



# U1. Conceptos básicos de algoritmos

Algoritmos y Estructuras de Datos

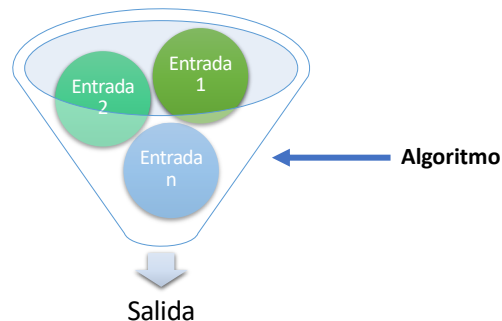
Por: Violeta Ocegueda

1



## Qué es un algoritmo?

- Es una **secuencia de pasos computacionales** que transforman un valor o conjunto de valores de **entrada** en un valor o conjunto de valores de **salida**.



2

## Ejemplo:

a:

5	6	4	9	7	1	8
---	---	---	---	---	---	---

k: 7

Queremos saber si hay dos elementos en a, tal que:

$$a[i] + a[j] = k$$

Algunas posibles soluciones son:

- Probar todos los posibles pares.
- Ordenar el arreglo, poner las variables izq y der, y mover estos valores.
- Guardar los elementos que ya hemos encontrado  $k - a[i]$ .



Todas funcionan

**Cuál elegir?**

3

## Cómo elegir un algoritmo?

- Comparar las diferentes soluciones:
  - # de líneas de código.
  - Qué tan entendible es el código.
  - Conjunto de criterios propios.
- Cantidad de **recursos requeridos** para implementar la solución:
  - Tiempo
  - Espacio

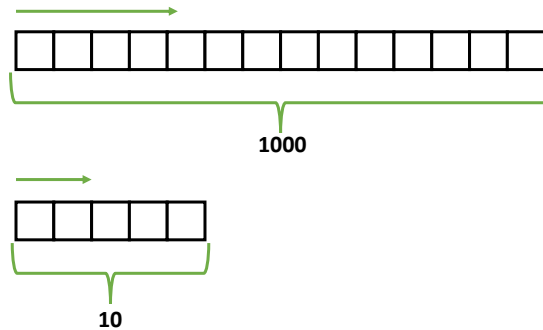
**Análisis de la complejidad de los algoritmos**

4

## Análisis de la complejidad de los algoritmos



- Los algoritmos requieren **tiempo** y **espacio**, comúnmente asociados con el tamaño de la entrada.



No es lo mismo recorrer un arreglo de 1000 elementos que uno de 10 elementos.

## Análisis de la complejidad de los algoritmos



- Describe la cantidad de **recursos requeridos** en términos del tamaño de la **entrada**.
- Dos tipos:
  - Análisis de la complejidad del tiempo.
  - Análisis de la complejidad del espacio.



## Análisis de la complejidad del tiempo

- En la ejecución de un algoritmo se ejecutan diferentes operaciones:
  - Asignación de valores
  - Comparación de valores
  - Creación de variables
  - Etc.
- Objetivo:
  - Conocer cómo se comportan la cantidad de **operaciones ejecutadas** por el algoritmo con respecto al **tamaño de la entrada**.



## Ejemplo 1:

Suma todos los números desde 1 hasta n.

```
int suma(int n){
    int sum = 0;
    while(n>0){
        sum += n;
        n -= 1;
    }
    return sum;
}
```

Si n = 5:

$$5+4+3+2+1 = 15$$

```
int suma(int n){
    return n*(n+1)/2;
}
```

Si n = 5:

$$5*(5+1)/2 = 15$$



## Ejemplo 1:

Suma todos los números desde 1 hasta n.

```
int suma(int n){
    int sum = 0;
    while(n>0){
        sum += n;
        n -= 1;
    }
    return sum;
}
```

Si n = 7:

$$7+6+5+4+3+2+1 = 28$$

```
int suma(int n){
    return n*(n+1)/2;
}
```

Si n = 7:

$$7*(7+1)/2 = 28$$



## Ejemplo 1:

Suma todos los números desde 1 hasta n.

