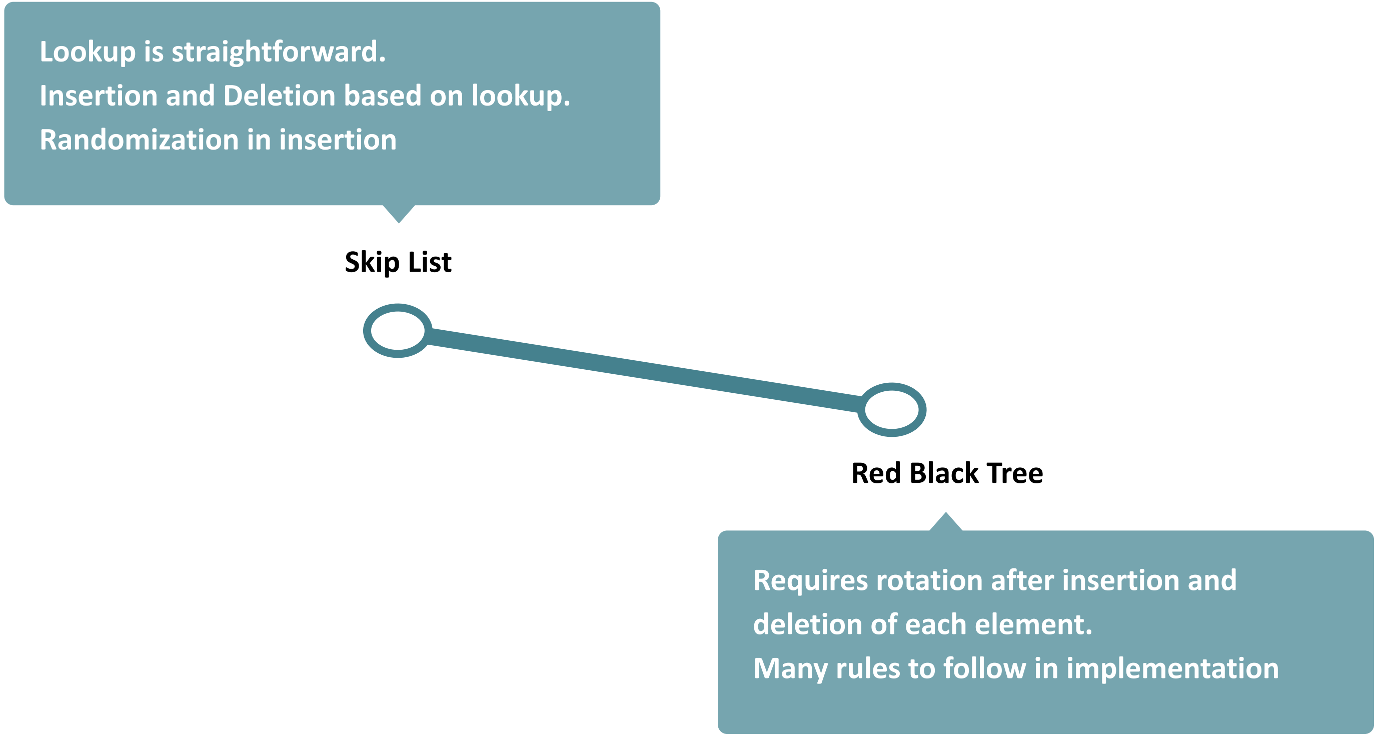
Skip List
Probabilistic Data Structure
Organized in layers
Coin Flipping technique

Red Black Tree
Binary Search Tree
Self Balancing
Red or Black nodes colors





Implementation Comparison



Lookup is straightforward.
Insertion and Deletion based on lookup.
Randomization in insertion

Skip List

Red Black Tree

Requires rotation after insertion and deletion of each element.
Many rules to follow in implementation

Space and Time Complexity

	Skip List	Red Black Tree
Insertion	O(logn) Worst case O(n)	O(logn) Worst case O(logn)
Search	O(logn) Worst case O(n)	O(logn) Worst case O(logn)
Deletion	O(logn) Worst case O(n)	O(logn) Worst case O(logn)
Space Complexity	O(n)	O(n)

