ALGORITMIA

Grado en Ingeniería Informática del Software - Curso 2025-2026

GUIÓN DE LA PRÁCTICA 0

0) INTRODUCCIÓN

Como es natural, cuando se resuelve un problema mediante un programa de ordenador interesa implementarlo de la mejor forma posible.

Los parámetros fundamentales que interesa optimizar son estos dos:

*** tiempo de ejecución**: tiempo (p.e. milisegundos) que tarda el ordenador en resolver o ejecutar el programa que se diseñe para resolver el problema.

*** memoria ocupada**: memoria (p.e. Bytes) que ocupa el programa más la que ocupan las diferentes estructuras de datos que nos sirven para modelizar el problema.

Entre los dos factores anteriores (tiempo y memoria), en la práctica suele ser más interesante tender a minimizar el tiempo de ejecución; al menos mientras la memoria no se desborde por no caber en ella las estructuras de datos que utilice el programa realizado.

Pues bien, en esta sesión inicial vamos a ir viendo diferentes factores que influyen en el tiempo de ejecución y a través de un sencillo problema ejemplo se irá analizando en qué medida afectan al tiempo de ejecución.

***PROBLEMA EJEMPLO*:**

"calcular todos los números primos que hay entre 2 y un entero n dado"

(aquí, n es el tamaño de problema)

1) FACTOR1: TAMAÑO DEL PROBLEMA

Mientras más grande sea el tamaño del problema, mayor será el tiempo que se tarde en resolverlo para cualquier algoritmo que se diseñe. Desafortunadamente, la mayoría de los problemas interesantes en la vida real tienen un tamaño considerable y no es posible disminuir esa magnitud.

Para el problema ejemplo antes propuesto, si nos retrotraemos a la asignatura Fundamentos de Informática del primer curso, está claro que procederíamos a abordar su análisis y codificación utilizando el lenguaje de programación Python.

Se adjuntan el siguiente módulo Python:

***PythonA1.py**: se resuelve en Python de una determinada forma (algoritmo **A1**) el *PROBLEMA EJEMPLO* propuesto, mediante su ejecución podemos comprobar como el tiempo de ejecución va aumentando a medida que el problema a resolver aumenta de tamaño (n crece).

SE LE PIDE:

Haga una tabla en la que refleje los tiempos de ejecución del módulo **PythonA1.py** para los valores de n expuestos (5000,10000, 20000, 40000, 80000, …). Si para alguna n tardara más de 60 segundos puede poner FdT ("Fuera de Tiempo"), tanto en este apartado como en los apartados posteriores.

2) FACTOR2: POTENCIA DEL ORDENADOR

Es evidente que un mismo programa va a tardar en ejecutarse más o menos tiempo dependiendo de la potencia del ordenador en el que se ejecute. Esa potencia de un ordenador (trabajo o instrucciones que ejecuta por unidad de tiempo) depende de sus componentes; sin duda el más importante es la CPU que disponga, si bien influyen también la cantidad memoria principal y secundaria, programa de energía del sistema operativo, …

Si tenemos acceso a varios ordenadores, para minimizar el tiempo obviamente nos interesa ejecutar cualquier programa en el ordenador más potente.

SE LE PIDE:

Haga una tabla en la que refleje, **al menos para dos ordenadores** a los que tenga acceso, los tiempos de ejecución del módulo **PythonA1.py** para los valores de n expuestos (5000, 10000, 20000, 40000, 80000, …). Referencie claramente para cada ordenador qué CPU es la existente.

3) FACTOR3: ENTORNO DE IMPLEMENTACIÓN

Ahora vamos a comparar lo hecho antes en el entorno del intérprete Python con otro, en concreto se va a resolver lo mismo en Java y así podremos comparar los tiempos obtenidos para ambos casos.

La conclusión trivial que al final sacaremos es: si para la resolución de un problema tenemos la posibilidad de hacer el desarrollo en varios entornos, para minimizar el tiempo nos interesa seleccionar el más eficiente.

De forma preliminar, se va a recordar cómo compilar y ejecutar programas en Java.

3.1) JAVA con LÍNEA DE COMANDOS (cmd)

La **ejecución básica de programas en Java** se hace desde la **ventana DOS**. Para lograr dicha ventana tiene que ejecutar el comando:

```
CMD
```

Cuando queramos cambiar de unidad sólo tenemos que poner:

```
x:                  // irá a la unidad x:
```

Cuando estemos en un directorio y nos queramos colocar en un hijo haremos:

```
CD hijo
```

Para subir de un directorio al padre:

```
CD ..
```

Una forma de configurar el entorno es crear con un editor un **fichero de lotes (rutalaboratorio.bat)** con el PATH (apuntando al directorio bin de JDK) y el CLASSPATH (apuntando a los directorios de clases Java donde vayamos a trabajar) adecuados al ordenador concreto. Ese fichero de lotes se debe ejecutar cada vez que abramos una ventana DOS.

El comando **SET PATH** permite saber los PATH activos en ese momento.
El comando **SET CLASSPATH** permite saber los CLASSPATH activos en ese momento**.**

Si se cierra la ventana DOS desaparecen las rutas establecidas, por lo que cuando abramos otra nueva ventana DOS existe la necesidad de volver a establecer la ruta.

Cuando creamos un programa o clase JAVA, lo *normal* es que forme parte de un paquete (conjunto de clases JAVA con finalidad común). La primera sentencia válida de la clase establece precisamente a qué paquete pertenece esa clase. Además, el directorio en el que se almacena esa clase debe coincidir exactamente con el nombre del paquete.

