

PRÓLOGO

Dos de las disciplinas clásicas en todas las carreras relacionadas con la Informática y las Ciencias de la Computación son *Estructuras de Datos* y *Algoritmos*, o bien una sola disciplina si ambas se estudian integradas: “*Algoritmos y Estructuras de Datos*”. El estudio de estructuras de datos y de algoritmos es tan antiguo como el nacimiento de la programación y se ha convertido en estudio obligatorio en todos los currículos desde finales de los años 70 y sobre todo en esa misma década cuando apareció el lenguaje Pascal de la mano del profesor Niklaus Wirtz, y posteriormente en la década de los ochenta con la aparición de su obra —ya clásica— *Algorithms and Data Structures* (1986).

Muchas facultades y escuelas de Ingeniería, así como institutos tecnológicos, comienzan sus cursos de Estructuras de Datos con el soporte de Java. Existen muchas razones por las cuales pensamos que Java es apropiado para la formación en estructuras de datos. Una de ellas es que Java, es un lenguaje más moderno que C o C++, con mejores funcionalidades, orientado a objetos, a la programación en Web,... Además, a partir de Java 1.5 permite diseñar clases genéricas, de forma similar a las plantillas (*templates*) de C++.

El primer problema que se suele presentar al estudiante de *Estructura de Datos* que, probablemente, procederá de un curso de nivel básico, medio o avanzado de *introducción o fundamentos de programación* o bien de *iniciación en algoritmos*, es precisamente el modo de afrontar información compleja desde el principio. Aunque es verdad que Java¹ tiene muchas ventajas sobre un lenguaje procedimental, por ejemplo C, muchas de estas ventajas no se hacen evidentes hasta que un programa se “vuelve” o “hace” más complejo. En este caso el *paradigma orientado a objetos* es una herramienta de programación y organización muy poderosa y con grandes ventajas para la enseñanza y posterior tarea profesional.

A primera vista, Java es más interesante que un lenguaje procedimental por su enfoque orientado a objetos, aunque puede parecer, en el caso del análisis y diseño de algoritmos y estructuras de datos, que esta propiedad añade una complejidad inherente y no es así, la implementación en clases y objetos puede darle una nueva potencialidad. Pensando en esta transición se ha incluido un capítulo dedicado a conceptos teórico-prácticos de orientación a objetos. En cualquier caso, el curso está soportando la comprensión del tipo abstracto de datos (TAD) de modo que el estilo de programación empleado en el texto se basa en el estudio de tipos abstractos de datos como base para la formación en orientación a objetos.

Se estudian estructuras de datos con un objetivo fundamental: aprender a escribir programas más eficientes. También cabe aquí hacerse la pregunta: ¿Por qué se necesitan programas más eficientes cuando las nuevas computadoras son más rápidas cada año? La razón tal vez resida en el hecho de que nuestras metas no se amplían a medida que se aumentan las características de las computadoras. La potencia de cálculo y las capacidades de almacenamiento aumentan la eficacia y ello conlleva un aumento de los resultados de las máquinas y de los programas desarrollados por ellas.

¹ Véanse otras obras de los autores, publicadas también en McGraw-Hill, tales como *Programación en C++*, *Programación en Java 2* o *Programación en C*.

La búsqueda de la eficiencia de un programa no debe chocar con un buen diseño y una codificación clara y legible. La creación de programas eficientes tiene poco que ver con “trucos de programación” sino, al contrario, se basan en una buena organización de la información y buenos algoritmos. Un programador que no domine los principios básicos de diseños claros y limpios probablemente no escribirá programas eficientes. A la inversa, programas claros requieren organizaciones de datos claras y algoritmos claros, precisos y transparentes.

La mayoría de los departamentos informáticos reconocen que las destrezas de buena programación requieren un fuerte énfasis en los principios básicos de ingeniería de software. Por consiguiente, una vez que un programador ha aprendido los principios para diseñar e implementar programas claros y precisos, el paso siguiente es estudiar los efectos de las organizaciones de datos y los algoritmos en la eficiencia de un programa.

