

# Apuntes de Análisis de Algoritmos

Leonardo H. Añez Vladimirovna

*Universidad Autónoma Gabriel René Moreno,  
Facultad de Ingeniería en Ciencias de la Computación y Telecomunicaciones,  
Santa Cruz de la Sierra, Bolivia*

3 de junio de 2018

Estos apuntes fueron realizados durante mis clases en la materia INF210 (Programación II), acompañados de referencias de libros, fuentes y código que use a lo largo del curso, en el período I-2018 en la Facultad de Ingeniería en Ciencias de la Computación y Telecomunicaciones.

Para cualquier cambio, observación y/o sugerencia pueden enviarme un mensaje al siguiente correo:

`toborochi98@outlook.com`

## 1. Algoritmia

### 1.1. Conceptos

#### 1.1.1. Algoritmo

Secuencia de pasos a seguir para resolver un problema. Un algoritmo tiene las siguientes características:

- Finitud
- Exactitud
- Sin Ambigüedades

Esquema de un algoritmo:

- **Entrada:** Conjunto de Datos a ser procesado. Donde la cantidad de datos es representada por:  $n$ .
- **Salida:** Conjunto de datos resultantes del proceso.
- **Proceso:** Conjunto de Instrucciones para transformar la entrada en la salida.

### 1.2. Tiempo de Ejecución

Denotado por:

$$T_A(n)$$

Es el tiempo que le toma a un algoritmo  $A$  procesar una entrada de tamaño  $n$ . Para determinar esto es necesario diferenciar: *Cantidad de Operaciones* y *Cantidad de Instrucciones*.

#### 1.2.1. Cantidad de Operaciones

Para entender este punto basta ver el siguiente ejemplo:

```
1 x = a + b*c;
```

Donde la cantidad de operaciones es de tres, estas son el producto de  $b*c$ , esto con la suma de  $a$  forman dos operaciones la tercera operación es todo eso asignado a  $x$ .

### 1.2.2. Cantidad de Instrucciones

Para el mismo ejemplo anterior:

```
1 x = a + b*c;
```

En este caso la cantidad de *Instrucciones* es de una.

### 1.3. Cálculo de $T(n)$

Para el cálculo de tiempo tendremos dos clasificaciones:

1. **Algoritmos Iterativos:** Usando tablas de conteo.
2. **Algoritmos Recursivos:** Mediante Ecuaciones de Recurrencia.

### 1.4. Clasificación de Algoritmos por su Grado de Complejidad

- |                           |                                |
|---------------------------|--------------------------------|
| ■ Algoritmos Constantes   | ■ Algoritmos $n$ -logarítmicos |
| ■ Algoritmos Lineales     | ■ Algoritmos Cúbicos           |
| ■ Algoritmos Logarítmicos | ■ Algoritmos Exponenciales     |
| ■ Algoritmos Cuadráticos  | ■ Algoritmos Factoriales       |

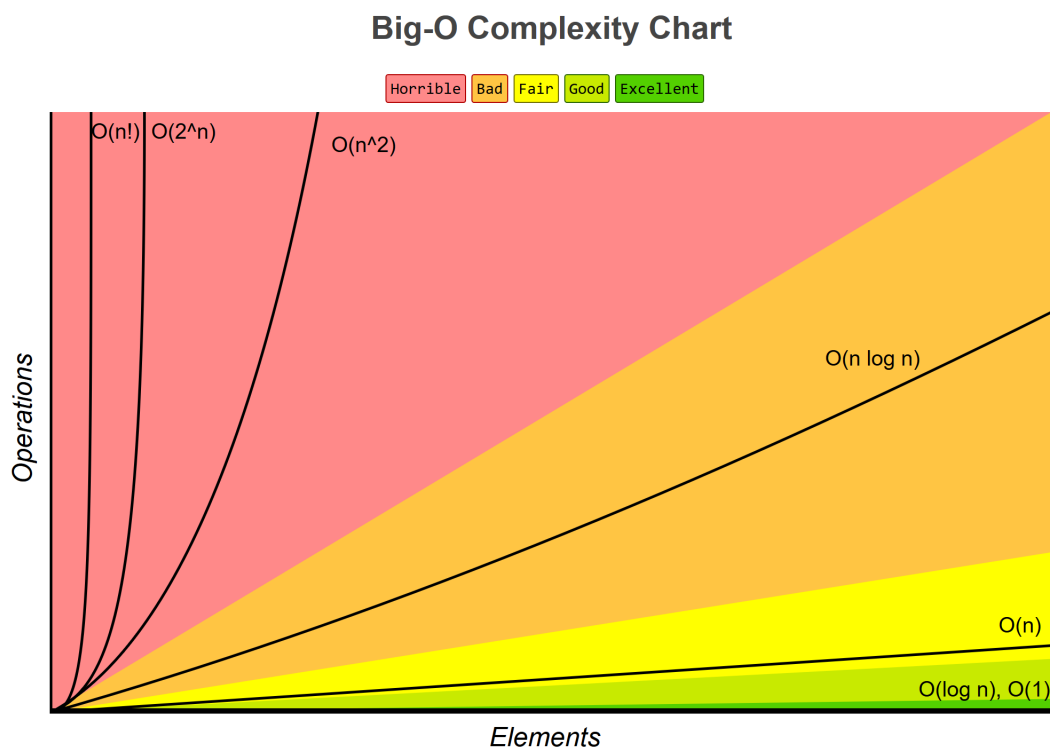


Figura 1: Gráfica que expresa el crecimiento de las operaciones para cada Grado de Complejidad.  
Fuente: <http://bigocheatSheet.com>

## 2. Cálculo de Tiempo

### 2.1. Algoritmos Constantes

Tiene la siguiente forma:

$$T(n) = C \quad \text{donde } C \text{ es una Constante}$$

Son algoritmos que no contienen ciclos (que se ejecuten) y tarda siempre el mismo tiempo.

### 2.1.1. Ejemplos

1. Función que Intercambia dos variables por referencia:

```
1 void intercambiar (int &x, int &y)
2 {
3     int w = x;
4     x = y;
5     y = w;
6 }
```

Podemos ver que las únicas líneas que contienen instrucciones son las líneas 3, 4 y 5. Y estas siempre serán las instrucciones que se ejecutaran, sin importar los datos de la función. Por lo tanto:

$$T_{intercambiar}(2) = 3$$

2. Función que devuelve el mayor de los datos de un vector ordenado:

```
1 int mayor (int v[], int n)
2 {
3     return v[n-1];
4 }
```

En esta función no hay mucho misterio, si vemos la línea 3 sabemos que:

$$T_{mayor} = 1$$

## 2.2. Algoritmos Lineales

Tienen la siguiente forma:

$$T(n) = An + B$$

### 2.2.1. Ejemplos

1. Función que devuelva la suma de los elementos de un arreglo:

```
1 int SumaElementos (int v[], int n)
2 {
3     int i, s;
4     i=0, s=0;
5     while (i<n)
6     {
7         s = s + v[i];
8         i++;
9     }
10    return s;
11 }
```