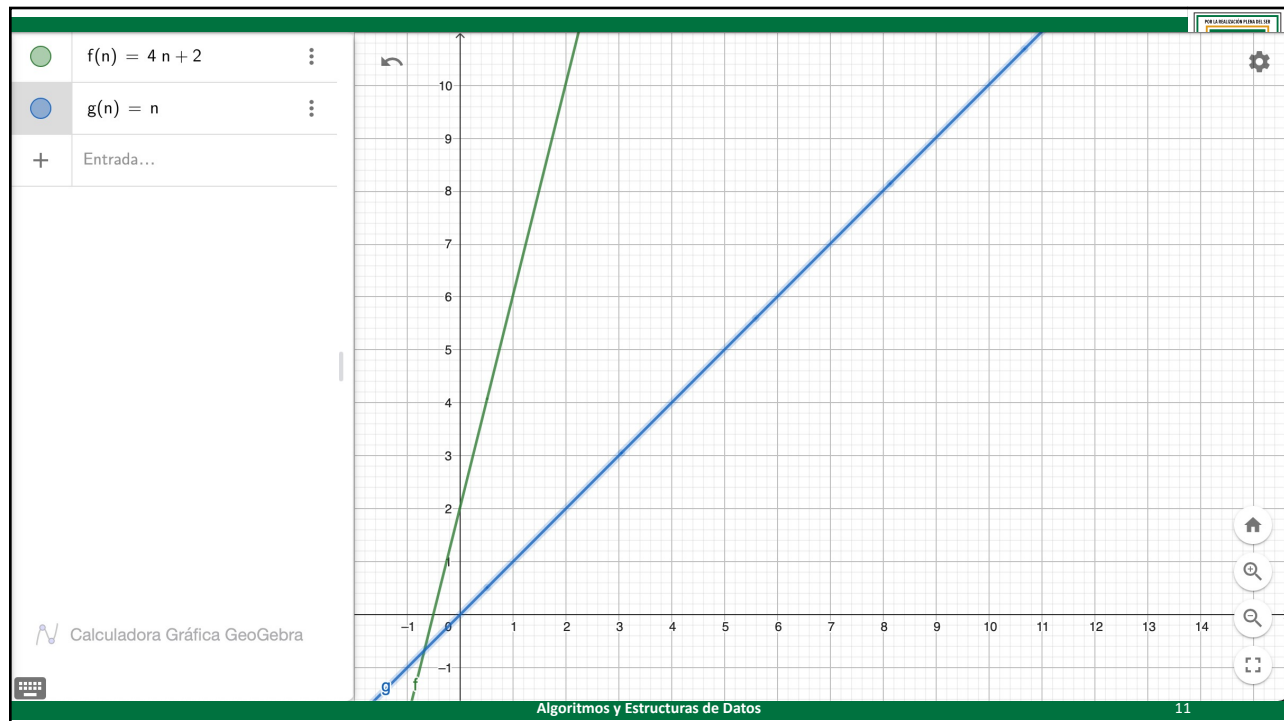
```
int suma(int n){
    int sum = 0;
    while(n>0){
        sum += n;
        n -= 1;
    }
    return sum;
}
```

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

$$T(n) = \cancel{1n} + \cancel{1}$$

$$\longrightarrow T(n)$$

```
int suma(int n){
    return n*(n+1)/2;
}
```



11

## Ejemplo 1:

Suma todos los números desde 1 hasta n.

```

int suma(int n){
    int sum = 0;
    while(n > 0){
        sum += n;
        n -= 1;
    }
    return sum;
}

```

```

int suma(int n){
    return n*(n+1)/2;
}

```

$T(n) = 1 + n(2+2) + 1 \rightarrow T(n)$   
 $T(n) = \cancel{4n} + \cancel{4} \rightarrow T(n)$

$T(n) = 4 \rightarrow T(1)$

Algoritmos y Estructuras de Datos

12



## Ejemplo 2:

Recorre una matriz cuadrada.