EL ENFOQUE DEL LIBRO

En esta obra se muestran numerosas técnicas de representación de datos. En su contexto las mismas se engloban en los siguientes principios:

1. Cada estructura de datos tiene sus costes y sus beneficios. Los programadores y diseñadores necesitan una comprensión rigurosa y completa de cómo evaluar los costes y beneficios para adaptarse a los nuevos retos que afronta la construcción de la aplicación. Estas propiedades requieren un conocimiento o comprensión de los principios del análisis de algoritmos y también una consideración práctica de los efectos significativos del medio físico empleado (p.e. datos almacenados en un disco frente a memoria principal).
2. Los temas relativos a costes y beneficios se consideran dentro del concepto de elemento de compensación. Por ejemplo, es bastante frecuente reducir los requisitos de tiempo en beneficio de un incremento de requisitos de espacio en memoria, o viceversa.
3. Los programadores no deben reinventar la rueda continuamente. Por consiguiente, los estudiantes necesitan aprender las estructuras de datos utilizadas junto con los algoritmos correspondientes.
4. Los datos estructurados siguen a las necesidades. Los estudiantes deben aprender a evaluar primero las necesidades de la aplicación y, a continuación, encontrar una estructura de datos en correspondencia con sus funcionalidades.

Esta edición describe, fundamentalmente, *estructuras de datos*, métodos de organización de grandes cantidades de datos y *algoritmos* junto con el *análisis de los mismos*, en esencia estimación del tiempo de ejecución de algoritmos. A medida que las computadoras se vuelven más y más rápidas, la necesidad de programas que pueden manejar grandes cantidades de entradas se vuelve más críticas y su eficiencia aumenta a medida que estos programas pueden manipular más y mejores organizaciones de datos. Analizando un algoritmo antes de que se codifique realmente, los estudiantes pueden decidir si una determinada solución será factible y rigurosa. Por ejemplo, se pueden ver cómo diseños e implementaciones cuidadosas pueden reducir los costes en tiempo y memoria. Por esta razón, se dedican dos capítulos, en exclusiva a tratar los conceptos fundamentales de *análisis de algoritmos*, y en un gran número de algoritmos se incluyen explicaciones de tiempos de ejecución para poder medir la complejidad y eficiencia de los mismos.

El método didáctico que sigue nuestro libro ya lo hemos seguido en otras obras nuestras y busca preferentemente enseñar al lector a pensar en la resolución de un problema siguiendo un determinado método ya conocido o bien creado por el propio lector. Una vez esbozado el método, se estudia el algoritmo correspondiente junto con las etapas que pueden resolver el problema. A continuación, se escribe el algoritmo, en la mayoría de las veces en lenguaje Java. Para que el lector pueda verificar su programa en la computadora, se incluyen los códigos fuente en la página web del libro.

Uno de los objetivos fundamentales es enseñar al estudiante, simultáneamente, buenas reglas de programación y análisis de algoritmos de modo que puedan desarrollar los programas con la mayor eficiencia posible.

Aunque se ha tratado de que el material de este texto sea lo menos dependiente del lenguaje, la programación, como el lector sabe, requiere el uso de un lenguaje específico. En nuestro caso se ha elegido Java como espina dorsal de programación

Java ya es un lenguaje de programación muy extendido en el mundo de la programación y de la ingeniería de software, tal vez el más utilizado por su gran vinculación con Internet y la web, y también en comparación con el otro gran lenguaje de programación C++. Java ofrece muchos beneficios y los programadores suelen verlo como más seguro, más portable y más fácil de utilizar que C++. Por estas propiedades, es un lenguaje idóneo para el examen e implementación de estructuras de datos fundamentales. Otras características importantes de Java, tales como hilos (*threads*) y sus *GUIs* (Interfaces gráficas de usuario), aunque importantes, no se suelen necesitar en este texto y, por consiguiente, no se han examinado.

