# Taller en sala 5: Notación O Recursiva

Por:

Pablo Alberto Osorio Marulanda

Verónica Mendoza Iguarán

Datos y algoritmos I

Universidad Eafit

2018

**1.**

RECORDAR QUE EL PROGRAMA SE HA DETENIDO UN SEGUNDO POR ITERACIÓN (PARA PODER TOMAR LOS DATOS)

**1.1 Código en Word**

```
public static int [] sort(int [] arr){
        for(int i=0;i<arr.length;i++){
            for(int j=i;j>0;j--){
                if(arr[j]<arr[j-1]){
                    int temp=arr[j];
                    arr[j]=arr[j-1];
                    arr[j-1]=temp;
                }
            }
        return arr;
    }
```

**1.2 Etiquetar cuánto se demora cada línea**

```
public static int [] sort(int [] arr){
        int n = arr.length; //c
        for(int i=0;i<n;i++){ //c_1 * (n+1) + c_2
            for(int j=i;j>0;j--){  //c_3 *( i + 1) * n
                if(arr[j]<arr[j-1]){  //c_4 * i * n
                    int temp=arr[j]; //c_5 * i * n
                    arr[j]=arr[j-1]; //c_6 * i * n
                    arr[j-1]=temp; //c_7 * i * n
                }
            }
        return arr; // c_8
    }
```

### 1.3 Solución de la ecuación con Wolfram Alpha

T(n,i) = c_1*(n+1) + c_2 + c_3*(i+1)*n + (c_4+c_5+c_6+c_7)*(i*n)

T(n,i) = c_1*(n+1) + (1+ i) * (c_3*n) + i*(c_4+c_5+c_6+c_7)*n +c_2

### 1.4 Notación O

T(n,i) es O(c_1*(n+1) + (1+ i) * (c_3*n) + i*(c_4+c_5+c_6+c_7)*n +c_2), por definición de O

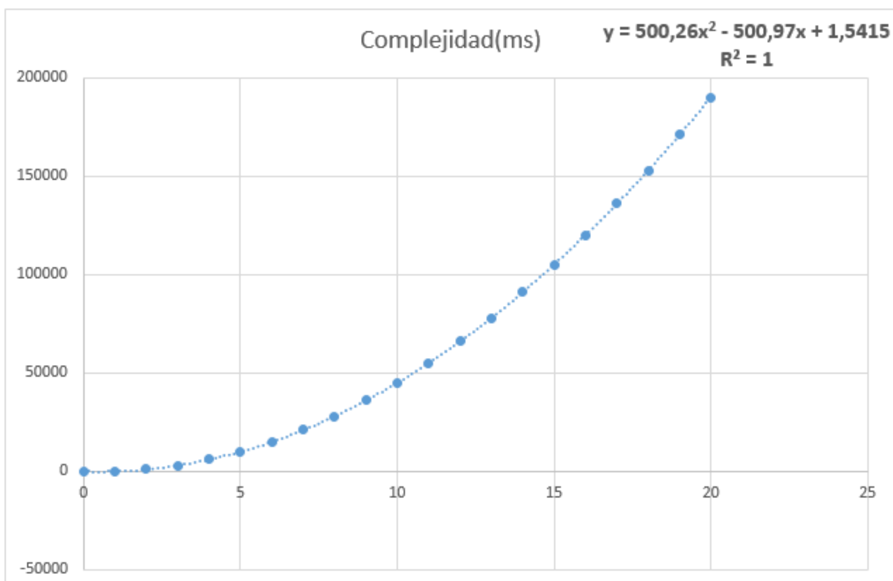
T(n,i) es O(c_1*(n+1)) + O((1+ i) * (c_3*n)) + O(i*(c_4+c_5+c_6+c_7)*n)+ O(C_2), por regla de la suma

T(n,i) es O(i*(c_4+c_5+c_6+c_7)*n), por regla de la suma

T(n,i) es O(i*n), por regla del producto

### 1.5 Gráfica

| n | Tiempo(ms) |
|---|---|
| 0 | 0 |
| 1 | 0 |
| 2 | 1001 |
| 3 | 3001 |
| 4 | 6001 |
| 5 | 10004 |
| 6 | 15007 |
| 7 | 21009 |
| 8 | 28011 |
| 9 | 36013 |
| 10 | 45025 |
| 11 | 55022 |
| 12 | 66030 |
| 13 | 78030 |
| 14 | 91042 |
| 15 | 105042 |
| 16 | 120047 |
| 17 | 136056 |
| 18 | 153066 |
| 19 | 171094 |
| 20 | 190085 |




