Vaibhav Patel

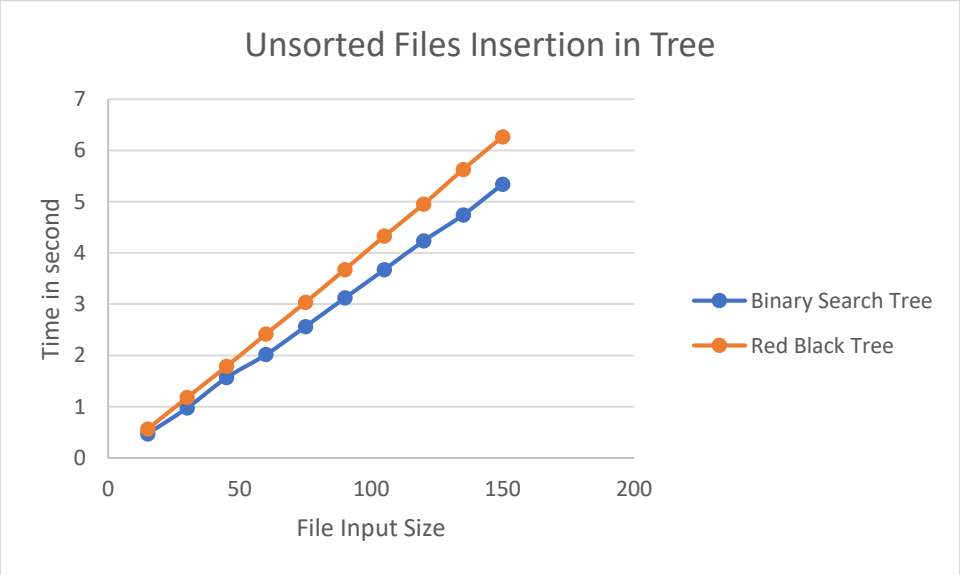
CS-340 Project II

Report and Plots

All Algorithms are implemented from pseudo code provided on class slides.

Here is a table for inserting unsorted files into tree.

File Size(in K Unit)	Binary Search Tree	Red Black Tree
15	0.47	0.562
30	0.975	1.177
45	1.564	1.782
60	2.014	2.415
75	2.561	3.033
90	3.119	3.673
105	3.671	4.325
120	4.234	4.952
135	4.741	5.628
150	5.34	6.264

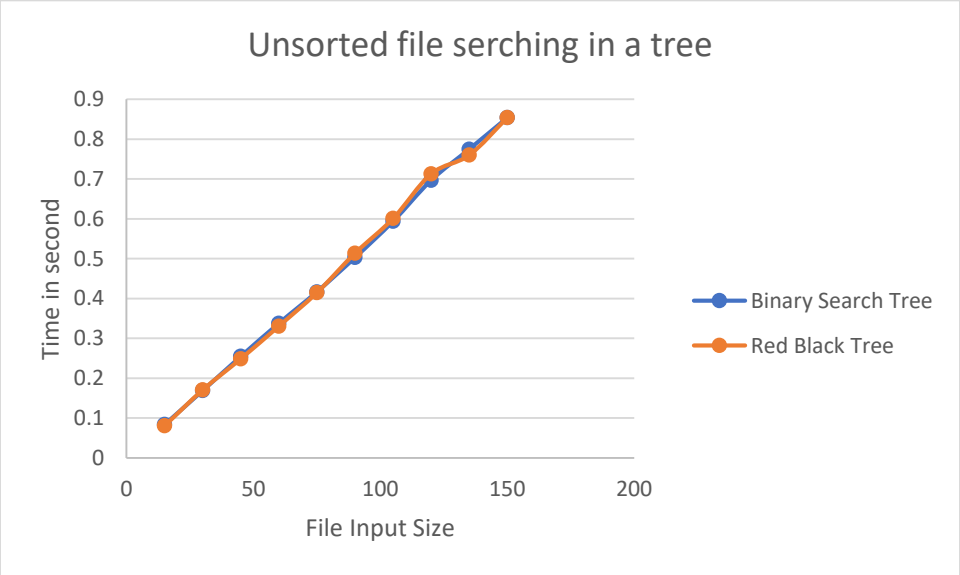


Binary Search Tree: It took Θ (log N) time to insert which was expected too. (Average Case)

Red Black Tree: It took Θ (log N) time to insert which was expected too. (Average Case)

Here is Table for searching in a unsorted files.

File Size(in K Unit)	Binary Search Tree	Red Black Tree
15	0.084	0.081
30	0.169	0.171
45	0.255	0.249
60	0.338	0.331
75	0.417	0.415
90	0.503	0.514
105	0.594	0.601
120	0.697	0.713
135	0.775	0.76
150	0.854	0.854

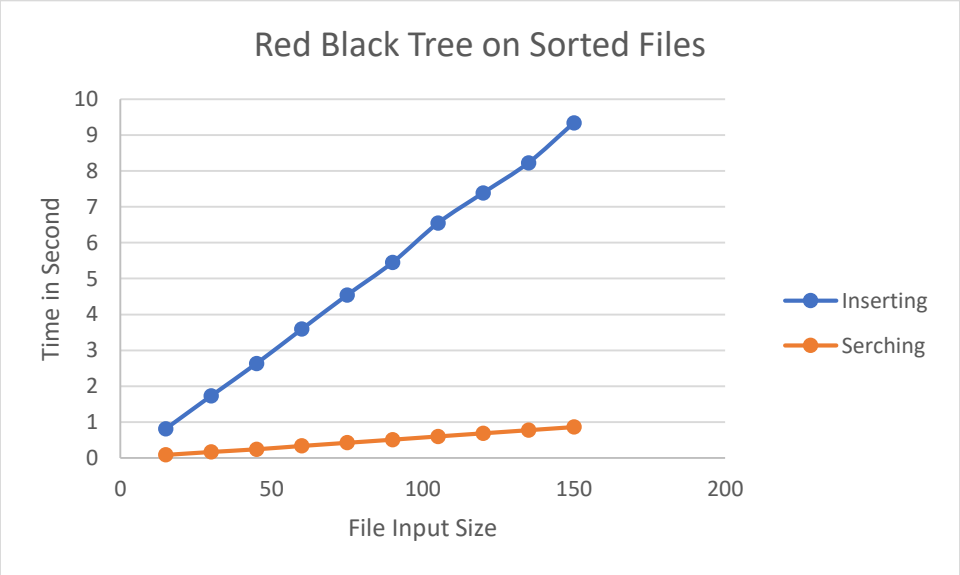


Binary Search Tree: It took Θ (log N) time to search which was expected too. (Average Case)

Red Black Tree: It took Θ (log N) time to search which was expected too. (Average Case)

Here is table for running red black tree on sorted file.

File Size(in K Unit)	Inserting	Searching
15	0.813	0.086
30	1.73	0.168
45	2.631	0.243
60	3.591	0.336
75	4.541	0.424
90	5.448	0.511
105	6.544	0.601
120	7.389	0.689
135	8.229	0.776
150	9.341	0.863



Red Black Tree: It took Θ (log N) time to search and insert which was expected too. (Worst Case)

Binary Search Tree: I did implement iterative insertion for BST but it took more than 1 hour to insert 150K word file so implemented recursively. Since it is recursively implement it will give stack overflow when running BST on sorted files because too many recursive calls.