**n**

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

**n**

**n = 4**

```
void imprimeMatriz(int mat[4][4], int n){
    int i, j;
    for(i=0; i<n; i++){
        for(j=0; j<n; j++){
            printf("%d", mat[i][j]);
        }
    }
}
```

13

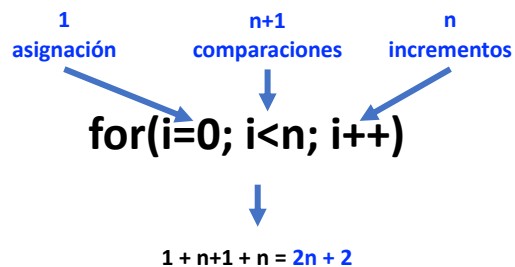


## Ejemplo 2:

Recorre una matriz cuadrada.

**n = 4**

```
void imprimeMatriz(int mat[4][4], int n){
    int i, j;
    for(i=0; i<n; i++){
        for(j=0; j<n; j++){
            printf("%d", mat[i][j]);
        }
    }
}
```



14



## Ejemplo 2:

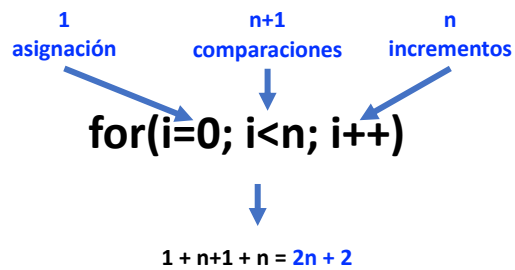
Recorre una matriz cuadrada.

```

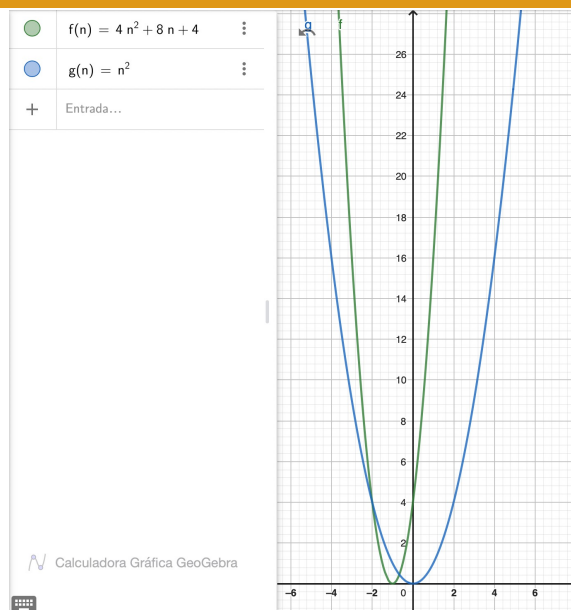
n = 4
void imprimeMatriz(int mat[4][4], int n){
    int i, j;
    for(i=0; i<n; i++){
        for(j=0; j<n; j++){
            printf("%d", mat[i][j]);
        }
    }
}

```

$$\begin{aligned}
 T(n) &= (2n+2)(2n+2)(1) \\
 &= 4n^2 + 4n + 4n + 4 \\
 &= \cancel{n^2} + 8n + 4 \longrightarrow T(n^2)
 \end{aligned}$$



15



16





## Ejemplo 3:

Recorre una matriz no cuadrada.

**m**

	1	2	3	4
<b>n</b>	5	6	7	8
	9	10	11	12

**n = 3, m = 4**

```
void imprimeMatriz(int mat[3][4], int n, int m){
    int i, j;
    for(i=0; i<n; i++){
        for(j=0; j<m; j++){
            printf("%d", mat[i][j]);
        }
    }
}
```

17



## Ejemplo 2:

Recorre una matriz no cuadrada.

**n = 3, m = 4**

```
void imprimeMatriz(int mat[3][4], int n, int m){
    int i, j;
    for(i=0; i<n; i++){
        for(j=0; j<m; j++){
            printf("%d", mat[i][j]);
        }
    }
}
```

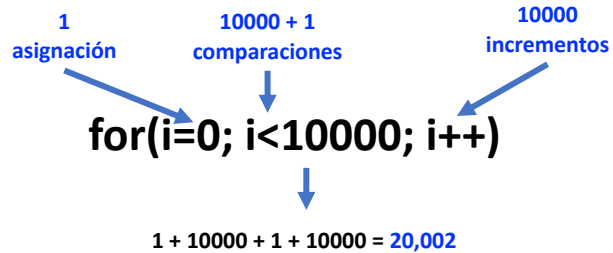
$$T(n) = (2n+2)(2m+2)(1) \\ = 4nm + 4n + 4m + 4 \longrightarrow T(nm)$$

18

### Ejemplo 3:

Imprime n 10,000 veces.

