HOJA 1. Análisis de la eficiencia

1. Calcular la eficiencia de los siguientes subalgoritmos:

```
funcion suma_datos(A:tmatriz; n:entero) devuelve entero
variables
    i,j,suma:entero
principio
    suma←0
    para i←1 hasta n hacer
        para j←1 hasta n hacer
            suma←suma+A[i,j]
        fpara
    fpara
    devuelve(suma)
fin

funcion producto(A,B:tmatriz; n:entero) devuelve tmatriz
variables
    i,j,k:entero
    C:tmatriz
principio
    para i←1 hasta n hacer
        para j←1 hasta n hacer
            C[i,j] ←0
            para k←1 hasta n hacer
                C[i,j] ←C[i,j]+A[i,k]*B[k,j]
            fpara
        fpara
    fpara
    devuelve(C)
fin
funcion traspuesta(A:tmatriz; n:entero) devuelve tmatriz
variables
   i,j:entero
   B:tmatriz
principio
    para i←1 hasta n hacer
        para j←1 hasta n hacer
            B[i,j]←A[j,i]
        fpara
    fpara
    devuelve(B)
fin
```

```
accion traspuesta(e/s A:tmatriz; ent n:entero)
variables
   i,j,aux:entero
principio
    para i←1 hasta n hacer
        para j←i hasta n hacer
            aux←A[i,j]
            A[i,j]←A[j,i]
            A[j,i] ← aux
        fpara
    fpara
fin
```

2. El siguiente algoritmo calcula la moda de un vector de enteros. Estima razonadamente el orden de su función complejidad en tiempo (para el caso peor):

```
función calculaModa (v:tvector; n:entero) devuelve entero
variables
    w:tvector
    i,j,cont,max,x,moda: entero
principio
     para i←1 hasta n hacer
         cont ← 1
         x←v[i]
         para j←i+1 hasta n hacer
             si (v[j]=x) entonces
               cont←cont+1
             fsi
         fpara
         w[i] ←cont
     fpara
     moda←v[1]
     max←w[1]
     para j←2 hasta n hacer
         si (w[j]>max) entonces
           max←w[j]
           moda←v[j]
         fsi
     fpara
     devuelve (moda)
fin
```

3. Calcular el coste computacional (orden) del siguiente algoritmo:

```
acción ejemplo (ent v:tvector; ent n:entero)
variables
  i,j: entero
principio
  para i←1 hasta n hacer
    j←n
    mientras que (j>0) hacer
     v[j] ←v[j]+1
     j←j div 2
   fmq
  fpara
  para i←1 hasta n hacer
     escribir(v[i])
  fpara
fin
```

4. Calcular el coste computacional (el **orden**) de la siguiente acción:

```
acción calcula_mi_coste (ent n:entero)
variables
     i,j,k,m,dato:entero
principio
     leer(dato)
     i←2
     mientras que (i<=n) hacer
          para j←1 hasta n hacer
               dato←((dato+1)*3)
               para k←3 hasta n hacer
                    escribir(k)
               fpara
          fpara
          i←i+1
     fmq
     para m←1 hasta n hacer
          escribir("¿Cuál es mi coste computacional?")
     fpara
fin
```

5. Calcular el coste computacional (el **orden**) de la siguiente acción:

```
acción accion1 (ent x : entero, sal y : entero)
variables
  i:entero
principio
  i ← 1
  mientras que i <= 30000 hacer
            ejemplo(x,y)
            i ← 2 * i
  fmq
fin

acción ejemplo (ent a : entero, sal b : entero)
variables
  i:entero
principio
  i ← 1
  b ← 0
  mientras que i <= a hacer
       b ← b+i
       i ← i*10
  fmq
fin
```

6. Sean las funciones

$$f_1(n) = n^3$$

$$f_2(n) = 30000n^2 + 8000n$$

$$f_3(n) = \begin{cases} n & \text{si } n \text{ par} \\ n^3 & \text{si } n \text{ impar} \end{cases}$$

$$f_4(n) = \begin{cases} n & \text{si } n \le 100 \\ n^3 & \text{si } n > 100 \end{cases}$$

$$f_5(n) = n^3 \log n$$

Estudiar si $f_i \in O(f_j)$ y si $f_i \in \Theta(f_j)$, para todo i, j.

7. Sean las funciones

$$f(n) = \begin{cases} n^4 & \text{si } n \text{ } es \text{ } par \\ n^2 & \text{si } n \text{ } es \text{ } impar \end{cases} \qquad g(n) = \begin{cases} n^2 & \text{si } n \text{ } es \text{ } par \\ n^3 & \text{si } n \text{ } es \text{ } impar \end{cases}$$

Calcula una función h tal que $O(f) + O(g) = O(h)$.