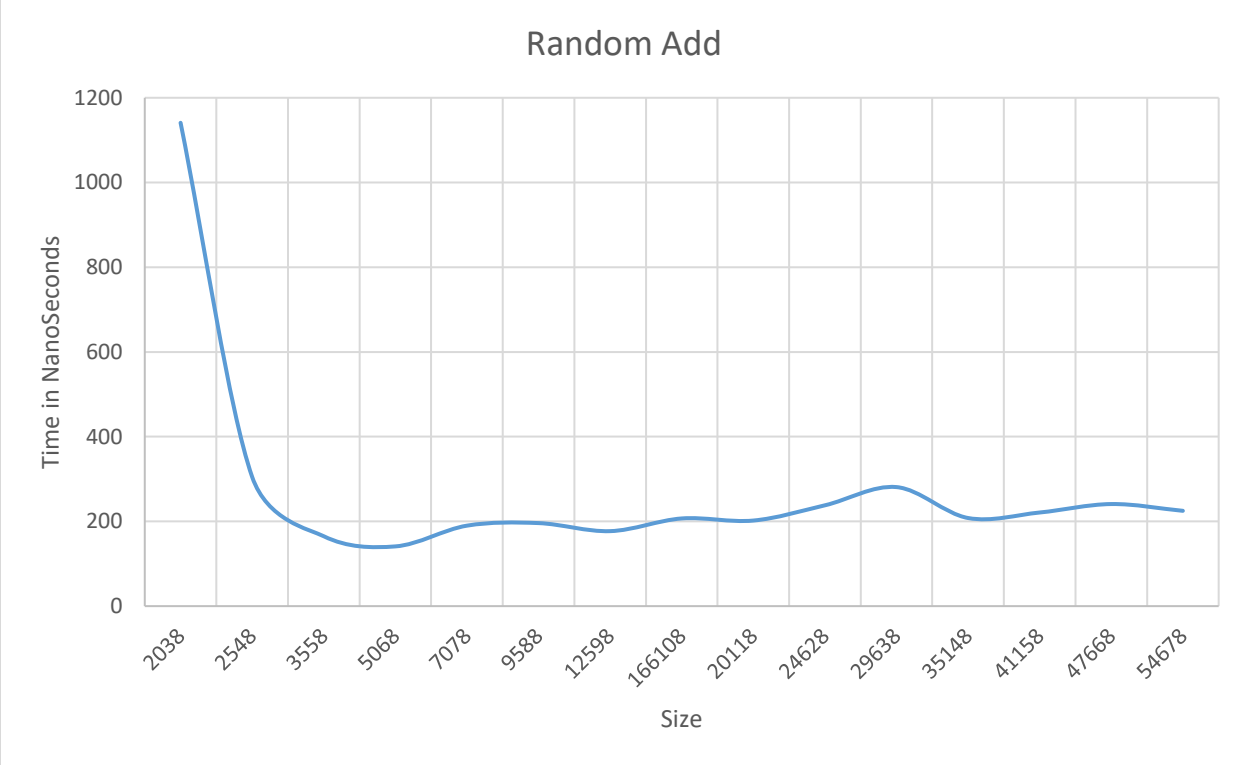
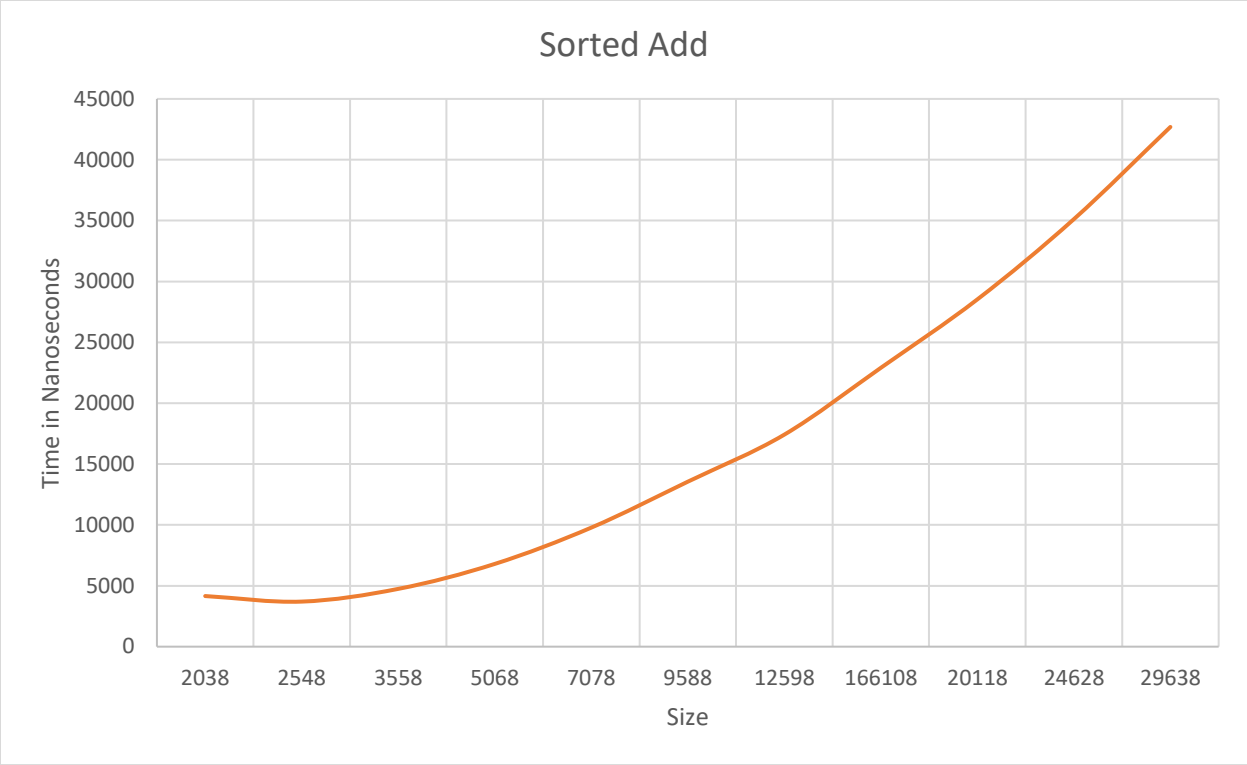
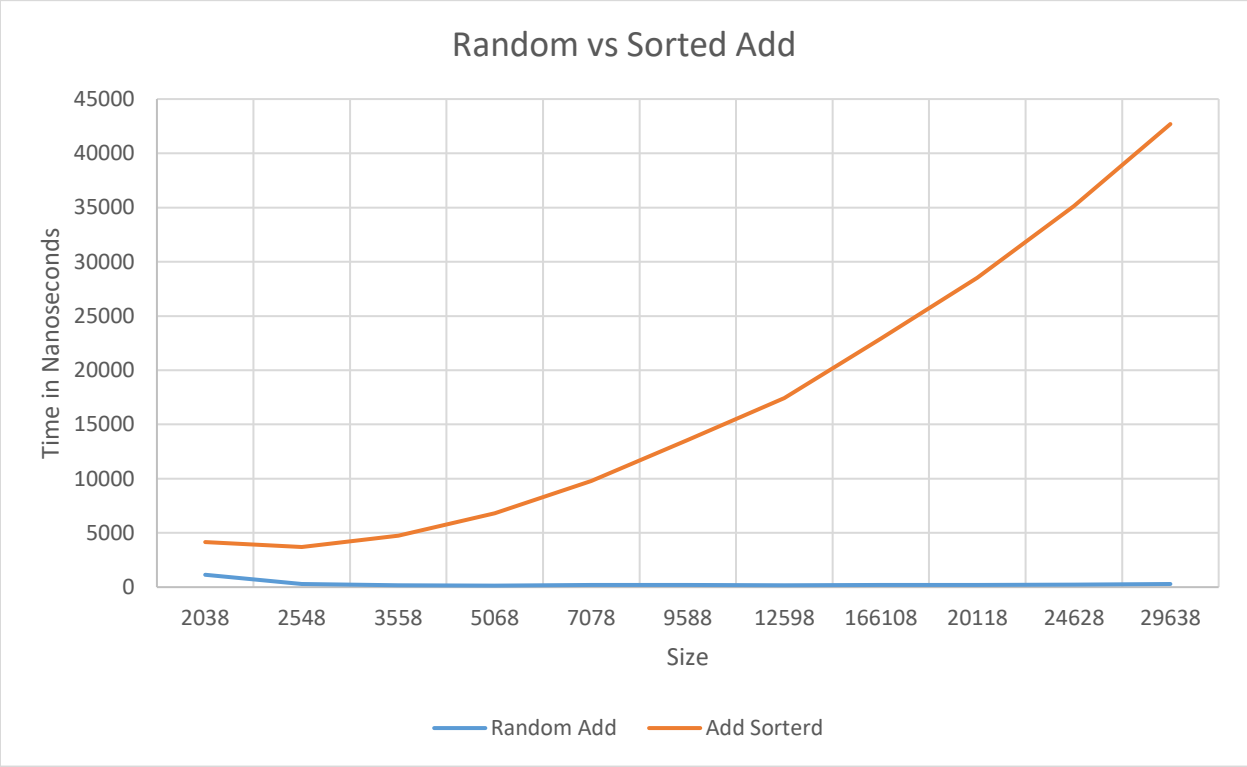
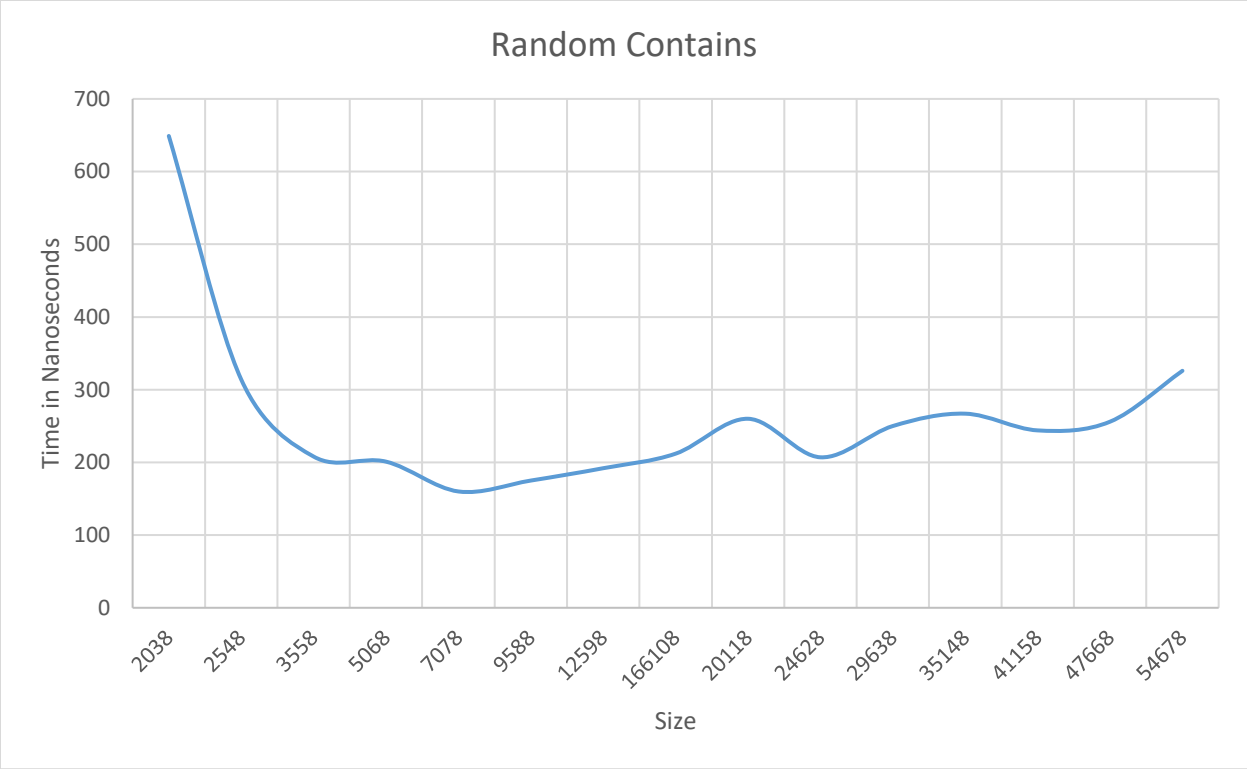
1. My partner is Brayden wright (u0895942), he will be submitting the assignment.
2. My partner was great and very helpful, he did his job and helped explain the parts that I did not understand. I will work with him again on the next assignment.
3. The experiment for adding and finding items random and sorted ordered items in to the tree was done, by:
    a. Creating an array list that contains numbers from 1 to the desired range.
    b. For the sorted add, all the items in the list are added to the tree in order by using the method addAll(), while timing each addAll(), then getting the average time by dividing it by the array size.
    c. For the random order, the array list is simply shuffled before step (b)the addAll() method.

