Raul Cervantes

RVCERVAN

Start of Report

hermod.ics.uci.edu - PuTTY

```
using namespace std;
#include <string>
#include <iostream>
#include <fstream>
#include "Timer.h"
#include "Sorter.h"
#define NWords 45293
        Sorter::Sorter(int max_len)
                :buf(new string[max_len]), capacity(max_len)
        {
        }

        void Sorter::insertAllFromFile(int partition, string fileName, string Sorter)
        //Big-O = O(N)
        {
                ifstream in(fileName);
                Timer t1;
                double t1Time;

                string word;
                for(int i = 0; i < capacity; ++ i)
                {
                        in >> word;
                        buf[i] = word;
                }

                t1.start();
                sort();
                t1.elapsedUserTime(t1Time);
                in.close();

                cout << "File: "<< fileName<< ". Partition: "<<
                partition<< "/10. Sorter: "<< Sorter << "  Time: "<< t1Time<< endl;
        }

        void Sorter::print(ostream& out)
        {
                out << buf[0] << endl;
        }
```

hermod.ics.uci.edu - PuTTY

```
using namespace std;
#include <string>
#include <iostream>
#include "QuickSorter.h"
#include "Sorter.h"
#define K 16
        QuickSort::QuickSort(int max_len)
                :Sorter(max_len){}

        void QuickSort::printAll()
        //Big-O = O(N)
        {
                for(int i = 0; i < capacity; ++i)
                {
                        cout << buf[i] << endl;
                }
        }

        void QuickSort::sort()
        //Big-O = O(N)
        {
                quick_sort(buf, 0, capacity-1);
                //printAll();
        }

        void QuickSort::quick_sort(string A[], int low, int high)
        //Big-O = O(N)
        {
                int slice = high - low;
                if(slice < K)
                {
                        QuickSort::InsertionSort(A, slice);//O(N)
                }

                else
                {
                        int i = partition(A, low, high);
                        quick_sort(A, low, i-1);
                        quick_sort(A, i+1, high);
                }
        }

        int QuickSort::partition(string A[], int low, int high)
        //Big-O = O(N^2)
        {
                string pivot = median_of_three(A, low, high);
                int below = low, above = high;
                for(;;)
                {
                        while(A[below] < pivot) { ++below;}
                        while(pivot < A[above]) { --above;}
                        if(below < above)
                        {
                                swap(A[below++], A[above--]);
                        }
                        else
                        {
                                break;
                        }
                }
                swap(A[below], A[high]);
                return below;
        }
                                                                    18,2-9        Top
```

