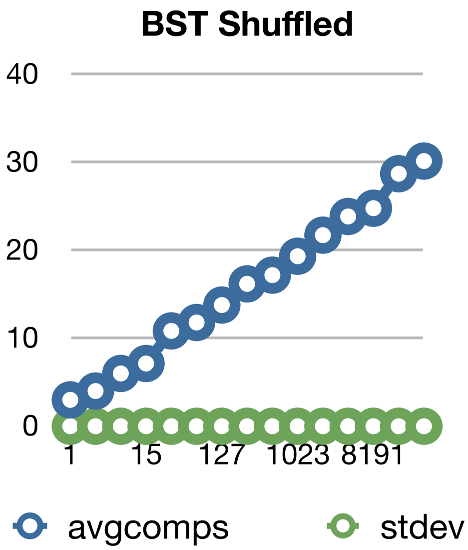
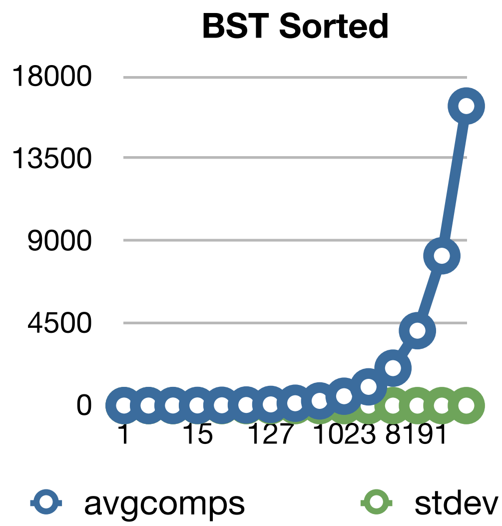
Benchtree Report

Yan Zhang,Chaoteng Liu

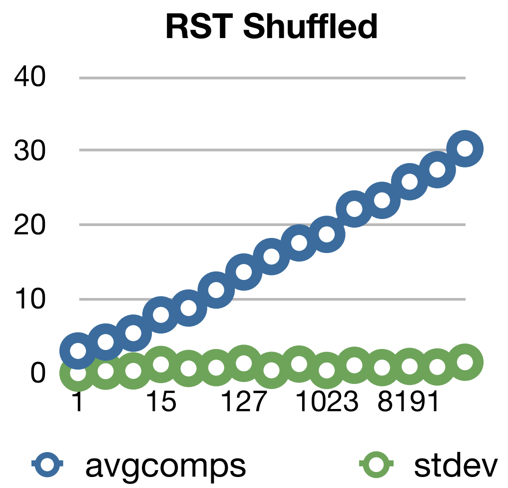




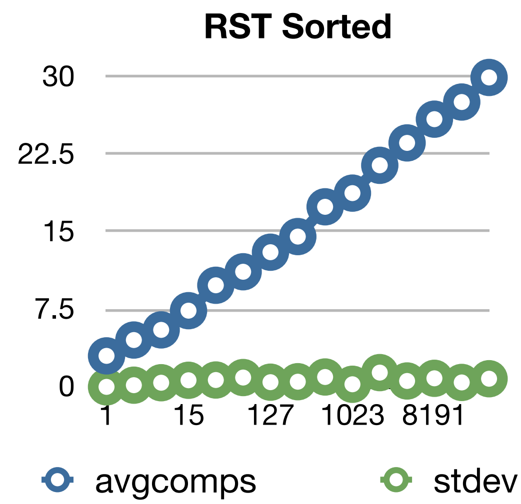
Number of Nodes	avgcomps	stdev
1	3	0
3	4	0
7	6	0
15	7.13333	0
31	10.8387	0
63	11.7937	0
127	13.7795	0
255	16.1882	0
511	17.18	0
1023	19.3275	0
2047	21.7255	0
4095	23.8493	0
8191	24.7857	0
16383	28.7096	0
32767	30.1497	0

Number of Nodes	avgcomps	stdev
1	3	0
3	4	0
7	6	0
15	10	0
31	18	0
63	34	0
127	66	0
255	130	0
511	258	0
1023	514	0
2047	1026	0
4095	2050	0
8191	4098	0
16383	8194	0
32767	16386	0

Form the plot and table above, we can find when data is shuffled, the average number of comparisons of BST is O(log(N)), and when data is sorted it is O(N), which is the worst case.
Therefore, it agrees with the theoretical big-O time cost analysis. BST performed as we expected. And the tree is fixed with the input order and the standard deviation is 0.