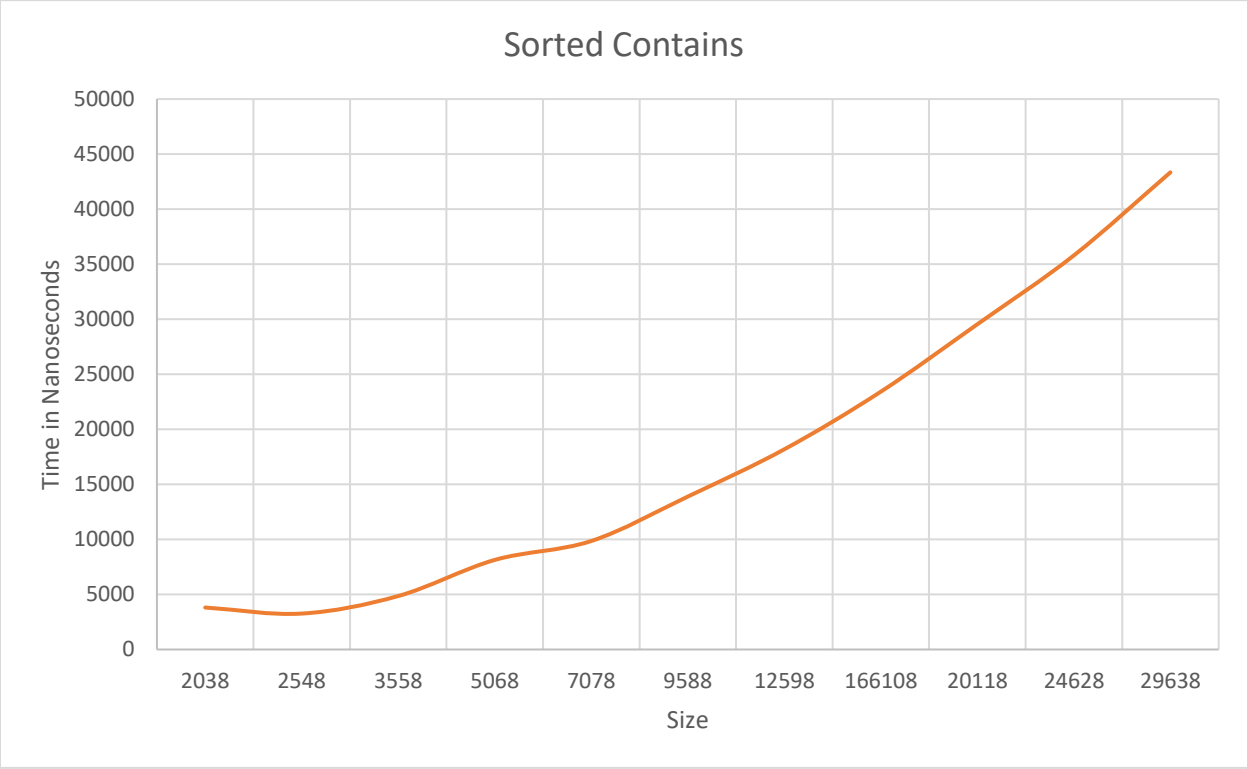
    The contains methods is done similarly:
    a. Shuffle the array list previously created
    d. Check use the containsAll() method to check if the tree contains all elements in the shuffled array list, then getting the average time by dividing it by the array size.
    e. For the random order, use the random array list created previously.
    b.

