ASSIGNMENT-3

AVL vs BST vs Treap

Roll-No-24101018

For Comparing I use five test files –

test0.txt, test1.txt, test2.txt, test3.txt, test4.txt

test0.txt contain –

- 10000 operations
- 70% insert
- 30 % delete

test1.txt contain –

- 8000 Operations
- 80% insert
- 20% delete

test2.txt contain –

- 12000 Operations
- 75% insert
- 25% delete

test3.txt contain –

- 14000 Operations
- 70% insert
- 30% delete

test4.txt contain –

- 11000 Operations
- 75% insert
- 25% delete

	A	B	C	D	E	F	G	H	I	J
1	filename	tree_type	total_ope	insert	count_no	hight	comparisi	rotation	average_hight	
2	test0.txt	traep	10000	7000	4000	28	123861	19499	2	
3	test0.txt	avl	10000	7000	4000	13	101764	5714	1	
4	test0.txt	bst	10000	7000	4000	23	119676	0	2	
5	test1.txt	traep	8000	6400	4800	29	99273	16012	2	
6	test1.txt	avl	8000	6400	4800	14	84768	4904	1	
7	test1.txt	bst	8000	6400	4800	24	99493	0	2	
8	test2.txt	traep	12000	9000	6000	27	153737	23846	2	
9	test2.txt	avl	12000	9000	6000	14	130779	7146	1	
10	test2.txt	bst	12000	9000	6000	25	148066	0	2	
11	test3.txt	traep	14000	9800	5600	28	185280	27807	2	
12	test3.txt	avl	14000	9800	5600	14	148814	7931	1	
13	test3.txt	bst	14000	9800	5600	26	166171	0	2	
14	test4.txt	traep	11000	8250	5500	27	148232	21615	2	
15	test4.txt	avl	11000	8250	5500	14	117873	6531	1	
16	test4.txt	bst	11000	8250	5500	24	131392	0	2	
17										
18										

Height Comparison –

Height of a tree is largest number of edge in a path from root node to leaf node.

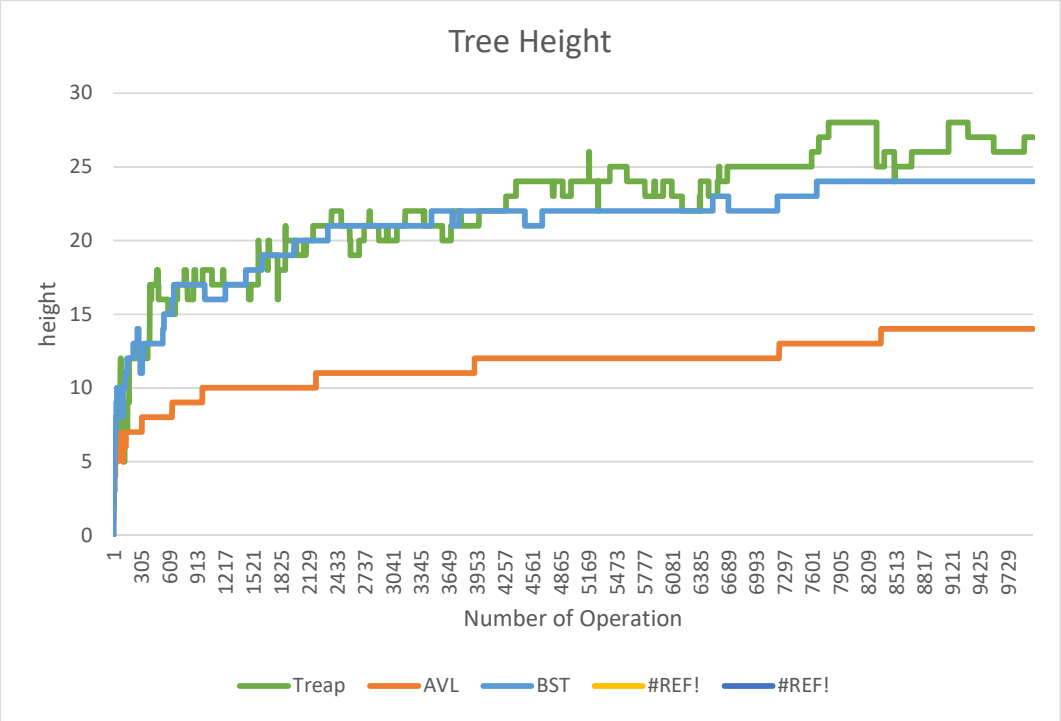
Operations –

Let we are doing 10000 operations out of which 70000 is insert operations and 3000 is delete operation

Total number of available node = 7000-3000=4000

Here we compare height of **AVL, BST and Treap**.

