selection sort algorithm (complexity analysis)

```
for(i=1 to n-1)  ⟶ (n-1) times
{
    value = A[i];
    loc = i;
    for(j= i+1 to n) → n(n-1)/2 times
    {
        if(A[j] < value)
        {
            value = A[j];
            loc = j;
        }
    }
    temp= A[i];
    A[i] = value;      } (n-1) times
    A[loc] = temp;
}
```

(n-1) + (n-2) + ... + 1 = n(n-1)/2

$\therefore C_1(n-1) + C_2\frac{n^2-n}{2} + C_3(n-1)$

If we reduce this equation we will get like this polynomial, an^2+bn+c where a, b, c are constants and n^2 is the biggest term so, the function is proportional to n^2

$\therefore f(n) \propto n^2$

The complexity is $O(n^2)$.

Insertion sort algorithm (complexity analysis)

```
for(i = 2 to n) ---> (n-1) times
{
    value = A[i];
    loc = i;
    while(loc > 1 && A[loc-1] > value)   ---> (n-1)+(n-2)+...+2+1 = n(n-1)/2
    {
        A[loc] = A[loc-1];
        loc = loc-1;
    }
    A[loc] = value; ---> (n-1) times
}
```

$\therefore c_1(n-1) + c_2\left(\frac{n(n-1)}{2}\right) + c_3(n-1)$

If we reduce this we will get like this; an^2+bn+c [a, b, c are constants]

where n^2 is the highest order term. If

∴ Complexity : $O(n^2)$

Bubble sort Algorithm (Complexity Analysis)

```
for( i = 1 ; i < n ; i++)          ---> (n-1) times
{
    for( j = 1 ; j < n ; j++)      ---> (n-1)(n-1) times
    {
        if( arr[j] > arr[j+1] )
        {
            temp = arr[j];
            arr[j] = arr[j+1];
            arr[j+1] = temp;
        }
    }
}
```

~~$c_1 n + c_2 n^2$~~

$c_1(n-1) + c_2(n-1)(n-1)$

$cn - c_1 + c_2 n^2 - 2c_2 n + c_2$

If we reduce this we will get like this polynomial, $an^2 + bn + c$ where n^2 is the biggest term $\therefore O(n^2)$ a, b, c are constants