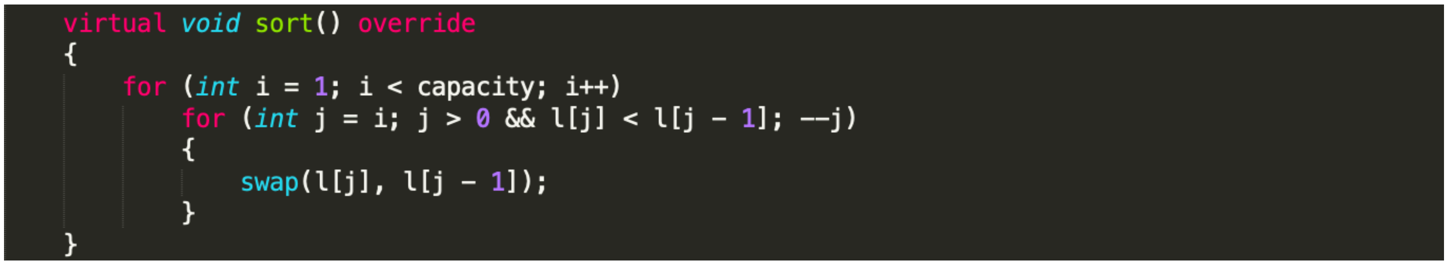
HOMEWORK 7

YANG TANG ID: 53979886

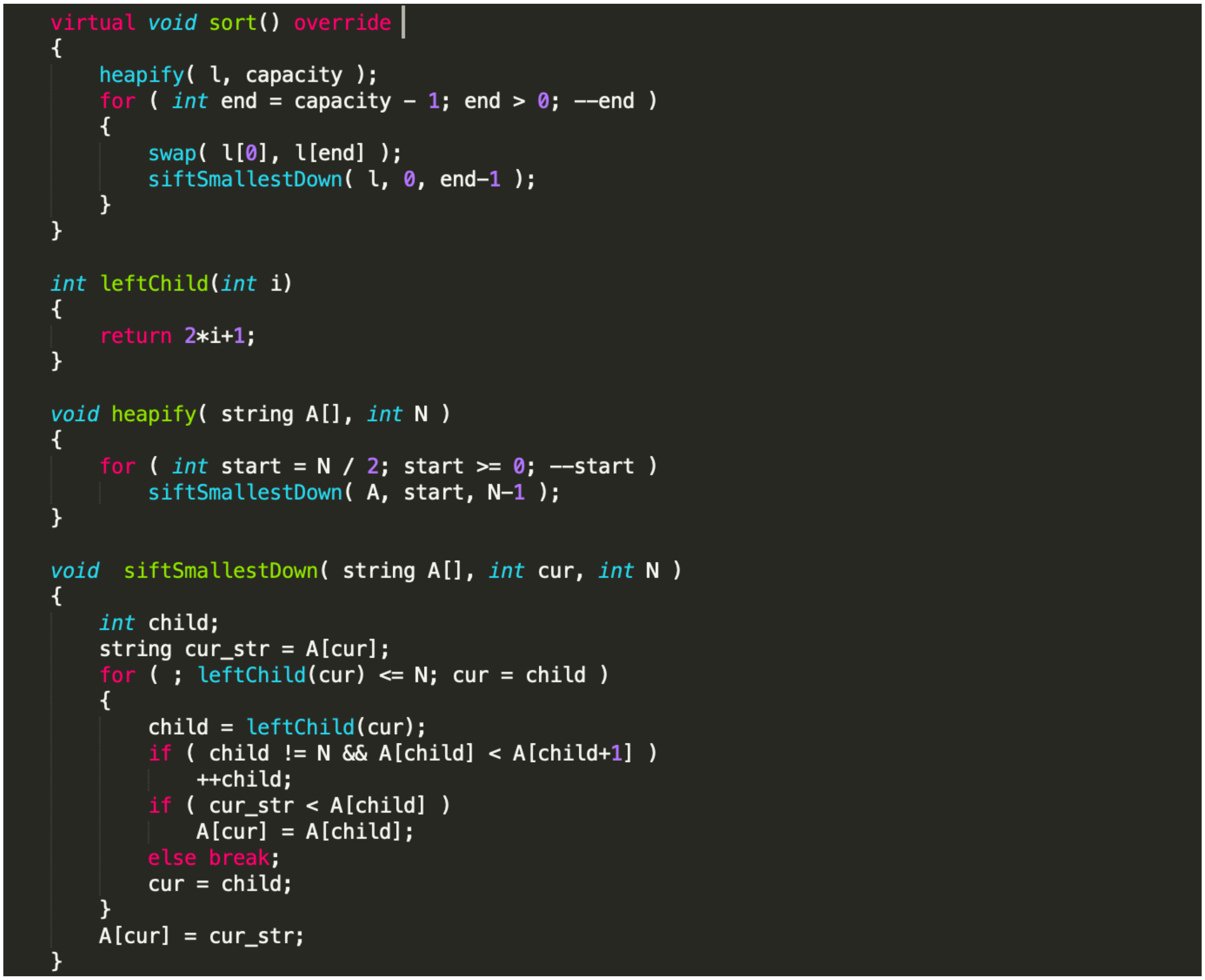
Part 1: Code Implementation & Big-O Time complexity Analysis

void InsertionSorter::sort() O(N^2)



