Ian Almutawa, Scott Daniel

SKIPLIST PERFORMANCE SUMMARY

CS-610 / 07 DECEMBER 2022

TEST DESCRIPTION

The skiplist data structure creates a key-value dictionary utilizing multiple linked lists with pointers that can quickly navigate through the keys. A skiplist can be an efficient algorithm for searching, inserting, and deleting elements in a dictionary. Using the Python programming language, we constructed a basic skiplist and then tested various operations 1,000 times with randomly-generated entries to evaluate the average run time for each. We repeated the tests for half-filled dictionaries of increasing size.

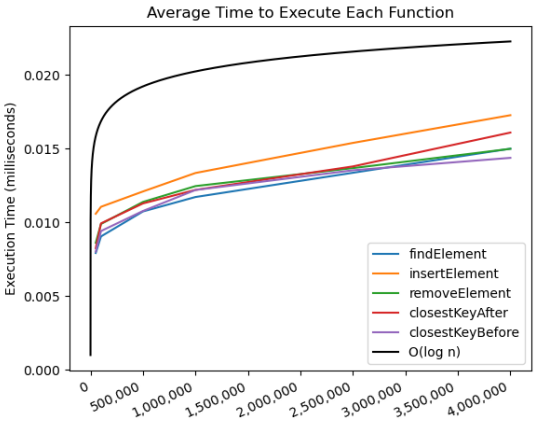
RESULTS TABLE

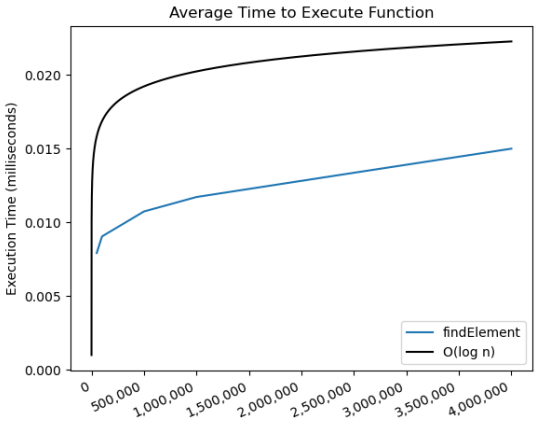
Average Run Time of Each Operation by Dictionary Size in Milliseconds

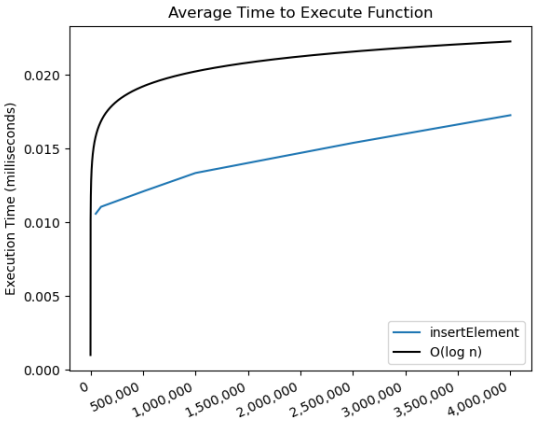
(time in milliseconds)	Dictionary Size (# of existing keys)					
	50,000	100,000	500,000	1,000,000	2,500,000	4,000,000
findElement	0.007932	0.009055	0.010748	0.011727	0.013371	0.015004
insertElement	0.010592	0.011064	0.012103	0.013355	0.015395	0.017272
removeElement	0.008625	0.009900	0.011394	0.012466	0.013686	0.014994
closestKeyAfter	0.008260	0.009940	0.011292	0.012210	0.013801	0.016096
closestKeyBefore	0.008026	0.009420	0.010780	0.012202	0.013539	0.014382

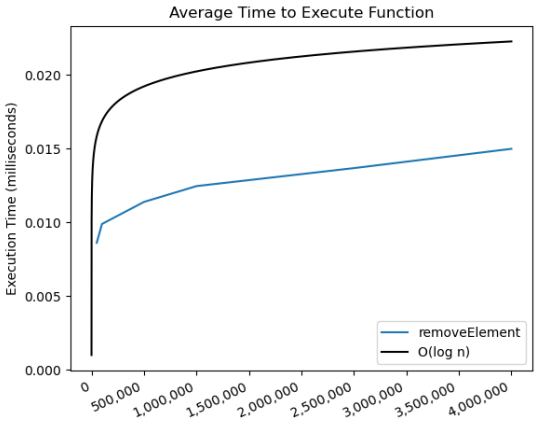
RESULTS GRAPHS

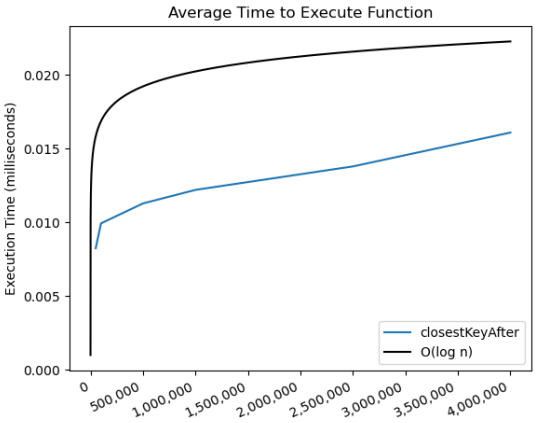
Graph of Average Run Times by Dictionary Size Compared to Log(n) Curve

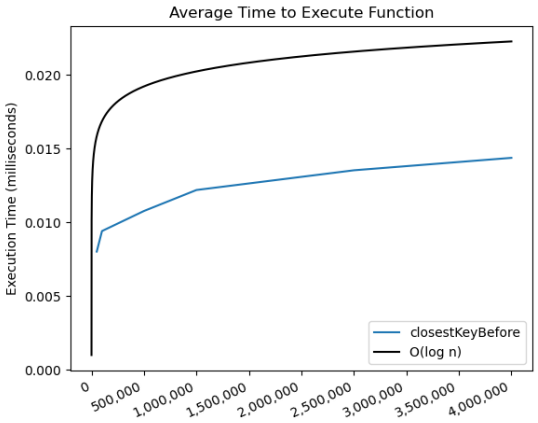












CONCLUSION

The time complexity of the skiplist's basic operations were consistent with the expected O(log n). These tests showed that a skiplist can be an efficient data structure for dictionary search, insertion, and deletion operations.