ALGORITMIA

Grado en Ingeniería Informática del Software - Curso 2025-2026

GUION DE LA PRÁCTICA 1.2 (directorio p12)

0)INTRODUCCIÓN

Veremos en esta sesión la medición de tiempos de ejecución de algoritmos iterativos (sobre algunos modelos ejemplo) y cómo comparar entre sí dos algoritmos o dos entornos de desarrollo.

1)ALGUNOS MODELOS ITERATIVOS

Se le proporcionan las clases **Bucle1.java, Bucle2.java, Bucle3.java y Bucle4.java**, son modelos iterativos sobre los que inicialmente hay que analizar sobre el papel su complejidad temporal teórica.

Una vez determinada su complejidad temporal, desde línea de comandos según ya vimos:

```
javac *.java                    // compila todos los fuentes .java
dir
java -Xint p12.Bucle1 1 // 10,100,… ir probando escalas
java -Xint p12.Bucle2 1 // 10,100,… ir probando escalas
java -Xint p12.Bucle3 1 // 10,100,… ir probando escalas
java -Xint p12.Bucle4 1 // 10,100,… ir probando escalas
```

Si utilizamos el entorno Eclipse, creamos un proyecto y se procede como en la práctica anterior.

SE LE PIDE:
Tras medir los tiempos, rellenar la tabla:

TABLA1 (tiempos en milisegundos y SIN_OPTIMIZACIÓN):
Pondremos "FdT" para tiempos superiores al minuto

n	*TBucle1*	*TBucle2*	*TBucle3*	*TBucle4*
100				
200				
400				
800				
1600				
3200				
6400				
12800				
25600				
51200				

Razone si los diferentes tiempos obtenidos concuerdan con lo esperado, según la complejidad temporal de los cuatro casos.

2)CREACIÓN DE MODELOS ITERATIVOS DE UNA COMPLEJIDAD DADA

SE LE PIDE:
Implementar tres nuevas clases Bucle5, Bucle6 y Bucle7, que simulen algoritmos iterativos con una complejidad $O(n^2 \log^2 n)$, $O(n^3 \log n)$ y $O(n^4)$ respectivamente.

Tras implementar las clases, mida sus tiempos de ejecución y rellene la tabla:

TABLA2 (tiempos en milisegundos y SIN_OPTIMIZACIÓN):
Pondremos "FdT" para tiempos superiores al minuto

n	*TBucle5*	*TBucle6*	*TBucle7*
100			
200			
400			
800			
1600			
3200			
6400			

Razone si los diferentes tiempos obtenidos concuerdan con lo esperado, según la complejidad temporal de los tres casos.

3)RELACION TEMPORAL DE ALGORITMOS

En este apartado se pretende fundamentalmente ver cómo podemos proceder para comparar entre sí dos algoritmos, lo que es interesante si se da el caso de que ambos algoritmos están resolviendo el mismo problema.

CASO1: DOS ALGORITMOS DE DIFERENTE COMPLEJIDAD

Para el caso de dos algoritmos con diferente función de complejidad, independientemente del entorno de implementación y ejecución, si calculamos el cociente de tiempos que tardan para el mismo tamaño del problema, veremos que a medida que crece el tamaño de problema, ese cociente va tendiendo a 0 si ponemos en el numerador el de menor complejidad y en el denominador el de mayor (evidentemente, si hacemos al revés lo del numerador y el denominador, ese cociente de tiempo tenderá a infinito, al ir creciendo n).

Lo anterior significa que mientras más crece el tamaño de un problema, más interesa tener algoritmos de menor complejidad.

SE LE PIDE:

Tras medir los tiempos, rellenar la tabla:

TABLA3 (tiempos en milisegundos y SIN_OPTIMIZACIÓN):
Pondremos "FdT" para tiempos superiores al minuto

n	*TBucle1(t1)*	*TBucle2(t2)*	*t1/t2*
100			
200			
400			
800			
1600			
3200			
6400			
12800			
25600			
51200			

Razone si los diferentes tiempos y su cociente concuerda con lo esperado.

CASO2: DOS ALGORITMOS CON LA MISMA COMPLEJIDAD

Para el caso de dos algoritmos con la misma función de complejidad, su comparación hace necesario que tengamos que implementarlos en el mismo entorno de desarrollo y de ejecución. Una vez hecho lo anterior, al medir los tiempos y calcular su cociente para los mismos tamaños del problema, veremos que la relación gira alrededor de un valor constante, que relaciona ambos algoritmos. Si esa constante es menor a 1, es tanto mejor el que hemos puesto en el numerador y si es mayor que 1, es tanto mejor el que hemos puesto en el denominador.

SE LE PIDE:

Tras medir los tiempos, rellenar la tabla:

TABLA4 (tiempos en milisegundos y SIN_OPTIMIZACIÓN):
Pondremos "FdT" para tiempos superiores al minuto

n	*TBucle3(t3)*	*TBucle2(t2)*	*t3/t2*
100			
200			
400			
800			
1600			
3200			
6400			
12800			
25600			
51200			

Razone si los diferentes tiempos y su cociente concuerda con lo esperado.

CASO3: MISMO ALGORITMO EN ENTORNOS DE DESARROLLO y/o EJECUCIÓN DIFERENTES

Para finalizar, vamos a implementar el mismo algoritmo en dos entornos diferentes con el objetivo de comparar, para ese algoritmo, dichos entornos (**Bucle4.py** y **Bucle4.java**). Posteriormente, mediremos los tiempos en Python y Java SIN OPTIMIZACIÓN. El cociente de tiempos para los mismos tamaños del problema oscilará alrededor de un valor (constante que relaciona ambos entornos). Si esta constante es menor a 1, es tanto mejor el que hemos puesto en el numerador y si es mayor que 1, es tanto mejor el que hemos puesto en el denominador.

SE LE PIDE:

Tras medir los tiempos, rellenar la tabla:

TABLA5 (tiempos en milisegundos):
Pondremos "FdT" para tiempos superiores al minuto

n	***Bucle4-Python***	***Bucle4-Java SIN OPTIMIZACION***	*tPython/tJava*
200			
400			
800			
1600			
3200			
6400			

Razone si los diferentes tiempos y sus cocientes concuerda con lo esperado.

Se ha de entregar en un **.pdf** el trabajo que se le pide y además las clases **.java** que ha programado. Todo ello lo pondrá en una carpeta, que es la que entregará comprimida en un fichero **p12ApellidosNombre.zip**.

La entrega de esta práctica se realizará, en tiempo y forma, según las indicaciones dadas por el profesor de prácticas.