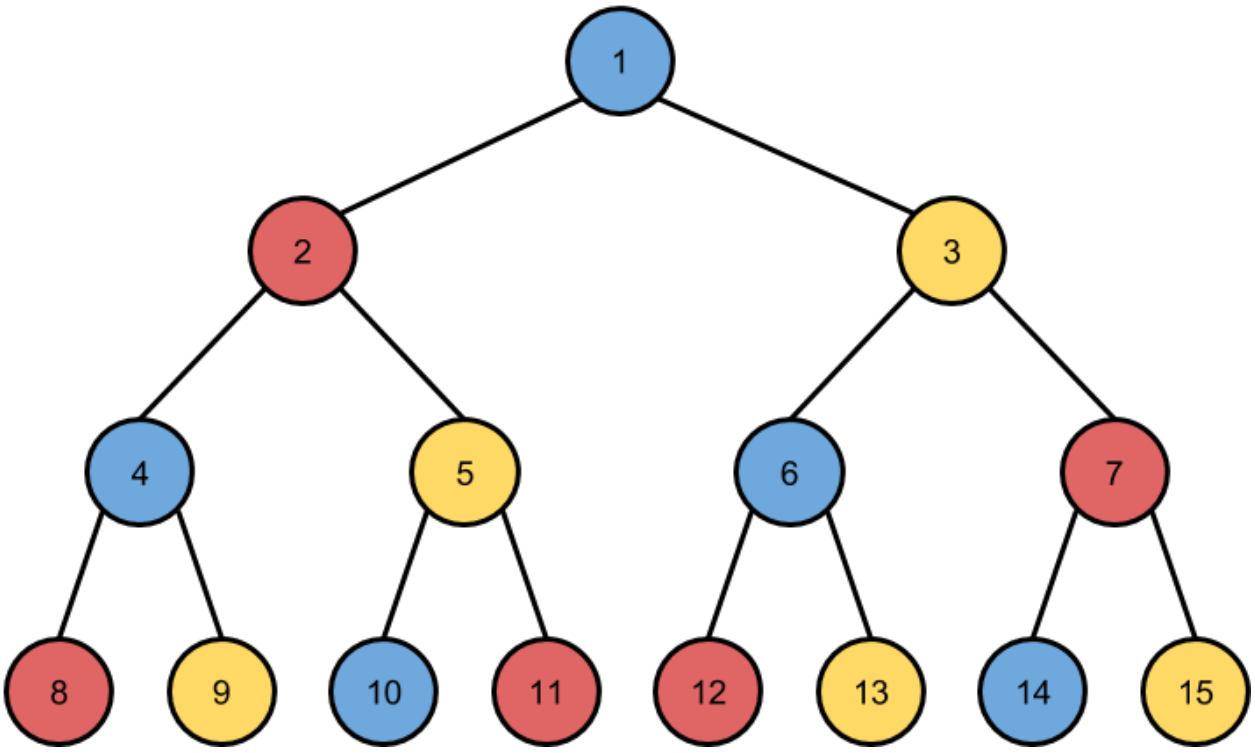
HOMEWORK 2 REPORT

Binary Search Tree



Tyler Adam Martinez

03.04.2022

CSCE 3110 – Data Structures & Algorithms

INTRODUCTION

The assignment was to implement a binary search tree (BST) data structure from scratch and perform a series of tests on the BST data structure. The functions of the BST class include insert, size, depth, search, and remove. The goal of the tests is to find the worst and average run times for all the functions of the BST class.

There are two distinct groups of tests unsorted inputs and sorted inputs; nevertheless, both groups of tests randomly generate floating-point values between a finite pre-set range and use these values as the inputs for the tests. The pre-set ranges of both groups of tests include hundred, a thousand, and ten thousand unique floating-point values. The difference between the unsorted and sorted tests is that the sorted test sorts all the randomly generated floating-point values from lowest to highest then precedes to input the values into the BST, whereas the unsorted tests do not.

