Assignment 2 – Report

Mike Kim

October 31st, 2022

Experiment 1:

T1: Long and skinny binary search tree

T2: More realistic binary search tree (realistic/natural insertion)

Number of nodes used: 20000

Unit for time: milliseconds

- Data:

	Average time for membership test	Average time for membership test:	The average depth of search for membership test
T1	852	980	9998
T2	410	993	4999

- Findings from experiment 1:
    - When comparing the average time for the membership test, the binary tree with realistic insertion is more efficient than the long and skinny binary search tree (t1). There is a 207.8 percent increase in time consumption compared to t2.
    - When comparing the average time for the membership test, there were minuscule differences between the two binary search trees. The first tree was more effective by a very small marginal amount.
    - When comparing the average depth of search for membership tests, the long and skinny binary search tree goes through twice as much in-depth compared to the binary search tree that went through a natural insertion procedure. With 20000 keys in both trees, on average, t1 would search to the 9998th depth whereas t2 would search to the 4999th depth, which is significantly more efficient.
    - From these three points, it is evident that t2 is more efficient when comparing the aspects of average time consumed and the average level of depths searched for membership tests as there was a minuscule difference for time consumption in membership tests.

Experiment 2:

Number of nodes used: 20000

Unit for time: milliseconds

- Data

	Avg. insertion time	Height	Average key depth	Minimum time for membership checks	Maximum time for membership checks	Mean time for membership checks
BST	871.08	20000	9998	0.001	0.201	0.050493
AVL	6335.61	15	12	0.001	0.024	0.00058325

	Minimum depths of search	Maximum depths of search	Mean depths of search
BST	399	18768	9998
AVL	13	14	12

- Findings from experiment 2:
    - When comparing the insertion time between a binary search tree and an AVL tree, the AVL tree significantly takes more time compared to the BST due to the nature of the complex insertion algorithm.
    - While comparing height and average key depth, the AVL tree is significantly shorter (15) compared to BST (20,000). This is because the AVL tree's algorithm ensures that the height is always O(log n). Due to the result of being exponentially shorter than the BST, the AVL tree's average key depth (12) also is exponentially shorter than the BST's (9998).
    - Time comparison for membership check:
        - Minimum time for membership check was equivalent to 0.001. This is because the time complexity to access the first node(the root of the tree) would be O(1).
        - For maximum time for membership check, BST significantly consumed more time (0.201 ms) compared to AVL trees (0.024 ms).
        - For mean time for membership checks the AVL tree was significantly faster than the BST.
        - Overall, the AVL tree is way more efficient in time when it comes down to membership checks.
    - Depths of search:
        - Minimum + Maximum
            - Binary search tree significantly had a higher depth of search (minimum: 399 maximum:18768) compared to AVL trees (minimum: 13 maximum: 14). This was inevitable as BST's depth was significantly higher than the AVL tree.
        - Mean
            - As mentioned above, BST's amount of nodes are exponentially larger than the AVL tree's node amount. Due to this result, the difference between the mean value of depths search is very significant (BST: 9998, AVL: 12).
        - Overall, AVL tree has to traverse less amount of depths than the BST trees.