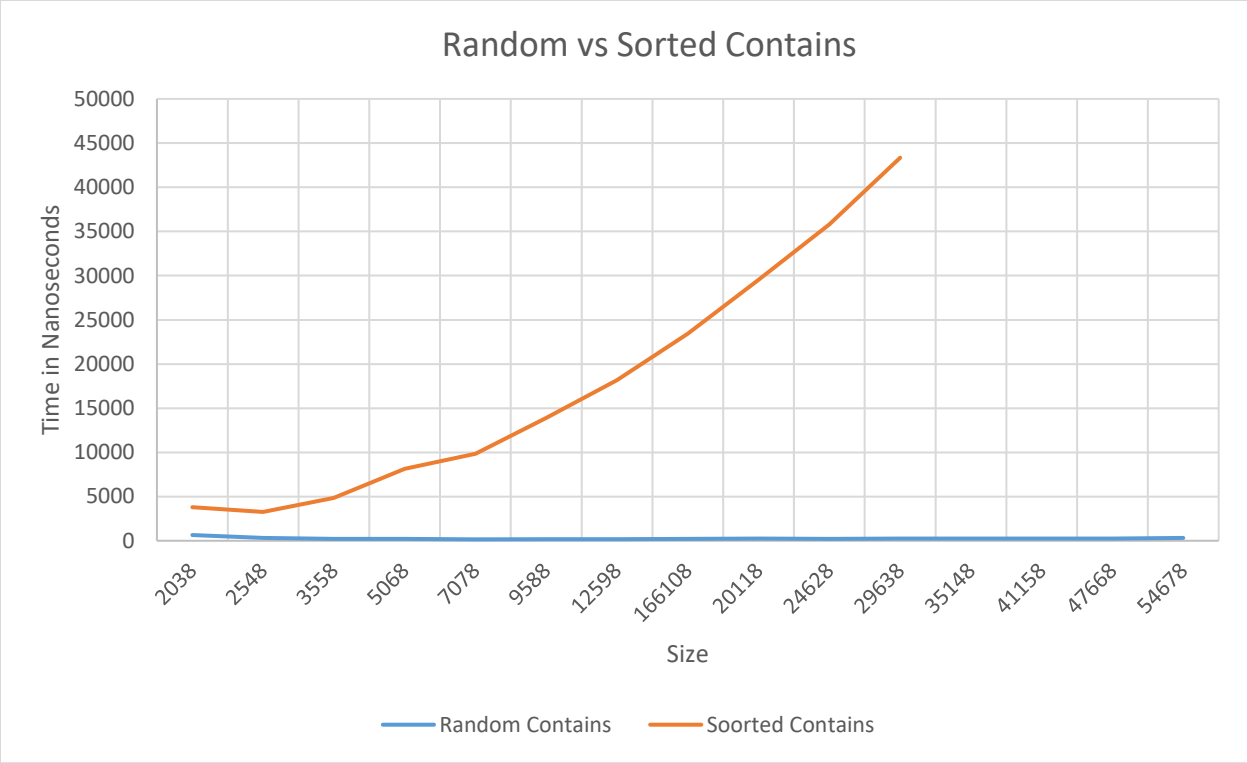



Sorted Add
45000
40000
35000
30000
25000
20000
15000
10000
5000
0
Time in Nanoseconds
2038
2548
3558
5068
7078
9588
12598
166108
20118
24628
29638
Size


Random vs Sorted Add
45000
40000
35000
30000
25000
20000
15000
10000
5000
0
Time in Nanoseconds
2038
2548
3558
5068
7078
9588
12598
166108
20118
24628
29638
Size
Random Add
Add Sorterd


Random Contains
Time in Nanoseconds
700
600
500
400
300
200
100
0
2038
2548
3558
5068
7078
9588
12598
166108
20118
24628
29638
35148
41158
47668
54678
Size


Sorted Contains
Time in Nanoseconds
50000
45000
40000
35000
30000
25000
20000
15000
10000
5000
0
2038
2548
3558
5068
7078
9588
12598
166108
20118
24628
29638
Size