This Graph shows height of tree vs Number of operations. I have taken 70% operation as insertion and 30% deletion. After inserting 70 elements 30 element is randomly deleted (deleted only that element present which is present in tree) for remember the key values which is present in tree I use **"set"** data structure.



This graph shows that as we increase number of node in graph height of tree increases. Height of **Treap** and **BST** increases faster than **AVL**.

This graph shows the final height of tree

Observation and Comparison –

Binary tree –

In BST height of tree is varies from

$$O(\log_2 n) \text{ to } O(n)$$

So for 4000 nodes(test0) height of tree should be

$$\log_2 4000 \approx 12$$

So height will be from 12to 4000. Minimum height of tree may be 12 if tree is balanced.

Similarly for 4800 node(test1) height of tree should be

$$\log_2 4800 \approx 13$$

Similarly for 6000 node(test1) height of tree should be

$$\log_2 6000 \approx 13$$

Similarly for 5600 node(test1) height of tree should be

$$\log_2 5600 \approx 13$$

Similarly for 5500 node(test1) height of tree should be

$$\log_2 5500 \approx 13$$

Observations –

1. Height of tree is between n to ($\log_2 N$)
2. Height of BST is same as treap but much greater then AVL
3. Height of tree increasing when we insert the element in tree.

Conclusion –

In the graph height of BST is 23,24,25,26,24 which means we get correct result.

AVL tree –

In AVL height of tree should be

Minimum =($\log_2 n$) '

Maximum 1.44*$\log_2 n$

So for 4000 nodes height of tree should be

$$\log_2 4000 \approx 12$$

Similarly, for 4800 node(test1) height of tree should be

$$\log_2 4800 \approx 13$$

Similarly, for 6000 node(test1) height of tree should be

$$\log_2 6000 \approx 13$$

Similarly, for 5600 node(test1) height of tree should be

$$\log_2 5600 \approx 13$$

Similarly, for 5500 node(test1) height of tree should be

$$\log_2 5500 \approx 13$$

Observations –

1. Height of tree should be from (**$\log_2 N$) to 1.44*($\log_2 N$)**
2. Height of **AVL** is much less then **Treap** and **BST**
3. Height of tree increasing when we insert the element in tree. But it remains constant for some time.

Conclusion –

We get height of tree 13 and 14. So our empirical result matches with theoretical result. Then we can say that our implementation is correct.

Treap –

Treap is a Binary Search Tree, but not guaranteed to have height as O(Log n). The idea is to use Randomization and Binary Heap property to maintain balance with high probability.

Expected Height of treap is

$$\mathbf{O(\log_2 n)}$$

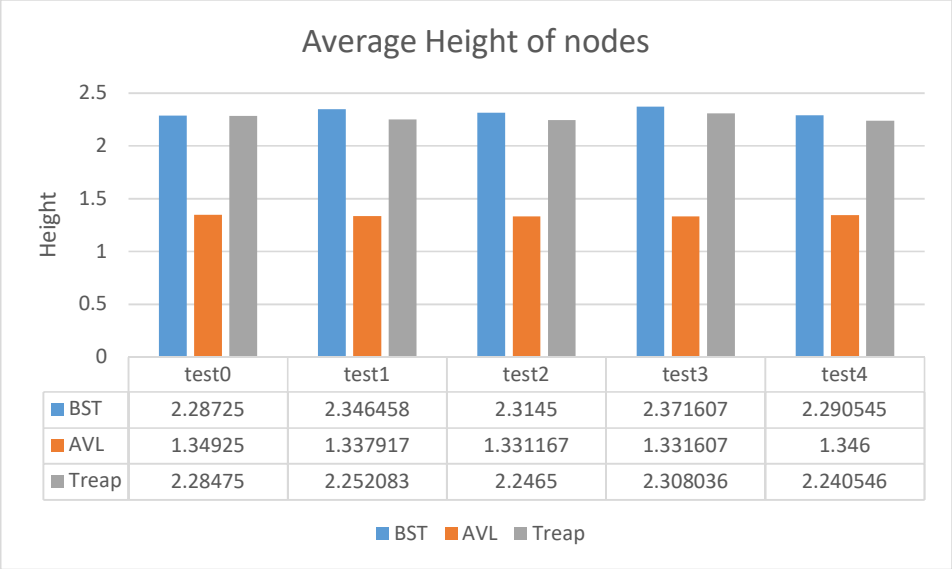
Observations-

1. Height of **treap** is 2* $\log_2$ N which is O($\log_2$ N)
2. Height of treap is nearly equal to height of BST.
3. Height of treap is greater than **AVL.**
4. When we delete the key of tree height of tree decreases

Conclusion –

Theoretical result will match with empirical result.