Java es un lenguaje de programación sencillo, orientado a objetos, distribuido, interpretado, robusto, seguro, neutro ante la arquitectura, portable, altas prestaciones, multihilo y dinámico

Por el contrario, Java tiene pocas pero algunas desventajas comparado con C++: no soporta bien programación genérica, características de E/S menos potentes, ... Pensando en aquellos lectores que deseen conocer las características del lenguaje C++, en la página oficial del libro, el lector, el profesor y el maestro, podrán encontrar amplia información de C++ y también en obras nuestras, “hermanas” de este texto, tales como *Programación en Java* y *Estructuras de datos en C++*, que se encuentran en dicha página.

Los programas han sido compilados en diversos entornos tales como *JCreator* y *NetBeans* utilizando la plataforma J2SE5 conocida popularmente como versión 5. La reciente aparición de Java SE 6, también conocida como Java 6 puede ser utilizada también para el desarrollo de programas. Para consultar las nuevas características de la versión 6 de Java, presentada a finales del 2006, visite la página de Sun:

```
//java.sun.com/javase/6
```

Si desea descargarse alguna de las versiones de esta nueva versión de Sun acceda a la siguiente dirección web:

```
//java.sun.com/javase/downloads/index.jsp
```

EL LIBRO COMO TEXTO DE REFERENCIA UNIVERSITARIA Y PROFESIONAL

El estudio de *Algoritmos* y de *Estructuras de Datos* son disciplinas académicas que se incorporan a todos los planes de estudios universitarios de Ingeniería e Ingeniería Técnica en Informática, Ingeniería de Sistemas Computacionales y Licenciatura en Informática, así como a los planes de estudio de Formación Profesional e institutos politécnicos. Suele considerarse también a estas disciplinas como ampliaciones de las asignaturas de *Programación*, en cualquiera de sus niveles.

En el caso de España, los actuales planes de estudios de Ingeniería Técnica en Informática e Ingeniería Informática, y los futuros, contemplados en la Declaración de Bolonia, tienen materias troncales relativas tanto a *Algoritmos* como a *Estructuras de Datos*. Igual sucede en los países iberoamericanos donde también es común incluir estas disciplinas en los currículos de carreras de Ingeniería de Sistemas y Licenciaturas en Informática. ACM, la organización profesional norteamericana más prestigiosa a nivel mundial, incluye en las recomendaciones de sus diferentes currículos de carreras relacionadas con informática el estudio de materias de algoritmos y estructuras de datos. En el conocido *Computing Curricula* de 1992 se incluyen descriptores recomendados de *Programación y Estructura de Datos*, y en los últimos currículos publicados, *Computing Curricula* 2001 y 2005, se incluyen en las áreas **PF** de *Fundamentos de Programación (Programming Fundamentals, PF1 a PF4)* y **AL** de *Algoritmos y Complejidad (Algorithms and Complexity, AL1 a AL3)*. En este libro se han incluido los descriptores más importantes tales como *Algoritmos y Resolución de Problemas, Estructuras de datos fundamentales, Recursión, Análisis de algoritmos básicos y estrategias de algoritmos*. Además se incluye un estudio de algoritmos de estructuras discretas tan importantes como *Árboles y Grafos*. Por último, se desarrolla y se ponen ejemplos de todas las colecciones presentes en Java.

Organización del libro

Esta obra está concebida como un libro didáctico y eminentemente práctico. Se pretende enseñar los principios básicos requeridos para seleccionar o diseñar las estructuras de datos que ayudarán a resolver mejor los problemas y a no memorizar una gran cantidad de implementaciones. Por esta razón, se presentan numerosos ejercicios y problemas resueltos en su totalidad, siempre organizados sobre la base del análisis del problema y el algoritmo correspondiente en Java. Los lectores deben tener conocimientos a nivel de iniciación o nivel medio en programación. Es deseable haber cursado al menos un curso de un semestre de introducción a los algoritmos y a la programación, con ayuda de alguna herramienta de programación, preferentemente, y se obtendrá el mayor rendimiento si además se tiene conocimiento de un lenguaje estructurado como C.