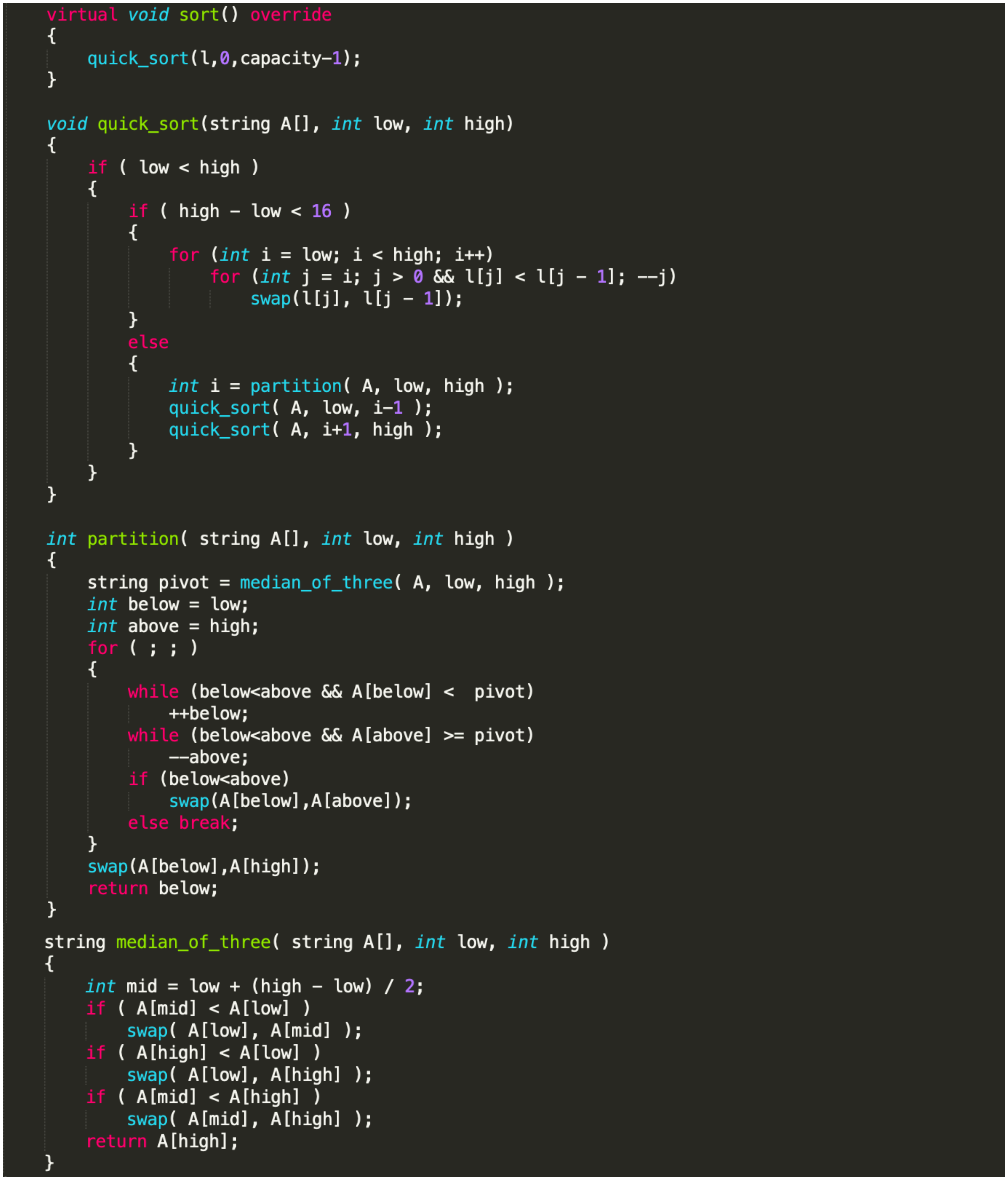
```
virtual void sort() override
{
    for (int i = 1; i < capacity; i++)
        for (int j = i; j > 0 && l[j] < l[j - 1]; --j)
        {
            swap(l[j], l[j - 1]);
        }
}
```

void HeapSorter::sort() O(NlogN)



```
virtual void sort() override
{
    heapify( l, capacity );
    for ( int end = capacity - 1; end > 0; --end )
    {
        swap( l[0], l[end] );
        siftSmallestDown( l, 0, end-1 );
    }
}

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

void heapify( string A[], int N )
{
    for ( int start = N / 2; start >= 0; --start )
        siftSmallestDown( A, start, N-1 );
}

void  siftSmallestDown( string A[], int cur, int N )
{
    int child;
    string cur_str = A[cur];
    for ( ; leftChild(cur) <= N; cur = child )
    {
        child = leftChild(cur);
        if ( child != N && A[child] < A[child+1] )
            ++child;
        if ( cur_str < A[child] )
            A[cur] = A[child];
        else break;
        cur = child;
    }
    A[cur] = cur_str;
}
```

void QuickSorter::sort()   average: O(NlogN)   worst: O(N^2)