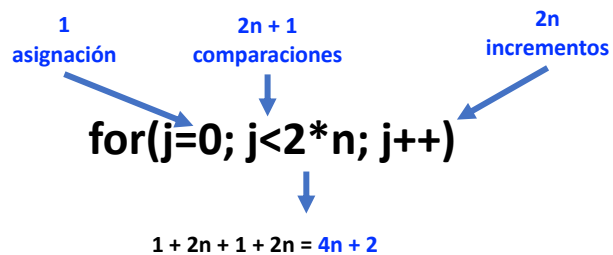
```
void imprime(int n){
    int i;
    for(i=0; i<10000; i++){
        printf("%d", n);
    }
}
```



$$T(n) = (20002)(1) = 20002 \longrightarrow T(1)$$

### Ejemplo 4:

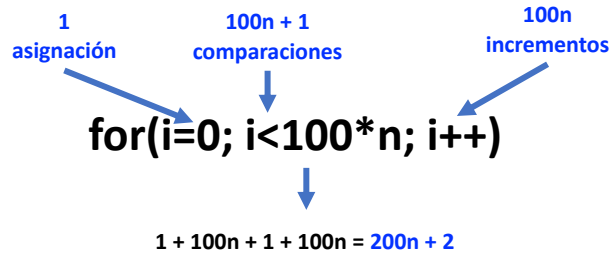
```
int funcion(int n){
    int i, j, suma;
    suma = 0;
    for(i=0; i<n; i++){
        for(j=0; j<2*n; j++){
            suma += j;
        }
        for(i=0; i<100*n; i++){
            suma += i;
        }
        for(i=0; i<4; i++){
            for(j=0; j<3; j++){
                suma += j;
            }
        }
    }
    return suma;
}
```



## Ejemplo 4:



```
int funcion(int n){
    int i, j, suma;
    suma = 0; ➔ 1
    for(i=0; i<n; i++) ➔ 2n + 2
        for(j=0; j<2*n; j++) ➔ 4n + 2
            suma += j; ➔ 2
    for(i=0; i<100*n; i++) ➔ 200n + 2
        suma += i; ➔ 2
    for(i=0; i<4; i++)
        for(j=0; j<3; j++)
            suma += j;
    return suma;
}
```

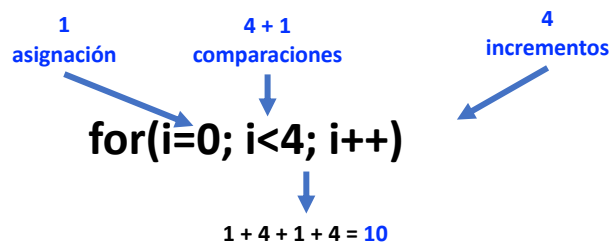


21

## Ejemplo 4:



```
int funcion(int n){
    int i, j, suma;
    suma = 0; ➔ 1
    for(i=0; i<n; i++) ➔ 2n + 2
        for(j=0; j<2*n; j++) ➔ 4n + 2
            suma += j; ➔ 2
    for(i=0; i<100*n; i++) ➔ 200n + 2
        suma += i; ➔ 2
    for(i=0; i<4; i++) ➔ 10
        for(j=0; j<3; j++)
            suma += j;
    return suma;
}
```

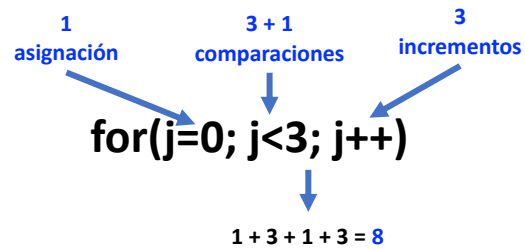


22



## Ejemplo 4:

```
int funcion(int n){
    int i, j, suma;
    suma = 0; ➤ 1
    for(i=0; i<n; i++) ➤ 2n + 2
        for(j=0; j<2*n; j++) ➤ 4n + 2
            suma += j; ➤ 2
    for(i=0; i<100*n; i++) ➤ 200n + 2
        suma += i; ➤ 2
    for(i=0; i<4; i++) ➤ 10
        for(j=0; j<3; j++) ➤ 8
            suma += j; ➤ 2
    return suma; ➤ 1
```