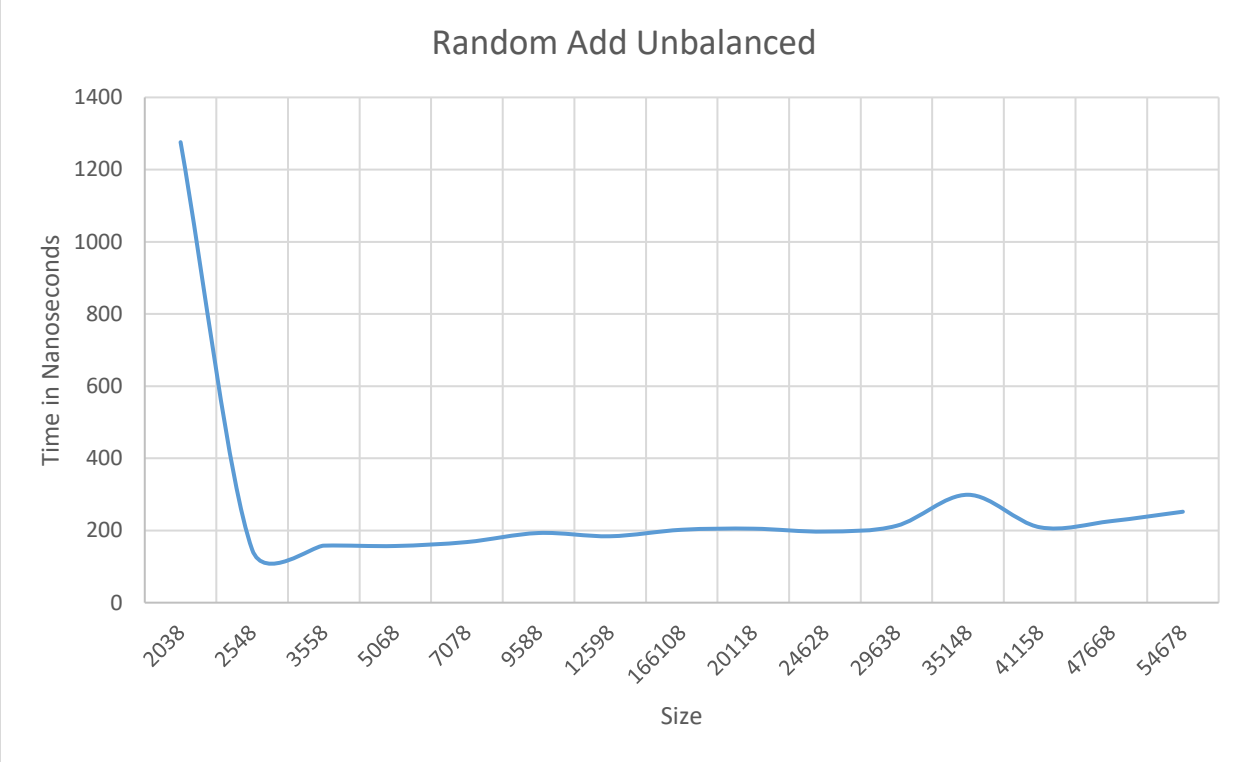


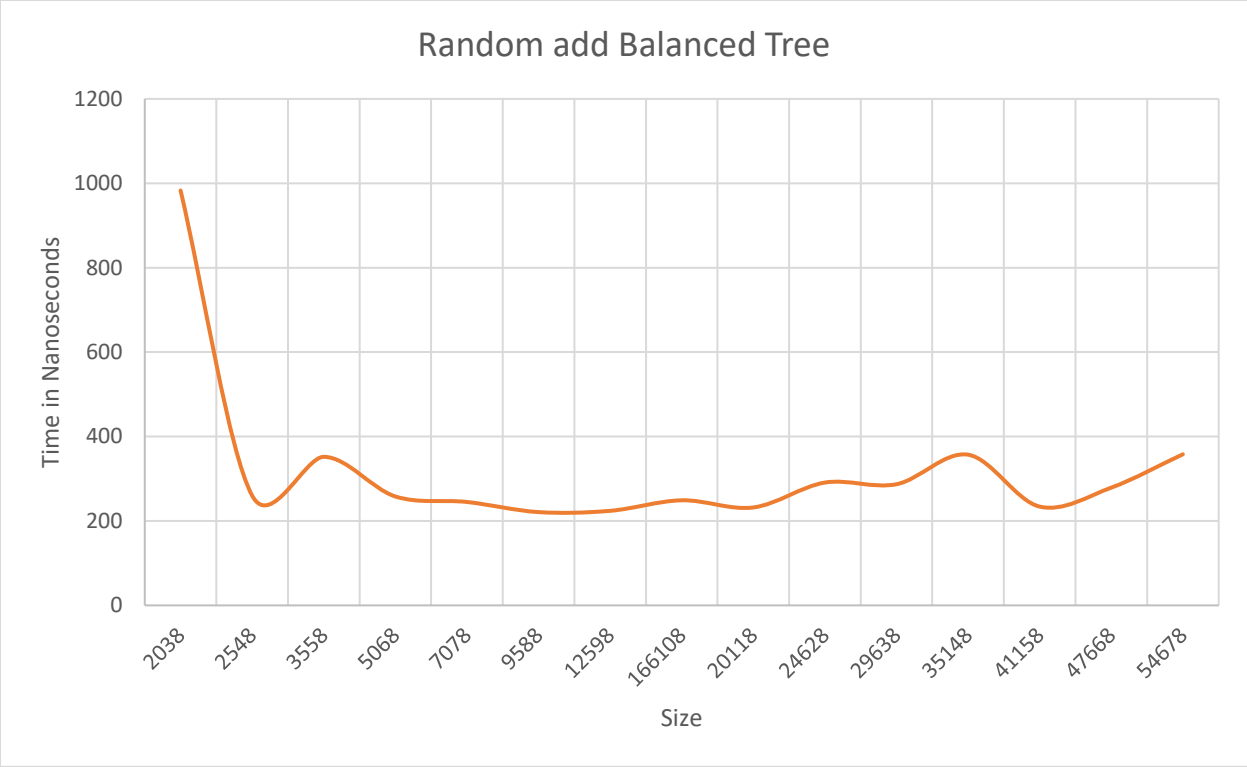
The graphs behaviors' is as expected. The randomly inserted array is O(log n). The sorted tree operations are the worst case, and they are supposed to have O(n), however, the graph looks closer to a O(n log n). Still the random insertion is much faster and much more efficient than the sorted set. Since it utilizes the insertion, and search of the binary tree.