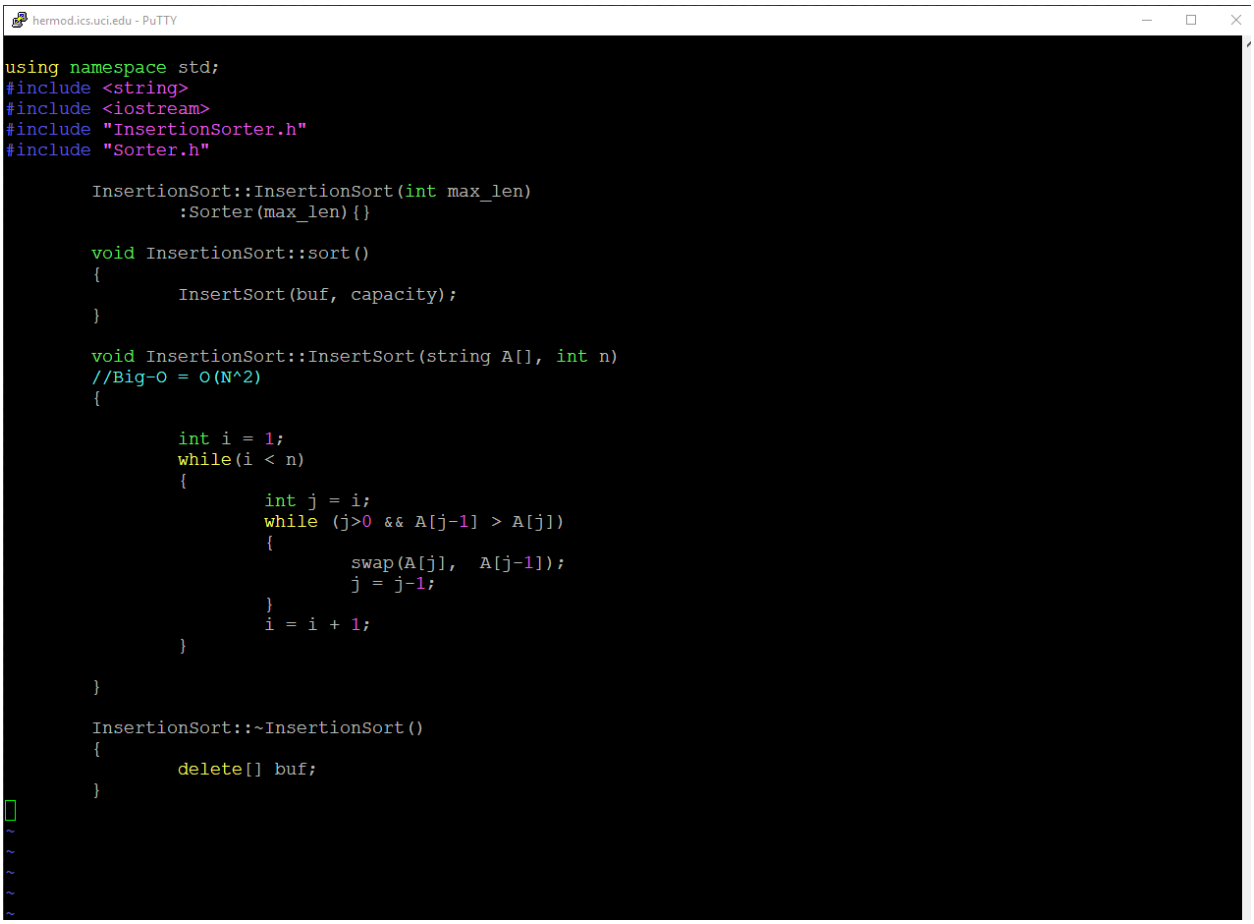
```
string QuickSort::median_of_three(string A[], int low, int high)
//Big-O = O(1)
{
        int mid = low + (high - low) / 2;
        if(A[mid] < A[low]) swap(A[low], A[mid]);
        if(A[high] < A[low]) swap(A[low], A[high]);
        if(A[mid] < A[high]) swap(A[mid], A[high]);
        return A[high];
}

QuickSort::~QuickSort()
{
        delete[] buf;
}
                                                                    64,2-9        Bot
```

hermod.ics.uci.edu - PuTTY

```
using namespace std;
#include <string>
#include <math.h>
#include <iostream>
#include "HeapSorter.h"
#include "Sorter.h"

        HeapSort::HeapSort(int max_len)
                :Sorter(max_len){}


        void HeapSort::printAll()
        //Big-Oh = O(N)
        {
                for(int i = 0; i < capacity; ++i)
                {
                        cout << buf[i] << endl;
                }
        }

        void HeapSort::sort()
        //Big-Oh = O(N)
        {
                heapSort(buf, capacity);
        //      printAll();
        }

        int HeapSort::leftChild(int i)
        //Big-Oh = O(1)
        {
                return 2*i + 1;
        }

        int HeapSort::rightChild(int i)
        //Big-Oh = O(1)
        {
                return 2*i + 2;
        }

        int HeapSort::Parent(int i)
        //Big-Oh = O(1)
        {
                return floor((i-1) / 2);
        }

        void HeapSort::heapSort(string A[], int N)
        //Big-Oh = O(N^2)
        {
                heapify(A, N);

                int end = N-1;
                while(end > 0)
                {
                        swap(A[end], A[0]);
                        end = end - 1;
                        siftSmallestDown(A, 0, end);
                }
        }
```

```
void HeapSort::heapify(string A[], int count)
//Big-Oh = O(N^2)
{
        int start = Parent(count-1);

        while(start >= 0)
        {
                siftSmallestDown(A, start, count - 1);
                start = start - 1;
        }
}


void HeapSort::siftSmallestDown(string A[], int start, int end)
//Big-Oh = O(N)
{
        int root = start;
        while(leftChild(root) <= end)
        {
                int child = leftChild(root);
                int to_swap = root;
                if(A[to_swap] < A[child])
                {
                        to_swap = child;
                }
                if(child + 1 <= end && A[to_swap] < A[child + 1])
                {
                        to_swap = child + 1;
                }
                if(to_swap == root)
                {
                        return;
                }
                else
                {
                        swap(A[root], A[to_swap]);
                        root = to_swap;
                }
        }
}

HeapSort::~HeapSort()
{
        delete[] buf;
}
```