23



## Ejemplo 4:

```
int funcion(int n){
    int i, j, suma;
    suma = 0; ➤ 1
    for(i=0; i<n; i++) ➤ 2n + 2
        for(j=0; j<2*n; j++) ➤ 4n + 2
            suma += j; ➤ 2
    for(i=0; i<100*n; i++) ➤ 200n + 2
        suma += i; ➤ 2
    for(i=0; i<4; i++) ➤ 10
        for(j=0; j<3; j++) ➤ 8
            suma += j; ➤ 2
    return suma; ➤ 1
```

$$\begin{aligned}
 T(n) &= 1 + (2n + 2)(4n + 2)(2) + (200n + 2)(2) + (10)(8)(2) + 1 \\
 &= 1 + (8n^2 + 4n + 8n + 4)(2) + 400n + 4 + (80)(2) + 1 \\
 &= 1 + 16n^2 + 8n + 16n + 8 + 400n + 4 + 160 + 1 \\
 &= 16n^2 + 424n + 174
 \end{aligned}$$

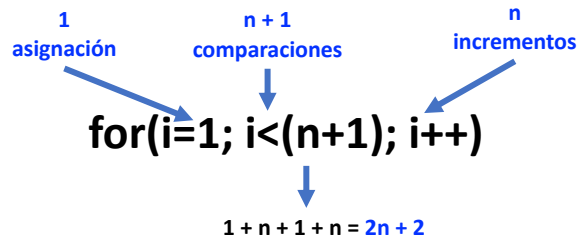
↓  
 **$T(n^2)$**

24

## Ejemplo 5:



```
void funcion(int n){
    int i,j;
    for(i=1; i<(n+1); i++) 2n + 2
        for(j=0; j<i; j++)
            printf("%d",j);
}
```



25

## Ejemplo 5:



```
void funcion(int n){
    int i,j;
    for(i=1; i<(n+1); i++) 2n + 2
        for(j=0; j<i; j++)
            printf("%d",j);
}
```

La cantidad de iteraciones que realice depende de los valores de i.

26

## Ejemplo 5: Primero analizamos las comparaciones



```
void funcion(int n){
    int i,j;
    for(i=1; i<(n+1); i++)
        for(j=0; j<i; j++)
            printf("%d",j);
}
```

n	i	j se compara	
4	1	0 ? → 1 ? X	2 veces
	2	0 ? → 1 ? → 2 ? X	3 veces
	3	0 ? → 1 ? → 2 ? → 3 ? X	4 veces
	4	0 ? → 1 ? → 2 ? → 3 ? → 4 ? X	5 veces
	...	...	

Entonces...

Para n=1 → j se compara 2 veces

Para n=2 → j se compara 5 veces

Para n=3 → j se compara 9 veces

Para n=4 → j se compara 14 veces

27

## Ejemplo 5:



```
void funcion(int n){
    int i,j;
    for(i=1; i<(n+1); i++)
        for(j=0; j<i; j++)
            printf("%d",j);
}
```

n	i	j se compara	
4	1	0 ? → 1 ? X	2 veces
	2	0 ? → 1 ? → 2 ? X	3 veces
	3	0 ? → 1 ? → 2 ? → 3 ? X	4 veces
	4	0 ? → 1 ? → 2 ? → 3 ? → 4 ? X	5 veces
	...	...	

El comportamiento anterior se parece en parte a la **Sumatoria de Gauss**:

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

Qué le podemos agregar para que nos de el valor exacto?

$$\begin{aligned} n = 1 &\rightarrow 1(1+1)/2 = 1 + 1 = 2 \\ n = 2 &\rightarrow 2(2+1)/2 = 3 + 2 = 5 \\ n = 3 &\rightarrow 3(3+1)/2 = 6 + 3 = 9 \\ n = 4 &\rightarrow 4(4+1)/2 = 10 + 4 = 14 \end{aligned}$$



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

28

## Ejemplo 5: Ahora analizamos los incrementos

Entonces...

```
void funcion(int n){
    int i,j;
    for(i=1; i<(n+1); i++)
        for(j=0; j<i; j++)
            printf("%d",j);
}
```

n	i	j se incrementa	
4	1	0 → + 1 X	1 vez
	2	0 → + 1 → + 2 X	2 veces
	3	0 → + 1 → + 2 → + 3 X	3 veces
	4	0 → + 1 → + 2 → + 3 → + 4 X	4 veces
	...	...	