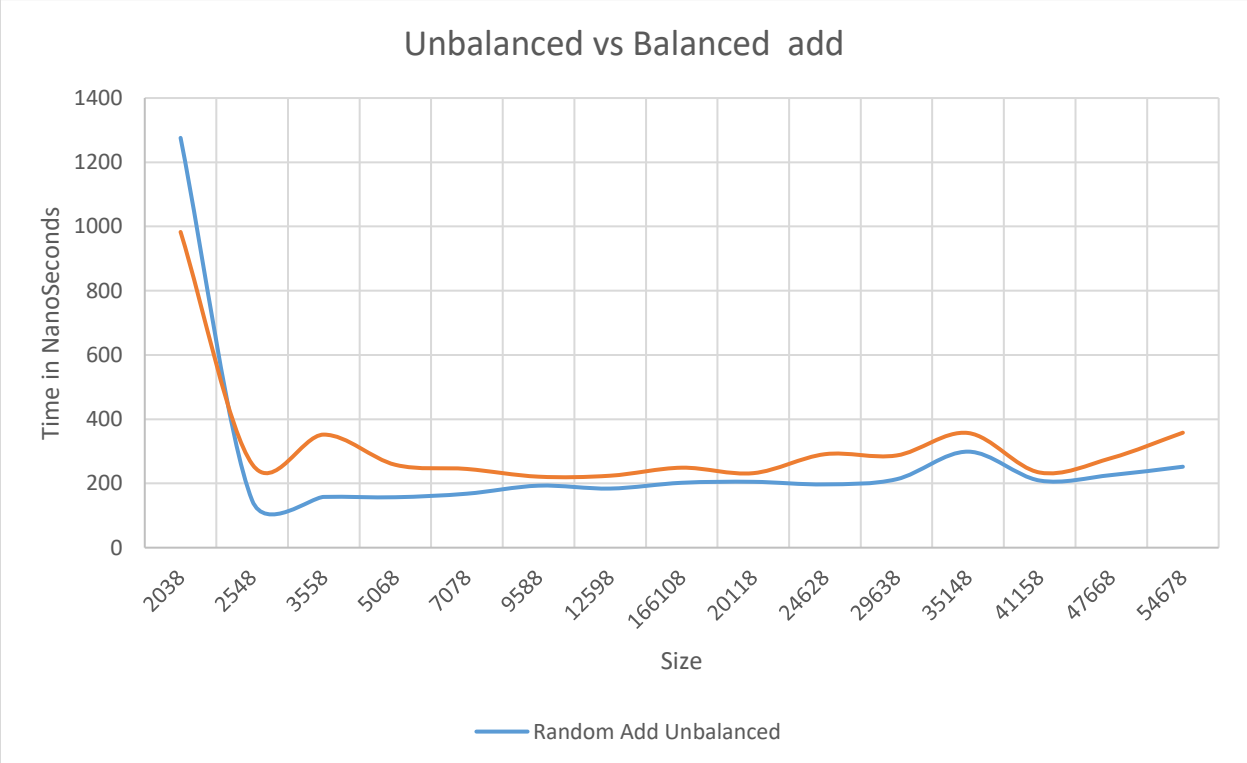
Something worth mentioning, is that while timing the tree I found that as soon as the sorted array reaches over 30,000 elements the stack over flows. However, if the elements are shuffled before being added to the tree, there is no problem. So, I had to adjust the range accordingly.

