Computer & Systems Engineering Department

Data Structures and Algorithms

Implementing Self Balanced BSTs (AVL & Red Black)

Contributors:

	Name	ID
1	Adel Mahmoud Mohamed Abdelrahman	20010769
2	Amr Ahmed Abdelazim	20011037
3	Marwan Essam Eldin	20011859
4	Mohamed Amr Abdelfattah	20011675

Time Analysis of the project parts:

1. Time Complexity of AVL basic functions:

Function	Time Analysis
Insert	O ($\log_2 n$)
Delete	O ($\log_2 n$)
Search	O ($\log_2 n$)
Size	O (1)
Height	O (1)

2. Time Complexity of Red-Black basic functions:

Function	Time Analysis
Insert	O ($\log_2 n$)
Delete	O ($\log_2 n$)
Search	O ($\log_2 n$)
Size	O (1)
Height	O (n)

3. Time Complexity of English dictionary basic operations:

Function	Time Analysis
Initialize	O (n)
Insert	O ($\log_2 n$)
Delete	O ($\log_2 n$)
Search	O ($\log_2 n$)
Batch insert	O (n $\log_2 n$)
Batch delete	O (n $\log_2 n$)
Size	O (1)
Height	O (n)

Comparison between AVL and RB under different test sizes measuring different aspects:

1. **Time taken for insertion, deletion, and height of n elements:**

N	AVL tree		RB tree	
100	Insertion	**11982 µs**	Insertion	**9386.8 µs**
	Deletion	**2472.8 µs**	Deletion	**2228 µs**
	Height	**8**	Height	**8**
1000	Insertion	**22543.2 µs**	Insertion	**15842.4 µs**
	Deletion	**3755.9 µs**	Deletion	**5587.3 µs**
	Height	**12**	Height	**12**
5000	Insertion	**31261 µs**	Insertion	**25717.2 µs**
	Deletion	**21753.7 µs**	Deletion	**11199.6 µs**
	Height	**15**	Height	**15**
10000	Insertion	**45668.5 µs**	Insertion	**30798.9 µs**
	Deletion	**23836.5 µs**	Deletion	**15571.9 µs**
	Height	**16**	Height	**16**
50000	Insertion	**131903 µs**	Insertion	**71527.2 µs**
	Deletion	**87064.9 µs**	Deletion	**31281.8 µs**
	Height	**19**	Height	**22**
100000	Insertion	**171664 µs**	Insertion	**102510.5 µs**
	Deletion	**114667µs**	Deletion	**53099.4 µs**
	Height	**20**	Height	**26**
200000	Insertion	**283562µs**	Insertion	**172087 µs**
	Deletion	**158707 µs**	Deletion	**102067.4 µs**
	Height	**21**	Height	**25**

400000	Insertion	**545516 µs**	Insertion	**316833.6 µs**
	Deletion	**291288 µs**	Deletion	**135412.3 µs**
	Height	**22**	Height	**27**

500000	Insertion	**583974 µs**	Insertion	**378528.4 µs**
	Deletion	**339629 µs**	Deletion	**176426 µs**
	Height	**23**	Height	**28**
600000	Insertion	**749134 µs**	Insertion	**454226.7 µs**
	Deletion	**427172 µs**	Deletion	**229001.5 µs**
	Height	**23**	Height	**28**
800000	Insertion	**927828 µs**	Insertion	**526490.6 µs**
	Deletion	**591314 µs**	Deletion	**269388.7 µs**
	Height	**23**	Height	**29**
1000000	Insertion	**1048402 µs**	Insertion	**624711.4 µs**
	Deletion	**675250 µs**	Deletion	**343832.9 µs**
	Height	**24**	Height	**30**

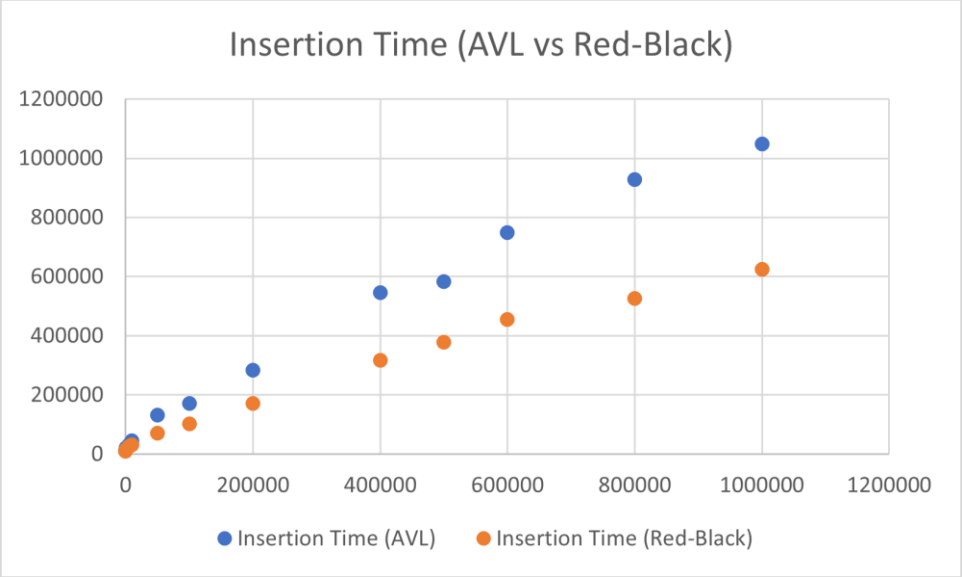
2. Time taken for search in n elements:

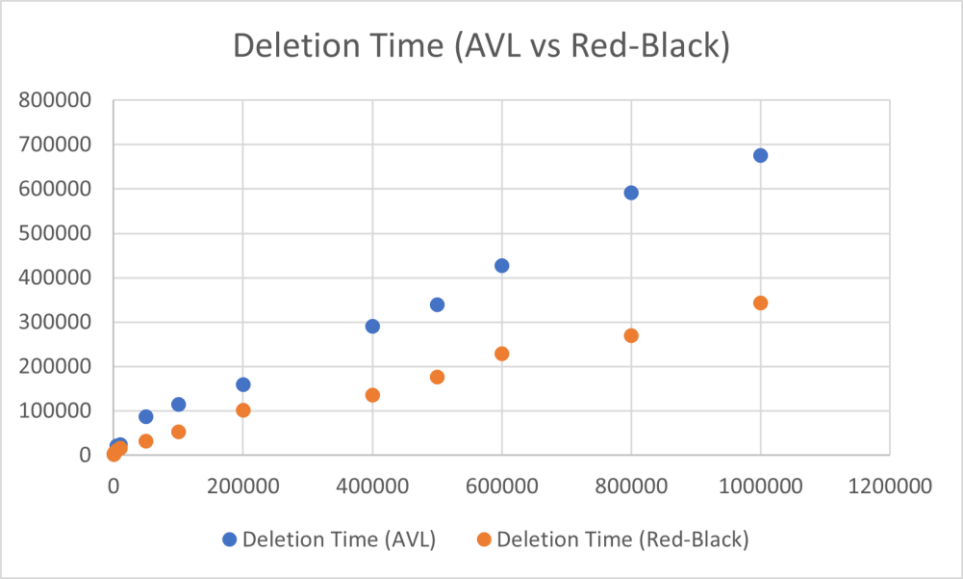
N	AVL tree	RB tree
100	**68.275 µs**	**67.55 µs**
1000	**55.525 µs**	**63.35 µs**
10000	**54.6 µs**	**65.35 µs**

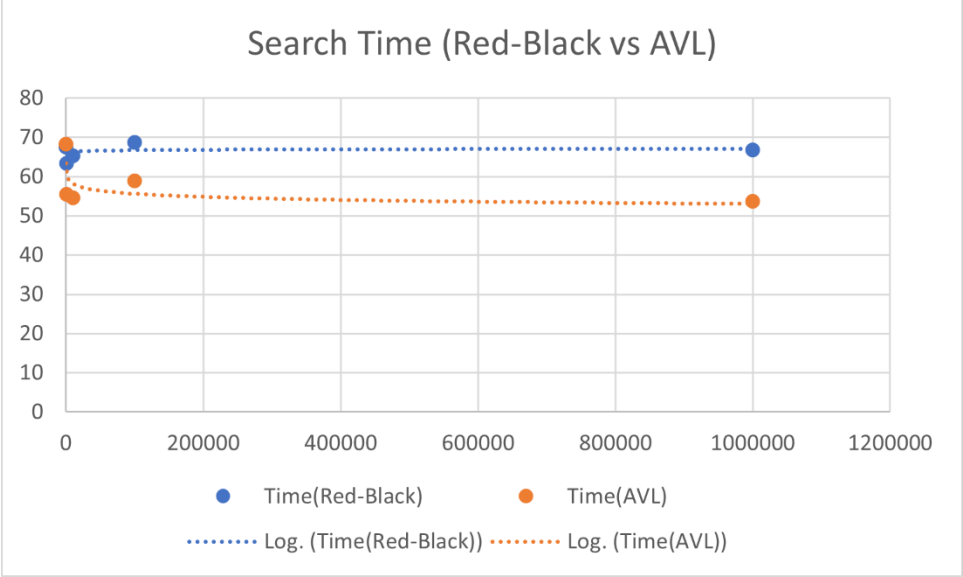
So now we can get the weighted avg of the time as follows:

	AVL	Red-Black
Insertion	**379,453.325 µs**	**193,321.7333 µs**
Deletion	**474,304.6894 µs**	**236,389.5467 µs**
Searching (for one element)	**54.14 us**	**66.89 us**

Statistical confirmation of the results we got:




Deletion Time (AVL vs Red-Black)
800000
700000
600000
500000
400000
300000
200000
100000
0
0
200000
400000
600000
800000
1000000
1200000
Deletion Time (AVL)
Deletion Time (Red-Black)


Search Time (Red-Black vs AVL)
80
70
60
50
40
30
20
10
0
0
200000
400000
600000
800000
1000000
1200000
Time(Red-Black)
Time(AVL)
Log. (Time(Red-Black))
Log. (Time(AVL))

Conclusion:

From the above results, it has been evident that the AVL tree involves more rotations in insertion/deletion while preserving a shorter height. Therefore, AVL trees are well suited for situations where a lot of search operations are required while having occasional insertion/deletion operations. This was evident in the dictionary application as the AVL tree was faster at searching but slower at insertion/deletion.

On the other hand, Red-Black trees involve less rotation operations in insertion/deletion as Red-Black tree conditions are more relaxed. Due to the same reason, the red-black tree has a longer height, where one path can reach at most twice the length of the other path in the tree. This makes the Red-Black tree a candidate for situations where a lot of insertion/deletion operations are required with less frequent search operations. This appeared in the dictionary application when the Red-Black tree insertion/deletion operations were faster but slower at searching.

We can say that there is a tradeoff between maintaining a minimal tree height and minimizing the number of rotation operations. This depends on the nature of the application in which we are using the trees, i.e., the frequency of deletions/insertions [writes] vs the frequency of searches [reads].