Algoritmos
Estructuras de Datos I

Facultad de Ciencias Exactas y Tecnología
Universidad Nacional de Tucumán
2024

Complejidad
Notación O grande(2)

Notación O grande
Ejercicios:

Verdadero o Falso

- $4n+1 \in O(n^2)$?
- $n - 1 \in O(1)$?
- $\log_2 n \in O(n)$?
- $2n+ \log_4 n \in O(n^2)$?
- $\sqrt{n} \in O(n)$?
- $3^n \in O(2^n)$?

Tiempo de ejecución de un algoritmo

Sea T(n) el tiempo de ejecución de un algoritmo para una entrada de tamaño n.

Se dice que ***T(n) es de orden f(n)*** o que ***T(n) es O(f(n))*** si existen dos constantes: c(real) y n_0(natural) tales que:

$$T(n) \leq c.\ f(n) \text{ cuando } n \geq n_0$$

Cuando se dice que T(n) es O(f(n)) se está garantizando que la función f(n) es una cota superior para el crecimiento de T(n).

Notación O grande

Funciones de complejidad notables:

$O(1)$	*Complejidad constante*
$O(\log n)$	*Complejidad logarítmica*
$O(n)$	*Complejidad lineal*
$O(n \log n)$	
$O(n^2)$	*Complejidad Cuadrática*
$O(n^3)$	*Complejidad Cubica*
$O(n^a)$	*Complejidad Polinomial*
$O(2^n)$	*Complejidad Exponencial*
$O(c^n)$	*Complejidad Exponencial, $c \geq 2$*
$O(n!)$	*Complejidad Factorial*

Notación O grande

Se verifica que:

$$O(1) \subset O(\log n) \subset O(n) \subset O(n \log n) \subset O(n^2) \subset O(n^3) \subset O(c^n) \subset O(n!)$$

RAM
(RANDOM ACCESS MACHINE)

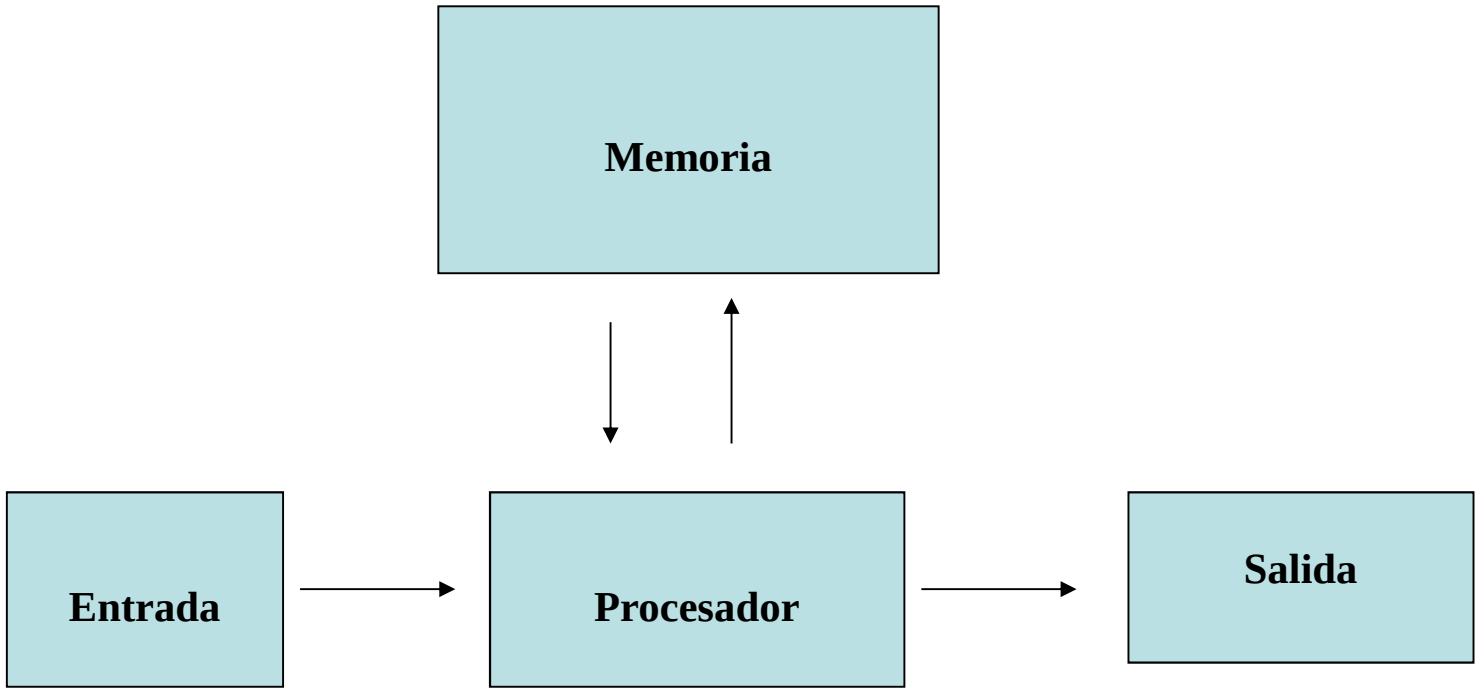
Cómo medir el tiempo de ejecución en función del tamaño de la entrada de los datos del problema?

