```
cuenta ← 0
resto ← dividendo
Repetir
	resto← resto – divisor
	cuenta← cuenta +1
Hasta que (resto <divisor)
Coeficiente ← cuenta
```

```
producto← y
cuenta ← 1
Mientras cuenta < x hacer
	producto ← producto + y
	cuenta ← cuenta + 1
Fin Mientras
```

O(1)
O(1)
O(n)
    O(1)
    O(1)

O(1)+O(1)+ O(n)*2*O(1)

2*O(1)+ O(n)*c

C_1 + O(n)

O(n)

1. Decidir el parámetro n, que indica el tamaño de la entrada

2. Identificar la operación básica del algoritmo

3. Determinar si el número de veces que la operación básica es ejecutada puede variar para diferentes entradas del mismo tamaño(analizar el peor caso, caso medio y el mejor caso)

4. Expresar como una relación de recurrencia, con una condición inicial, el número total de ejecuciones de la operación básica.

5. Resolver la relación de recurrencia, para encontrar la fórmula Explícita para el término genérico que satisfaga la recurrencia Y la condición inicial o probar que no existe

```
Algoritmo factorial(n):
    Si n = 0 Entonces
        devolver 1
    Sino
        devolver factorial(n-1)*n
    Fin Si
Fin
```

1. n es el número del cual se calcula el factorial
2.La operación básica es la multiplicación: M(n)
3.No hay variaciones
4. $M(0) = 0$ para $n = 0$

$$M(n) = \underbrace{M(n-1)}_{\text{factorial}(n-1)} + \underbrace{1}_{\text{factorial}(n-1)*1} \quad \text{para } n>0$$

5. $M(n) = M(n-1) + 1$ sustituir $M(n-1) = M(n-2) + 1$

$= [M(n-2) + 1] + 1$

$= M(n-2) + 2$ sustituir $M(n-2) = M(n-3) + 1$

$= [M(n-3) + 1] + 2$

$= M(n-3) + 3$

$$M(n) = M(n-i) + i$$

$M(n) = M(n-i) + i$

Condición inicial $M(0) = 0$
$i=n$

$$
\begin{aligned}
M(n) &= M(n-i) + i \\
&= M(n-n) + n \\
&= M(0) + n \\
&= n
\end{aligned}
$$

lineal

```
hanoi(n,inicial, destino, auxiliar):
	hanoi(n-1, inicial, auxiliar, destino)
	mover(inicial, destino)
	hanoi(n-1, auxiliar, destinio , origen)
```

1. n es el número de discos
2. Operación básica mover un disco: $M(n)$
3. no existen variaciones
4. $M(1) = 1$
   $M(n) = M(n-1) + 1 + M(n-1)$
   $\quad = 2M(n-1) + 1$
5. $M(n)=2M(n-1)+1$ sust. $M(n-1) = 2M([n-1]-1) + 1$
   $= 2[2M(n-2)+ 1] + 1$
   $=2^2M(n-2)+ 2 + 1$ sust. $M(n-2) = 2M([n-2]-1) + 1$
   $=2^2[2M(n-3)+1]+2+1$
   $=2^3M(n-3)+2^2+2^1 +1$

$M(n)= 2^iM(n-i)+2^{i-1}+2^{i-2}+...+2^1 + 1$
$\quad = 2^iM(n-i)+2^i-1$

Sustituyendo $i = n-1$ y $M(1) =1$

$M(n) =2^iM(n-i)+2^i-1$
$=2^{n-1}M(n-(n-1))+2^{n-1}-1$
$=2^{n-1}M(n-n+1)+2^{n-1}-1$
$=2^{n-1}M(1)+2^{n-1}-1$
$=2^{n-1}+2^{n-1}-1$
$=2^n-1$

Es de orden exponencial

```
Algoritmo S(n):
    Si n = 1 Entonces
        devolver 1
    Sino
        devolver S(n-1)*n*n*n
    FinSi
Fin
```

n=1 S(1) = 1
N=2 S(2) = 8
N=3 S(3) =216
N=4 S(4)=13824

1. n el número para calcular su factorial elevado al cubo
2. Operación baśica: multiplicación M(n)
3. No hay variación
4. Condición inicial M(1) = 0
   M(n) = M(n-1)+ 3

5. M(n) = M(n-1)+ 3 sustituir M(n-1) = M(n-2)+ 3
   =[M(n-2)+3]+ 3
   =M(n-2) + 6 sustituir M(n-2) = M(n-3)+ 3
   =[M(n-3)+3]+ 6
   =M(n-3) + 9

M(n)=M(n-i)+ 3i

Sustituir M(1) =0 e i=n-1

M(n) = M(n-n+1)+ 3n-3
= M(1) + 3n -3
= 3n -3
= n

El algoritmo es lineal

a) $x(n) = x(n-1) + 5$ para $n > 1$, $x(1) = 0$

b) $x(n) = 3x(n-1)$ para $n > 1$, $x(1) = 4$

c) $x(n) = x(n-1) + n$ para $n > 0$, $x(0) = 0$

d) $x(n) = x(n/2) + n$ para $n > 1$, $x(1) = 1$ (resolver para $n = 2^{k)}$

```
Algoritmo misterioso(n):
    s ← 0
    Para i ← 1 hasta n hacer
        s ← s + i*i
    Fin Para
    devolver s
Fin
```

¿Qué calcula el algoritmo?
¿Cuál es la operación básica?
¿Cuántas veces se ejecuta la operación básica?
¿Cuál es la eficiencia del algoritmo?
Sugerir una mejora al algoritmo.