Ejercicios. Eficiencia de algoritmos 1

1. De las siguientes afirmaciones, indicar cuáles son ciertas y cuáles no:

a) $n^2 \in O(n^3)$
b) $n^3 \in O(n^2)$
c) $2^{n+1} \in O(2^n)$
d) $(n+1)! \in O(n!)$
e) $f(n) \in O(n) \Rightarrow 2^{f(n)} \in O(2^n)$
f) $3^n \in O(2^n)$
g) $\log n \in O(n^{1/2})$
h) $n^{1/2} \in O(\log n)$
i) $n^2 \in \Omega(n^3)$
j) $n^3 \in \Omega(n^2)$
k) $2^{n+1} \in \Omega(2^n)$
l) $(n+1)! \in \Omega(n!)$
m) $f(n) \in \Omega(n) \Rightarrow 2^{f(n)} \in \Omega(2^n)$
n) $3^n \in \Omega(2^n)$
o) $\log n \in \Omega(n^{1/2})$
p) $n^{1/2} \in \Omega(\log n)$

2. Sea a una constante real, $0 \leq a \leq 1$. Usar las relaciones $\subset$ y = para ordenar los órdenes de complejidad de las siguientes funciones:

- $n \log n$
- $n^2 \log n$
- n^8
- n^{1+a}
- $(1+a)^n$
- $(n^2 + 8n + \log^3 n)^4$
- $n^2 / \log n$
- 2^n

3. Supongamos que tenemos el siguiente algoritmo:

```
algoritmo uno (ENT-SAL a:vector)
variables i,j,temp: entero
principio
        para i=1 a n-1 hacer
                para j=n a i+1 hacer
                        si a[j-1]>a[j] entonces
                                temp ← a[j-1]
                                a[j-1] ← a[j]
                                a[j] ← temp
                        fin si
                fin para
        fin para
fin
```

a) Calcula los tiempos de ejecución en el caso mejor, peor y caso medio.
b) Da las cotas asintóticas O, Ω y Θ de las funciones anteriores.

4. Demuestras que $f(n) \in O(g(n)) \Leftrightarrow g(n) \in \Omega(f(n))$.

5. Ordena las siguientes funciones de acuerdo a su velocidad de crecimiento:

- n
- $\sqrt{n}$
- $\log n$
- $\log \log n$
- $\log^2 n$
- $n/\log n$
- $\sqrt{n}\log^2 n$
- $(1/3)^n$
- $(3/2)^n$
- 17
- n^2

6. Supongamos que tenemos el siguiente algoritmo:

```
algoritmo simetria (ENT A:matriz[1..n][1..n] SAL booleano)
variables f,c: entero; es_traspuesta: booleano
principio
        es traspuesta=verdadero
        c=1
        mientras c<= n y es_traspuesta hacer
                f=n
                mientras f>c y es_traspuesta hacer
                        si A[f,c] ≠ A[c,f] entonces
                                es_traspuesta=falso
                        fin si
                        f=f-1
                fin mientras
                c=c+1
        fin mientras
        devolver es_traspuesta
fin
```

Calcular los tiempos de ejecución en el caso mejor y peor, y dar sus cotas asintóticas.

7. Supongamos que tenemos el siguiente algoritmo:

```
algoritmo ecto (ENT a:vector[n], n:entero SAL entero)
variables i,j,x: entero; permuta: booleano
principio
        permuta=verdadero
        i=1
        mientras permuta hacer
                i=i+1
                permuta=falso
                para j=n hasta i hacer
                        si a[j] < a[j-1] entonces
                                x=a[j]
                                permuta=cierto
                                a[j]=a[j-1]
                                a[j-1]=x
                        fin si
                fin para
        fin mientras
        devolver i
fin
```

Calcular los tiempos de ejecución en el caso mejor y peor, y dar sus cotas asintóticas.