- RAM: modelo que se usa para evaluar la complejidad de un algoritmo.

- *Un modelo de máquina de acceso directo RAM es una computadora de un acumulador cuyas instrucciones no pueden modificarse a sí mismas.* Consta de:
  - unidad de memoria
  - unidad de entrada
  - procesador
  - unidad de salida
  - conjunto de instrucciones

- Un ***programa*** RAM es una *secuencia finita de estas instrucciones*

RAM



RAM

En este modelo se supone que:

- La ***entrada*** es una secuencia representada por una cinta de cuadrados en cada uno de los cuales está almacenado un número entero.

- Cuando un símbolo se lee de la cinta, la cabeza ***avanza*** al próximo cuadrado.

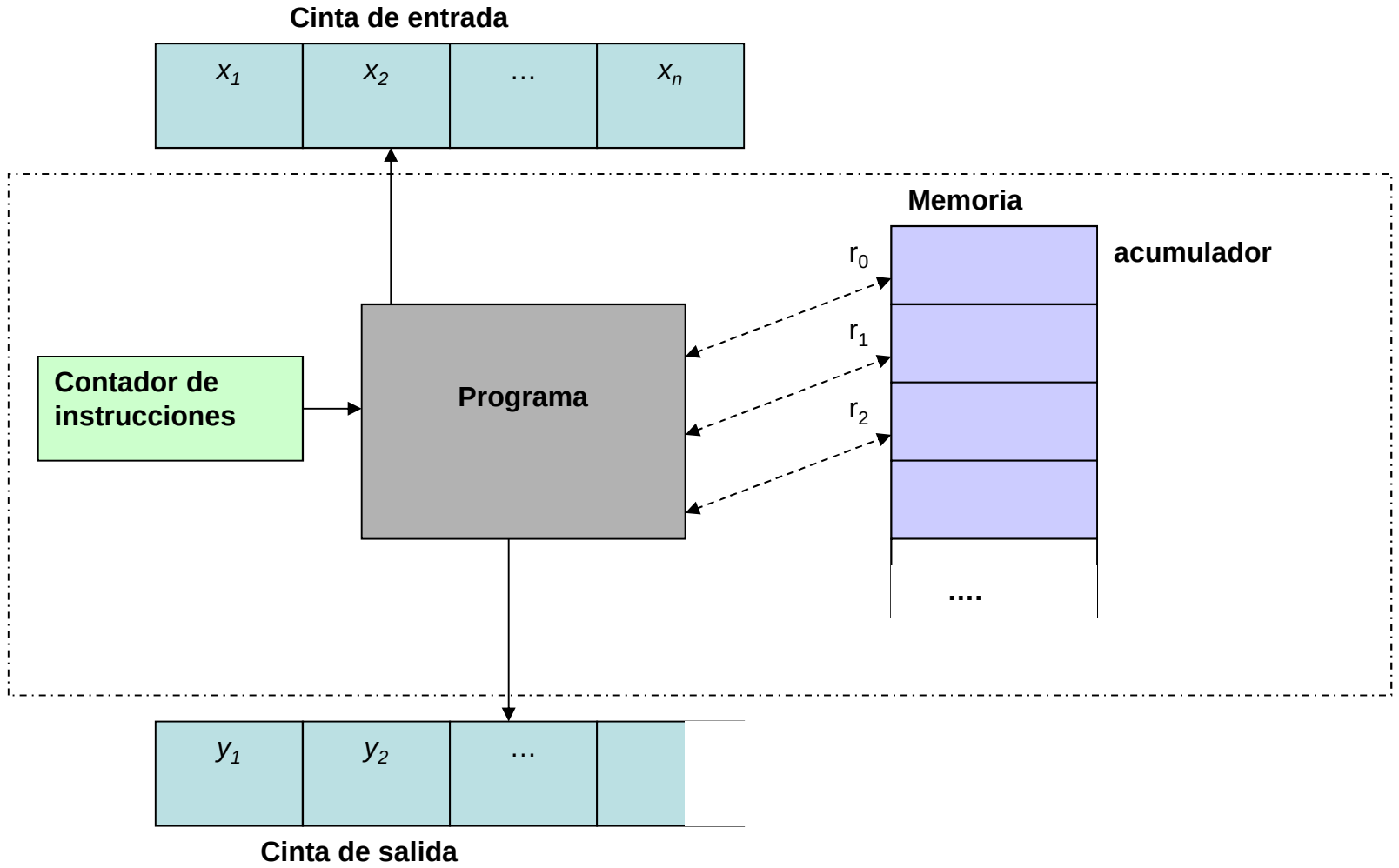
- La ***salida*** es una cinta de cuadrados, que está inicialmente en blanco, en cada uno de los cuales se puede escribir un entero.

- Cuando un símbolo se escribe en la cinta, la cabeza ***avanza*** al próximo cuadrado. Cuando ya se escribió un símbolo, no se puede cambiar.

RAM

- La ***memoria*** es una sucesión (cantidad ilimitada) de registros *r0,r1,..,* cada uno de los cuales puede almacenar un entero de tamaño arbitrario.
- Los cálculos se hacen en el primero de ellos, *r0* , llamado ***acumulador.***
- El ***programa*** es una secuencia de instrucciones.
- Cada instrucción tiene un ***código de operación*** y una ***dirección.***
