1. <u>Selection Sort:</u>

```
for(i = 0; i < size-1; ++i)
{
        originalSpot = i;
        for(next = i + 1; next < size; ++next)
        {
                if(data[next] < data[originalSpot])
                {
                    originalSpot = next;
                }
            }
}
```

For each i in outer loop inner loop executes for n-i times
Therefore, in total:

$= \sum_{i=1}^{n} n - i$

$= \frac{1}{2}(n-1)\, n$

Hence,
$W(n) = \theta(n^2)$

2. <u>Counting Sort</u>

```
for(i=0;i<100;i++)
{
        countingArray[i]=0;
}

for(i=0;i<size;i++)
{
        countingArray[data[i]]++;
}
int j=0;
for(i=0;i<100;i++)
{
        while(countingArray[i]>0)
        {
                data[j++] = i;
                countingArray[i]--;
        }
}
```

In first for loop there is one assignment operation which is constant
In second for loop there is increment operation which is constant time operation
In third for loop there is printing/copying element to original array, which is constant time operation

Hence,
Total count = $3 * \sum_{i=1}^{n} 1$
= 3 * n
Hence,
W(n) = θ(n)

3. Merge Sort

```
int mid = size/2;
mergeSort(arr1,mid);                          // ---------------------------------(1)
mergeSort(arr2,(size-mid));                   // ---------------------------------(2)
merge(data,arr1,arr2,mid,(size-mid));         // ---------------------------------(3)
```

Here these 3 are major barometric equations

Equation 1 & 2 run for T(n/2) operations and equation 3 runs for n times.
Therefor complexity in terms of equation count will be

T(n) = 1, for n=1
= T(n/2) + T(n/2) + n , otherwise
= 2T(n/2) + n

Using recurrence equations, reducing this we get,
T(n) = 2 T(n/2) + n
T(n) = 2 [2 T(n/4)+ n/2] + n
T(n) = 4 T(n/4) + 2n
T(n) = 4 [2 T(n/8) + n/4] + 2n
T(n) = 8 T(n/8) + 3n

Hence, we have
T(n) = 2^k T($n/2^k$) + k n ---------- (1)

T(1) = 1, n = 2^k or k = $\log_2 n$ = lg n

Substituting in equation 1,
T(n) = n T(1) + (lg n).n
T(n) = n + n lg n
T(n) = O(n lg n)