104,2-9 Bot



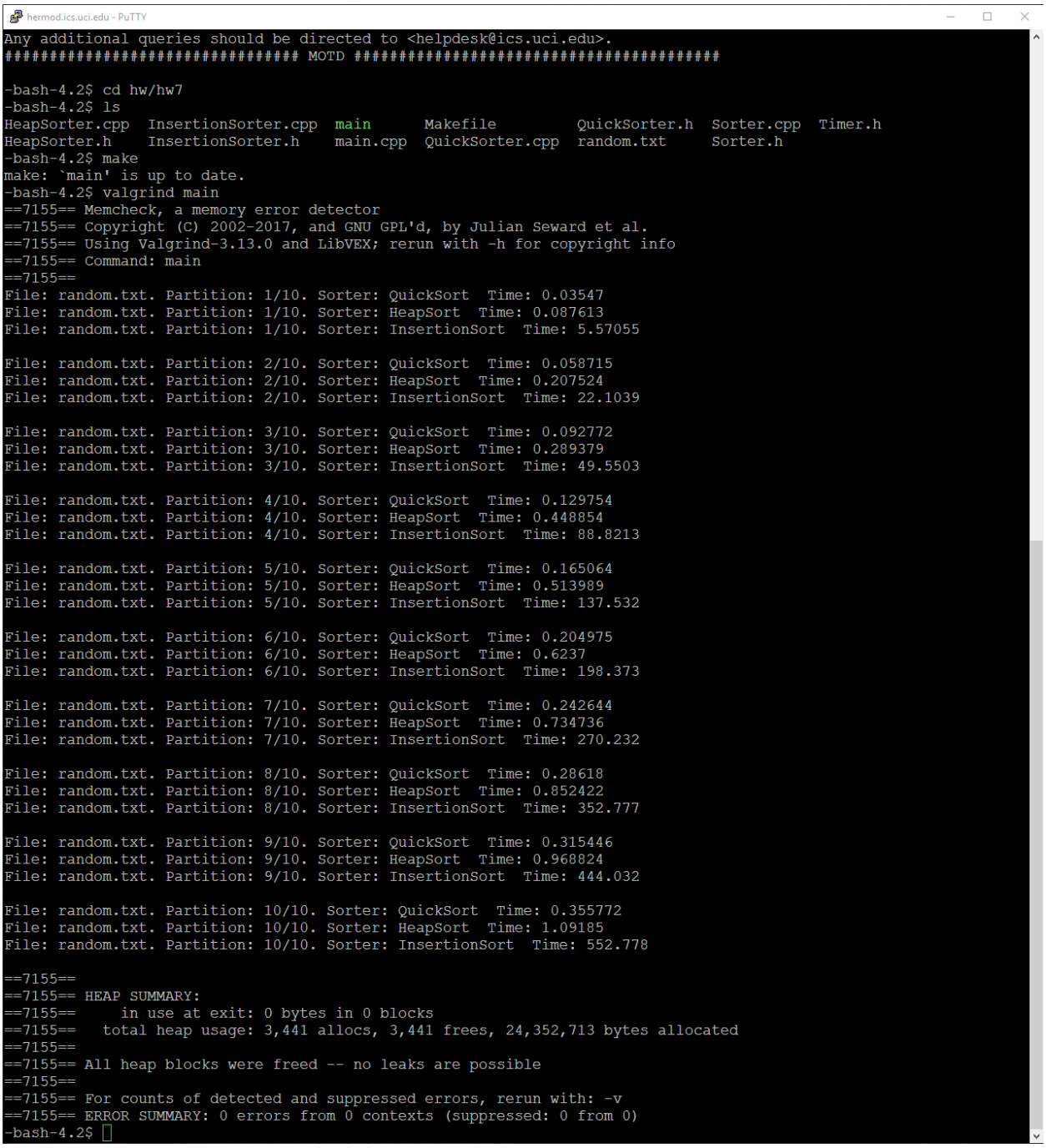
Above is all of my code relating to sort, which is the QuickSort, HeapSort, InsertionSort, and insertAllFromFile() with their Big-Oh notation.

