ALGORITMIA

Grado en Ingeniería Informática del Software - Curso 2018-2019

GUION DE LA PRÁCTICA 1.2

OBJETIVO:

- **Ejemplos de medición del tiempo de ejecución de algoritmos iterativos.**
- **Constante de Implementación de dos algoritmos.**

1. ALGUNOS MODELOS ITERATIVOS

Se le proporcionan las clases **Bucle1.java, Bucle2.java, Bucle3.java, Incognita.java**. Son modelos iterativos sobre los que hay que trabajar para determinar su complejidad temporal.

Para ello desde línea de comandos:

```
javac *.java
dir
java alg77777777.p12.Bucle1  nVeces   // probar escalas válidas
java alg77777777.p12.Bucle2  nVeces   // probar escalas válidas
java alg77777777.p12.Bucle3  nVeces   // probar escalas válidas
java alg77777777.p12.Incognita  nVeces   // probar escalas válidas
```

Si utilizamos el entorno Eclipse simplemente creamos un proyecto que llamaremos **prac01_Tiempos<UOpropio>** y arrastramos el paquete **alg77777777.p12** a la carpeta src. Renombramos el paquete a *alg<dni>* Para compilar y ejecutar utilizamos la opción "**Run as…**" como indicamos anteriormente. Para añadir los argumentos en la ejecución hay que configurarlos en "**Run configurations…**".

Realizar un análisis del código de cada fichero para plantear su **complejidad analítica**. Comprobar los tiempos para diferentes tamaños del problema, para concluir si los tiempos obtenidos son los que se esperaban de la complejidad de cada programa.

2. COMPARACIÓN DE DOS ALGORITMOS

Para comparar los tiempos de ejecución de dos algoritmos entre sí iremos calculando el cociente de los tiempos de ejecución que tardan esos algoritmos, para el mismo tamaño del problema.

Cuando el tamaño del problema crece y los algoritmos a comparar tienen diferente complejidad, se comprueba que dicho cociente va tendiendo a 0 (si en el numerador ponemos el menos complejo); o a infinito (si en el numerador ponemos el más complejo).

Cuando el tamaño del problema crece y los algoritmos a comparar tienen igual complejidad, se comprueba que dicho cociente va tendiendo a una constante, llamada **constante de implementación**. Esa constante es la que nos dice qué algoritmo, de esos dos que tienen la misma complejidad, es mejor; porque si es menor que 1 es mejor el que hemos puesto en el numerador; si es igual a 1 son iguales y si es mayor que 1 es mejor el que hemos puesto en el denominador.

3. TRABAJO PEDIDO

Tabla 1. Dos algoritmos con misma complejidad:

Vamos a comparar los dos algoritmos *bucle2* y *bucle3* y así obtener la constante *bucle2/bucle3*. Para ello irá rellenando la siguiente tabla:

N	*t bucle2*	*t bucle3*	*t bucle2/t bucle 3*
8	…..	…..	…..
16	…..	…..	…..
32	…..	…..	…..
64	…..	…..	…..
128	…..	…..	…..
256	…..	…..	…..
512	…..	…..	…..
1024	…..	…..	…..
2048	…..	…..	…..
4096	…..	…..	…..
……	…..	…..	…..

Recordad incluir en todas las tablas las unidades de tiempo y las características de la máquina donde se ha realizado la medida.

Tabla 2. Dos algoritmos con distinta complejidad:

Vamos a comparar los dos algoritmos *bucle1* y *bucle2* y así obtener la constante *bucle1/bucle2*. Para ello irá rellenando la siguiente tabla:

N	*t bucle1*	*t bucle2*	*t bucle1/t bucle 2*
8	…..	…..	…..
16	…..	…..	…..
32	…..	…..	…..
64	…..	…..	…..
128	…..	…..	…..
256	…..	…..	…..
512	…..	…..	…..
1024	…..	…..	…..
2048	…..	…..	…..
4096	…..	…..	…..
……	…..	…..	…..

Tabla 3. Complejidad del resto de los algoritmos

Implementar dos nuevas clases *bucle4* y *bucle5*, que simulen algoritmos iterativos con una complejidad $O(n^4)$ y $O(n^3 \log n)$ respectivamente.

Vamos a medir y comparar el resto de los algoritmos bucle4, bucle5, comprobando que los tiempos se corresponden con los valores esperados. Además, se debe medir los tiempos de la última clase proporcionada *incognita* y deducir su complejidad.

N	*t bucle4*	*t bucle5*	*t incognita*
8	…..	…..	…..
16	…..	…..	…..
32	…..	…..	…..
64	…..	…..	…..
128	…..	…..	…..
256	…..	…..	…..
512	…..	…..	…..
1024	…..	…..	…..
2048	…..	…..	…..
4096	…..	…..	…..
Complejidad	$O(n^4)$	$O(n^3 \log n)$	*O(¿?)*

Se entregará:

1. *Una hoja de cálculo con todas tablas y una justificación de las mismas (práctica 1.2).*
2. *Implementación de bucle4 y bucle5 pedidas, dentro del paquete alg<dnipropio>.p12 Si utiliza Eclipse llamar al proyecto prac01_Tiempos<UOpropio>. En el mismo proyecto, pero cad uno en su paquete se entregará el código de semanas anteriores.*
3. *Documentación de las semanas anteriores: hoja de cálculo benchmarking (práctica 0 y 1.1) y*
4. *Hoja de cálculo con las tablas y la justificación de la práctica 1.1*

La entrega se realizará en la tarea habilitada a tal efecto en el campus virtual.

*El **plazo límite es un día antes** de la próxima sesión de prácticas de tu grupo.*