- El programa no se almacena en la memoria y no se modifica a si mismo.
- El ***conjunto de instrucciones*** disponibles es similar al de las *computadoras reales* (operaciones aritméticas, de entrada/salida, de bifurcación*…)*

RAM


Cinta de entrada
x_1
x_2
…
x_n
Memoria
r_0
r_1
r_2
acumulador
Contador de instrucciones
Programa
....
y_1
y_2
…
Cinta de salida

Reglas Generales

Para determinar el crecimiento del costo del algoritmo en notación O grande, se van a analizar los distintos tipos de instrucciones:

- Instrucciones que no se consideran.
- Operaciones elementales.
- Secuencia.
- Selección.
- Iteración.
- Funciones no recursivas.
- Funciones recursivas.

Reglas Generales

Cero pasos:

Se desprecia todo lo que sea no ejecutable: comentarios, declaraciones, tipificaciones, descripciones, etc.

Operaciones elementales, se consideran O(1):

- leer un valor,
- escribir un valor o expresión simple,
- asignar, excepto de estructuras,
- evaluar un expresión que no tenga invocación a una función,
- evaluar la condición de una selección o iteración.

Reglas Generales

Secuencia

Hay que evaluar el tiempo de cada una de las instrucciones que componen la secuencia.

El tiempo de ejecución de toda la secuencia de instrucciones esta dado por la regla de la suma.

Esto implica el mayor tiempo de ejecución de cada instrucción de la secuencia.