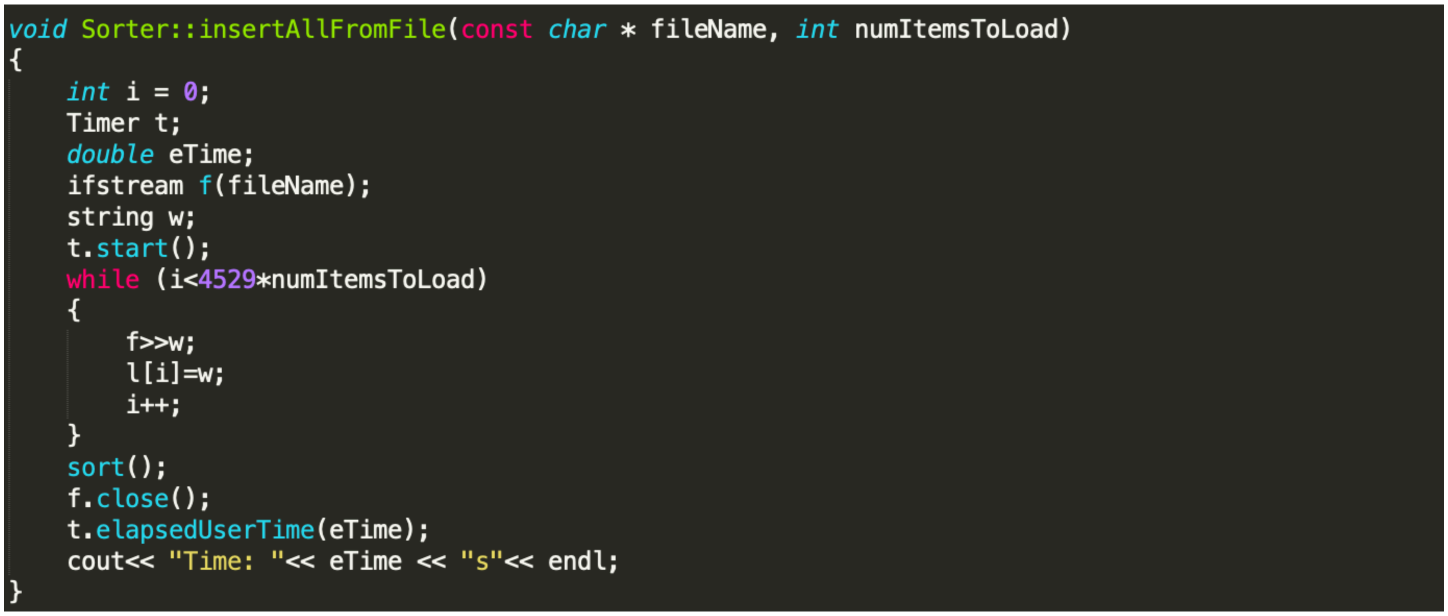


```
virtual void sort() override
{
    quick_sort(l,0,capacity-1);
}

void quick_sort(string A[], int low, int high)
{
    if ( low < high )
    {
        if ( high - low < 16 )
        {
            for (int i = low; i < high; i++)
                for (int j = i; j > 0 && l[j] < l[j - 1]; --j)
                    swap(l[j], l[j - 1]);
        }
        else
        {
            int i = partition( A, low, high );
            quick_sort( A, low, i-1 );
            quick_sort( A, i+1, high );
        }
    }
}

int partition( string A[], int low, int high )
{
    string pivot = median_of_three( A, low, high );
    int below = low;
    int above = high;
    for ( ; ; )
    {
        while (below<above && A[below] <  pivot)
            ++below;
        while (below<above && A[above] >= pivot)
            --above;
        if (below<above)
            swap(A[below],A[above]);
        else break;
    }
    swap(A[below],A[high]);
    return below;
}

string median_of_three( string A[], int low, int high )
{
    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];
}
```

Sorter::insertAllFromFile(int partition, char *fileName)

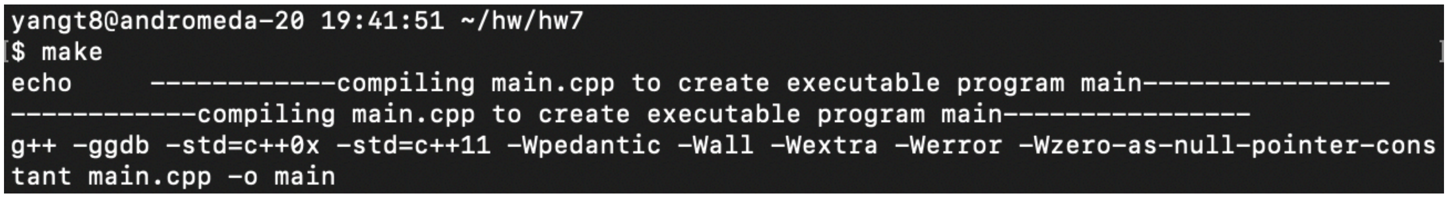
For heapsort & quicksort   O(NLogN)

For insertion sort   O(N^2)