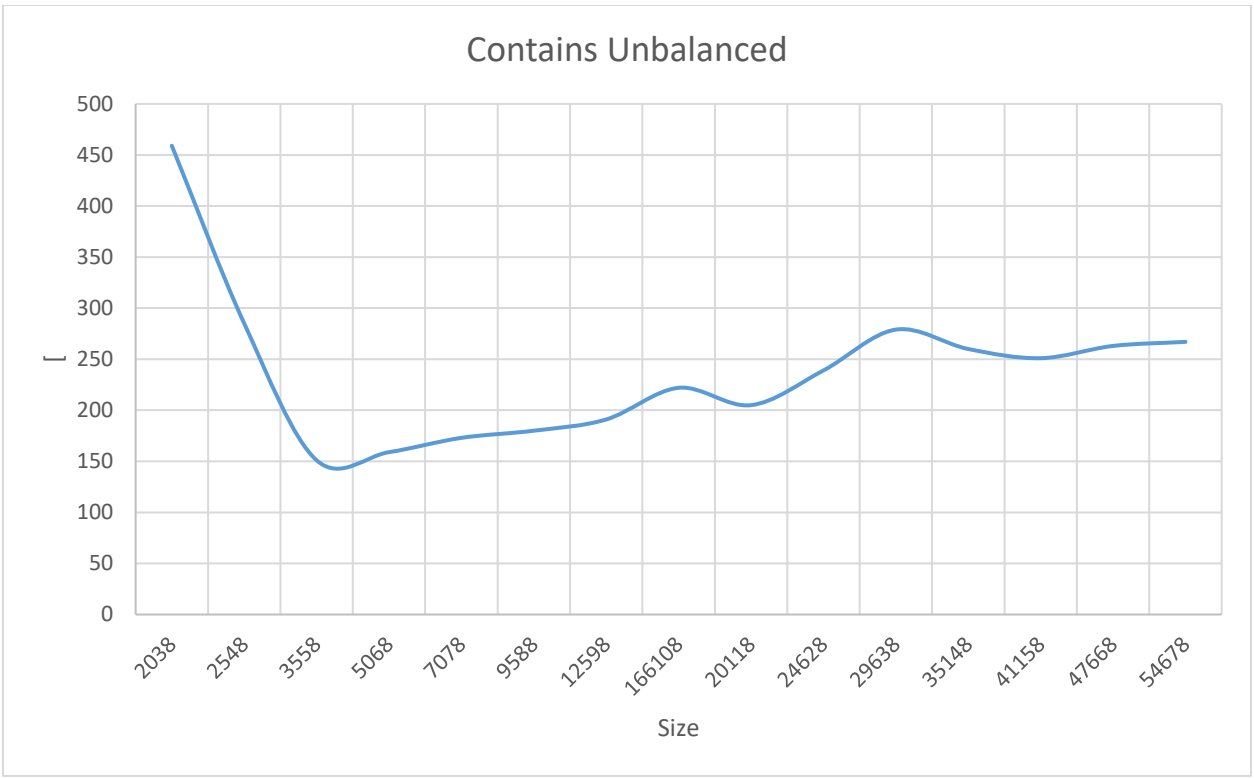
4. The experamint for balanced and unbalanced tree, was done very similar to the first expirament.

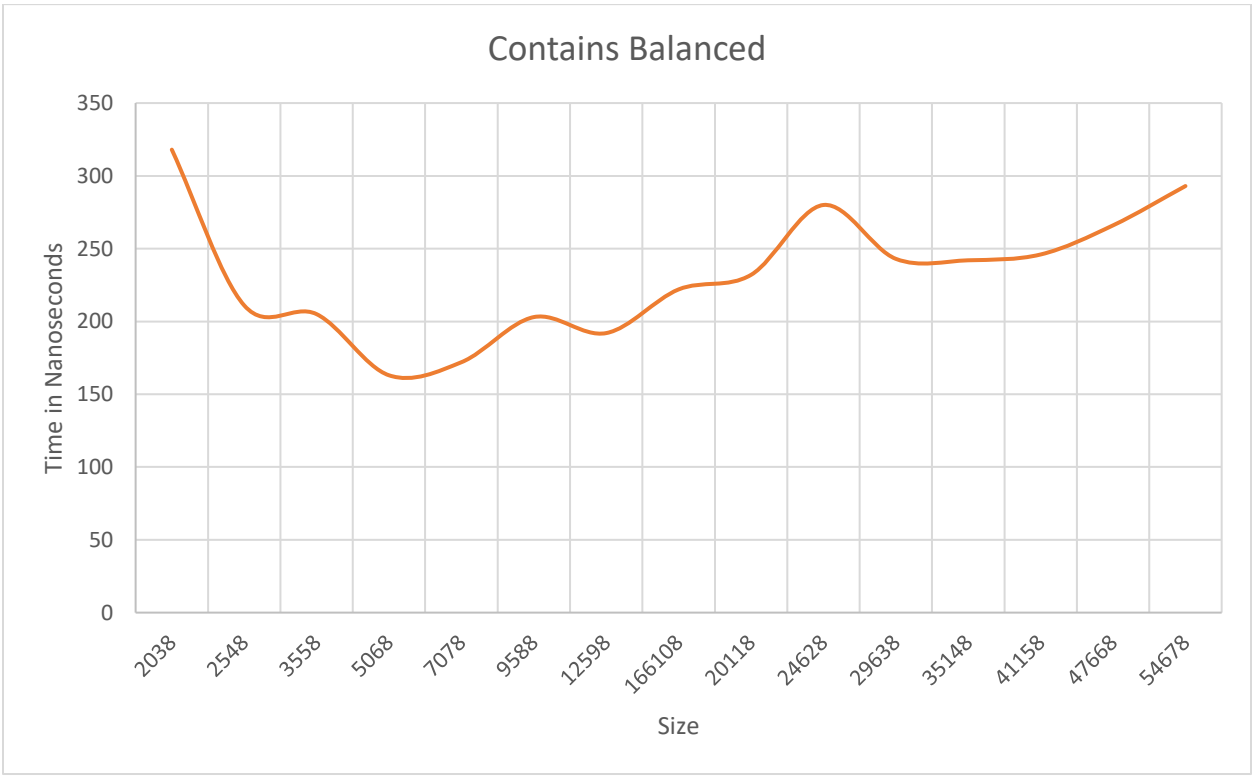
Create an array list with specified range, and createing a binary search tree as the unblalncing tree, and a TreeSet for the balanced tree. Shuffle the array list and addAll() to the tree set, in this expirament there were two adds, one for the balanced TreeSet, and one fore the BinarySearchTree. Each add was timed sepereatley. The contains is exactly the same, but the containsAll() method was timed for both trees.