### 1.6 Explicación:

No, este algoritmo no es apropiado para ordenar grandes volúmenes de datos, esto debido a que para un arreglo con una cantidad considerable de posiciones sería muy poco eficiente. Por ejemplo, para un arreglo de 1000000 de posiciones, el número de operaciones que tendrá que hacer sería 1000000^2, que sería igual 1000000000000 aproximadamente. Tal tamaño es considerable teniendo en cuenta la capacidad de un PC regular. Sin embargo, si esta cantidad la elevamos al cuadrado, es decir (1 trillon)^2, la cantidad de operaciones que debe realizar es abismal, equivalente a 1*10^24. Si hacemos 2.1 *10^9 operaciones por segundo, nos demoraríamos mas de 15 millones de años. Ahora, para un arreglo con n=100000000, nos demoraríamos aproximadamente 2777 horas. Asi que, en definitiva, este algoritmo no es muy eficiente para valores muy grandes.

RECORDAR QUE EL PROGRAMA SE HA DETENIDO UN SEGUNDO POR ITERACIÓN (PARA PODER TOMAR LOS DATOS)

**2.1 Código en Word:**

```
public static int suma(int[] a){
  int suma = 0;
  for(int i = 0; i < a.length; i++)
    suma += a[i];
  return suma;
}
```

**2.2 Etiquetar cuánto se demora cada línea**

```
public static int suma(int[] a){
  int suma = 0; // c_1
  for(int i = 0; i < a.length; i++) // c2 + sum c3, i=0 to n
    suma += a[i];  //sum c4, i=0 to n-1
  return suma; //c5
}
```

**2.3 Solución de la ecuación con Wolfram Alpha**

$T(n) = c1 + c2 + c3(n+1) + c4n + c5$

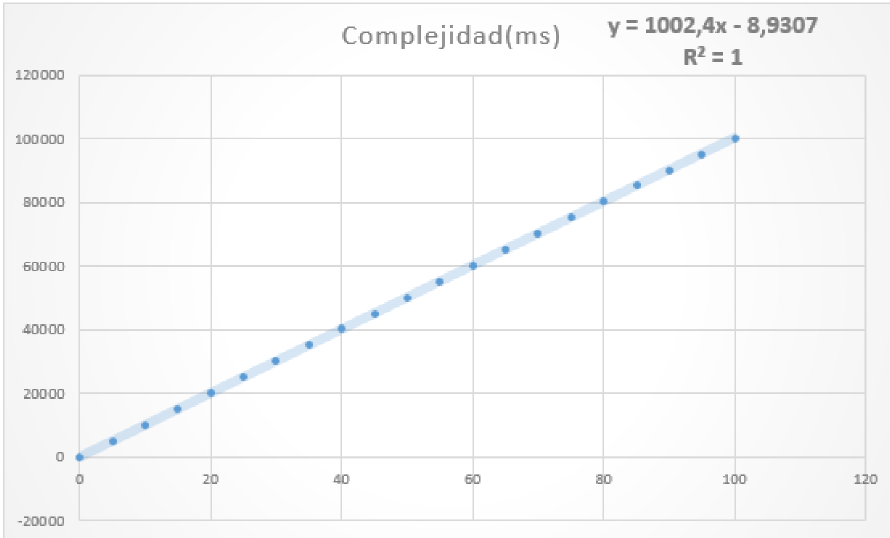
**2.4 Notación O**

T(n) es O(c3n), por regla de la suma

T(n) es O(n), por regla del producto

**2.5 Gráfica**

| n | Tiempo(ms) |
|---|---|
| 0 | 0 |
| 5 | 5002 |
| 10 | 10011 |
| 15 | 15012 |
| 20 | 20044 |
| 25 | 25059 |
| 30 | 30073 |
| 35 | 35064 |
| 40 | 40072 |
| 45 | 45071 |
| 50 | 50158 |
| 55 | 55093 |
| 60 | 60144 |
| 65 | 65121 |
| 70 | 70140 |
| 75 | 75199 |
| 80 | 80217 |
| 85 | 85162 |
| 90 | 90190 |
| 95 | 95252 |
| 100 | 100198 |




**2.6 Solución pregunta:**

Realmente no existe una gran diferencia en cuanto a los tiempos retornados cuando se suma el arreglo con recursión y cuando lo hace por medio de ciclos. Ambos, de hecho, pertenecen a la los O(n), y dado que para ambos su crecimiento es lineal, no hay mucho que esperar en comparación con los tiempos respectivos que arrojan cada método de solución.

