Prácticas de tablero – Sesión 2

EJERCICIO 1

Para cada uno de los métodos posteriores se le pide analizar su complejidad temporal, tras lo que se le pide que si cada uno de ellos en la llamada ***metodox(100)*** tarda un tiempo de ejecución de 1 segundo, calcular cuánto tardará (cada uno de ellos) para la llamada ***metodox(200)***.

```
public static void metodo1 (int n)
{   if (n>0)
      { operacion_O(1);
        metodo1 (n-2);
        for (int i=1; i<=n; i++)
           for (int j=1; j<=n; j++)  operacion_O(n);
      }
}

public static void metodo2 (int n)
{   if (n>0)
          { operacion_O(1);
            metodo2 (n/2);
            for (int i=1; i<=n; i++)
              for (int j=1; j<=n; j++)  operacion_O(n);
           }
}

public static void metodo3 (int n)
{   if (n>0)
         { operacion_O(1);
           metodo3 (n-1);
           metodo3 (n-1);
           metodo3 (n-1);
           metodo3 (n-1);
           for (int i=1; i<=n; i++)  operacion_O(n);
          }
}

public static void metodo4 (int n)
{   if (n>0)
      { operacion_O(n);
        metodo4        (n/2);
        metodo4        (n/2);
        metodo4        (n/2);
        metodo4        (n/2);
        operacion_O(n);
       }
}
```

```
public static void metodo5 (int n)
{   if (n>0)
          { for (int i=1; i<=2*n; i+=3)
                  for (int j=n; j>=1; j-=3)  operacion_O(1);
            metodo5 (n/3);
            for (int k=0; k<=n; k++)  operacion_O(1);
          }
 }

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

EJERCICIO 2

Sea:

```
 … metodo1 (int n)
     { for (int i=0; i<n; i++)
          for (int j=n; j>i; j--)  operacion_O(n);
       metodo2 (n);
      }

 … metodo2 (int n)
      { if (n>0)
          {  for  (int  i=0;  i<n;  i++)
                for (int j=0; j<n; j++)
                     for (int k=0; k<n; k++)  operacion_O(1);
             metodo2 (n-2);
           }
        }
```

SE PIDE: Tras razonar la complejidad temporal de **metodo1**, calcule cuánto tarda para n=10.000, si para n=1.000 tarda 1seg.

EJERCICIO 3

Tras analizar el comportamiento temporal de los dos métodos siguientes, razone qué método de los dos tardará más para un tamaño n=1.000.000, si para un tamaño n=1.000 ambos tardaron el mismo tiempo de 1 segundo:

```
… metodo1 (int n)
    { if (n>0)
              {for (int i=0; i<=2*n; i+=3)
                   for (int j=n; j>=1; j-=3) operacion_O(1);
               metodo1 (n/3);
               for (int k=0; k<=n;k++)  operacion_O(1);
              }
      }
```

```
… metodo2 (int n)
    { if (n>0)
             {  for (int i=0; i<=2*n; i+=3)  operacion_O(1);
                metodo2 (n/2);
                metodo2 (n/2);
                metodo2 (n/2);
                metodo2 (n/2);
                for (int k=0; k<=n;k++)  operacion_O(1);
             }
      }
```

EJERCICIO 4

a) Razonar la complejidad temporal de:

```
… metodo1 (int n)
{ for (int i=1; i<=n; i+=3)
      for (int j=1; j<=n; j*=3)  operacion_O(1);
}
```

b) Razonar la complejidad temporal de:

```
… metodo2 (int n)
{ if (n>0)
   { metodo2(n/3); operacion_O(n);
     metodo2(n/3); operacion_O(n);
     metodo2(n/3); operacion_O(n);
   }
}
```

c) ¿Es mayor la complejidad de ***metodo1*** o la de ***metodo2***?

EJERCICIO 5

Sobre el algoritmo de ordenación por **selección** se le pide:

a) Realice la traza de ejecución sobre los n=8 elementos siguientes: 9,1,8,3,5,4,7,2.
b) Razone: ¿qué complejidad tiene el algoritmo si el vector está inicialmente ordenado? ¿y si está inicialmente en orden inverso?
c) A partir del apartado anterior, calcule lo que tarda el algoritmo para un array inicialmente en orden inverso de n=1.000.000 elementos, si para n=10.000 tarda 3seg.

EJERCICIO 6

Para la siguiente secuencia de números: 5, 9, 1, 4, 3; Se pide:

a) Realizar la traza **QuickSort**.
b) Describir y realizar la traza **MergeSort**
c) Brevemente: similitudes y diferencias entre ambos métodos.

EJERCICIO 7

En el algoritmo de **ordenación rápido** (QuickSort), razonar:

a) ¿cuándo se da el caso mejor y su complejidad?
b) ¿cuándo se da el caso peor y su complejidad?
c) ¿qué caso se da si el vector está inicialmente ordenado y seleccionamos como pivote el primer elemento para cada partición?

d) ¿qué caso se da si el vector está inicialmente ordenado y seleccionamos como pivote el elemento en posición central para cada partición?

SOLUCIONES FINALES

EJERCICIO 1

metodo1 es $O(n^4)$, luego 16 seg.

metodo2 es $O(n^3)$, luego 8 seg.

metodo3 es $O(4^n)$, luego 4^{100} seg, = 2^{200} seg. = aprox. 10^{60} seg.

metodo4 es $O(n^2)$, luego 4 seg.

metodo5 es $O(n^2)$, luego 4 seg.

metodo6 es idem a metodo3.

EJERCICIO 2

metodo2 es $O(n^4)$, luego metodo1 es $O(n^4)$, luego 10^4 seg.

EJERCICIO 3

Ambos son métodos $O(n^2)$, luego 10^6 seg.

EJERCICIO 4

Son iguales, al ser ambos métodos O(n log n).

EJERCICIO 5

a) Hacer traza
b) Selección es para cualquier orden inicial del vector a ordenar $O(n^2)$ …
c) Luego $3*10^4$ seg. =30000 seg.

EJERCICIO 7

a) Cuando el esquema divide y vencerás se comporta por división (a=2; b=2; k=1), por lo tanto O(n log n).
b) Cuando el esquema divide y vencerás se comporta por sustracción (a=1; b=1; k=1), por lo tanto $O(n^2)$.
c) Estamos en el caso visto en el apartado b).
d) Estamos en el caso visto en el apartado a).