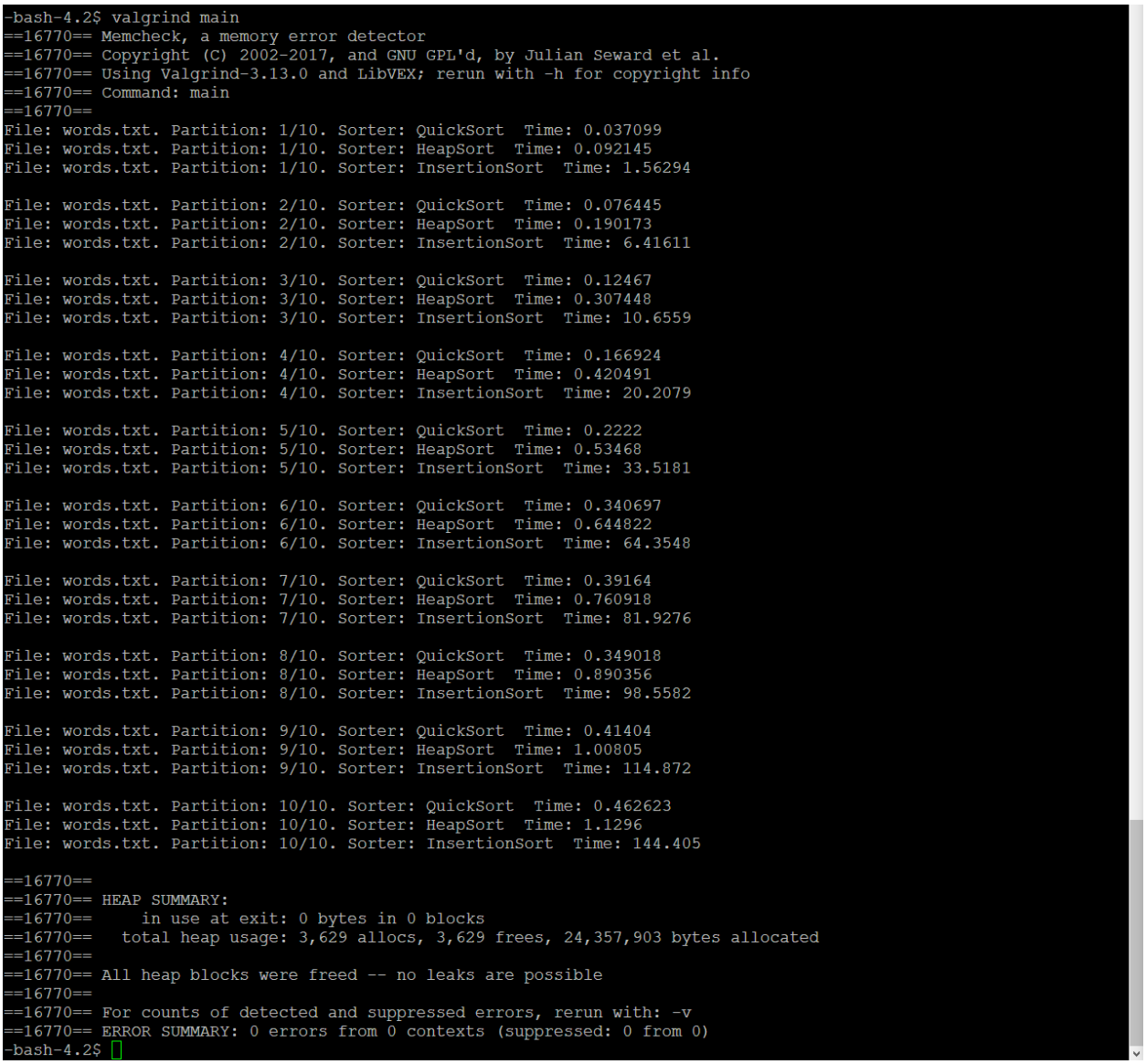
hermod.ics.uci.edu - PuTTY

```
using namespace std;
#include <string>
#include "QuickSorter.h"
#include "HeapSorter.h"
#include "InsertionSorter.h"
#include "Sorter.h"
#include <iostream>
#define NWords 45293
void measureAll(string fileName)
//Big-O = O(N^2)
{

        for(int i = 1; i<=10; ++i)
        {
                int word_count = i*NWords/10;
                QuickSort T1(word_count);
                HeapSort T2(word_count);
                InsertionSort T3(word_count);

                T1.insertAllFromFile(i,fileName,"QuickSort");
                T2.insertAllFromFile(i,fileName,"HeapSort");
                T3.insertAllFromFile(i,fileName,"InsertionSort");
                cout << endl;

        }
}

int main()
{
        string file = "random.txt";
        measureAll(file);
        return 0;
}
~
```