Average Height of nodes –



	test0	test1	test2	test3	test4
BST	2.28725	2.346458	2.3145	2.371607	2.290545
AVL	1.34925	1.337917	1.331167	1.331607	1.346
Treap	2.28475	2.252083	2.2465	2.308036	2.240546

$$\text{Average height of tree} = \frac{\text{sum of height of all nodes in tree}}{\text{\# of nodes}}$$

Observations –

1. In case of AVL average height of node is minimum.
2. In case of BST and Treap average height of nodes is same.

We know that height of AVL nodes is minimum and height of BST node and Treap node is high so average height of nodes should be greater in case of BST and Treap.

So average height of nodes should be minimum in case of AVL

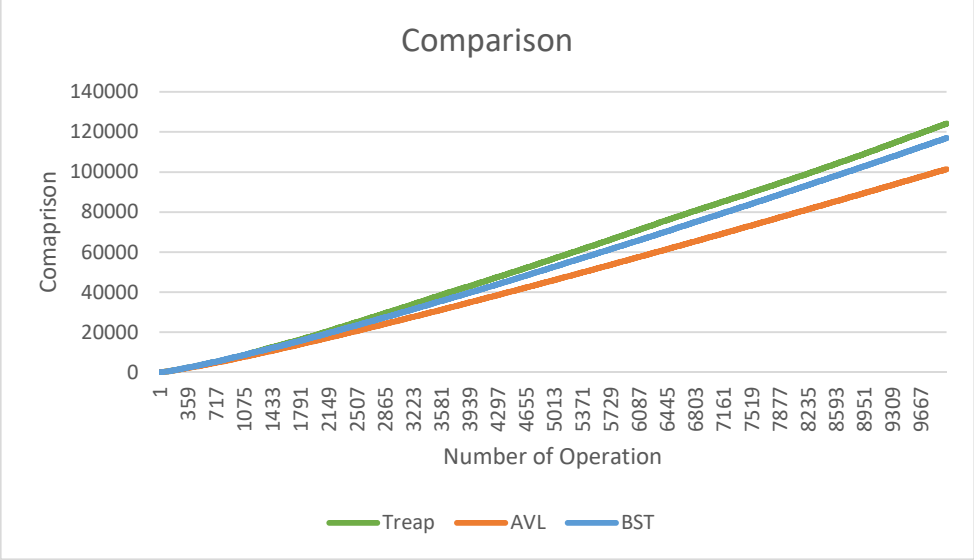
AVL<BST ≈ Treap

By analysing the graph, we can say that our empirical result is correct.

Number of Comparison –

Total number of comparison is number of key comparison which is occur during **insertion and deletion** of key elements in tree.

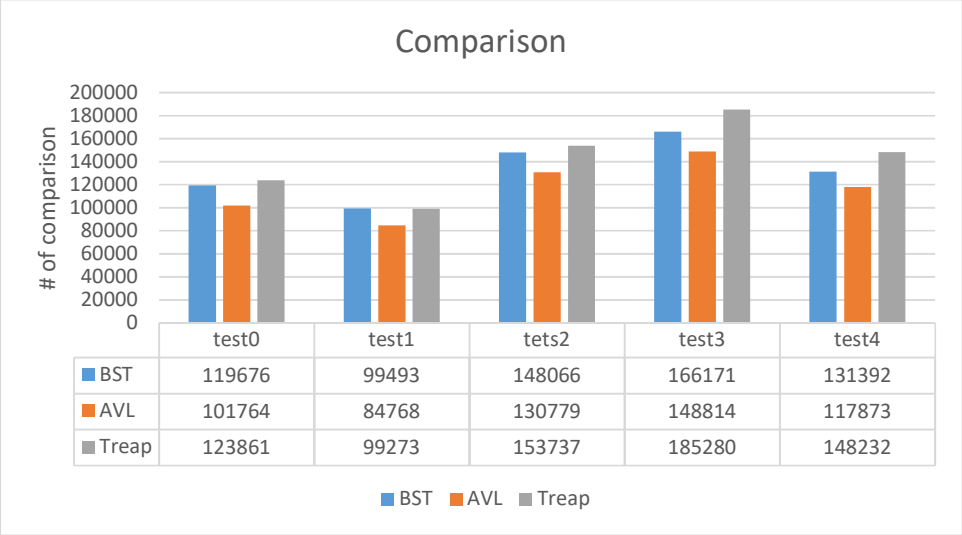
For insertion and deletion compare the key values for searching element. So we can say that in worst case we need to compare the O(h) elements so number of comparison depends upon height of tree and we already discuss height of Heap and BST greater then height of AVL. So the number of comparison should be minimum in case of AVL.