Reglas Generales

Selección

Primero evaluar los tiempos que se necesitan para cada una de las distintas alternativas que podrían ejecutarse.

Luego sumar el tiempo de evaluar la condición de selección, mas el mas grande de los tiempos que se necesitan para cada una de las distintas alternativas.

La condición de selección en general es O(1).

Reglas Generales

Iteración de un numero fijo de veces

Primero evaluar cada uno de los tiempos que necesitan cada una de las instrucciones que abarca el *cuerpo* de la iteración.

Sumar el tiempo de todas las instrucciones que abarca la iteración (el cuerpo), multiplicarlo por el *numero total de iteraciones* que se efectúen.

Reglas Generales

Iteración condicional

Primero evaluar cada uno de los tiempos que necesitan cada una de las instrucciones que abarca el *cuerpo* de la iteración.

Sumar el tiempo de todas las instrucciones que abarca la iteración (el cuerpo), mas el tiempo de evaluar la *condición* de terminación y todo multiplicarlo por el *numero total de iteraciones* que se efectúen.

La condición de terminación en general es O(1).

Reglas Generales

Funciones no recursivas

El tiempo de cada función se debe evaluar por separado, comenzando por las funciones que no invoquen a otras.

Se evalúa el tiempo de ejecución de la función según las estructuras de control que tengan (secuencia, selección, iteración).

Usando esos tiempos se evalúan las funciones que las invocaron y así hasta llegar al programa principal.

Reglas Generales

Funciones recursivas

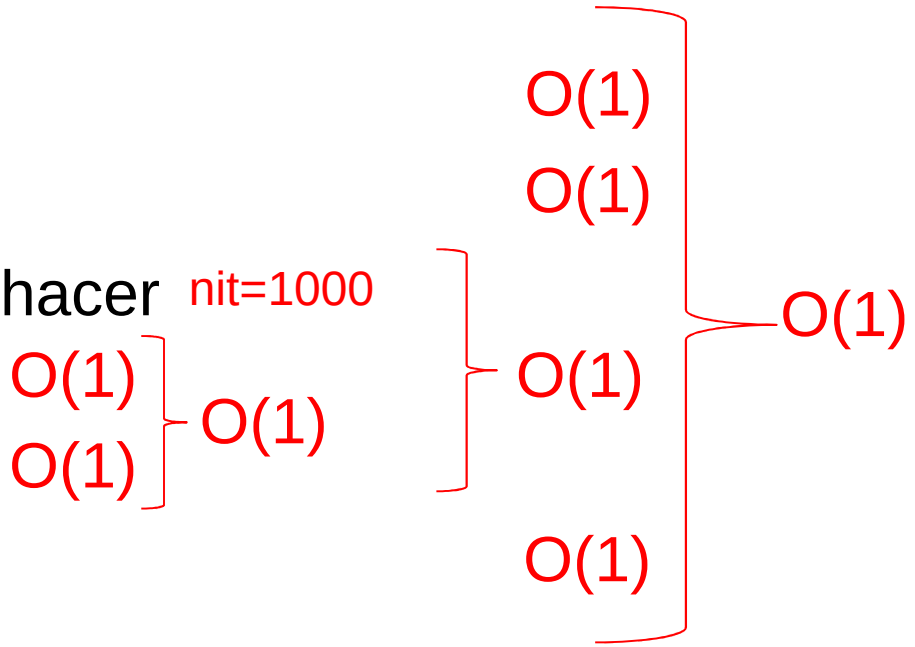
Se asocia una función de tiempo desconocida T(n).

A continuación se plantea la recurrencia de T(n) donde n mide el tamaño de la entrada.

Luego se desarrolla la recurrencia, esto es una ecuación para T(n) en términos de T(k) para distintos valores de k.