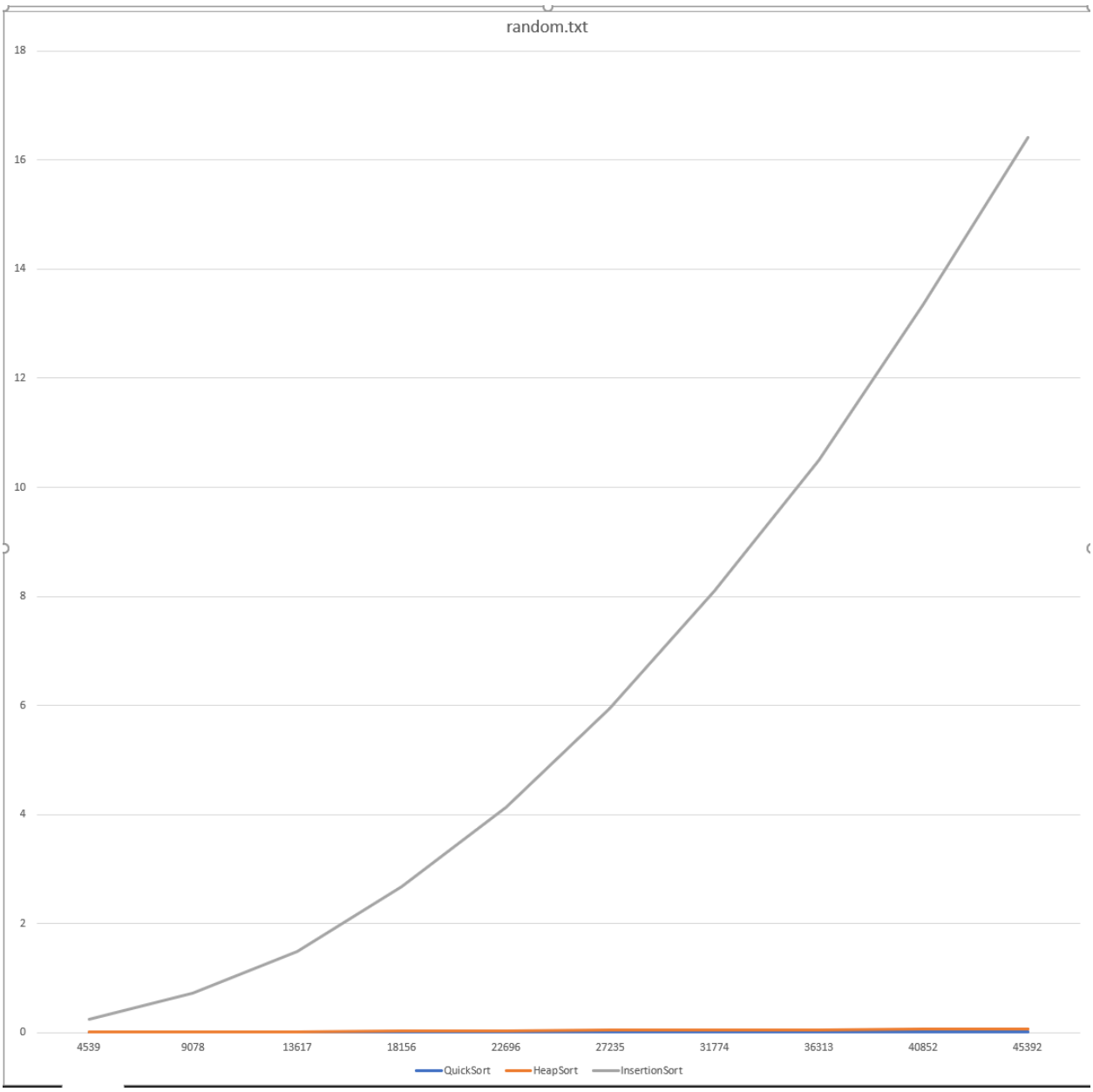
```
-bash-4.2$ main
File: random.txt. Partition: 1/10. Sorter: QuickSort  Time: 0.002093
File: random.txt. Partition: 1/10. Sorter: HeapSort  Time: 0.005197
File: random.txt. Partition: 1/10. Sorter: InsertionSort  Time: 0.229842

File: random.txt. Partition: 2/10. Sorter: QuickSort  Time: 0.002986
File: random.txt. Partition: 2/10. Sorter: HeapSort  Time: 0.007622
File: random.txt. Partition: 2/10. Sorter: InsertionSort  Time: 0.701853

File: random.txt. Partition: 3/10. Sorter: QuickSort  Time: 0.00471
File: random.txt. Partition: 3/10. Sorter: HeapSort  Time: 0.012068
File: random.txt. Partition: 3/10. Sorter: InsertionSort  Time: 1.47436

File: random.txt. Partition: 4/10. Sorter: QuickSort  Time: 0.006656
File: random.txt. Partition: 4/10. Sorter: HeapSort  Time: 0.016658
File: random.txt. Partition: 4/10. Sorter: InsertionSort  Time: 2.64597

File: random.txt. Partition: 5/10. Sorter: QuickSort  Time: 0.008204
File: random.txt. Partition: 5/10. Sorter: HeapSort  Time: 0.021415
File: random.txt. Partition: 5/10. Sorter: InsertionSort  Time: 4.10909

