HOMEWORK 3 - MASTER THEOREM

- DAKSH BHUVA
- 10475468

(1) $T(n) = 3T(n/2) + n^2$

→ General form : $T(n) = aT(n/b) + f(n)$ ......(i)
Comparing with eq (i) ($\because$ a is constant)

$\therefore a = 3,\ b = 2,\ f(n) = n^2$
So $\log_b a = \log_2 3 = 1.58 < 2$ ......(ii)

→ From (ii), we can say $f(n) < n^{\log_b a + \epsilon}$
; ϵ is constant.

→ Case 3 is implied.
$\therefore$ This means $T(n) = f(n) = \Theta(n^2)$

(2) $T(n) = 2^n T(n/2) + n^n$

→ Comparing with general form
$a = 2^n,\ b = 2,\ f(n) = n^n$

→ But, here according to given condition, 'a' must be constant. Thus, 2^n is constant

→ Master theorem doesn't apply.

(3) $T(n) = 3T(n/4) + n\log n$

→ Comparing with general form
$a = 3,\ b = 4,\ f(n) = n\lg n$

→ Now, $n^{\log_b a} \Rightarrow n^{\log_4 3} = n^{0.8}$
If we take $\epsilon = 0.2$, then
$3f(n/4) \leq 3/4\ f(n)$ for $c = 3/4$

⇒ Here, Case 3 is implied | $f(n) = \Omega(n^{\log_b a + \epsilon})$

$T(n) = \Theta(n\lg n)$

(4) $T(n) = 2T(n/2) + n/\log n$

→ Comparing with general form
$a = 2,\ b = 2,\ f(n) = n/\log n$

→ Now, $f(n) = n/\log n$ is not smaller than $n^{\log_b a + \epsilon}$ (Case - 1)
$n^{\log_2 2 - \epsilon} > n/\log n$ for any $\epsilon > 0$

→ Master Theorem doesn't apply.

(5) $T(n) = 0.5\, T(n/2) + 1/n$

→ Comparing with general form
$a = 0.5$, $b = 2$, $f(n) = 1/n$

→ we would see that 'a' is less than 1.

→ Master Theorem doesn't apply.

Assignment 2: Master Theorem, Binary-Search and Red-Black Trees

Abstract:

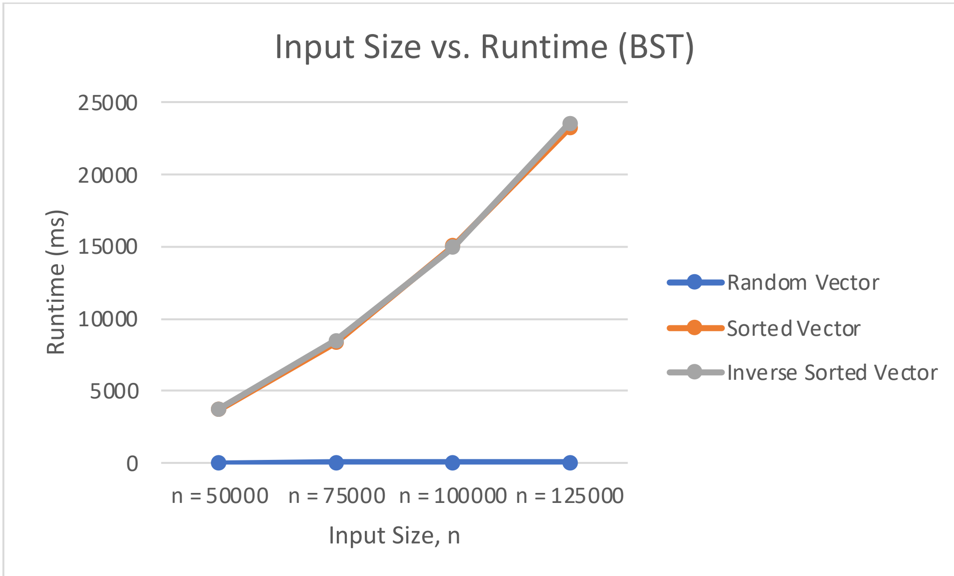
I have implemented and analyzed the time-complexity of Binary-Search Tree and Red-Black Tree. By comparing the run-times for different values 'n', I have documented the results and plotted the graphs of runtime for a better understanding of these algorithm's running times.

Result:

A. Binary-Search Tree (BST)

Table 1: BST Runtime in ms

	Input Type	Time (ms)	Duplicates
n = 50000	Random Vector	18.3	0.4
	Sorted Vector	3726.1	0
	Inverse Sorted Vector	3772.9	0
n = 75000	Random Vector	25.5	1.5
	Sorted Vector	8362.6	0
	Inverse Sorted Vector	8508.2	0
n = 100000	Random Vector	36.1	2.1
	Sorted Vector	15062.7	0
	Inverse Sorted Vector	14977.4	0
n = 125000	Random Vector	43.9	2.8
	Sorted Vector	23218.3	0
	Inverse Sorted Vector	23573.6	0
n = 250000	Random Vector	73.5	13.1
	Sorted Vector	-	-
	Inverse Sorted Vector	-	-
n = 500000	Random Vector	153.1	55.7
	Sorted Vector	-	-
	Inverse Sorted Vector	-	-
n = 1000000	Random Vector	369.2	239.1
	Sorted Vector	-	-
	Inverse Sorted Vector	-	-