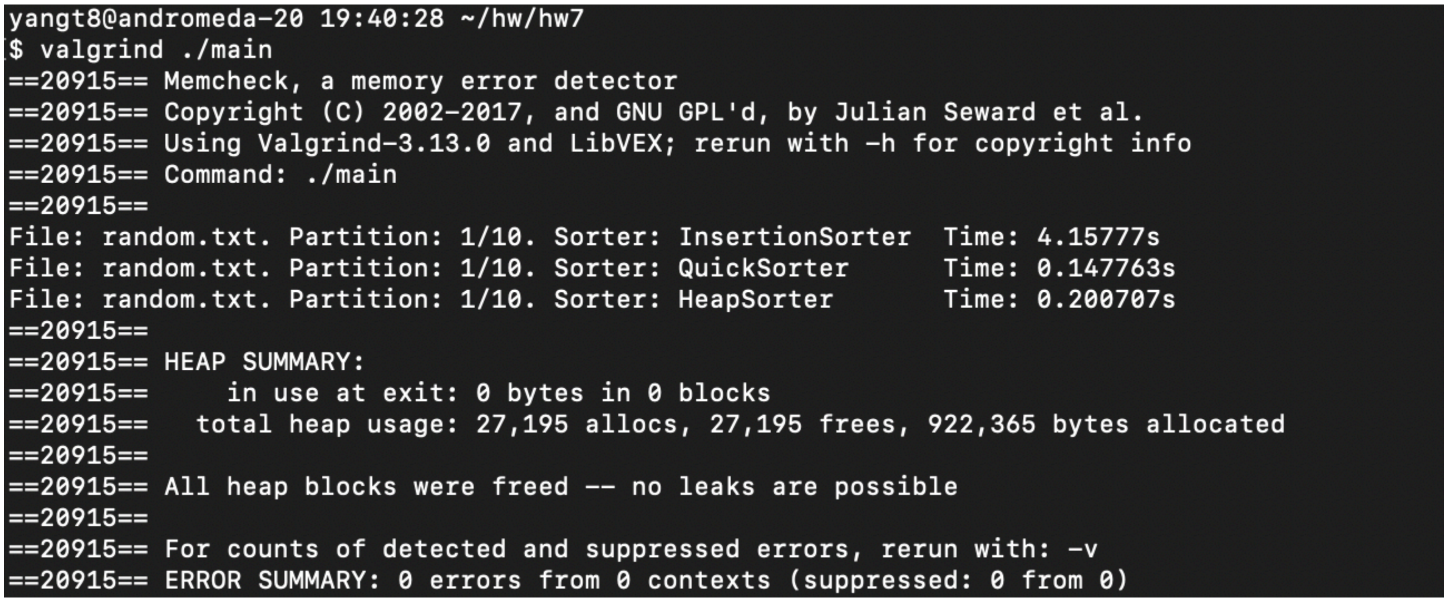


```
void Sorter::insertAllFromFile(const char * fileName, int numItemsToLoad)
{
    int i = 0;
    Timer t;
    double eTime;
    ifstream f(fileName);
    string w;
    t.start();
    while (i<4529*numItemsToLoad)
    {
        f>>w;
        l[i]=w;
        i++;
    }
    sort();
    f.close();
    t.elapsedUserTime(eTime);
    cout<< "Time: "<< eTime << "s"<< endl;
}
```

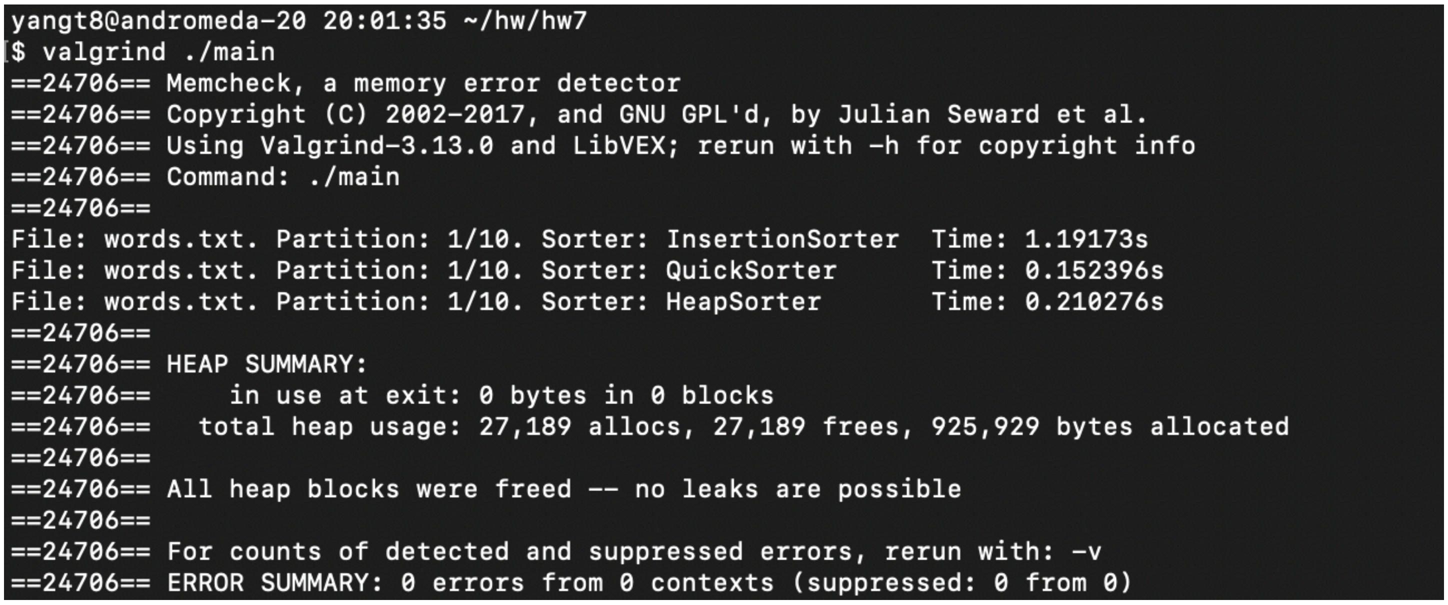
Part 2: Testing each sort function under valgrind with no memory leaks