By analysing this graph, we can say that maximum comparison happens in treap and minimum comparison happen in AVL tree.

Total number of Comparison-

AVL<BST<Treap



	test0	test1	tets2	test3	test4
BST	119676	99493	148066	166171	131392
AVL	101764	84768	130779	148814	117873
Treap	123861	99273	153737	185280	148232

Observation and Comparison –

For insertion and deletion, we compare the key value with tree node value as for inserting an element we need to find the correct position of element and elements can only insert at leaf node so we can say that number of comparison depends on height of tree. If height of tree is high, then number of comparison will be high.

# of comparison $\propto$ height of tree

By theoretically we can say that –

1. Comparison in heap should be higher in Treap then BST and AVL.
2. Comparison in BST should be higher in AVL.

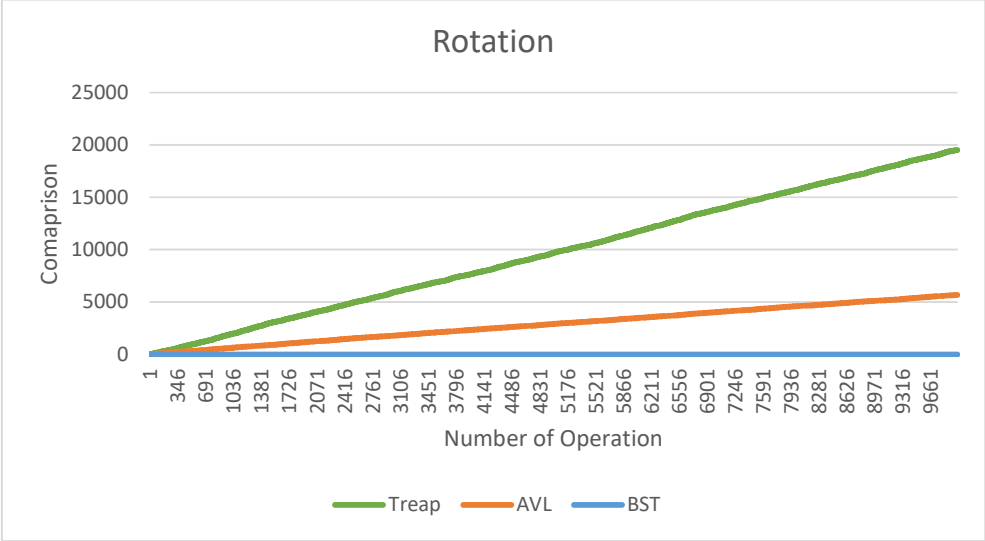
By analysing the graph, we can get that

Total number of comparison in Treap is higher than BST and AVL and lowest in AVL.

So our theoretical result will match with empirical result.

Number of Rotation –

A tree rotation moves one node up in the tree and one node down. It is **used to change the shape of the tree.** When tree fail the property rotation happen for rearranging the tree so that tree will follow their respective properties.

