COSC520 Assignment2 Advanced Data Structures

1. Complexity Analysis

#elements: n

Insert & Search

	Average time	Average auxiliary space	reference
Sorted Linkedlist	O(n) (sorting)	O(1)	Linked list - Wikipedia
Skiplist	O(logn)	O(logn) (record the path)	Skip list - Wikipedia
Binary Search Tree	O(logn)	O(1)	Binary search tree - Wikipedia

2. Measured Performance

There were 20 tests in the numerical experiment on the datasets of size between 500 and 1500. Here, to be concise I extracted only part of the results that could be representative.

Runtime of **inserting n elements and then search** for one element:

Test#	Dataset size(#n)	LinkedList	SkipList	BST
1	500	0.2500	0.0705	0.0246
2	750	0.5208	0.1171	0.0374
3	1000	0.9161	0.1692	0.0520
4	1250	1.4185	0.2384	0.0611
5	1500	2.0794	0.3042	0.0695

Runtime of only **inserting** n elements:

Test#	Dataset size(#n)	LinkedList	SkipList	BST
1	500	0.2489	0.0683	0.0243
2	750	0.5184	0.1147	0.0370
3	1000	0.9131	0.1666	0.0517
4	1250	1.4168	0.2355	0.0606
5	1500	1.9804	0.2956	0.0686

Runtime of only **searching** for one element:

Test#	Dataset size(#n)	LinkedList	SkipList	BST
1	500	0.0011	0.0023	0.0003
2	750	0.0024	0.0024	0.0004
3	1000	0.0029	0.0027	0.0003
4	1250	0.0017	0.0029	0.0005
5	1500	0.0030	0.0025	0.0003

BST is the fastest, then Skip List and LinkedList. This proves the time complexity. When searching, linked list at the beginning is better than skip list but skip list outperforms linked list in the long term.

If you rerun numerical_test.m, you may not get the exact same runtime results as above but are very close (the randomly generated test sets are the same using the same seed).