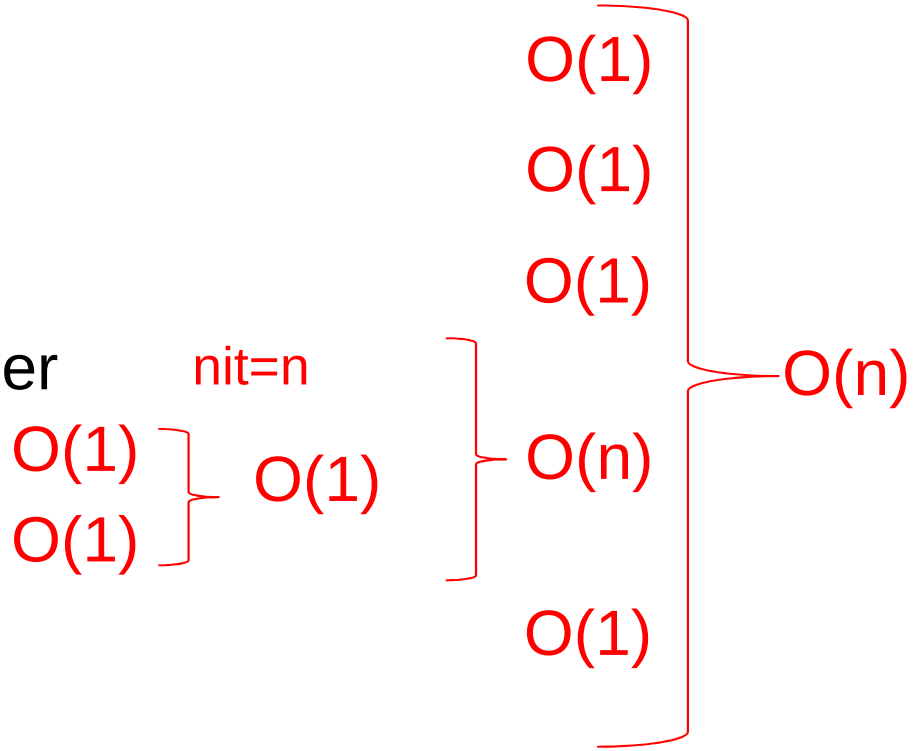
Ejemplo 1

```
...
acc←0
cont←0
Para k desde 1 hasta 1000 hacer
    acc←2*acc+k
    cont←cont+1
Escribir (acc,cont)
...
```



Ejemplo 2

```
Leer(n)
acc←0
cont←0
Para k desde 1 hasta n hacer
        acc←2*acc+k
        cont←cont+1
Escribir (acc,cont)
...
```



Ejemplo 3

…

Leer (n)

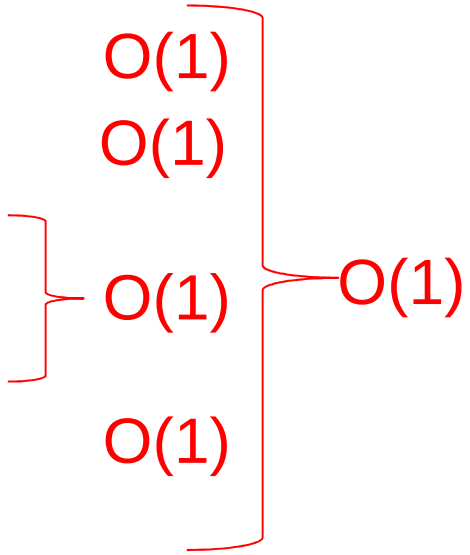
k←1

Para i desde **n** hasta **n+999** hacer

k←k*i

Escribir (k)

