CS575 -01 Assignment 1

“I have done this assignment completely on my own. I have not copied it, nor have I given my solution to anyone else. I understand that if I am involved in plagiarism or cheating I will have to sign an official form that I have cheated and that this form will be stored in my official university record. I also understand that I will receive a grade of 0 for the involved assignment for my first offense and that I will receive a grade of “F” for the course for any additional offense.”

Shiva Subramanian

Time Complexity Analysis of Insertion Sort

Code snippet:

```
for(i=1;i<size;i++){
        int hole = i;
        int value = array[i];
        while(hole>0 && array[hole-1] > value){
                array[hole]=array[hole-1];
                hole = hole-1;
        }
        array[hole] = value;
        }
```

Analysis:
The outer loop will execute from 1 to N
Considering the worst case scenario the inner loop will run N times.
Taking these into consideration we arrive at the following equation

$\Sigma^{n}_{i=1} (i)$
$= (n)(n+1) / 2$ (as per summation rule $1+2+3..+n = n(n+1)/2$)
$= (n^2+n)/2$

Hence instruction count for Insertion sort Θ is Θ(n^2)

Time Complexity Analysis of Counting Sort

Code Snippet

```
for(i=0;i<size;i++){
        array1[array[i]]++;------------------------→Array 1 is auxiliary array
}

for(i=0;i<sizeOfArray1;i++){
        array1[i]=array1[i-1] +array1[i];------------→ Array 1 is auxiliary array
}
```

Analysis:
The above code snippet is taken for analysis using barometer operation.
We can see that the addition operation will be done at the most N-1 times which is linear
Hence we come to the below equation.
$\Sigma^{n}_{i=0}$ (1) = n (as per summation 1+1+1..+1 (n times) = N)

Hence Counting sort instruction count is Ɵ(n)

Time Complexity Analysis of Merge Sort

Code Snippet:

```
if(size>=2){

        int mid;int i;
        mid = (size/2);                                   c1
        int left [mid];int right[size-mid];

        for(i=0;i<mid;i++){
            left[i]=array[i];                             n/2
        }
                                                                   c2*n
        for(i=mid;i<size;i++){
            right[i-mid]=array[i];                        n/2
        }

        merge_sort(left,mid,limit); ---------------------------→T(n/2)
        merge_sort(right,(size-mid),limit);-------------------→T(n/2)
        merge(left,mid,right,(size-mid),array,limit);--------→c3.n + c4

    }
```

The equation is:

$T(n) = c1 + c2*n + T(n/2) + T(n/2) + c3*n + c4$

$T(n) = 2T(n/2) + (c1+c4) + (c2+c3)*n$

$T(n) =2T(n/2)+n$ (for large input we can neglect the constants)

$=2[2T(n/4)+ (n/2)]+n$

$=4T(n/4)+2n$

$=4[2T(n/8)+(n/4)]+2n$

$=8T(n/8) + 3n$

$=2^{k}T(n/2^{k})+n+n+....+n$

$=2^{k}T(n/2^{k})+kn$

$=2^{k}T(2^{k} /2^{k})+kn$

$=2^{k}T(1)+nlgn$ $n=2^{k}$ so we can say that lg n=klg2

$=2^{k}(1)+nlgn$

$=n+nlgn$

Hence T(n)= Ɵ(nlgn) for Merge Sort