Para n=1 → j se incrementa 1 vez

Para n=2 → j se incrementa 3 veces

Para n=3 → j se incrementa 6 veces

Para n=4 → j se incrementa 10 veces

29

## Ejemplo 5: Ahora analizamos los incrementos

```
void funcion(int n){
    int i,j;
    for(i=1; i<(n+1); i++)
        for(j=0; j<i; j++)
            printf("%d",j);
}
```

n	i	j se incrementa	
4	1	0 → + 1 X	1 vez
	2	0 → + 1 → + 2 X	2 veces
	3	0 → + 1 → + 2 → + 3 X	3 veces
	4	0 → + 1 → + 2 → + 3 → + 4 X	4 veces
	...	...	

Sumatoria de Gauss:  
 $n(n+1)/2$

$$n = 1 \rightarrow 1(1+1)/2 = 1$$

$$n = 2 \rightarrow 2(2+1)/2 = 3$$

$$n = 3 \rightarrow 3(3+1)/2 = 6$$

$$n = 4 \rightarrow 4(4+1)/2 = 10$$

30

## Ejemplo 5:

```
void funcion(int n){
    int i,j;
    for(i=1; i<(n+1); i++)
        for(j=0; j<i; j++)
            printf("%d",j);
}
```

$$\begin{aligned}
 &= 1 + n(n+1)/2 + n + n(n+1)/2 \\
 &= 1 + (n^2+n)/2 + n + (n^2+n)/2 \\
 &= 1 + n^2/2 + n/2 + n + n^2/2 + n/2 \\
 &= 1 + n^2 + 2n \\
 &= n^2 + 2n + 1
 \end{aligned}$$



$T(n^2)$

31

## Ejemplo 6:

```
p = 0;
for(i=1; p<=n; i++)
```

```
    p = p+i;
```

Este ciclo **NO** se va a repetir **n** veces.

La cantidad de iteraciones del ciclo está dada por la operación que modifica el valor de **p** para que le permita llegar a **n**.

Otra forma de calcular la complejidad de un ciclo es identificar en qué momento se va a detener.

32





## Ejemplo 6:

```
p = 0;
for(i=1; p<=n; i++)
    p = p+i;
```

n	i	p
9	1	0 + 1 = 1
	2	0 + 1 + 2 = 3
	3	0 + 1 + 2 + 3 = 6
	4	0 + 1 + 2 + 3 + 4 = 10
	...	...
	k	0 + 1 + 2 + 3 + 4 + ... + k

