Análisis de Algoritmos

1. Indicar el orden de complejidad de este algoritmo en el peor caso.

   Algoritmo CheckOrder Dado un array A, comprueba si está ordenado de forma ascendente.

   CheckOrder(A, n)

   INPUT: **A**: array de enteros $A[1, .., n]$
   OUTPUT: **True**, si $A[1] \leq A[2] \leq \cdots \leq A[n]$

   1. FOR $i := 1$ to n
      2. IF $A[i] > A[i+1]$ Devolver $False$
   3. Devolver $True$

2. Indicar el orden de complejidad de este algoritmo que realiza la multiplicación de dos matrices por el método de "fuerza bruta".

   ProductMatrix(A, B, n)

   INPUT: **A**, **B**: matrices a multiplicar de dimensión $n \times n$
   OUTPUT: **C**: matriz resultante

   1. FOR $i := 1$ to n
   2. FOR $j := 1$ to n
   3. $suma \leftarrow 0$
   4. FOR $k := 1$ to n
   5. $suma \leftarrow suma + A[i,k]B[k,j]$
   6. $C[i,j] \leftarrow suma$
   7. Devolver C

3. Indicar el orden de complejidad de este algoritmo.

   Algoritmo PrefixMedias Dado un array X, la i-ésima media de prefijo A_i es la media de los $(i+1)$ primeros elementos del vector. Por ejemplo:

$$A_i = \frac{X[0] + X[1] + \cdots + X[i]}{i+1}$$

PrefixMedias1(X, n)

INPUT: $\mathbf{X}$: array de enteros
OUTPUT: $\mathbf{A}$: array de medias prefijas: A_i

a) $A \leftarrow$ Incializar (array de n double)
b) FOR $i := 0$ to $n - 1$
   1) $s \leftarrow X[0]$
   2) FOR $j := 1$ to i
      $s \leftarrow s + X[j]$
   3) $A[i] \leftarrow s/(i + 1)$
c) Devolver A

4. Indicar el orden de complejidad para esta nueva versión del algoritmo de cálculo de medias, en la que se elimina el bucle interno.

PrefixMedias2(X, n)

INPUT: $\mathbf{X}$: array de enteros
OUTPUT: $\mathbf{A}$: array de medias prefijas: A_i

a) $A \leftarrow$ Inicializar (array de n double)
b) $s \leftarrow 0$
c) FOR $i := 0$ to $n - 1$
   1) $s \leftarrow s + X[i]$
   2) $A[i] \leftarrow s/(i + 1)$
d) Devolver A

5. Indicar el orden de complejidad de este algoritmo en el peor caso:

Calcular(X, n)

INPUT: $\mathbf{X}$: array de enteros
OUTPUT: $\mathbf{A}$: array de valores calculados: A_i

a) $A \leftarrow$ Inicializar (array de n double)
b) FOR $i := 0$ to $n - 1$
   1) $s \leftarrow X[0]$
   2) FOR $j := 1$ to 8
      $s \leftarrow s + X[j]$
   3) $A[i] \leftarrow s/(i + 1)$
c) Devolver A

6. Calcular el número de instrucciones que se realizan en este código en el caso peor:

```
    int i,j;
    for( i=0; i<n; i++)
       for( j=i+1; j<n; j++)
          if( a[i] + a[j] == 0) contador++;
```

7. Calcular orden de complejidad para este algoritmo:

ParImpar(n: **int**)

INPUT: **n**: número entero

1. $x \leftarrow 0$
2. $y \leftarrow 0$
3. FOR $i := 1$ to n
4.   IF esPar(i)
5.   FOR $j := i$ to n
6. $x \leftarrow x + 1$
7.   ELSE
8.   FOR $j := 1$ to $i - 1$
9. $y \leftarrow y + 1$

End

8. Calcular $\mathcal{O}$ y Ω para este algoritmo.

Algoritmo LongMaxSecuencia Dado un array X, calcula la máxima longitud de una subsecuencia ordenada (ascendente) dentro del array.

LongMaxSecuencia(X, n**)**

INPUT: **X**: array de longitud n

1. $max \leftarrow 0$
2. FOR $i := 1$ to n
3. $cont \leftarrow 1$
4. $j \leftarrow i + 1$
5.   WHILE $X[i] \leq X[j]$ AND $j \leq n$
6. $j := j + 1$
7. $cont := cont + 1$
8.   IF $cont > max$
9. $max \leftarrow cont$

End

Sugerencia: obtenerlos calculando el **número de veces que se ejecuta el bucle WHILE** en el caso peor (para la notación $\mathcal{O}$) y en el caso mejor (para la notación Ω).

9. (Examen Final Junio 2014). **Obtener y demostrar** el orden de complejidad de este algoritmo en el peor caso.

BuscaElementos(M, array, n)

INPUT: **M**: matriz de enteros de dimensión $n \times n$,
array: array de enteros de dimensión n
OUTPUT:
busca cada elemento de la matriz en el array y devuelve el número de coincidencias.

a) $suma \leftarrow 0$;
b) **FOR** $i \leftarrow 1$ **to** n **do**
c) **FOR** $j \leftarrow 1$ **to** n **do**
d) $valor \leftarrow$ BusquedaBinaria(array, n , M[i][j]) ;
e) IF $valor \geq 0$ THEN $suma \leftarrow suma + 1$ //está
f) Devolver $suma$;

10. (Examen Final Junio 2013). **Obtener y demostrar**, usando la notación asintótica, el orden $\mathcal{O}$ de este algoritmo.

funcion Productos($A[1..n]$)
//A es una array de n elementos

a) $prod \leftarrow 1$;
b) **FOR** $i \leftarrow 1$ **to** n **do**
c) **FOR** $j \leftarrow i+1$ **to** n **do**
d) **FOR** $k \leftarrow 1$ **to** 5 **do**
$prod \leftarrow A[i] * A[j] * prod$;
e) Devolver $prod$;