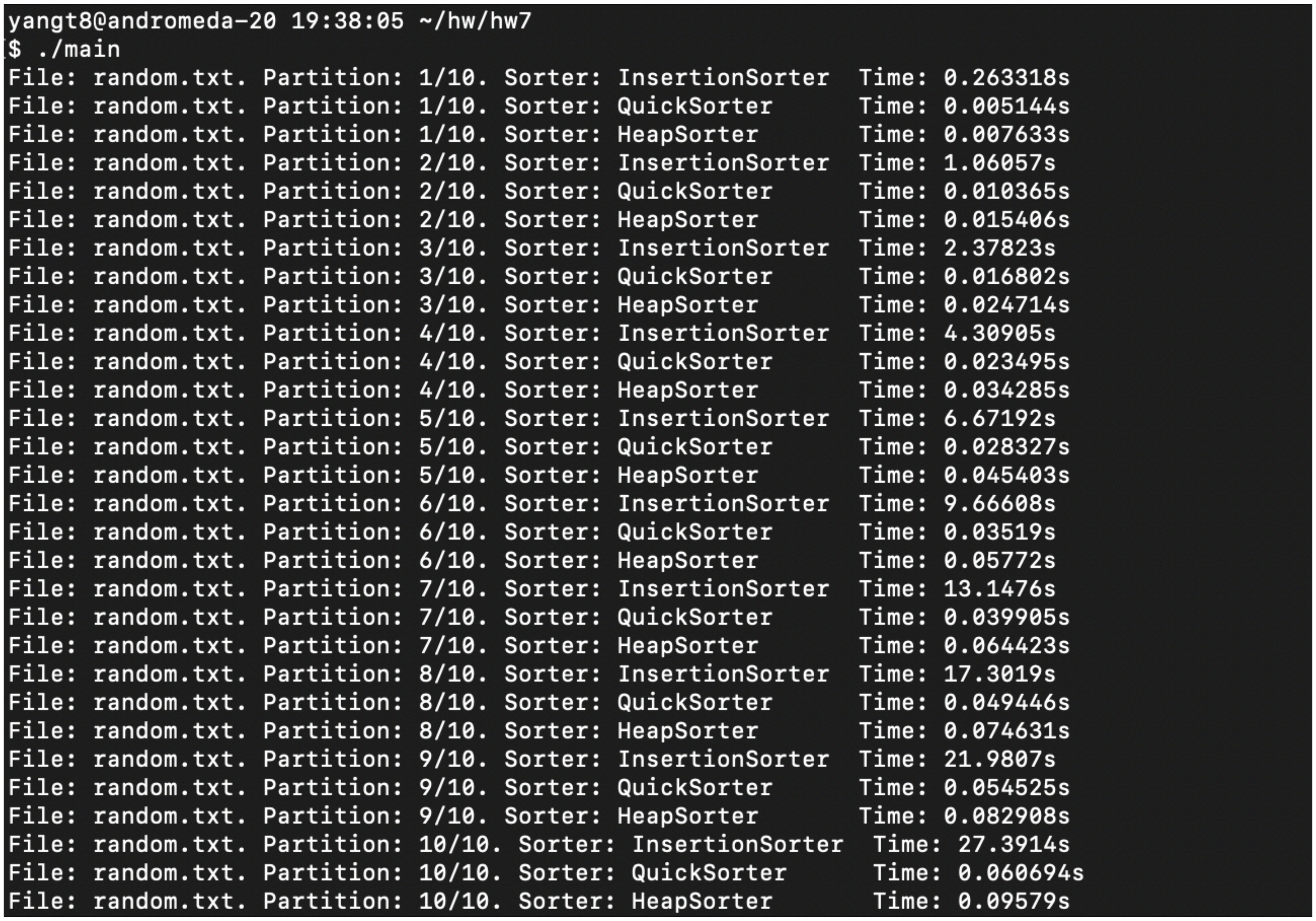
```
yangt8@andromeda-20 19:41:51 ~/hw/hw7
$ make
echo     ------------compiling main.cpp to create executable program main----------------
------------compiling main.cpp to create executable program main----------------
g++ -ggdb -std=c++0x -std=c++11 -Wpedantic -Wall -Wextra -Werror -Wzero-as-null-pointer-cons
tant main.cpp -o main
```