Strengths and Weakness Comparison

Skip List Advantages

- Fast Insertion - no rotations
- Simpler to implement
- Better Cache Locality

Red Black Tree Advantages

- Predictable Behaviour
- Guaranteed Worst case O(logn)
- Efficient memory usage

Strengths and Weakness Comparison

Skip List Disadvantages	Red Black Tree Disadvantages
Higher space complexity	Complex implementation
Slower search performance	More storage (color of node)
Non-deterministic behavior	Needs to maintain balance

Clock Speed of PC used

Time complexity for inserting,deleting n elements in skip list and red black tree: **O(nlogn)**

Clock speed = **1.8GHz AMD A6 processor**

Insert 100,000 elements = O(100,000log2(100,0000)) = growth with the factor of 1660964.04

Estimated clock cycles = 15 to execute a single insertion operation

Total clock cycles = Number of elements * Clock cycles per operation

= 100,000 * 15 = 1,500,000

Time it would take to insert 100,000 elements in seconds:

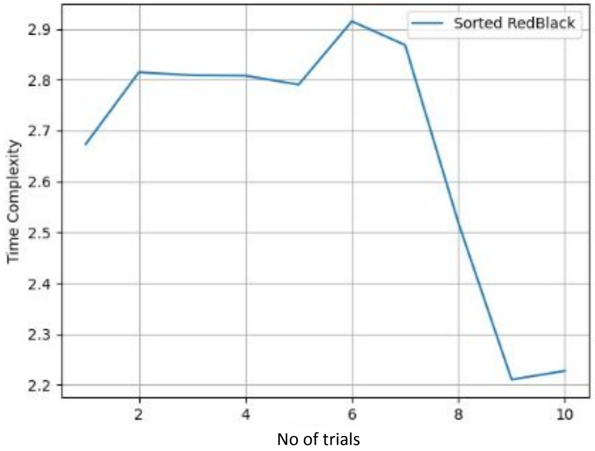
Time in seconds = Total clock cycles / Clock speed

= 1,500,000 / 1.8e9

= 0.00083333 seconds

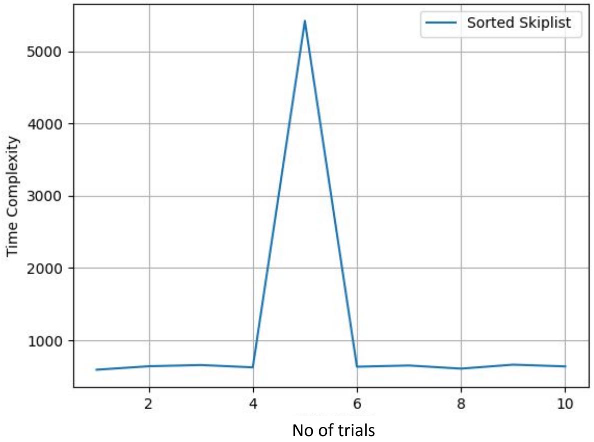
Sorted Dataset - 100,000

Red Black Trees



Average Time: 2.855s

Skip List



Average Time: 1114.03s

Red Black Tree is highly efficient for sorted dataset

Sorted Dataset

Red Black Tree is highly efficient for sorted dataset

Red Black Tree

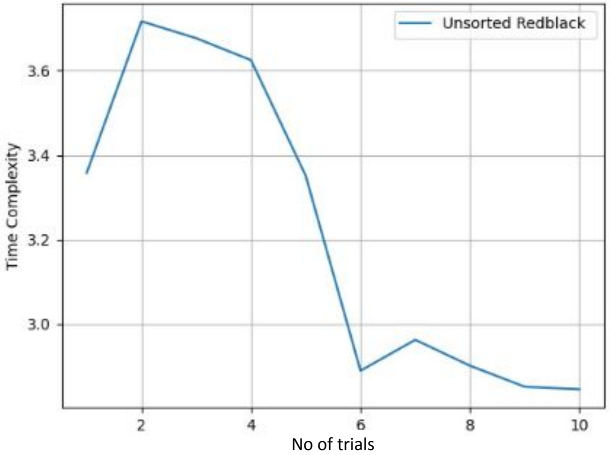
- It has a balanced structure so efficient for handling sorted data set.
- O(logn) insertion

