Grado en Ingeniería Informática
Análisis y Diseño de Algoritmos

Problemas del tema 1 – Hoja 1

1. Un programador ha implementado un algoritmo que tarda, en su antiguo ordenador de 100 Mhz, una milésima de segundo en resolver un problema de tamaño $n = 20$. ¿Cuánto tardaría en resolver un problema de tamaño $n = 30$ en los siguientes casos?
   a) El algoritmo es de orden $O(n^3)$
   b) El algoritmo es de orden $O(2^n)$
   c) El algoritmo es de orden $O(n!)$

   ¿Por que factor debería multiplicar la velocidad de su ordenador para que volviera a tardar una milésima de segundo? **Nota**: 1 año = 31.557.600 segundos.

2. ¿Es posible que un algoritmo de orden $O(n^2)$ se ejecute más rápidamente que un algoritmo de orden $O(n)$ en los siguientes casos?
   a) Para un tamaño n = 100.
   b) Para un determinado valor de la entrada.
   c) Para el mejor caso del primer algoritmo.
   d) Para todos los valores de la entrada.
   e) Si se ejecutan en máquinas distintas y podemos elegir aquella en que se ejecuta el primer algoritmo.

3. Al medir experimentalmente un algoritmo se encuentra que nunca emplea más de $k_1 \cdot n^3$ operaciones ni menos de $k_2 \cdot n^2$, siendo n el tamaño de los datos de entrada y k_1 y k_2 constantes. ¿Cuál sería la hipótesis más adecuada al interpretar estos datos?
   - El algoritmo es $O(n^2)$ y $\Omega(n^3)$
   - El algoritmo es $O(n^3)$ y $\Omega(n^2)$
   - El algoritmo es $\Theta(n^{2.5})$
   - El algoritmo es $\Theta(n^2 + n^3)$

4. Analizar cual sería el orden de complejidad del algoritmo que usamos habitualmente para multiplicar dos números enteros. Se supondrá que los dos números tienen el mismo número de dígitos, n, y que este valor define el tamaño de la entrada. Contar únicamente las operaciones de producto y suma de dígitos individuales.

5. Analizar el número de operaciones en las que interviene un elemento del vector en el siguiente algoritmo (ordenación por selección):

```
type TVector = array[1..N] of real;


procedure ordenar(var V: TVector);
var
  I,J,Min : integer;
  Temp : real;
begin
  for I := 1 to N-1 do
  begin
    Min := I;
    for J := I+1 to N do
      if V[J] < V[Min] then Min := J;
    Temp := V[I];
    V[I] := V[Min];
    V[Min] := Temp
  end
end
```

Encontrar las fórmulas exactas para el peor y el mejor caso, identificar cuales son los tipos de entradas que producen esos casos y repetir el análisis utilizando notación asintótica.

6. Calcular el número de operaciones de incremento de *x* que realizan los siguientes fragmentos de programas, teniendo en cuenta que sólo deseamos conocer exactamente el término de mayor crecimiento, y el resto de términos se pueden expresar usando notación asintótica:

```
for i := 1 to n do
  for j := 1 to n do
    if j = i then
      x := x+1
```

```
for i := 1 to n do
  for j := n downto i do
    for k := i to j do
      x := x+1
```

```
for i := 1 to n do
  for j := 1 to i*i do
    for k := 1 to j*j do
      x := x+1
```

7. Calcular el orden de complejidad de las siguientes funciones recursivas (contar sólo las operaciones producto):

```
function f1(n: integer): integer;
begin
  if n < 1 then f1 := 1 else
    f1 := f1(n div 2)+f1(n div 2)*f1(n div 2)
end;
```

```
function f2(n: integer): integer;
var i,x: integer;
begin
  if n < 1 then f2 := 1 else
  begin
    x := 1;
    for i := 1 to n*n do x := x*2;
    f2 := x*f2(n div 2)
  end
end
```

8. Demostrar si los siguientes enunciados son ciertos o falsos:

   a) ¿$\log 3^n \in \Theta(n)$?

   b) ¿$2^{n+1} \in O(2^n)$?

   c) ¿$3^n \in \Omega(2^n)$?

   d) ¿$2n^2 - n + 1 \in \Omega(n)$?

9. Calcular el orden de complejidad respecto al tiempo (suponiendo que deseamos contar las operaciones producto) y al espacio de la siguiente función:

```
function f(n: integer) : integer;
var i,x: integer;
begin
  if n < 1 then f := 1 else
  begin
    x := n;
    for i := 1 to n do x := x+1;
    f := f(n div 2)*f(n div 2)+f(n div 3)+x
  end
end;
```

10. Al analizar la eficiencia de una operación de inserción sobre una estructura de datos se encuentra que a veces la inserción tarda un tiempo $\Theta(n^3)$ pero eso implica que las siguientes n inserciones tardarán un tiempo de $O(1)$. Analizar el comportamiento del algoritmo en el peor y mejor caso, el caso promedio y el tiempo amortizado.