Para editar vale cualquier **editor de programas** (vale hasta el Bloc de Notas de Windows, en cuyo caso hay que grabar el programa con la **opción de "Todos los Archivos"** y la extensión .java, para que no se añada automáticamente la extensión **.txt**). En los laboratorios del centro también se encuentran instalados otros editores más potentes que proporcionan ventajas como el coloreado de sintaxis o completado de código.

Tras ejecutar cmd y preparar "ad hoc" el PATH y el CLASSPATH pertinente, para la clase que se le proporciona **JavaA1.java**, se compilará:

javac JavaA1.java

Y se ejecutará:

java -Xint p0.JavaA1 *// JIT desactivado,*
// también se podría en lugar de ***-Xint***
// esto: ***-Djava.compiler=NONE***

o

java p0.JavaA1 *// JIT activado*

Java es un lenguaje de programación que data de 1996, pero que ha ido adaptándose y mejorando con el paso del tiempo, lo cual hace que sea uno de los lenguajes de programación más utilizados y potentes. Para compilar los programas, Java utiliza el compilador **javac**. Dicha herramienta convierte el programa en código máquina (también llamado bytecode). Además, Java cuenta con un componente interno que ha ido mejorando con el paso del tiempo: **Java-In-Time Compiler** (**JIT**).

Este componente **optimiza** la ejecución de los programas consiguiendo mejoras de tiempo muy importantes. Por ejemplo, si el JIT se da cuenta que hemos implementado un bucle en nuestro código pero que ese bucle no cambia ninguna variable que vayamos a utilizar después en el mismo código, es muy posible que el JIT "elimine" ese bucle en tiempo de compilación y así optimice la ejecución del programa. Es **muy importante** tener esto en cuenta porque cuando midamos tiempos puede ser que en algunas ocasiones se produzcan mediciones muy inferiores a lo esperado. Eso significaría que el JIT ha encontrado código "optimizable" y lo ha cambiado para que se ejecute lo más rápido posible. Así, es muy recomendable utilizar el JIT, pero para evitar inconsistencias debidas a optimizaciones realizadas por el JIT en fragmentos de código que no hemos escrito adecuadamente, podemos desactivarlo y así comprobar que el código escrito siempre tiene la complejidad teórica esperada.

En lo sucesivo, denominaremos tiempos "JAVA SIN OPTIMIZACIÓN" a los obtenidos con el JIT desactivado y tiempos "JAVA CON OPTIMIZACIÓN" a los obtenidos con el JIT activado.

3.2) JAVA con ENTORNOS DE DESARROLLO INTEGRADOS (EDI o IDE)

Además de en forma básica (directamente con el JDK), se puede trabajar en JAVA con un IDE de los distintos que existen (Eclipse, NetBeans, BlueJ, IntelliJ Idea, Codenvy, jGrasp, Visual Studio Code, Kawa, …), que permiten integrar múltiples características dentro de un entorno gráfico.

En los laboratorios de prácticas disponemos del **entorno Eclipse** que ya conoce y puede utilizar en estas prácticas.

SE LE PIDE:

Ejecutar la clase que se le proporciona de nombre **JavaA1.java**, que utiliza el mismo algoritmo **A1** para ver resolver el *PROBLEMA EJEMPLO*.

Hacer una tabla en la que refleje los tiempos de ejecución de **JavaA1.java** para los valores de n expuestos (5000, 10000, 20000, 40000, 80000, …).

Los tiempos se han de obtener de la doble forma: ***JAVA SIN OPTIMIZACIÓN y JAVA CON OPTIMIZACIÓN.***

Para finalizar, compare esos tiempos entre sí y con los obtenidos en Python (en un apartado anterior), para ese mismo **algoritmo A1**.

4) FACTOR4: ALGORITMO EMPLEADO

El tiempo de ejecución también va a depender frecuentemente de forma acusada del algoritmo y las estructuras de datos empleadas en resolver el problema. Este factor es determinante y es el objetivo básico de esta asignatura Algoritmia: ***("mejor algoritmo que resuelva un problema dado")***.

Para ilustrar lo anterior, vamos a resolver nuestro problema de otras tres formas (algoritmos A2, A3 y A4):

SE LE PIDE:

Tras estudiar **PythonA2.py, PythonA3.py y PythonA4**, ejecútelas y ponga en una tabla sus tiempos de ejecución, para los valores de n ya antes expuestos (5000, 10000, 20000, 40000, 80000…).

Realice la codificación de esos mismos algoritmos **A2, A3 y A4** en Java, en tres clases de nombres respectivamente **JavaA2.java, JavaA3.java y Java A4.java**.

Posteriormente, realice una tabla en la que refleje los tiempos de ejecución (de ambas formas: ***JAVA SIN OPTIMIZACIÓN y JAVA CON OPTIMIZACIÓN***) de las clases **JavaA2.java, JavaA3.java y JavaA4.java,** para los valores de n expuestos (5000, 10000, 20000, 40000, 80000, …).

Para finalizar, compare esos tiempos con los antes obtenidos en el entorno Python para esos mismos algoritmos.

De forma opcional, puede implementar (en Python y/o en Java) algún otro algoritmo A5 que resuelva el mismo problema, y tras medir sus tiempos de ejecución concluir si su algoritmo A5 destrona o no al mejor de los anteriores (ese que tiene más de 2000 años).

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* ***.zip****.*

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