RECORDAR QUE EL PROGRAMA SE HA DETENIDO UN SEGUNDO POR ITERACIÓN (PARA PODER TOMAR LOS DATOS)

**3.**

**3.1 Código en Word**

```
public static void tablas(int n, int m){
    for (int i=1;i<=n;i++){
        for(int j=1;j<=m;j++){
            System.out.println(i+"x"+j+"="+(i*j));
        }
    }
}
```

**3.2 Etiquetar cuánto se demora cada línea**

```
public static void tablas(int n, int m){
      for (int i=1;i<=n;i++){  //c_1 * n + c_2
         for(int j=1;j<=m;j++){ //c_3 * m * n
            System.out.println(i+"x"+j+"="+(i*j)); //c_4 * m * n
         }
      }
   }
```

### 3.3 Solución de la ecuación con Wolfram Alpha

T (n,m) = c_1 * n + c_2 + c_3 * m * n + c_4 * m * n

T(n,m) = c_1 * n + c_2 + (c_3+c_4)*m*n

### 3.4 Notación O

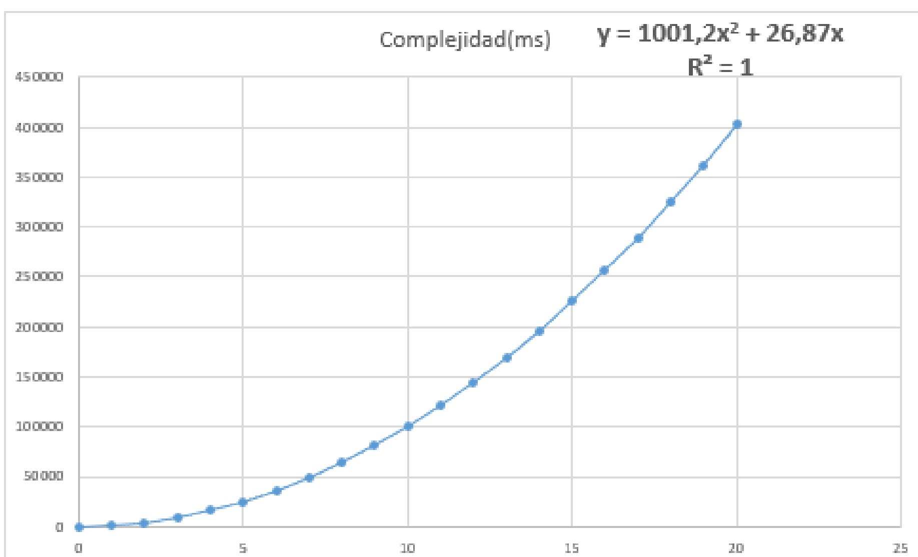
T(n,m) es O(c_1 * n + c_2 + (c_3+c_4)*m*n), por definición de O

T(n,m) es O((c_3+c_4)*m*n), por regla de la suma

T(n,m) es O(n*m), por regla del producto

### 3.5 Gráfica

| n | Tiempo(ms) |
|---|---|
| 0 | 0 |
| 1 | 1013 |
| 2 | 4023 |
| 3 | 9057 |
| 4 | 16083 |
| 5 | 25198 |
| 6 | 36239 |
| 7 | 49370 |
| 8 | 64459 |
| 9 | 81538 |
| 10 | 100732 |
| 11 | 121828 |
| 12 | 144391 |
| 13 | 169209 |
| 14 | 196208 |
| 15 | 225426 |
| 16 | 256959 |
| 17 | 289666 |
| 18 | 324311 |
| 19 | 361135 |
| 20 | 402356 |




### 3.6 Interpretación

Los resultados encontrados a partir de la teoría si se adaptan a los encontrado experimentalmente. A partir de los resultados teóricos encontramos que la complejidad del problema pertenece al orden de los O(n*m), es decir, es polinómica. Y, como se puede observar en la gráfica, los tiempos y el n tomado también cumplen tal característica.