3. Performance Profile

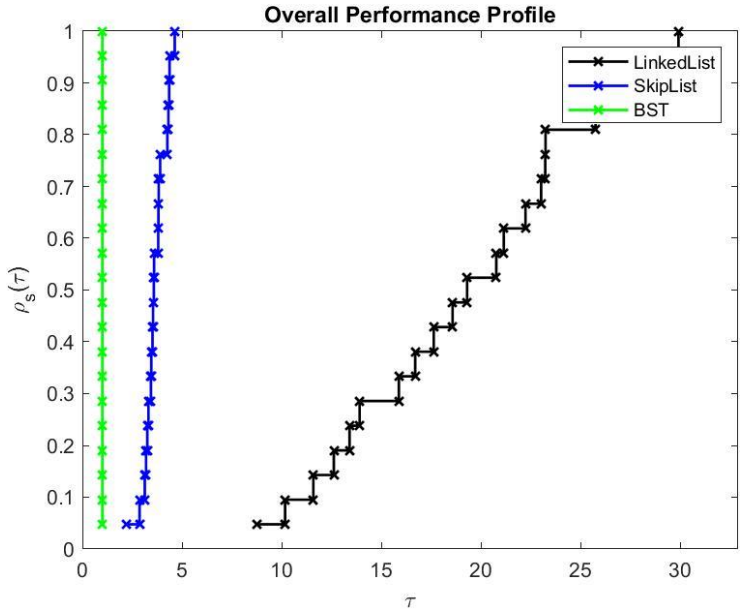

Fig 1: all.jpg

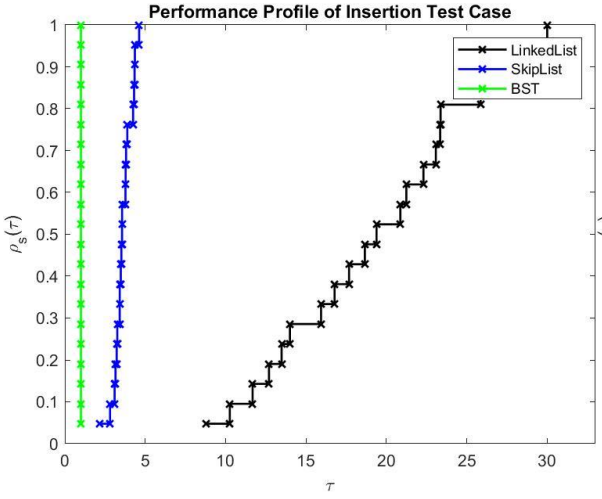

Fig 2: insert.jpg

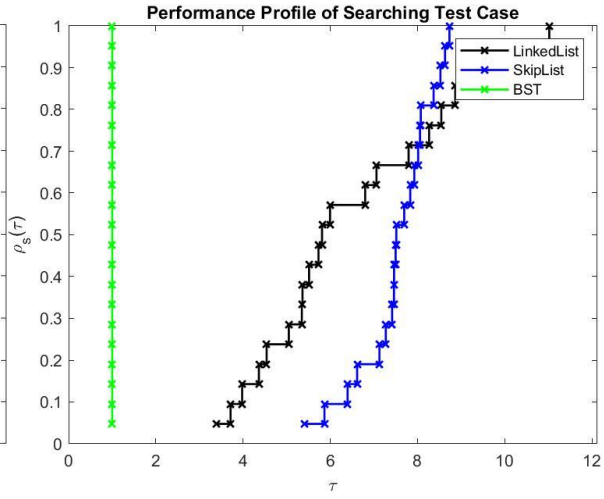

Fig 3: search.jpg

4. Limitation

Switching phenomenon after removing the best solver didn't happen in the overall profile but did happen in the searching plot. Due to the large difference in the overall performance, Skip list and linked list keep the trend after removing BST. However, since they intersected in searching plot, it's likely that they would switch in searching. Therefore, I made the second plot of the two solvers on only searching tasks and it proves.

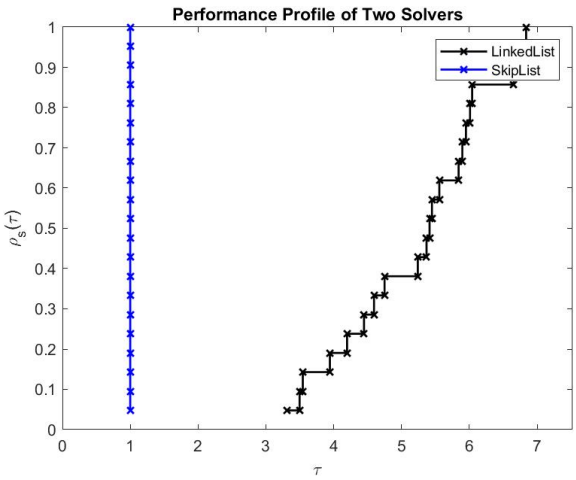

Fig 4: two.jpg

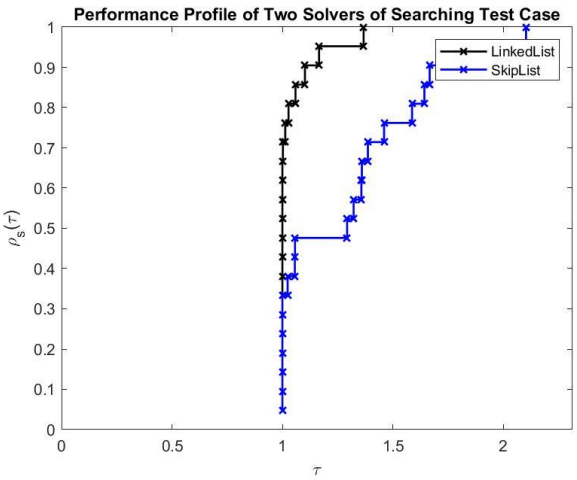

Fig 5: twosearch.jpg

5. Explanation

I chose these data structures because they have something similar in inserting and searching. Using Linked list, we can only linearly read or insert data. Using Binary Search Tree, we can skip nearly half of data to search for one specific element. And using Skip list we can choose a parameter and skip some proportion of data.

I'm also interested in how an un-balanced search tree would actually work in solving problems. Because we often consider its worst "linear tree" case when analysing, I'd like to know whether that's a common phenomenon or not, and how useful an unbalanced search tree is. It turns out that it still performs well at least in my experiment.

Skip List | Set 1 (Introduction) - GeeksforGeeks