# Prácticas de tablero – Sesión 1

## EJERCICIO1

Para cada uno de los métodos siguientes, se le pide analizar su complejidad temporal. Posteriormente calcule (para cada uno de ellos) si la llamada ***metodox(1 000)*** tardase un tiempo de ejecución de 1 seg., cuánto tardaría la llamada ***metodox(10 000)***.

```
public static void metodo1 (int n)
{
      for (int i=1; i<=n/2; i+=2)
        for (int j=n; j>=1; j-=2)
           operacion_O(n);
}

public static void metodo2 (int n)
{
      for (int i=1; i<=2*n; i+=3)
        for (int j=n; j<=n*n; j+=2)
           operacion_O(1);
}

public static void metodo3 (int n)
{
      for (int i=1; i<=n*n; i+=2)
        for (int j=n; j>=-n; j--)
          operacion_O(1);
}

public static void metodo4 (int n)
{
    for (int i=1; i<=n;i++)
         for (int j=1; j<=i; j++)
            for (int k=1; k<=j; k++)
               for (int l=1; l<=k; l++)
                   operacion_O(1);
}

public static void metodo5 (int n)
{
      for (int i=log2n; i<n*n; i+=2) // log 2 n es pseudocódigo
        for (int j=n; j>=n/2; j--)
           operacion_O(1);
}

public static void metodo6 (int n)
{
      metodo1(n);
      metodo2(n);
      metodo3(n);
      metodo4(n);
      metodo5(n);
}
```

## EJERCICIO 2

Sea el bucle:

```
for (int i=1; i<=n/2;i++)
    for (int j=2; j<=n*n; j*=2)
       for (int k=2*n; k>=n/2; k-=3)
          operacion_O(1);
```

a) Determinar su complejidad temporal

b) Si para n=100 000 tarda 1 minuto, razonar si tardará más o menos de 500 minutos para n=1 000 000.

## EJERCICIO 3

a) Un algoritmo $O(n^4)$ tarda 1 segundo para n=10, calcular su tiempo de ejecución para n=20.

b) Un algoritmo $O(2^n)$ tarda 1 segundo para n=10, calcular su tiempo de ejecución para n=20.

c) Un algoritmo O(n!) tarda 1 segundo para n=10, calcular su tiempo de ejecución para n=20.

## EJERCICIO 4

Tras razonar la complejidad temporal de los dos algoritmos siguientes, concluya cuál tiene menor complejidad.

```
public static void metodo1 (int n)
{  for (int i=n*n; i>n/3; i--)
     {  j=1;
        while (j<n*n*n)
          { operacion_O(1);
             j=j*4;
          }
     }
 }

public static void metodo2 (int n)
     { i=8*n*n;
        while (i>1)
          {   for (int j=2; j<=2*n; j*=3)
                   operacion_O(1);
         i=i-5;
          }
}
```

## EJERCICIO 5

Una operación que suma todos los elementos de una matriz cuadrada de orden n, la ejecutamos en un ordenador para n=100 y tarda 0,4 segundos. Razonar lo que tardará para n=10.000. ¿Tarda más o menos de una hora?

## EJERCICIO 6

Calcule de complejidad temporal de los dos métodos siguientes:

```
public static void metodo1 (int n)
{
       for (int i=n*n; i>=1; i--)
          for (int j=1; j<=n*n; j++)
              if (i==j) operacion_O(n⁴logn);
            else operacion O(n²)
}

public static void metodo2 (int n)
{
       for (int i=n*n; i>=1; i--)
          for (int j=1; j<=n*n; j++)
               if (i!=j)   operacion_O(1);
           else operacion O(n)
}
```

## EJERCICIO 7

Tenemos un algoritmo de complejidad temporal cuadrática $O(n^2)$ y comprobamos en nuestro ordenador que para n=1.000 tarda 1 seg.

a) Calcular qué tamaño del problema podemos resolver si disponemos de una hora en el mismo ordenador.
b) Calcular qué tamaño del problema podemos calcular si disponemos de una hora en un superordenador 1.000.000 veces más potente.

## EJERCICIO 8

Tenemos un algoritmo de complejidad temporal exponencial $O(2^n)$ y comprobamos en nuestro ordenador que para n=100 tarda 1 seg.

a) Calcular qué tamaño del problema podemos resolver si disponemos de una hora en el mismo ordenador ¿y en un día?
b) Calcular qué tamaño del problema podemos calcular disponemos de una hora en un superordenador 1.000.000 veces más potente.