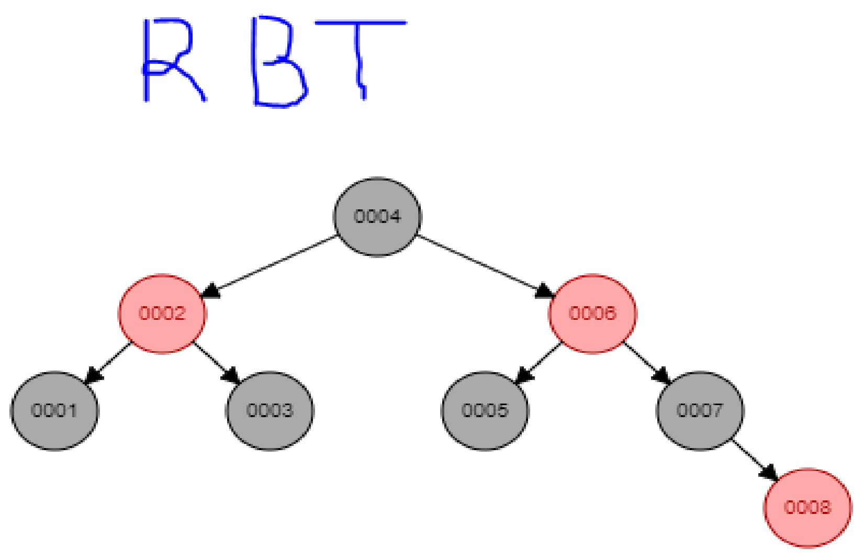
BINARY SEARCH TREE Report

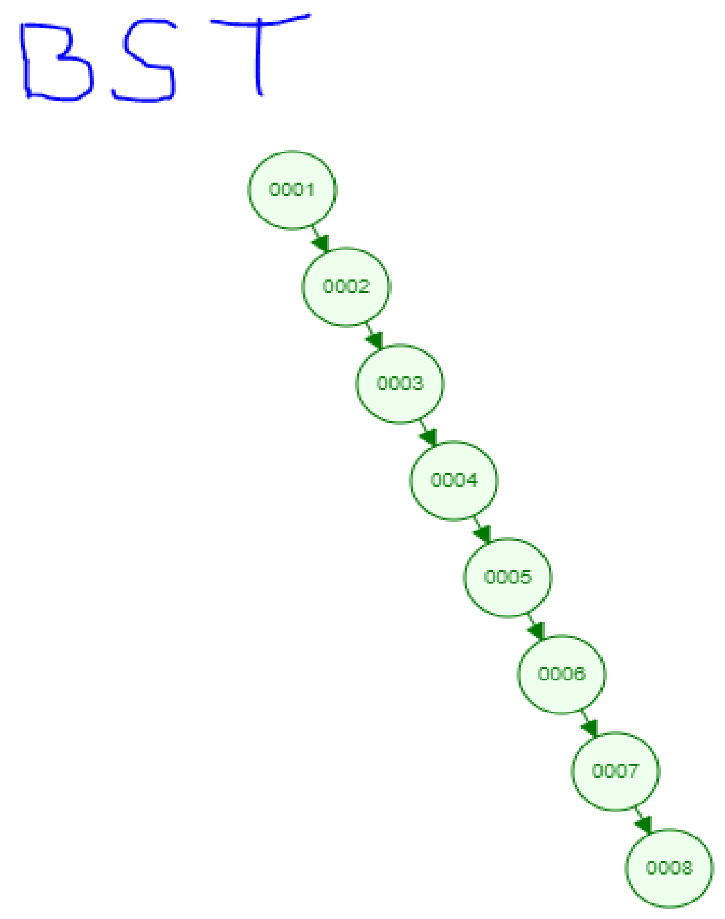
Input Size (n)	Time(t) for Direction(d)=random(0) (in ms)	Time(t) for Direction(d)=Sorted(1) (in ms)	Time(t) for Direction(d)=Reversed(-1) (in ms)	Height of the binary search tree =random(0)	Height of the binary search tree =Sorted(1)	Height of the binary search tree =Reversed(-1)	No Of Duplicates =Random(0)	No of Duplicates =Sorted(1)	No of Duplicates =Reversed(-1)
50000 (This reading has been taken by adding n_array[i] %= 5000 in sort.cpp)	0.3	2	2.6	29.1	50000	50000	45000	0	0
100000 (This reading has been taken by adding n_array[i] %= 5000 in sort.cpp)	0.3	5.2	6.5	28.8	100000	100000	95000	0	0
250000 (This reading has been taken by adding n_array[i] %= 5000 in sort.cpp)	0.5	X	X						
500000 (This reading has been taken by adding n_array[i] %= 5000 in sort.cpp)	0.2	X	X	29					
1000000(This reading has been taken by adding n_array[i] %= 5000 in sort.cpp)	0	X	X	28.1					
2500000 **(This reading has been taken by removing n_array[i] %= 5000 from sort.cpp)**	321.8	X	X	52.7			1460.5		
5000000 **(This reading has been taken by removing n_array[i] %= 5000 from sort.cpp**	553	X	X	56.3			5808.5		

