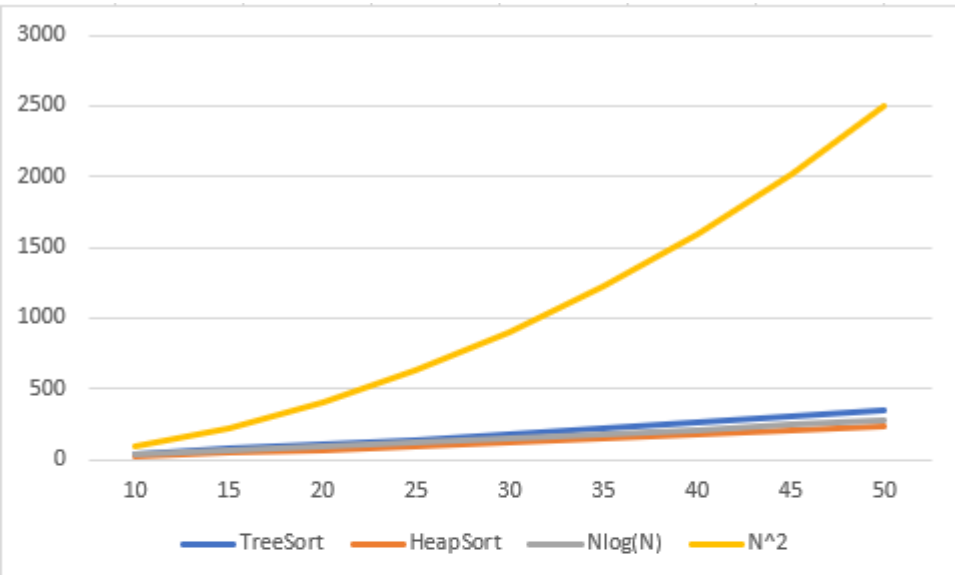
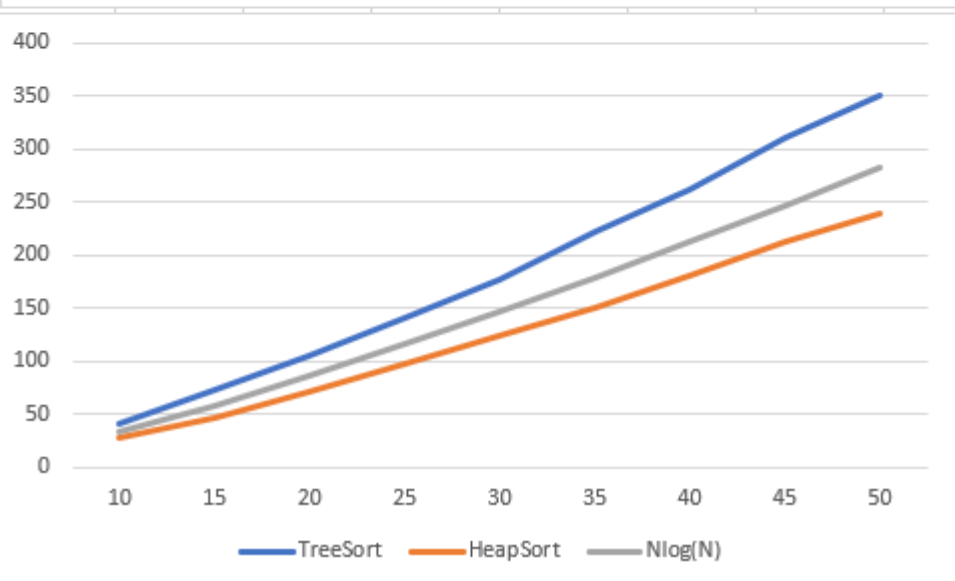
```
// Taylor Pedretti - 005488635 – Homework 4
// TreeSortHeapSort.h
#ifndef TSORTHSORT_H_
#define TSORTHSORT_H_

#include <vector>
#include "BST_HW4.h"
#include "BinaryHeap_HW4.h"

using namespace std;

extern int CLUMSY_COUNT;

template <typename C>
void TreeSort(vector<C>& data, int& comps)
{
	CLUMSY_COUNT = 0;
	BinarySearchTree<C> bst;

	for (int i = 0; i < data.size(); i++)
	{
		bst.insert(data[i]);
	}

	int i = 0;
	typename BinarySearchTree<C>::iterator itr = bst.begin();

	for (; itr != bst.end(); ++itr)
	{
		data[i] = *itr;
		i++;
	}
	comps = CLUMSY_COUNT;
}

template <typename C>
void HeapSort(vector<C>& data, int& comps)
{
	CLUMSY_COUNT = 0;
	// HW4: fill in to implement HeapSort;

	for (int i = data.size() / 2 - 1; i >= 0; --i) /* buildHeap */
		percDown(data, i, data.size());
	for (int j = data.size() - 1; j > 0; --j)
	{
		std::swap(data[0], data[j]); /* deleteMax */
		percDown(data, 0, j);
		CLUMSY_COUNT++;
	}

	comps = CLUMSY_COUNT;
}

inline int leftChild(int i)
{
	return 2 * i + 1;
}

template <typename C>
void percDown(vector<C> & a, int i, int n)
{
	int child;
	C tmp;
```

```
        for (tmp = std::move(a[i]); leftChild(i) < n; i = child)
        {
                child = leftChild(i);
                if (child != n - 1 && a[child] < a[child + 1])
                        ++child;
                if (tmp < a[child])
                        a[i] = std::move(a[child]);
                else
                        break;

                CLUMSY_COUNT++;
        }
        a[i] = std::move(tmp);
}

#endif
```

----Data Plots





------Output

```
, " << (int)avg1 << ", " << (int)avg2 << ", " << (int)nlogn <
```

C:\Users\Nex\source\repos\CSE-330 - HW4\Debug\CSE-

```
10, 39, 27, 33, 100

15, 68, 46, 58, 225

20, 105, 71, 86, 400

25, 143, 97, 116, 625

30, 191, 122, 147, 900

35, 220, 149, 179, 1225

40, 263, 180, 212, 1600

45, 308, 213, 247, 2025

50, 348, 241, 282, 2500
```