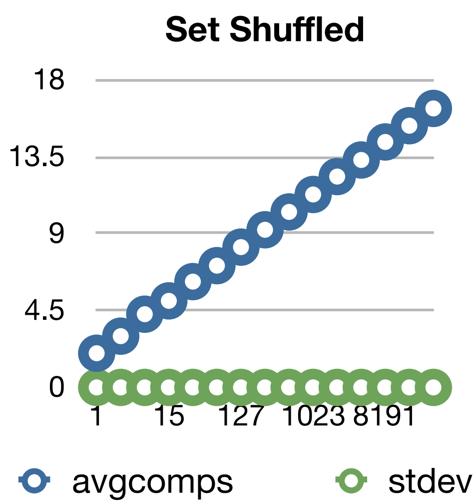
Number of Nodes	avgcomps	stdev
1	3	0
3	4.2	0.266667
7	5.4	0.305059
15	7.86667	1.17379
31	8.7871	0.634296
63	11.2159	0.743128
127	13.6504	1.32563
255	15.731	0.3807
511	17.5773	1.22625
1023	18.7298	0.36604
2047	22.1574	1.06313
4095	23.3227	0.749262
8191	25.8366	0.910267
16383	27.4658	0.836168
32767	30.2948	1.47619



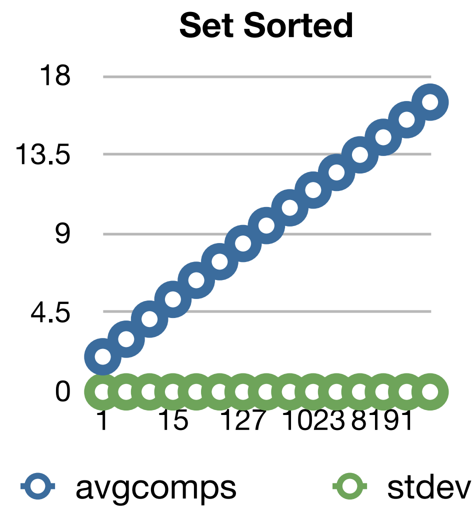
Number of Nodes	avgcomps	stdev
1	3	0
3	4.53333	0.163299
7	5.51429	0.4
15	7.37333	0.638888
31	9.83226	0.664046
63	11.1397	0.902526
127	13.0315	0.453587
255	14.5443	0.499192
511	17.4258	0.955894
1023	18.7501	0.251659
2047	21.4411	1.3798
4095	23.6097	0.571222
8191	25.9069	0.828953
16383	27.5654	0.428569
32767	29.9428	0.787897

Form the plot and table above, the data agrees with the theoretical O(log(N)) performance for insert operation of randomized search tree. Whether the data is sorted or shuffled, RST has almost the same performance. It has nothing to do with the order of input.

As a random data structure, we can see RST has a positive standard deviation. The running time of fixed data is not constant, but keeps O(log(N)) in average.



Number of Nodes	avgcomps	stdev
1	2	0
3	3	0
7	4.28571	0
15	5.06667	0
31	6.19355	0
63	7.12698	0
127	8.2126	0
255	9.22745	0
511	10.272	0
1023	11.2933	0
2047	12.3425	0
4095	13.314	0
8191	14.3682	0
16383	15.3187	0
32767	16.3312	0



Number of Nodes	avgcomps	stdev
1	2	0
3	3	0
7	4.14286	0
15	5.26667	0
31	6.35484	0
63	7.4127	0
127	8.44882	0
255	9.47059	0
511	10.4834	0
1023	11.4907	0
2047	12.4949	0
4095	13.4972	0
8191	14.4985	0
16383	15.4992	0
32767	16.4996	0

The std::set in C++ STL is implemented with Red-black tree, which is a very highly optimized data structure. The std::set has a stable running time whether the data is sorted or shuffled, and the plot is more smooth than RST. Moreover, the std::set has about double performance compared to our RST.
The deviation of std::set is 0, which indicates that the form of the tree is fixed with the order of the input.