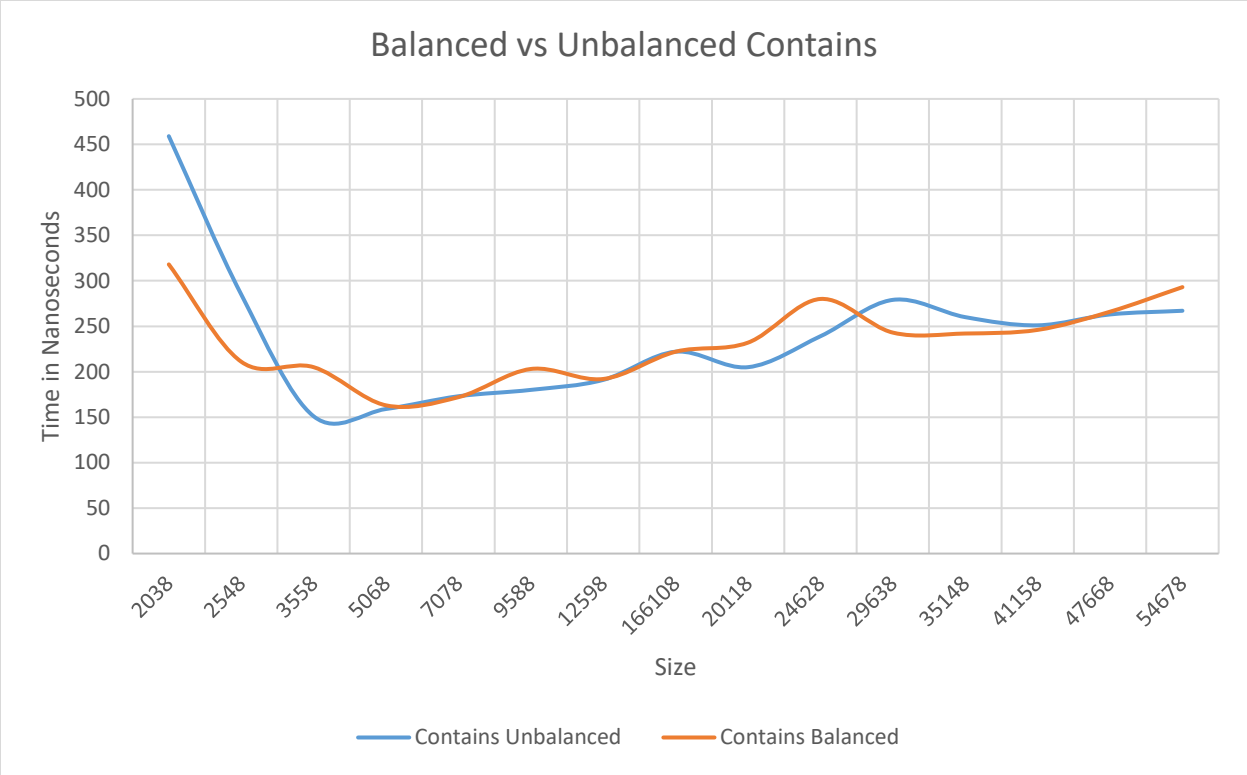



Random add Balanced Tree
Time in Nanoseconds
1200
1000
800
600
400
200
0
2038
2548
3558
5068
7078
9588
12598
166108
20118
24628
29638
35148
41158
47668
54678
Size


Unbalanced vs Balanced add
Time in NanoSeconds
1400
1200
1000
800
600
400
200
0
2038
2548
3558
5068
7078
9588
12598
166108
20118
24628
29638
35148
41158
47668
54678
Size
Random Add Unbalanced


Contains Unbalanced
500
450
400
350
300
250
200
150
100
50
0
[
2038
2548
3558
5068
7078
9588
12598
166108
20118
24628
29638
35148
41158
47668
54678
Size


Contains Balanced
350
300
250
200
150
100
50
0
Time in Nanoseconds
2038
2548
3558
5068
7078
9588
12598
166108
20118
24628
29638
35148
41158
47668
54678
Size



The graphs of both the balanced are almost identical. I assumed they would be different since the balanced tree must balance itself every occasionally. However, it seems like they are both identical.

Another point that is worth mentioning, is that the first few points in both experiments, take much longer time, for some reason.

5. To fix the dictionary problem, the words must be shuffled before insertion. Since they will be entered at a random order, the binary tree will start performing at O(log n). If they are entered alphabetically, this will create a worst case for a binary search tree, which performs at O(n).

6. I've worked on this assignment for about 10 hours.