# AVL Tree and Skip List Performance Comparison

Junyi Ye

## Performance testing

Test condition:
Size: [128, 512, 1024, 2048, 4096, 8192, 16384, 32768, 65535, 131072]
Log(size): [7, 8, 9, 10, 11, 12, 13, 14, 15, 16]
Key range: (1, 1_000_000)
Number of times: 100_000 for every size

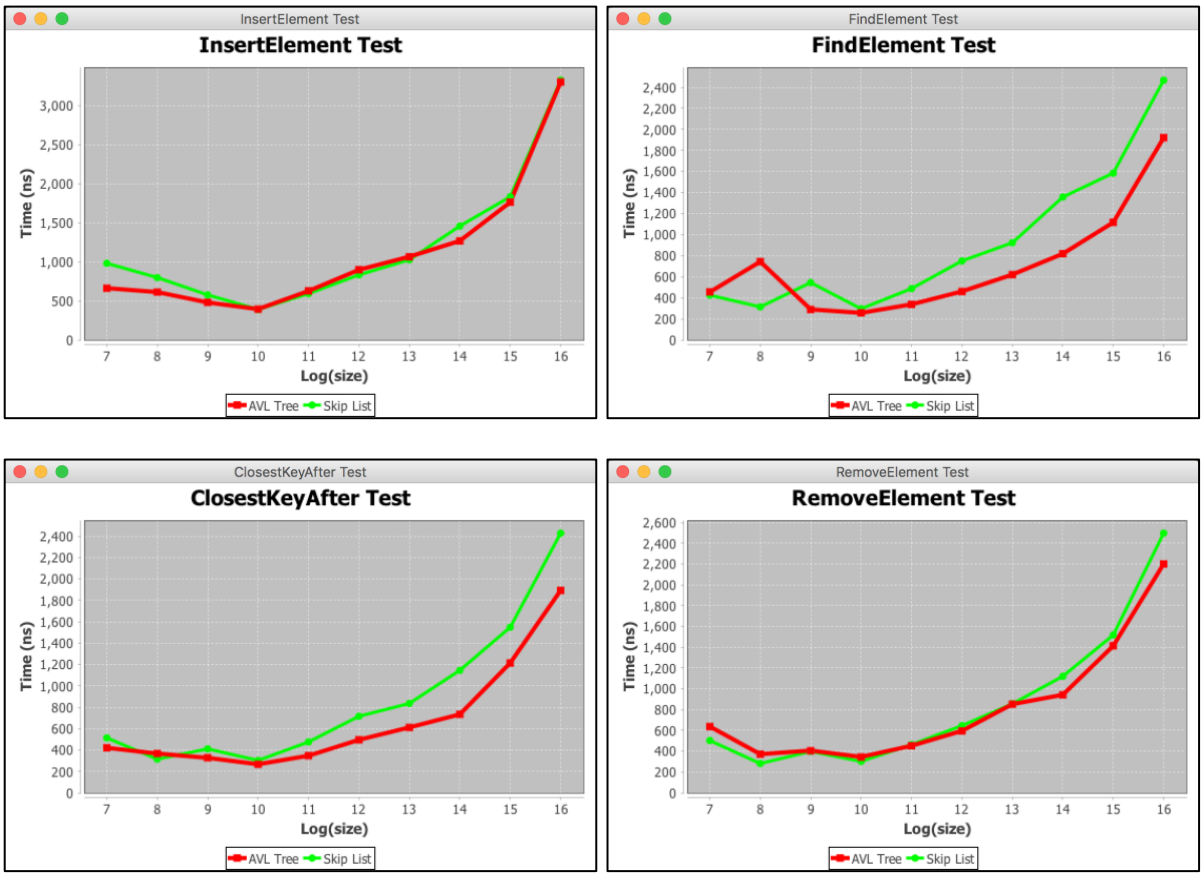


**Figure 1 Performance of four operations of AVL Tree and Skip List**
**(Red lines: AVL Tree; Green lines: Skip List)**

I used an open source Java plotting library called JfreeChart to plot the above line charts. The x coordinates were converted to logarithmic coordinates which is log(size). And the y

coordinates represent the average running time of each operation in nanosecond. The theoretical time complexity of all four operations in both AVL Tree and Skip List should be O(log(n)). That means all operations' average running time should grows linearly in the above logarithmic coordinate charts. However, the results of my program are greater than O(log(n)).

Before I do the performance testing, some small samples were tested on both AVL Tree and Skip List and they all passed. That means all the operations should be correct. In my opinion, there could be two reasons that caused the above issue. One is that some logic in my code may have some small problems. Even small ones under a large number of iterations or recursive calls will make the time complexity much greater. The other one could be the inhomogeneous utilization of the computer's CPU for each calculation. We can see there are some abnormal points in the small size cases in the above charts.