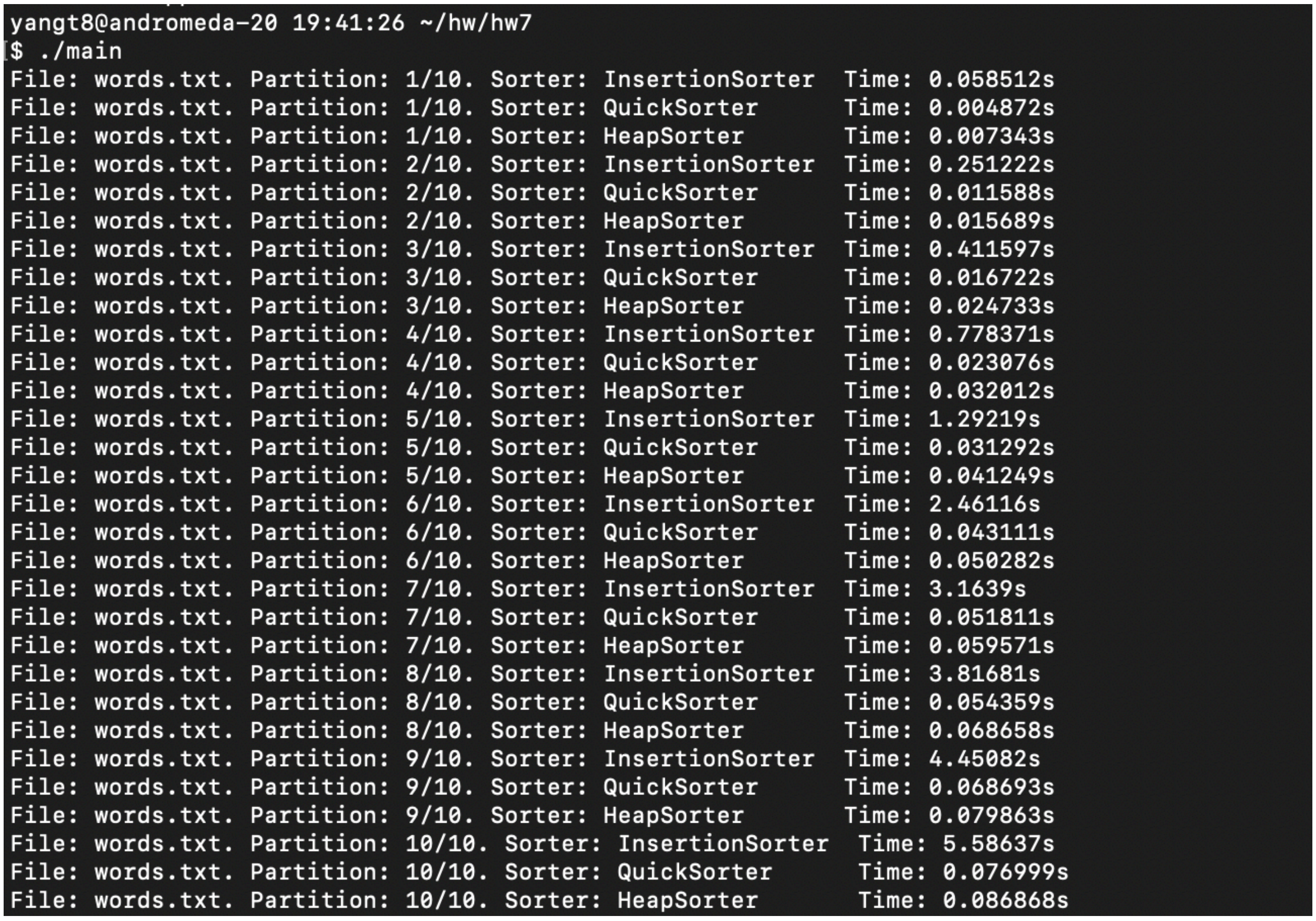
```
yangt8@andromeda-20 19:40:28 ~/hw/hw7
$ valgrind ./main
==20915== Memcheck, a memory error detector
==20915== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==20915== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==20915== Command: ./main
==20915==
File: random.txt. Partition: 1/10. Sorter: InsertionSorter  Time: 4.15777s
File: random.txt. Partition: 1/10. Sorter: QuickSorter      Time: 0.147763s
File: random.txt. Partition: 1/10. Sorter: HeapSorter       Time: 0.200707s
==20915==
==20915== HEAP SUMMARY:
==20915==     in use at exit: 0 bytes in 0 blocks
==20915==   total heap usage: 27,195 allocs, 27,195 frees, 922,365 bytes allocated
==20915==
==20915== All heap blocks were freed -- no leaks are possible
==20915==
==20915== For counts of detected and suppressed errors, rerun with: -v
==20915== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```



```
yangt8@andromeda-20 20:01:35 ~/hw/hw7
$ valgrind ./main
==24706== Memcheck, a memory error detector
==24706== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==24706== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==24706== Command: ./main
==24706==
File: words.txt. Partition: 1/10. Sorter: InsertionSorter  Time: 1.19173s
File: words.txt. Partition: 1/10. Sorter: QuickSorter      Time: 0.152396s
File: words.txt. Partition: 1/10. Sorter: HeapSorter       Time: 0.210276s
==24706==
==24706== HEAP SUMMARY:
==24706==     in use at exit: 0 bytes in 0 blocks
==24706==   total heap usage: 27,189 allocs, 27,189 frees, 925,929 bytes allocated
==24706==
==24706== All heap blocks were freed -- no leaks are possible
==24706==
==24706== For counts of detected and suppressed errors, rerun with: -v
==24706== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