Observations :

1) Execution creates core dump for BST with input size >= 250000 for the direction sorted and reversed. However it doesn't occur for RBT because of it's self-balancing characteristics. In case of BST, execution can't reach the bottom most leaf,hence it fails.

2) In the below pictures, for the input size 8, with the values ranging from 1 to 8, below are the respetive tree structures. In RBT, we'll see a balanced tree where as for BST it's always one direction.





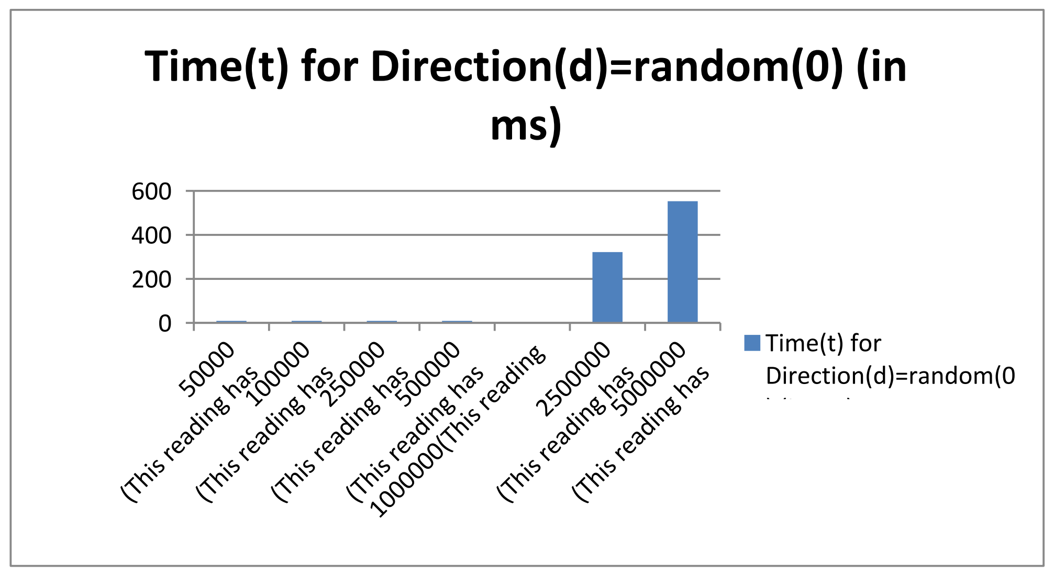
Analysis :

