

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

$$\begin{aligned} & \sum_{i=1}^n (i) \\ &= (n)(n+1) / 2 \quad (\text{as per summation rule } 1+2+3..+n = n(n+1)/2) \\ &= (n^2+n)/2 \end{aligned}$$

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

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.

$$\sum_{i=0}^n (1) = n \text{ (as per summation } 1+1+1..+1 \text{ (n times)} = N)$$

Hence Counting sort instruction count is $\Theta(n)$

Time Complexity Analysis of Merge Sort

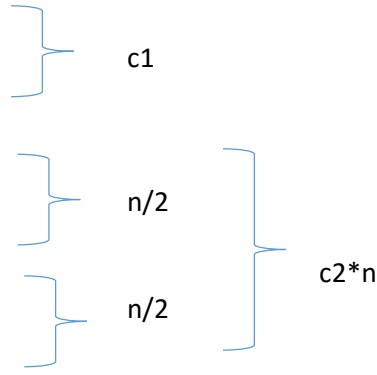
Code Snippet:

```
if(size>=2){
```

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

    for(i=0;i<mid;i++){
        left[i]=array[i];
    }
    for(i=mid;i<size;i++){
        right[i-mid]=array[i];
    }

    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:

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

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

$$T(n) = 2T(n/2) + n \quad (\text{for large input we can neglect the constants})$$

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

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

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

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

$$= 2^k T(n/2^k) + n + n + \dots + n$$

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

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

$$= 2^k T(1) + n \lg n \quad n = 2^k \text{ so we can say that } \lg n = k \lg 2$$

$$= 2^k (1) + n \lg n$$

$$= n + n \lg n$$

Hence $T(n) = \Theta(n \lg n)$ for Merge Sort