File: random.txt. Partition: 6/10. Sorter: QuickSort  Time: 0.010451
File: random.txt. Partition: 6/10. Sorter: HeapSort  Time: 0.026254
File: random.txt. Partition: 6/10. Sorter: InsertionSort  Time: 5.90961

File: random.txt. Partition: 7/10. Sorter: QuickSort  Time: 0.012028
File: random.txt. Partition: 7/10. Sorter: HeapSort  Time: 0.031166
File: random.txt. Partition: 7/10. Sorter: InsertionSort  Time: 8.05737

File: random.txt. Partition: 8/10. Sorter: QuickSort  Time: 0.014143
File: random.txt. Partition: 8/10. Sorter: HeapSort  Time: 0.036154
File: random.txt. Partition: 8/10. Sorter: InsertionSort  Time: 10.4517

File: random.txt. Partition: 9/10. Sorter: QuickSort  Time: 0.016043
File: random.txt. Partition: 9/10. Sorter: HeapSort  Time: 0.041431
File: random.txt. Partition: 9/10. Sorter: InsertionSort  Time: 13.3044

File: random.txt. Partition: 10/10. Sorter: QuickSort  Time: 0.018176
File: random.txt. Partition: 10/10. Sorter: HeapSort  Time: 0.046868
File: random.txt. Partition: 10/10. Sorter: InsertionSort  Time: 16.3582
```