…



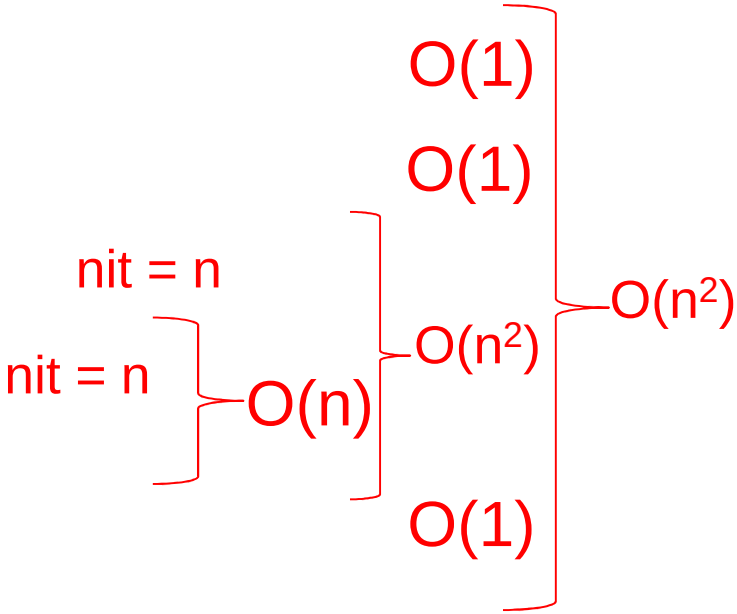
Ejemplo 4

```
…
Leer (n)                                  O(1)
suma←10                                   O(1)
Para i desde 1 hasta n hacer              O(n)   (nit = n)
    suma←suma+i*i+i                       O(1)
Escribir (suma)                           O(1)
…
```

O(n)

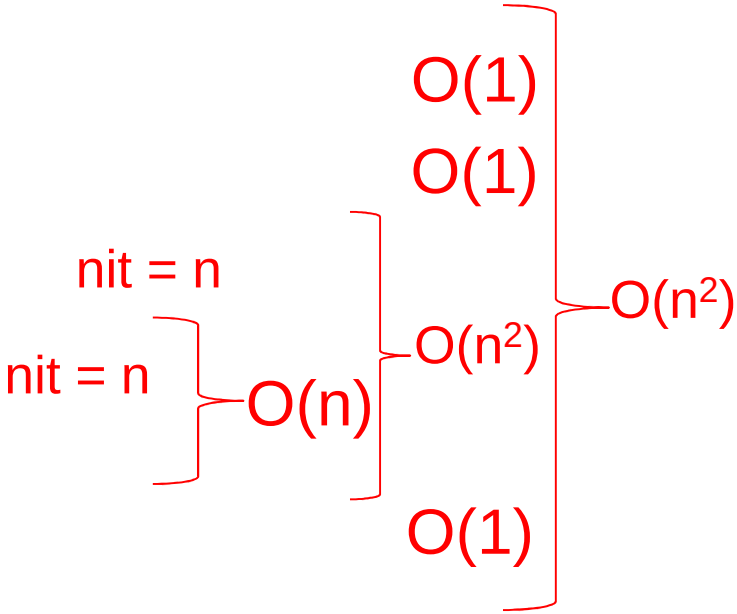
Ejemplo 5

```
…
Leer (n)
k←0
Para i desde 1 hasta n hacer
        Para j desde 1 hasta n hacer
                k←k+1    O(1)
Escribir (k)
…
```



Ejemplo 6

```
…
Leer (n)
k←0
Para i desde 1 hasta n hacer
        Para j desde 1 hasta i hacer
                k←k+1    O(1)
Escribir (k)
…
```



Por qué en la iteración de índice j se considera nit = n ????

Ejemplo 6 continuación

Para i desde 1 hasta n hacer

Para j desde 1 hasta i hacer

Por qué en la iteración de índice j se considera nit = n ?

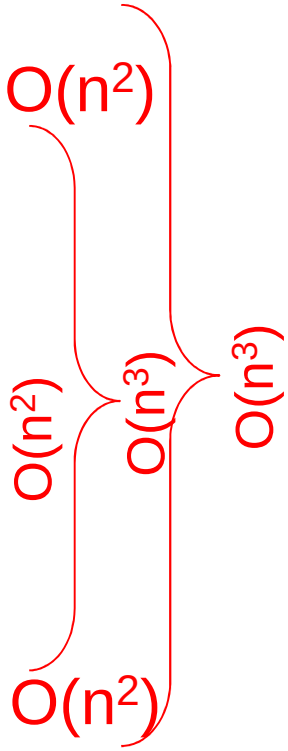
Dos posibilidades:

1) Se supone siempre el peor caso para el valor de i, que es i=n entonces: nit=n en la iteración de índice j.
2) Se puede calcular el valor exacto de iteraciones de las dos iteraciones anidadas:

$1+2+3+\ldots+n = \Sigma^{n}_{i=1}\ i = (1+n).\ n/2\ = n/2 + n^2/2 \in O(n^2)$

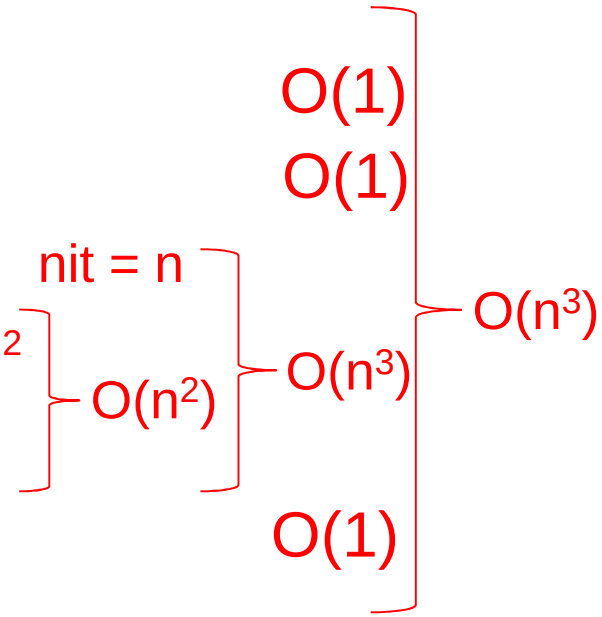
Ejemplo 7

```
...
Leer (n,B,C)
Para i desde 1 hasta n hacer
    Para j desde 1 hasta n  hacer
        A (i,j) ← 0
        Para k desde 1 hasta n  hacer
            A (i,j) ← A (i,j) + B (i,k) * C (k,j)
Escribir (A)
...
```



Ejemplo 8

```
…
Leer (n)
suma←0
Para i desde 1 hasta n hacer
        Para j desde 1 hasta n² hacer
                suma←suma + j
Escribir (suma)
…
```



Ejemplo 9

```
…
Leer (n)                                   O(1)
cuad←1                                     O(1)
Mientras cuad < n hacer    nit = ?
      cuad←cuad*2          O(1)
Escribir (cuad)                            O(1)
…
```

Cómo se calcula el número de iteraciones de la iteración condicional ?

Ejemplo 9 continuación

Cómo se calcula el número de iteraciones de la iteración condicional? …

```
cuad←1
Mientras cuad < n hacer
        cuad←cuad*2
```

Analizar los valores que va tomando cuad:

cuad=1,2,4,8,… $\Rightarrow$ cuad=2^0 , 2^1 , 2^2 , 2^3 , … $\Rightarrow$ cuad=2^x ,

Si itera x veces mientras (cuad<n), entonces la iteracion finaliza cuando (cuad $\geq$ n) ,

pero cuad=2^x, entonces: $2^x \geq n$,

de allí se deduce que: $x \geq \log_2 n$

Ejemplo 9

```
…
Leer (n)                                   O(1)
cuad←1                                     O(1)
Mientras cuad < n hacer   nit = log2n+1   O(log2n)   O(log2n)
    cuad←cuad*2           O(1)
Escribir (cuad)                            O(1)
…
```

Ejemplo 10

```
...
Leer (n)                                                    O(1)
r←0                                                         O(1)
Si n =100 Entonces
      r ← r+n    O(1)
Sino Si n=200  Entonces
      r ← r+2*n    O(1)                                                  O(1) ?
Sino Si n=300 Entonces                                                   O(n) ?
      Para i desde 1 a n  hacer   nit = n ?                 O(1)?
                  r ← r+i   O(1)                            O(n)?
Sino
      r ← -999      O(1)
Escribir (r)                                                O(1)
...
```