El libro busca de modo prioritario enseñar al lector técnicas de programación de algoritmos y estructuras de datos. Se pretende aprender a programar practicando el análisis de los problemas y su codificación en Java.

Está pensado para un curso completo anual o bien dos semestres, para ser estudiado de modo independiente —por esta razón se incluyen las explicaciones y conceptos básicos de la teoría de algoritmos y estructuras de datos— o bien de modo complementario, exclusivamente como apoyo de libros de teoría o simplemente del curso impartido por el maestro o profesor en su aula de clase. Pensando en su uso totalmente práctico se ha optado por seguir una estructura similar

al libro *Algoritmos y Estructura de Datos* publicado por McGraw-Hill por los profesores Joyanes y Zahonero de modo que incluye muchos de los problemas y ejercicios propuestos en esta obra. En caso de realizar su estudio conjunto, uno actuaría como libro de texto, fundamentalmente, y el otro como libro de prácticas para el laboratorio y el estudio en casa o en un curso profesional.

Contenido

El contenido del libro sigue los programas clásicos de las disciplinas *Estructura de Datos* y/o *Estructuras de Datos y de la Información* respetando las directrices emanadas de los *Currícula* de 1991 y las actualizadas del 2001 y 2005 de ACM/IEEE, así como de los planes de estudio de Ingeniero Informático e Ingeniero Técnico en Informática de España y los de Ingeniero de Sistemas y Licenciado en Informática de muchas universidades latinoamericanas.

La Parte I, *Análisis de algoritmos y estructuras de datos básicas* describe el importante concepto de análisis de un algoritmo y las diferentes formas de medir su complejidad y eficiencia; asimismo se describen las estructuras de datos más simples tales como arrays, cadenas o estructuras. La Parte II, *Diseño de algoritmos (Recursividad, ordenación y búsqueda)* examina los algoritmos más utilizados en la construcción de cualquier programa tales como los relativos a búsqueda y ordenación, así como las potentes técnicas de manipulación de la recursividad. La Parte III, *Estructuras de datos lineales (Abstracción de datos, listas, pilas, colas y tablas hash)* constituye una de las partes avanzadas del libro y suele formar parte de cursos de nivel medio/alto en organización de datos. Por último, la Parte IV, *Estructuras de datos no lineales (Árboles, grafos y sus algoritmos)* constituye también una de las partes avanzadas del libro; su conocimiento y manipulación permitirán al programador obtener el máximo aprovechamiento en el diseño y construcción de sus programas. La descripción más detallada de los capítulos correspondientes se reseñan a continuación.

Capítulo 1. Algoritmos y estructuras de datos. Los tipos de datos y la necesidad de su organización en estructuras de datos es la parte central de este capítulo. El estudio de los conceptos de algoritmos y programas y su herramienta de representación más característica, el *pseudocódigo*, son uno de los objetivos más ambiciosos de esta obra. El capítulo hace una revisión de sus propiedades más importantes: representación, eficiencia y exactitud. Se describe la notación *O grande* utilizada preferentemente en el análisis de algoritmos.

Capítulo 2. Tipos de datos: Clases y objetos. La programación orientada a objetos es, hoy en día, el eje fundamental de la programación de computadoras. Su núcleo esencial son los conceptos de clases y objetos. En el capítulo se consideran los conceptos teóricos de encapsulación de datos y tipos abstractos de datos como soporte de una clase y de un objeto; también se analiza el modo de construcción de objetos, así como conceptos tan importantes como la visibilidad de los miembros de una clase. Constructores, paquetes y recolección de objetos junto con la definición de tipos abstractos de datos en Java completan el capítulo.

Capítulo 3. Arrays (arreglos) y cadenas. Los diferentes tipos de *arrays* (*arreglos*) se describen y detallan junto con la introducción a las cadenas (*strings*). La importante clase `String` se describe con detalle, así como la especificación de la clase `Vector`.

