Big – Oh Notation:
Definition: T(n) = O(f(n)) if and only if there exist constance c , n_0 >0 such that T(n) ≤ cf(n) for all $n \geq n_0$.

Claim: if $T(n) = a_k n^k + \ldots\ldots + a_1 n + a_0$, then $T(n) = O(n^k)$
Proof:

$T(n) = a_k n^k + \ldots\ldots + a_1 n + a_0$
$\leq |a_k| n^k + \ldots\ldots\ldots + |a_1| n + a_0$
$\leq |a_k| n^k + \ldots\ldots\ldots + |a_1| n^k + |a_0| n^k$
$= (|a_k| + \ldots. + |a_1| + |a_0|)\, n^k$
$= c\, n^k$

$c = (|a_k| + \ldots. + |a_1| + |a_0|)$
$n_0 = 1$
therefore $T(n) = O(n^k)$

Big – Omega Notation:
Definition: T(n) = O(f(n)) if and only if there exist constance c , n_0 >0 such that T(n) ≥ cf(n) for all $n \geq n_0$.

Ex1.

```
Void max(int[]a){
        Int max = a[0];
        For(int I =1;i<a.length;i++){
                If(max < a[i]){
                        max = a[i];
                }
        }
}
```

O(n)

Ex2 .

```
Int sum(int a[]){
        Int sum = 0;
        For(int I =0;i< a.length;i++){
                Sum+= a[i];
        }
        Return sum;
}
```

O(n)

Ex3.

```
Int sum(int a[],int index){
        If(index == a.length)
                Return 0;
        Return a[i] + sum(a,index+1);
}
```

T(n) = T(n -1) +1
T(n -1) = T(n-2) +1
T(n – 2) = T(n-3) +1
.
.
.
T(1) =1

T(n) = T(n-1) +1
= T(n-2)+1 +1
= T(n-3)+ 1 +1 +1
.
.
.
=T(1)+ 1 + 1 +. . . . .+1
=n
T(n) = O(n)

Ex4

```
Boolean search(int a[],int t){
        for(int I =0; i< a.length; i++){
                If(a[i] == t){
                        Return true;
                }
        }
        return false;
}
```

O(n)

Ex5

```
Boolean search(int a[], int l ,int r,int t){
        If(r > l)
                Return Boolean;
        Int m = l + (r- l)/2;
        If( a[m] == t){return true;}
        If(a[m] < t)
```

```
                Return search(a, m+1, r,t);
        Return search(a, l, m-1,r);
}
```

$T(n) = T(n/2) + 1$

$T(n) = \log n$