Skip List

- Nodes Cluster together at the beginning of the skip list
- The number of levels increases linearly with number of elements in the list

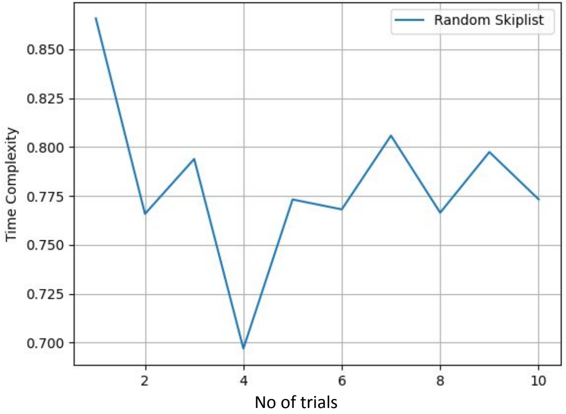
Random Dataset - 100,000

Red Black Trees



Average Time: 3.321s

Skip List

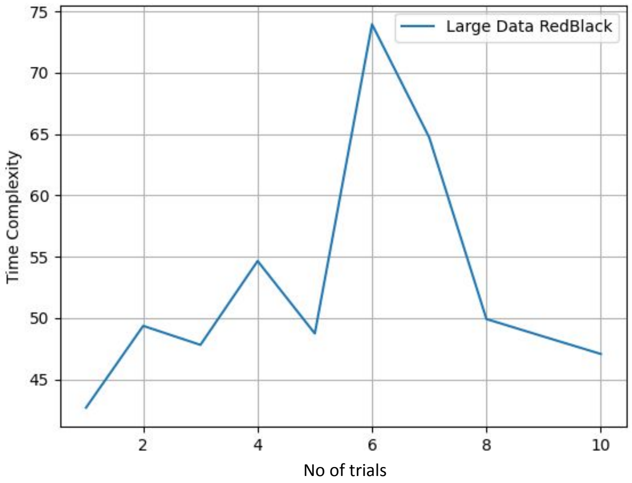


Average Time: 0.846s

Skip List is more efficient for random dataset

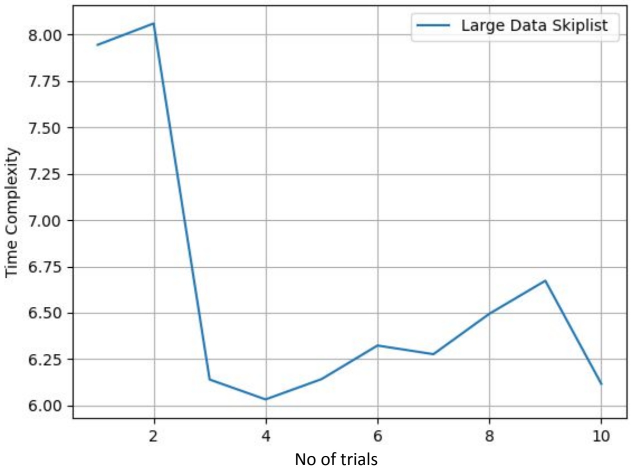
Large Dataset - 1000,000

Red Black Trees



Average Time: 59.854s

Skip List



Average Time: 6.605s

Skip List is more efficient for large dataset

Random and Large Dataset

Skip List is more efficient for random or large dataset

Red Black Tree

- Balance needs to be maintained by rotation and color changing so slower performance

Skip List

- Probabilistic nature ensures that nodes are evenly spread out leading to efficient insertion

Conclusion

- In conclusion, both Skip Lists and Red-Black Trees are highly efficient data structures that can be used for a variety of applications.
- Skip Lists are more efficient for large and random data sets, while Red-Black Trees are highly efficient for handling sorted data sets.
- Even though the average time complexities is the same for skip list and red black tree, the dataset being used makes a difference in the performance of both the data structures.
- The choice between Skip Lists and Red-Black Trees depends on the specific requirements of the application, the characteristics of the data being handled and hardware being used.