Capítulo 4. Clases derivadas y polimorfismo. Uno de los conceptos más empleados en programación orientada a objetos y que ayudará al programador de un modo eficiente al diseño de estructura de datos son las *clases derivadas*. La propiedad de *herencia*, junto con el *polimorfismo* ayudan a definir con toda eficacia las clases derivadas. Otro término fundamental

en POO son las *clases abstractas* que permitirán la construcción de clases derivadas. Java, soporta herencia simple y, por razones de simplicidad, no soporta herencia múltiple, como C++, al objeto de no añadir problemas al diseño. Los conceptos de interfaces también se describen en el capítulo.

Capítulo 5. Algoritmos recursivos. La recursividad es una de las características más sobresalientes en cualquier tipo de programación, Los algoritmos recursivos abundan en la vida ordinaria y el proceso de abstracción que identifica estos algoritmos debe conducir a un buen diseño de algoritmos recursivos. Los algoritmos más sobresalientes y de mayor difusión en el mundo de la programación se explican con detalle en el capítulo. Así, se describen algoritmos como *mergesort* (ordenación por mezclas), *backtracking* (vuelta atrás) y otros.

Capítulo 6. Algoritmos de ordenación y búsqueda. Las operaciones más frecuentes en el proceso de estructura de datos, son: ordenación y búsqueda de datos específicos. Los algoritmos más populares y eficientes de proceso de estructuras de datos internas se describen en el capítulo junto con un análisis de su complejidad. Así, se analizan algoritmos de ordenación básicos y algoritmos avanzados como *Shell*, *QuickSort*, *BinSort* o *RadixSort*; en lo relativo a métodos de búsqueda se describen los dos métodos más utilizados: *secuencial* y *binaria*.

Capítulo 7. Algoritmos de ordenación de archivos. Los archivos (*ficheros*) de datos son, posiblemente, las estructuras de datos más diseñadas y utilizadas por los programadores de aplicaciones y programadores de sistemas. Los conceptos de flujos y archivos de Java junto con los métodos clásicos y eficientes de ordenación de archivos se describen en profundidad en este capítulo.

Capítulo 8. Listas enlazadas. Una lista enlazada es una estructura de datos lineal de gran uso en la vida diaria de las personas y de las organizaciones. Su implementación mediante listas enlazadas es el objetivo central de este capítulo. Variantes de las *listas enlazadas simples* como *doblemente enlazadas* y *circulares*, son también, motivo de estudio en el capítulo.

Capítulo 9. Pilas. La pila es una estructura de datos simple cuyo concepto forma también parte, en un elevado porcentaje de la vida diaria de las personas y organizaciones, como las listas. El tipo de dato `Pila` se puede implementar con arrays o con listas enlazadas y describe ambos algoritmos y sus correspondientes implementaciones en Java.

Capítulo 10. Colas. Al igual que las pilas, las colas conforman otra estructura que abunda en la vida ordinaria. La implementación del TAD `Cola` se puede hacer con *arrays* (arreglos), listas enlazadas e incluso listas circulares. Asimismo, se analiza también en el capítulo el concepto de la *bicola* o cola de doble entrada.

Capítulo 11. Colas de prioridades y montículos. Un tipo especial de cola, la cola de prioridades, utilizada en situaciones especiales para la resolución de problemas, y el concepto de montículo (*heap*, en inglés) se analizan detalladamente, junto con un método de ordenación por montículos muy eficiente, sobre todo en situaciones complejas y difíciles.

Capítulo 12. Tablas de dispersión, funciones *hash*. Las tablas aleatorias *hash* junto con los problemas de resolución de colisiones y los diferentes tipos de direccionamiento conforman este capítulo.

Capítulo 13. Árboles. Árboles binarios y árboles ordenados. Los árboles son estructuras de datos no lineales y jerárquicas muy importantes. Estas estructuras son notables en programación avanzada. Los árboles binarios y los árboles binarios de búsqueda se describen con rigor

y profundidad por su importancia en el mundo actual de la programación tanto tradicional (fuera de línea) como en la Web (en línea).

