Instruction Counts

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.

Plash Sachdeva

CS 575

Insertion Sort

```
void insertion_sort()
{       int i,j,x,k,l;
        for(i=0;i<n;i++)
        {       x=a[i];
                j=i-1;
                while(j>=0 && a[j]>x)
                {       a[j+1]=a[j];
                        j=j-1;
                }
                a[j+1]=x;
        }
}
```

In the worst case, inputs to the array consist of distinct items in reverse sorted order:

a[0]>a[1]>a[2]>........>a[n-1].

The total worst case number of comparisons is:

1+2+.....+n-1 = $\frac{(n-1)n}{2}$= $\frac{1}{2}$ (n^2-n) ἐ Θ(n^2)

$$\sum_{i=1}^{n-1} i-1=\frac{n(n-1)}{2}$$

W (n) = $\frac{n(n-1)}{2}$ ἐ Θ (n^2)

Count sort

```
void count_sort()
{       int i,b[100],q;
        for(i=0;i<=99;i++)
        {       b[i]=0;
        }
        for(i=0;i<=n-1;i++)
        {       b[a[i]]+=1;
        }
        q=0;
        for(i=0;i<=99;i++)
        {       while(b[i]>0)
                {       a[q]=i;
                        b[i]-=1;
                        q=q+1;
                }
        }
}
```

$$\sum_{i=0}^{99} 1 = i$$

Since the instruction count is polynomial, the time complexity is O(n) where n is the range of data in this case and is equal to 99.

Merge Sort

```
void merge_sort(int l, int r)
{       int mid;
        if(l<r)
        {       mid=(l+r-1)/2;
                merge_sort(l,mid);
                merge_sort(mid+1,r);
                merge(l,mid,r);
        }
}

void merge(int left,int mid,int right)
{       int i,j,k,m1,m2,h,l,q;
        m1=mid-left+1;
        m2=right-mid;
        int b[m1],c[m2];
        for(i=0;i<m1;i++)
        {       b[i]=a[left+i];
        }
        for(i=0;i<m2;i++)
        {       c[i]=a[mid+1+i];
        }
        i=0;
        j=0;
        k=left;
        q=0;
        while(i<m1 && j<m2)
        {       if(b[i]<=c[j])
                {       a[k]=b[i];
                        i++;
                }
                else
                {       a[k]=c[j];
                        j++;
                }
                k++;
        }
        while(i<m1)
        {       a[k]=b[i];
                i++;
                k++;
        }
        while(j<m2)
        {       a[k]=c[j];
                j++;
                k++;
        }
}
```

The barometer is the merge method. As the merge method consists of for and while loops that are NOT nested inside of each other. Merge_sort split the array into left and right halves recursively. Because the splits occur recursively, the time complexity of those counts is combined.

Another way of looking at this is through a recurrence equation.

It is $T(n)=T(n/2)+T(n/2)+(n)$
$T(0)=T(1)=1$

Which is equal to $\Theta(n\lg n)$.