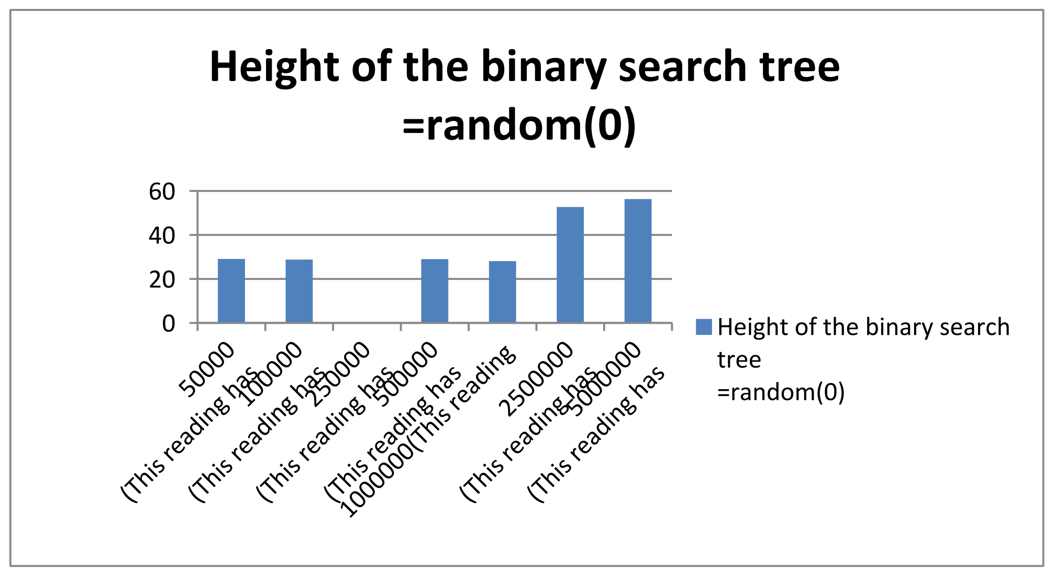
1) It can be inferred from the readings that, running time grows as the input size grows. The height of the tree grew as the input size grew.

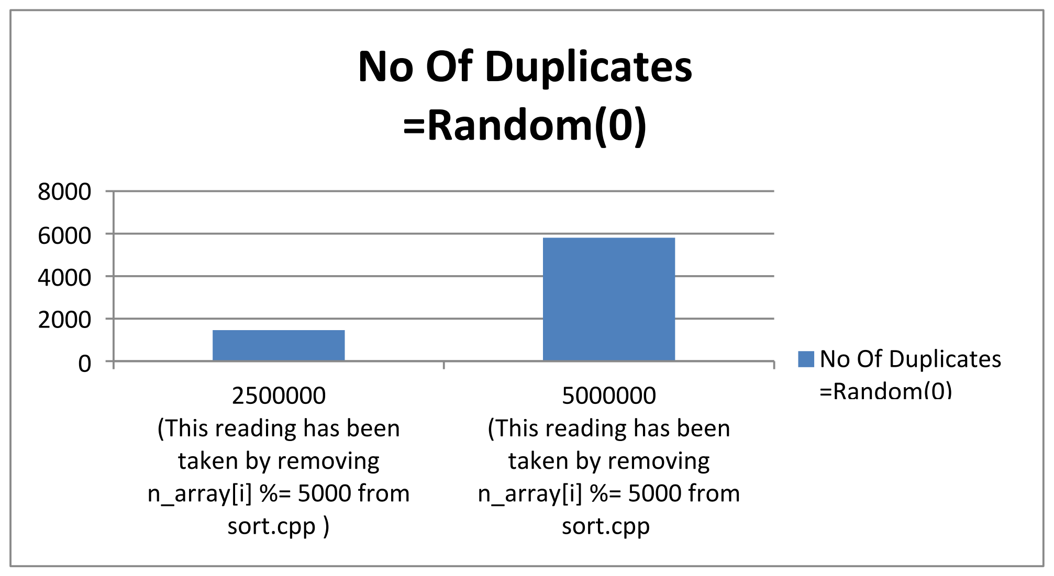
2) No of duplicate nodes increases as the input size increases.

3) Experiment is condcuted by remvoing modulo 5000 operations from sort.cpp for 2500000 and 5000000. Hence there are huge changes in the graph for those 2 values.

4) Running time for inorder tree traversal is theta(n) then, we can observe that, as input size increases, running time of inorder traversal increases


Time(t) for Direction(d)=random(0) (in ms)
50000 (This reading has
100000 (This reading has
250000 (This reading has
500000 (This reading has
1000000(This reading
2500000 (This reading has
5000000 (This reading has
Time(t) for Direction(d)=random(0


Height of the binary search tree =random(0)
50000 (This reading has
100000 (This reading has
250000 (This reading has
500000 (This reading has
1000000(This reading
2500000 (This reading has
5000000 (This reading has
Height of the binary search tree =random(0)


No Of Duplicates =Random(0)
2500000 (This reading has been taken by removing n_array[i] %= 5000 from sort.cpp)
5000000 (This reading has been taken by removing n_array[i] %= 5000 from sort.cpp
No Of Duplicates =Random(0)