Capítulo 14. Árboles de búsqueda equilibrados. Este capítulo se dedica a la programación avanzada de árboles de búsqueda equilibrada. Estas estructuras de datos son complejas y su diseño y construcción requiere de estrategias y métodos eficientes para su implementación; sin embargo, su uso puede producir grandes mejoras al diseño y construcción de programas que sería muy difícil y por otros métodos.

Capítulo 15. Grafos, representación y operaciones. Los grafos son una de las herramientas más empleadas en matemáticas, estadística, investigación operativa y numerosos campos científicos. El estudio de la teoría de grafos se realiza fundamentalmente como elemento importante de matemática discreta o matemática aplicada. El conocimiento profundo de la teoría de grafos junto con los algoritmos de implementación es fundamental para conseguir el mayor rendimiento de las operaciones con datos, sobre todo si éstos son complejos en su organización.

Capítulo 16. Grafos, algoritmos fundamentales. Un programador de alto nivel no puede dejar de conocer en toda su profundidad la teoría de grafos y sus aplicaciones en el diseño de redes; por estas razones se analizan los algoritmos de *Warshall*, *Dijkstra* y *Floyd* que estudian los caminos mínimos y más cortos. Se termina el capítulo con la descripción del árbol de expansión de coste mínimo y los algoritmos de *Prim* y *Kruskal* que resuelven su diseño e implementación.

Capítulo 17. Colecciones. El importante concepto de colección se estudia en este capítulo. En particular, los *contenedores* e *iteradores* son dos términos imprescindibles para la programación genérica; su conocimiento y diseño son muy importantes en la formación del programador.

El lector puede encontrar en la página web oficial del libro el Anexo con el estudio de *árboles B* y otros temas avanzados.

Código Java disponible

Los códigos en Java de todos los programas de este libro están disponibles en la web (Internet) <http://www.mhe.es/joyanes> —en formato Word para que puedan ser utilizados directamente y evitar su “teclado” en el caso de los programas largos, o bien simplemente, para seleccionar, recortar, modificar... por el lector a su conveniencia, a medida que avanza en su formación.

AGRADECIMIENTOS

Muchos profesores y colegas españoles y latinoamericanos nos han alentado a escribir esta obra, continuación/complemento de nuestra antigua y todavía disponible en librerías, *Estructura de Datos* cuyo enfoque era en el clásico lenguaje Pascal. A todos ellos queremos mostrarles nuestro agradecimiento y, como siempre, brindarles nuestra colaboración si así lo desean.

A los muchos instructores, maestros y profesores, tanto amigos como anónimos, de universidades e institutos tecnológicos y politécnicos de España y Latinoamérica que siempre apoyan nuestras obras y a los que desgraciadamente nunca podremos agradecer individualmente ese apoyo; al menos que conste en este humilde homenaje, nuestro eterno agradecimiento y reconocimiento por ese cariño que siempre prestan a nuestras obras. Como saben aquellos que nos conocen siempre estamos a su disposición en la medida que, físicamente, nos es posible. Gracias a todos, esta obra es posible, en un porcentaje muy alto, por vuestra ayuda y colaboración.

Y como no, a los estudiantes, a los lectores autodidactas y no autodidactas, que siguen nuestras obras. Su apoyo es un gran acicate para seguir nuestra tarea. También, gracias queridos lectores.

Pero si importantes son en esta obra nuestros colegas y lectores españoles y latinoamericanos, no podemos dejar de citar al equipo humano que desde la editorial siempre cuida nuestras obras y sobre todo nos dan consejos, sugerencias, propuestas, nos “soportan” nuestros retrasos, nuestros “cambios” en la redacción, etc. A Carmelo Sánchez, nuestro editor —y, sin embargo, amigo— de McGraw-Hill, que en esta ocasión, para no ser menos, nos ha vuelto a asesorar tanto en la primera fase de realización como en todo el proceso editorial y a nuestro nuevo editor José Luis García que nos ha seguido apoyando y alentando en nuestro trabajo.

Madrid, Mayo de 2007