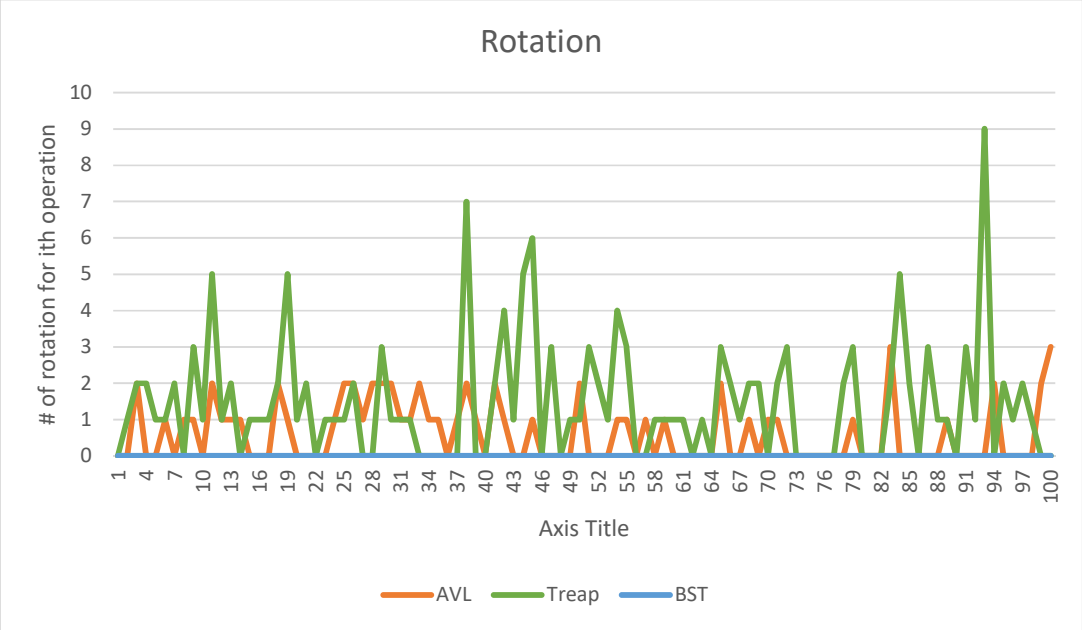


This graph shows the total number of rotation happen for different tree for 10000 operations

By analysing this graph, we can say that total number of comparison in Treap is very high then AVL and BST. And BST have zero rotation.

Total number of comparison –

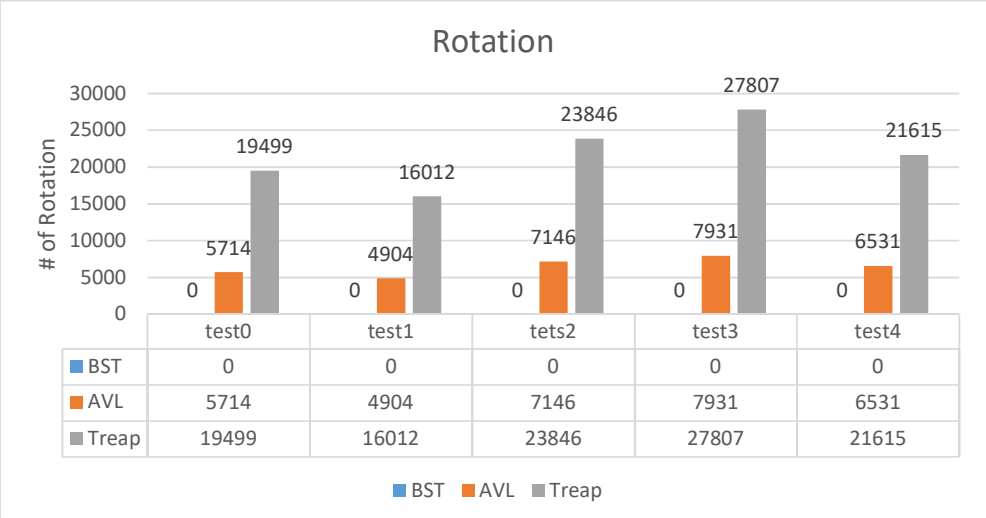
BST=0< AVL<Treap



Rotation vs i - th operation Graph

This graph shows the number of rotation vs first 100 operation

By analysing this graph, we can say that maximum number of rotation in AVL tree is two(R-L or L-R rotation) and maximum number of rotation in heap is O(h).



	test0	test1	tets2	test3	test4
BST	0	0	0	0	0
AVL	5714	4904	7146	7931	6531
Treap	19499	16012	23846	27807	21615

Observation and Comparison –

By analysing this graph, we can say that total number of rotation in Treap is very high as compare to AVL. Number of Rotation in BST is zero.

In AVL for balancing the height of tree we do 4 type of rotation L-Rotation, R-Rotation, L-R rotation, R-L rotation

1. L-Rotation increase count by 1
2. R-Rotation increases count by 1
3. L-R Rotation increase count by 2
4. R-L Rotation increase count by 2

In Treap-

We have only two type of rotation

1. L-Rotation
2. R-Rotation

Both rotation increase count by one.

Observations –

1. Rotation in Treap is very high
2. Rotation in BST is zero
3. In worst case number of rotation in Treap is O(h)
4. In worst case number of rotation in AVL is 2 otherwise 1

----------**XXXX**----------