HYPOTHESIS

Because BSTs searches are done by moving left or right depending on whether the search for value is greater or less than the current node, the sorted tests will probably run faster than the unsorted tests.

PROCEDURE

1. Create an implementation of a BST
2. Create a function to populate the BST with random floating-point sorted and unsorted of a hundred, a thousand, or ten thousand depending on the test.
3. Capture the time of each of the BST functions
4. Display the finding in an easy to read table

DATA

Unsorted Binary Tree Tests Average Case		
100 Elements	1k Elements	10k Elements
Insert Time: 0.000000483s	Insert Time: 0.000000273s	Insert Time: 0.000000395s
Size Time: 0.000001878s Size of Tree: 0	Size Time: 0.000016096s Size of Tree: 0	Size Time: 0.000171598s Size of Tree: 0
Depth Time: 0.000001282s Depth of Tree: 0	Depth Time: 0.000014511s Depth of Tree: 0	Depth Time: 0.000202194s Depth of Tree: 0
Search Time: 0.000000107s	Search Time: 0.000000099s	Search Time: 0.000000079s
Remove Time: 0.000023965s	Remove Time: 0.000284644s	Remove Time: 0.00386699s

Unsorted Binary Tree Tests Worst Case		
100 Elements	1k Elements	10k Elements
Insert Time: 0.000000709s	Insert Time: 0.000000346s	Insert Time: 0.000000521s
Size Time: 0.000003107s Size of Tree: 0	Size Time: 0.000019265s Size of Tree: 0	Size Time: 0.000186471s Size of Tree: 0
Depth Time: 0.00000138s Depth of Tree: 0	Depth Time: 0.000015501s Depth of Tree: 0	Depth Time: 0.000232913s Depth of Tree: 0
Search Time: 0.000000139s	Search Time: 0.000000142s	Search Time: 0.000000112s
Remove Time: 0.000033006s	Remove Time: 0.000514759s	Remove Time: 0.005445551s

Sorted Binary Tree Tests Average Case		
100 Elements	1k Elements	10k Elements
Insert Time: 0.000000382s	Insert Time: 0.000003496s	Insert Time: 0.000078149s
Size Time: 0.000000851s Size of Tree: 0	Size Time: 0.000005746s Size of Tree: 0	Size Time: 0.000098578s Size of Tree: 0
Depth Time: 0.000000501s Depth of Tree: 0	Depth Time: 0.00000507s Depth of Tree: 0	Depth Time: 0.000087646s Depth of Tree: 0
Search Time: 0.000000101s	Search Time: 0.000005246s	Search Time: 0.000059461s
Remove Time: 0.000123503s	Remove Time: 0.078842781s	Remove Time: 8.123707747s

Sorted Binary Tree Tests Worst Case		
100 Elements	1k Elements	10k Elements
Insert Time: 0.000000472s	Insert Time: 0.000003978s	Insert Time: 0.000129828s
Size Time: 0.00000146s Size of Tree: 0	Size Time: 0.000007882s Size of Tree: 0	Size Time: 0.00016377s Size of Tree: 0
Depth Time: 0.000000593s Depth of Tree: 0	Depth Time: 0.00000574s Depth of Tree: 0	Depth Time: 0.000137167s Depth of Tree: 0
Search Time: 0.000000137s	Search Time: 0.00000584s	Search Time: 0.000074228s
Remove Time: 0.000238591s	Remove Time: 0.081452735s	Remove Time: 8.186982344s

RESULTS

The unsorted tests outperformed the sorted tests in almost every category. For example, in the remove function, the difference between the worst case of the unsorted test and the sorted test is a 199.734% difference; however, the differences between the average unsorted case and worst unsorted case are not that comparable, and the same goes for the sorted cases as well. To conclude, the number of data points increases the runtime, and inputting sorted data also increases the runtime of each BST function except for size, which is an outliner by decreasing in some cases under the sorted tests.