// 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++;

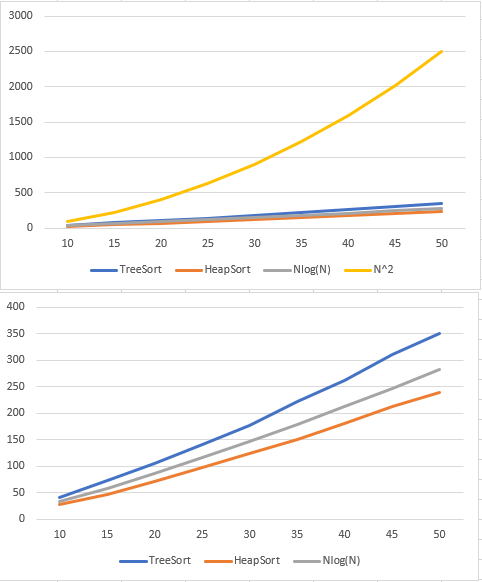
}

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

}

#endif

----Data Plots



------Output