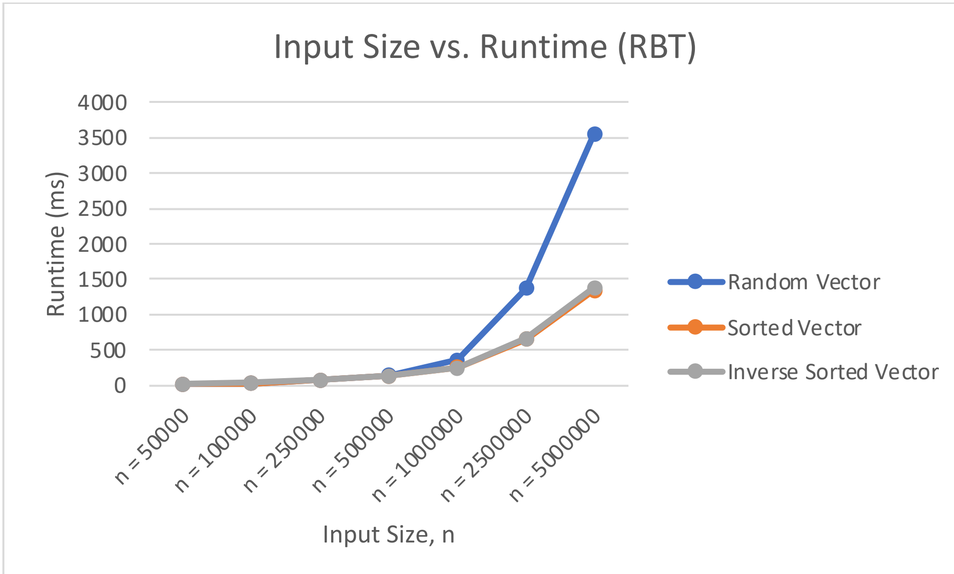


Graph 1

B. Red-Black Tree (RBT)

Table 2: RBT Runtime in ms

	Input Type	Time (ms)	Height	Case 1	Case 2	Case 3	Left Rotate	Right Rotate	Duplicates
n = 50000	Random Vector	20.5	11	25623.1	9859.5	19378.3	14580.9	14648.8	0.1
	Sorted Vector	17.1	16	49966	0	49971	49971	0	0
	Inverse Sorted Vector	17.3	16	49966	0	49971	0	49971	0
n = 100000	Random Vector	30.1	11	51277.5	19537.2	38972.2	29313.6	29196.1	3.5
	Sorted Vector	31.9	17	99964	0	99969	99969	0	0
	Inverse Sorted Vector	33.8	17	99964	0	99969	0	99969	0
n = 250000	Random Vector	76.2	12	128429.5	48495.9	96997.7	72592.1	72900.2	16.3
	Sorted Vector	73.5	18	249961	0	249967	249967	0	0
	Inverse Sorted Vector	73.4	18	249961	0	249967	0	249967	0
n = 500000	Random Vector	142.6	13	256443.1	97572.5	194559.9	145800.3	146331.3	56.1
	Sorted Vector	130.7	19	499959	0	499965	499965	0	0
	Inverse Sorted Vector	130.1	19	499959	0	499965	0	499965	0
n = 1000000	Random Vector	354.6	13	513583.3	194473.7	388433.1	291377.4	291529.6	220.7
	Sorted Vector	253.1	20	999957	0	999963	999963	0	0
	Inverse Sorted Vector	245.3	20	999957	0	999963	0	999963	0
n = 2500000	Random Vector	1367.2	14	1283750.6	484765.1	971365.9	728260.5	727870.4	1449.4
	Sorted Vector	651.5	21	2499952	0	2499960	2499960	0	0
	Inverse Sorted Vector	656.9	21	2499952	0	2499960	0	2499960	0
n = 5000000	Random Vector	3556.9	15	2566711.1	970502.5	1941789.8	1456650.7	1455641.3	5715.8
	Sorted Vector	1340.3	22	4999950	0	4999958	4999958	0	0
	Inverse Sorted Vector	1377.8	22	4999950	0	4999958	0	4999958	0



Graph 2

Discussion:

a. Binary-Search Tree

- As the value of n increases, there is exponential growth in the running time of the BST algorithm for Sorted & Inverse Sorted vectors and linear growth for Random vector. The same can be inferred from Table 1 and Graph 1.
- As the value of 'n' increases, the duplicates also increase for Random Vector input as seen from Table 1.
- Running time of Sorted & Inverse Sorted is almost equivalent to each other in all the cases. Hence their graph also coincides with each other.
- For Sorted & Inverse Sorted vectors the BST algorithm gives 'Segmentation Fault' for values of n > 170000 because for:
    - Sorted Vector - all the values will fall on the right side of the parent node for 'n' number of times.
    - Inverse Sorted vector - all the values will fall on the left side of the parent node for 'n' number of times.

b. Red-Black Tree

- As the value of n increases, there is exponential growth in the running time of the RBT algorithm. The same can be inferred from Table 2 and Graph 2.
- Running time of Sorted & Inverse Sorted is almost equivalent to each other in all the cases. Hence their graph also coincides with each other.
- I have also calculated the black height in *'check_black_height'* function and it satisfies the red-black property 5.

- As the value of ‘n’ increases, the height increases as seen from Table 2. And the height for Random vector input is always smaller than that of Sorted & Inverse Sorted vector input. Also, the height of the Sorted & Inverse Sorted vector input is the same.
- As the value of ‘n’ increases, the duplicates also increase for Random Vector input as seen from Table 2.
- In Table 2, ‘Case 2’ is zero for Sorted & Inverse Sorted vector input, non-zero for Random Vector input. Also, ‘Left Rotate’ is zero for Inverse Sorted vector input and ‘Right Rotate’ is zero for Sorted vector input.

Conclusion:

From the above results I conclude that that, Binary-search tree works well for smaller input sizes and if it is a random vector. Overall, Red-black tree reduces the running time to a great extent as compared to Binary-search tree for random, sorted, and inverse sorted vectors. Hence, Red-Black Tree is better amongst the given two.