1. Algoritmia

1.1. Conceptos

1.1.1. Algoritmo

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

- Finitud
- Exactitud
- Sin Ambiguedades

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 Ejecucin

Denotado por:

$$T_A(n)$$

Es el tiempo que le toma a un algoritmo A procesar una entrada de tamao 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:

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

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

1.2.2. Cantidad de Instrucciones

Para el msmo ejemplo anterior:

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

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

1.3. Clculo de $T(n)$

Para el clculo de tiempo tendremos dos clasificaciones:

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

1.4. Clasificacin de Algoritmos por su Grado de Complejidad

- Algoritmos Constantes
- Algoritmos Lineales
- Algoritmos Logartmicos
- Algoritmos Cuadrticos
- Algoritmos n-logartmicos
- Algoritmos Cbicos
- Algoritmos Exponenciales
- Algoritmos Factoriales

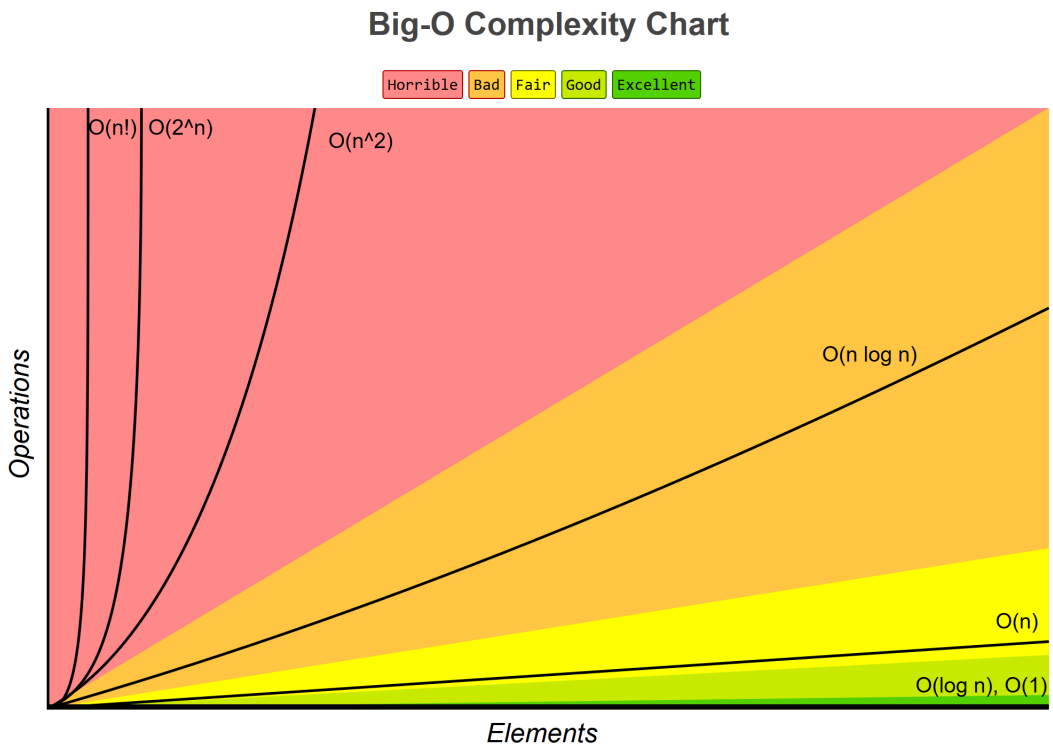

Figura 1: Grfica que expresa el crecimiento de las operaciones para cada Grado de Complejidad.
Fuente: `http://bigocheatsheet.com`

2. Clculo de Tiempo

2.1. Algoritmos Constantes

Tiene la siguiente forma:

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

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

2.1.1. Ejemplos

1. Funcin que Intercambia dos variables por referencia:

```
void intercambiar(int &x,int &y)
{
    int w = x;
    x = y;
    y = w;
}
```

Podemos ver que las nicas lneas que contienen instrucciones son las lneas `3`,`4` y `5`. Y estas siempre sern las instrucciones que se ejecutaran, sin importar los datos de la funcin. Por lo tanto:

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

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

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

En esta funcin no hay mucho misterio, si vemos la lnea `3` sabemos que:

$$T_{mayor}(n) = 1$$

2.2. Algoritmos Lineales

Tienen la siguiente forma:

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

2.2.1. Ejemplos

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

```
int SumaElementos(int v[],int n)
{
    int i,s;
    i=0, s=0;
    while(i<n)
    {
        s = s + v[i];
        i++;
    }
    return s;
}
```

Realizamos una tabla de conteo para cada lnea que contenga instrucciones:

l	$T(l)$
4	2
5	$n+1$
7	n
8	n
10	1

Finalmente el tiempo de este ejemplo es la suma de los tiempos de todas las lneas, esto es:

$$T_{SumaElementos}(n) = 3n + 4$$

Esto quiere decir que por ejemplo, si tuvieramos un vector con cuatro elementos, el tiempo de este algoritmo sera:

$$T_{SumaElementos}(4) = 3 \cdot 4 + 4 = 16$$

Esto significa que le tomara 16 instrucciones sumar un vector de cuatro elementos.