Sumatoria de Gauss

$$p = [i(i+1)]/2$$

Cuándo se va a detener el ciclo? cuando  $p > n$

$p = [i(i+1)] / 2 > n$  ← Aquí se detiene el ciclo

$[i^2 + i] / 2 > n$

$i^2 > n$

$i > n^{1/2}$

→  $O(n^{1/2})$

## Ejemplo 7:

```
for(i = 1; i < n; i*2)
    print(arreglo[i]);
```

n	i	
4	1	→ $2^0$
	1*2 = 2	→ $2^1$
	2*2 = 4	→ $2^2$
	4*2 = 8	→ $2^3$
	...	
	k	

$$i = 2^k$$

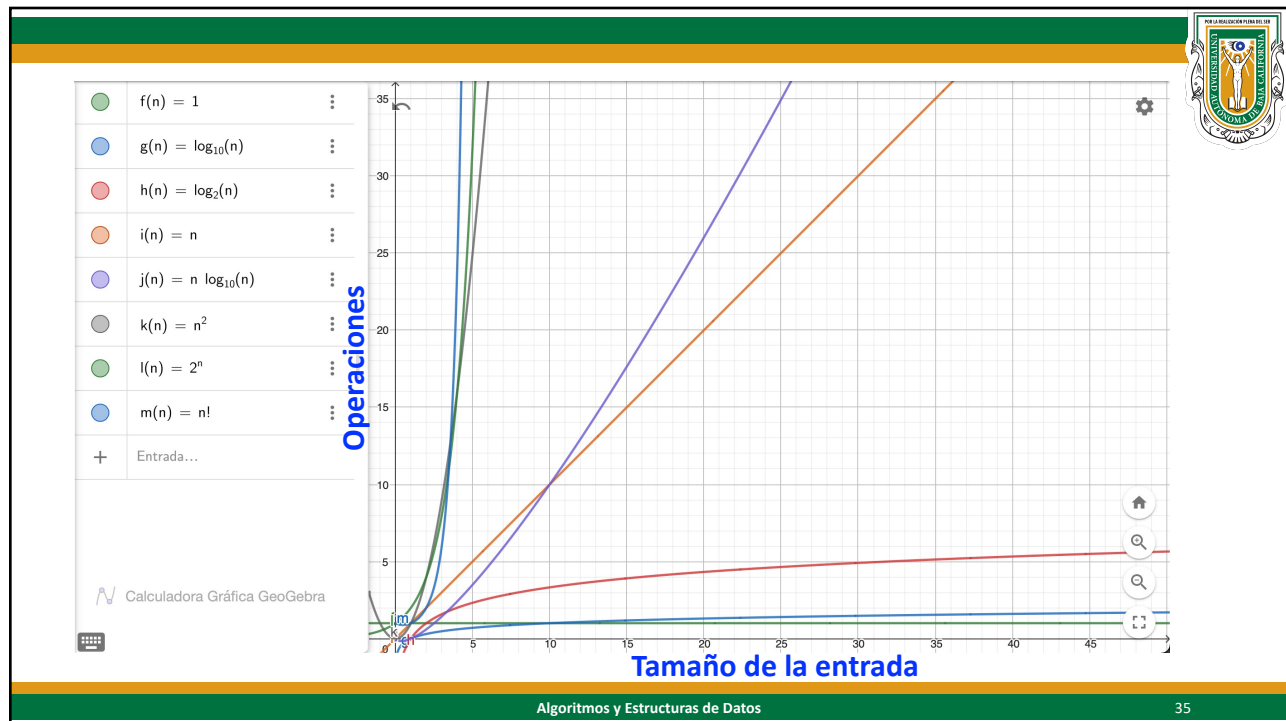
se detiene cuando:  $2^k \geq n$

$$\log_2(2^k) \geq \log_2(n)$$

$$\log_2(2^k) \geq \log_2(n)$$

$$k \geq \log_2(n)$$

↓  
 $O(\log_2 n)$



35

## Ejemplo 8:

```
while(m != n)
  if(m > n)
    m = m - n;
  else
    n = n - m;
```

m = 6	n = 3
3	3

m = 14	n = 3
11	3
8	3
5	3
2	3
2	1
1	1

m = 5	n = 5
14	2
12	2
10	2
8	2
6	2
4	2
2	2

m = 15	n = 7
8	7
1	7
1	6
1	5
1	4
1	3
1	2
1	1

6 iteraciones  
Aprox. 14/2  
m/2 iteraciones

7 iteraciones  
Aprox. 16/2  
m/2 iteraciones

8 iteraciones  
Aprox. 15/2  
m/2 iteraciones

$T(n) = m/2 \longrightarrow T(m)$

Algoritmos y Estructuras de Datos

36

## Algunas reglas...



- Las operaciones aritméticas son constantes
- Las asignaciones de variables son constantes
- Accesar un elemento de un arreglo es constante
- En un ciclo, la complejidad se da por la longitud del ciclo multiplicado por lo que se ejecuta en el ciclo

## Ejercicios:



- Elabora los algoritmos que se piden a continuación y luego analiza su complejidad.
  - Escribe una función que reciba como parámetro 2 arreglos, y que imprima sólo aquellos valores que se encuentren en ambos arreglos.
  - Escribe una función que reciba como parámetro 2 arreglos, y que imprima sólo aquellos elementos que están en el primer arreglo, pero que no están en el segundo arreglo.
  - Escribe una función que reciba como parámetro 2 arreglos, y que imprima todos los elementos que no estén en el primer arreglo.