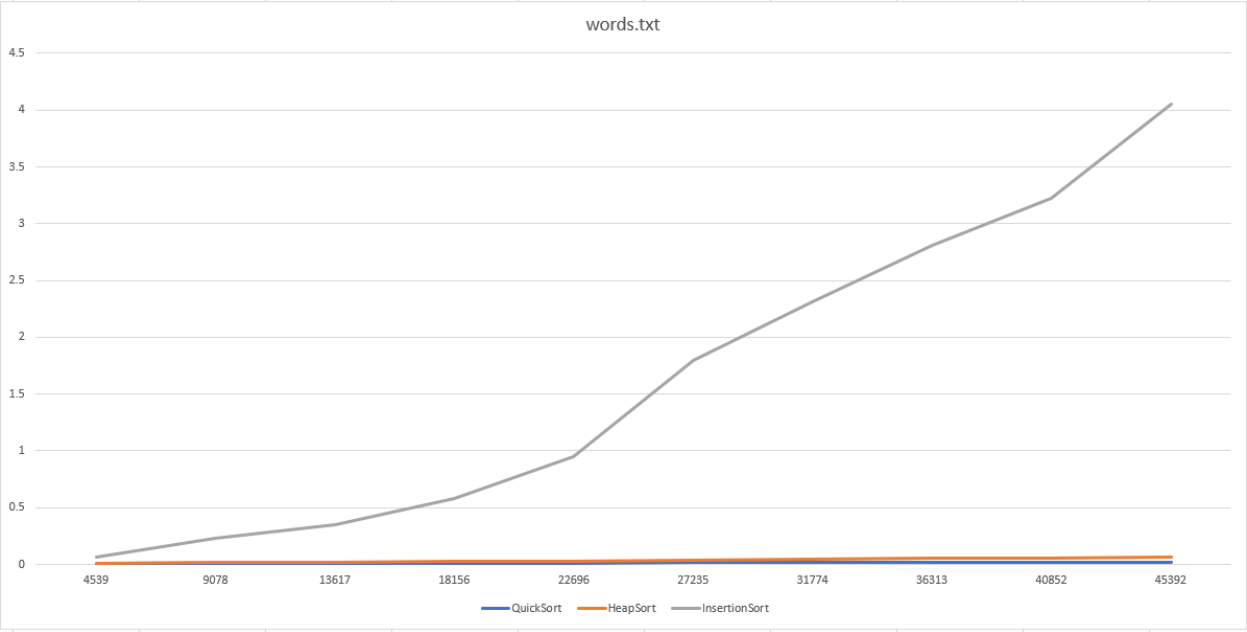
```
main
-bash-4.2$ main
File: words.txt. Partition: 1/10. Sorter: QuickSort  Time: 0.001942
File: words.txt. Partition: 1/10. Sorter: HeapSort  Time: 0.005015
File: words.txt. Partition: 1/10. Sorter: InsertionSort  Time: 0.061222

File: words.txt. Partition: 2/10. Sorter: QuickSort  Time: 0.006082
File: words.txt. Partition: 2/10. Sorter: HeapSort  Time: 0.011162
File: words.txt. Partition: 2/10. Sorter: InsertionSort  Time: 0.216006

File: words.txt. Partition: 3/10. Sorter: QuickSort  Time: 0.006019
File: words.txt. Partition: 3/10. Sorter: HeapSort  Time: 0.011651
File: words.txt. Partition: 3/10. Sorter: InsertionSort  Time: 0.336032

File: words.txt. Partition: 4/10. Sorter: QuickSort  Time: 0.008277
File: words.txt. Partition: 4/10. Sorter: HeapSort  Time: 0.015942
File: words.txt. Partition: 4/10. Sorter: InsertionSort  Time: 0.554542

File: words.txt. Partition: 5/10. Sorter: QuickSort  Time: 0.01124
File: words.txt. Partition: 5/10. Sorter: HeapSort  Time: 0.020765
File: words.txt. Partition: 5/10. Sorter: InsertionSort  Time: 0.920694

File: words.txt. Partition: 6/10. Sorter: QuickSort  Time: 0.017221
File: words.txt. Partition: 6/10. Sorter: HeapSort  Time: 0.0248
File: words.txt. Partition: 6/10. Sorter: InsertionSort  Time: 1.75737

File: words.txt. Partition: 7/10. Sorter: QuickSort  Time: 0.019437
File: words.txt. Partition: 7/10. Sorter: HeapSort  Time: 0.02926
File: words.txt. Partition: 7/10. Sorter: InsertionSort  Time: 2.26076

File: words.txt. Partition: 8/10. Sorter: QuickSort  Time: 0.017182
File: words.txt. Partition: 8/10. Sorter: HeapSort  Time: 0.034473
File: words.txt. Partition: 8/10. Sorter: InsertionSort  Time: 2.75414

File: words.txt. Partition: 9/10. Sorter: QuickSort  Time: 0.020475
File: words.txt. Partition: 9/10. Sorter: HeapSort  Time: 0.039082
File: words.txt. Partition: 9/10. Sorter: InsertionSort  Time: 3.1639

File: words.txt. Partition: 10/10. Sorter: QuickSort  Time: 0.022612
File: words.txt. Partition: 10/10. Sorter: HeapSort  Time: 0.043867
File: words.txt. Partition: 10/10. Sorter: InsertionSort  Time: 3.98229

-bash-4.2$
```