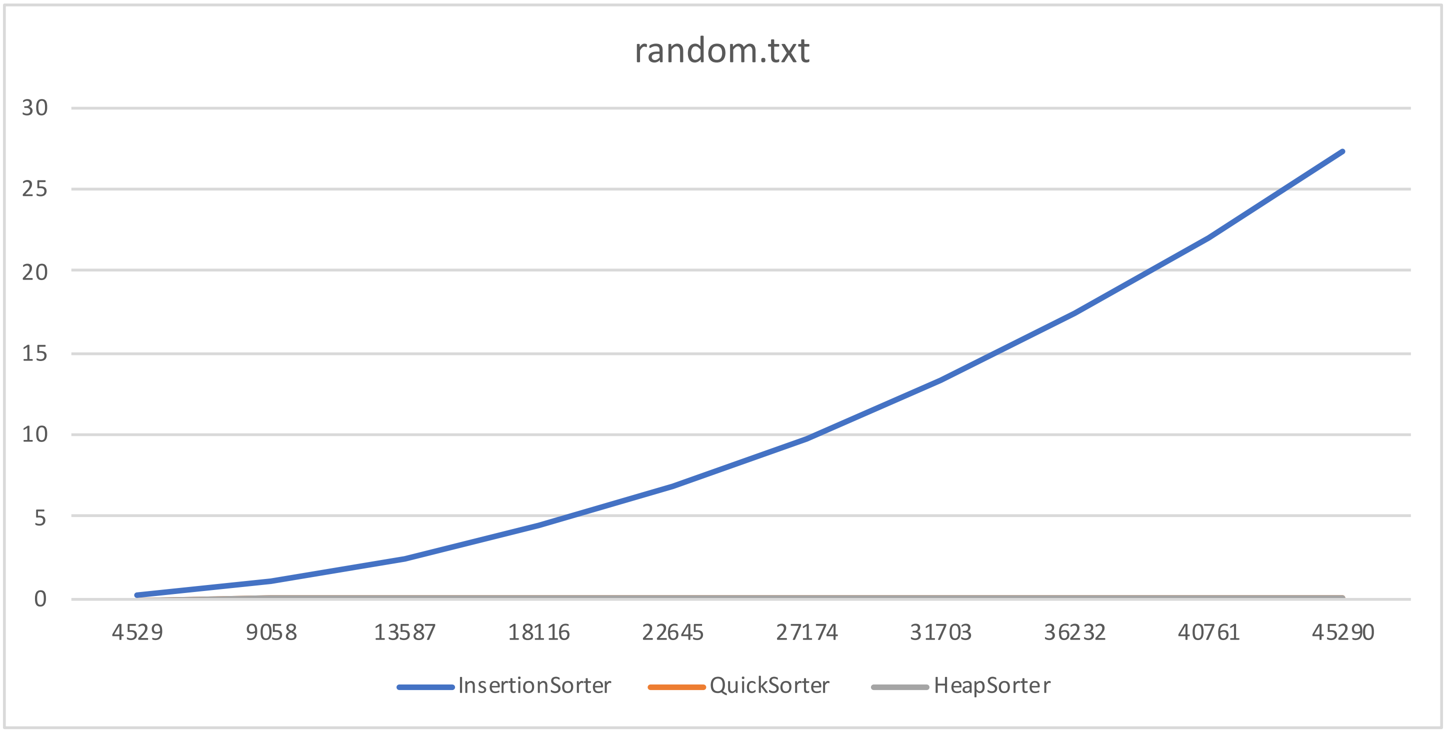


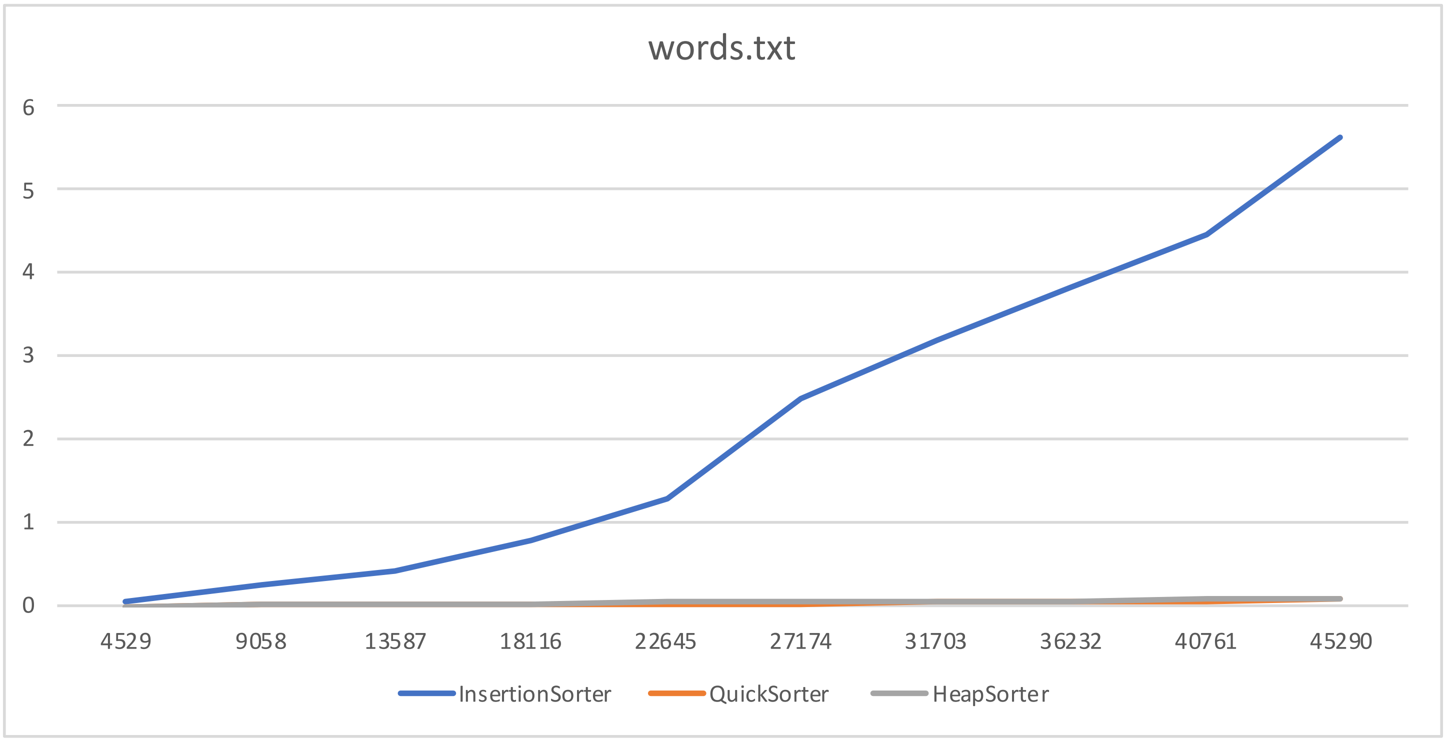
```
yangt8@andromeda-20 19:38:05 ~/hw/hw7
$ ./main
File: random.txt. Partition: 1/10. Sorter: InsertionSorter  Time: 0.263318s
File: random.txt. Partition: 1/10. Sorter: QuickSorter      Time: 0.005144s
File: random.txt. Partition: 1/10. Sorter: HeapSorter       Time: 0.007633s
File: random.txt. Partition: 2/10. Sorter: InsertionSorter  Time: 1.06057s
File: random.txt. Partition: 2/10. Sorter: QuickSorter      Time: 0.010365s
File: random.txt. Partition: 2/10. Sorter: HeapSorter       Time: 0.015406s
File: random.txt. Partition: 3/10. Sorter: InsertionSorter  Time: 2.37823s
File: random.txt. Partition: 3/10. Sorter: QuickSorter      Time: 0.016802s
File: random.txt. Partition: 3/10. Sorter: HeapSorter       Time: 0.024714s
File: random.txt. Partition: 4/10. Sorter: InsertionSorter  Time: 4.30905s
File: random.txt. Partition: 4/10. Sorter: QuickSorter      Time: 0.023495s
File: random.txt. Partition: 4/10. Sorter: HeapSorter       Time: 0.034285s
File: random.txt. Partition: 5/10. Sorter: InsertionSorter  Time: 6.67192s
File: random.txt. Partition: 5/10. Sorter: QuickSorter      Time: 0.028327s
File: random.txt. Partition: 5/10. Sorter: HeapSorter       Time: 0.045403s
File: random.txt. Partition: 6/10. Sorter: InsertionSorter  Time: 9.66608s
File: random.txt. Partition: 6/10. Sorter: QuickSorter      Time: 0.03519s
File: random.txt. Partition: 6/10. Sorter: HeapSorter       Time: 0.05772s
File: random.txt. Partition: 7/10. Sorter: InsertionSorter  Time: 13.1476s
File: random.txt. Partition: 7/10. Sorter: QuickSorter      Time: 0.039905s
File: random.txt. Partition: 7/10. Sorter: HeapSorter       Time: 0.064423s
File: random.txt. Partition: 8/10. Sorter: InsertionSorter  Time: 17.3019s
File: random.txt. Partition: 8/10. Sorter: QuickSorter      Time: 0.049446s
File: random.txt. Partition: 8/10. Sorter: HeapSorter       Time: 0.074631s
File: random.txt. Partition: 9/10. Sorter: InsertionSorter  Time: 21.9807s
File: random.txt. Partition: 9/10. Sorter: QuickSorter      Time: 0.054525s
File: random.txt. Partition: 9/10. Sorter: HeapSorter       Time: 0.082908s
File: random.txt. Partition: 10/10. Sorter: InsertionSorter  Time: 27.3914s
File: random.txt. Partition: 10/10. Sorter: QuickSorter      Time: 0.060694s
File: random.txt. Partition: 10/10. Sorter: HeapSorter       Time: 0.09579s
```



```