hermod.ics.uci.edu - PuTTY
Any additional queries should be directed to <helpdesk@ics.uci.edu>.
###################################### MOTD ##
-bash-4.2$ cd hw/hw7
-bash-4.2$ ls
HeapSorter.cpp InsertionSorter.cpp main   Makefile   QuickSorter.h Sorter.cpp Timer.h
HeapSorter.h   InsertionSorter.h   main.cpp QuickSorter.cpp random.txt   Sorter.h
-bash-4.2$ make
make: `main' is up to date.
-bash-4.2$ valgrind main
==7155== Memcheck, a memory error detector
==7155== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==7155== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==7155== Command: main
==7155==
File: random.txt. Partition: 1/10. Sorter: QuickSort Time: 0.03547
File: random.txt. Partition: 1/10. Sorter: HeapSort Time: 0.087613
File: random.txt. Partition: 1/10. Sorter: InsertionSort Time: 5.57055
File: random.txt. Partition: 2/10. Sorter: QuickSort Time: 0.058715
File: random.txt. Partition: 2/10. Sorter: HeapSort Time: 0.207524
File: random.txt. Partition: 2/10. Sorter: InsertionSort Time: 22.1039
File: random.txt. Partition: 3/10. Sorter: QuickSort Time: 0.092772
File: random.txt. Partition: 3/10. Sorter: HeapSort Time: 0.289379
File: random.txt. Partition: 3/10. Sorter: InsertionSort Time: 49.5503
File: random.txt. Partition: 4/10. Sorter: QuickSort Time: 0.129754
File: random.txt. Partition: 4/10. Sorter: HeapSort Time: 0.448854
File: random.txt. Partition: 4/10. Sorter: InsertionSort Time: 88.8213
File: random.txt. Partition: 5/10. Sorter: QuickSort Time: 0.165064
File: random.txt. Partition: 5/10. Sorter: HeapSort Time: 0.513989
File: random.txt. Partition: 5/10. Sorter: InsertionSort Time: 137.532
File: random.txt. Partition: 6/10. Sorter: QuickSort Time: 0.204975
File: random.txt. Partition: 6/10. Sorter: HeapSort Time: 0.6237
File: random.txt. Partition: 6/10. Sorter: InsertionSort Time: 198.373
File: random.txt. Partition: 7/10. Sorter: QuickSort Time: 0.242644
File: random.txt. Partition: 7/10. Sorter: HeapSort Time: 0.734736
File: random.txt. Partition: 7/10. Sorter: InsertionSort Time: 270.232
File: random.txt. Partition: 8/10. Sorter: QuickSort Time: 0.28618
File: random.txt. Partition: 8/10. Sorter: HeapSort Time: 0.852422
File: random.txt. Partition: 8/10. Sorter: InsertionSort Time: 352.777
File: random.txt. Partition: 9/10. Sorter: QuickSort Time: 0.315446
File: random.txt. Partition: 9/10. Sorter: HeapSort Time: 0.968824
File: random.txt. Partition: 9/10. Sorter: InsertionSort Time: 444.032
File: random.txt. Partition: 10/10. Sorter: QuickSort Time: 0.355772
File: random.txt. Partition: 10/10. Sorter: HeapSort Time: 1.09185
File: random.txt. Partition: 10/10. Sorter: InsertionSort Time: 552.778
==7155==
==7155== HEAP SUMMARY:
==7155==   in use at exit: 0 bytes in 0 blocks
==7155==   total heap usage: 3,441 allocs, 3,441 frees, 24,352,713 bytes allocated
==7155==
==7155== All heap blocks were freed -- no leaks are possible
==7155==
==7155== For counts of detected and suppressed errors, rerun with: -v
==7155== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
-bash-4.2$



Above is my valgrind run of both text files and regular run of both text files and also my test and measure function in my main.cpp file.


random.txt
18
16
14
12
10
8
6
4
2
0
4539
9078
13617
18156
22696
27235
31774
36313
40852
45392
QuickSort
HeapSort
InsertionSort



Above are my two graphs for words.txt and random.txt.

End of Report