yangt8@andromeda-20 19:41:26 ~/hw/hw7
$ ./main
File: words.txt. Partition: 1/10. Sorter: InsertionSorter  Time: 0.058512s
File: words.txt. Partition: 1/10. Sorter: QuickSorter      Time: 0.004872s
File: words.txt. Partition: 1/10. Sorter: HeapSorter       Time: 0.007343s
File: words.txt. Partition: 2/10. Sorter: InsertionSorter  Time: 0.251222s
File: words.txt. Partition: 2/10. Sorter: QuickSorter      Time: 0.011588s
File: words.txt. Partition: 2/10. Sorter: HeapSorter       Time: 0.015689s
File: words.txt. Partition: 3/10. Sorter: InsertionSorter  Time: 0.411597s
File: words.txt. Partition: 3/10. Sorter: QuickSorter      Time: 0.016722s
File: words.txt. Partition: 3/10. Sorter: HeapSorter       Time: 0.024733s
File: words.txt. Partition: 4/10. Sorter: InsertionSorter  Time: 0.778371s
File: words.txt. Partition: 4/10. Sorter: QuickSorter      Time: 0.023076s
File: words.txt. Partition: 4/10. Sorter: HeapSorter       Time: 0.032012s
File: words.txt. Partition: 5/10. Sorter: InsertionSorter  Time: 1.29219s
File: words.txt. Partition: 5/10. Sorter: QuickSorter      Time: 0.031292s
File: words.txt. Partition: 5/10. Sorter: HeapSorter       Time: 0.041249s
File: words.txt. Partition: 6/10. Sorter: InsertionSorter  Time: 2.46116s
File: words.txt. Partition: 6/10. Sorter: QuickSorter      Time: 0.043111s
File: words.txt. Partition: 6/10. Sorter: HeapSorter       Time: 0.050282s
File: words.txt. Partition: 7/10. Sorter: InsertionSorter  Time: 3.1639s
File: words.txt. Partition: 7/10. Sorter: QuickSorter      Time: 0.051811s
File: words.txt. Partition: 7/10. Sorter: HeapSorter       Time: 0.059571s
File: words.txt. Partition: 8/10. Sorter: InsertionSorter  Time: 3.81681s
File: words.txt. Partition: 8/10. Sorter: QuickSorter      Time: 0.054359s
File: words.txt. Partition: 8/10. Sorter: HeapSorter       Time: 0.068658s
File: words.txt. Partition: 9/10. Sorter: InsertionSorter  Time: 4.45082s
File: words.txt. Partition: 9/10. Sorter: QuickSorter      Time: 0.068693s
File: words.txt. Partition: 9/10. Sorter: HeapSorter       Time: 0.079863s
File: words.txt. Partition: 10/10. Sorter: InsertionSorter  Time: 5.58637s
File: words.txt. Partition: 10/10. Sorter: QuickSorter      Time: 0.076999s
File: words.txt. Partition: 10/10. Sorter: HeapSorter       Time: 0.086868s
```

Part 3: Graphing the data


random.txt
InsertionSorter
QuickSorter
HeapSorter


words.txt
InsertionSorter
QuickSorter
HeapSorter