Información
Ficha
Temario
Incompatibilidades
Profesorado
Ficha completa PDF
Materiales

Información correspondiente al curso 2010/2011

Justificación

La asignatura de Algorítmica entronca con otras asignaturas que tiene a la programación como eje: Metodología y Tecnología de la Programación, Estructuras de Datos y de la Información y Programación Avanzada. Estas asignaturas previas enseñan los fundamentos de la programacion, los conceptos básicos de estructuras de datos y el diseño orientado a objetos. Algorítmica parte de estos conocimientos y pone énfasis en el análisis y el diseño de algoritmos, sin descuidar los aspectos relacionados con la implementación. El análisis de los algoritmos permite estudiar el consumo de tiempo y espacio en función de la talla del problema, lo que resulta útil para hacer un pronósticio de su comportamiento y seleccionar el algoritmo más eficiente para resolver un problema. El diseño de algoritmos es un arte, pues no hay recetas universales que conduzcan del planteamiento de un problema a un algoritmo resolutivo eficiente. No obstante, hay algunos principios de diseño que pueden ser explotados con éxito en muchas familias de problemas. La asignatura se encarga de presentar estos principios a los estudiantes agrupando los algoritmos como instanciaciones de esquemas algorítmicos. En particular, se estudian las técnicas de "divide y vencerás", "búsqueda con retroceso", "estrategia voraz", "programación dinámica" y "ramificación y acotación". Por otra parte, todo ingeniero informático debe conocer una relación de algoritmos clásicos para la resolución de algunos problemas que se presentan en numerosos contextos. Se trata tanto de tener una cierta "cultura algorítmica" como de tener ocasión de estudiar con detalle algoritmos relevantes con los que reforzar los conceptos de análisis.

Todo ingeniero informático que trabaje en un área relacionada con la programación debe conocer los algoritmos clásicos, ser capaz de analizar los costes temporales y espaciales y diseñar algoritmos aplicando principios de diseño como los desarrollados en la asignatura. El alumno debe ser consciente de que una formación deficiente en las asignaturas II04, II13 e II17 supone una seria limitación para aprovechar el curso de Algorítmica, hasta el punto de que es desaconsejable que se matricule de la II24, Algorítmica, si una o más de una de dichas asignaturas no ha sido superada. Es conveniente, además, que el estudiante repase los conceptos básicos de la programación con Python, pues es el lenguaje en el que se presentan las implementaciones de los algoritmos.

La ubicación de la asignatura en el tercer curso permite beneficiarse de la formación previa de los estudiantes en el campo de la programación y que el estudiante pueda explotar los conocimientos adquiridos en las asignaturas de cursos posteriores en las que se le solicitará diseñar e implementar algoritmos que resuelvan una gran diversidad de problemas.

Contenidos

Introducción

Este bloque ofrece al estudiante una visión global de la asignatura y la enmarca en el contexto del resto del plan de estudios. Se repasan algunos conceptos básicos y se presentan de forma muy general los aspectos que se estudian en el resto de bloques.

Análisis de algoritmos

El tema repasa y profundiza los aspectos relacionados con el análisis de algoritmos. Parte de la idea de analizar el tiempo de ejecución experimentalmente. Tras ver las dificultades prácticas que supone esta estrategia, se introduce el análisis de complejidad (aunque el estudiante ya parte con algunos conocimientos básicos). Además de presentar ejemplos relevantes que permiten practicar el análisis de algoritmos y la comparación de estos en términos de eficiencia, se presentan conceptos como el de coste amortizado.

Algunas estructuras de datos

En este bloque se presentan, en un marco homogéneo, las estructuras de datos que resultarán instrumentales en los algoritmos que se presentan más adelante. Se repasan las estructuras básicas (pilas, colas, listas, árboles, colas de prioridad) y se presentan otras nuevas para el estudiante (diccionarios de prioridad, colas de prioridad con dos extremos, conuntos de unión-búsqueda). Este bloque permite poner en práctica los conceptos aprendidos en el bloque anterior.

Algoritmos sobre grafos

En este bloque se presentan algunos de los problemas y algoritmos clásicos sobre grafos. En particular, se estudia el recorrido de grafos, el cálculo de la arborescencia (o bosque) de recubrimiento, la arborescencia mínima de recubrimiento, el ordenamiento topológico de un grafo acíclico, la clausura de un digrafo y una gran variedad de problemas de cálculo del camino o los caminos más cortos en un digrafo. La soluciones algorítmicas de problemas de cálculo de caminos más cortos son especialmente relevantes en tanto que anticipan algunos esquemas algorítmicos y se utilizan profusamente en muchos de los bloques posteriores.

Divide y vencerás

El objetivo del bloque es doble: presentar el concepto de esquema algorítmico e introducir una estrategia de diseño de algoritmos fundamental (divide y vencerás). Se presenta por medio de un ejemplo paradigmático (la ordenación por fusión) y se amplía con la presentación constructiva de algoritmos divide y vencerás que resuelven diferentes problemas: ordenación rápida, búsqueda binaria, potencia entera, envolvente convexa rápida, etcétera.

Búsqueda y optimización en espacios de estados

Este es un bloque breve que pretende presentar un marco de referencia común a los siguiente 4 bloques, pues todos ellos abordan problemas de búsqueda y/o optimización en espacios de estados.

Búsqueda con retroceso

Aunque se trata de una estrategia que suele conducir a algoritmos de elevado coste, la búsqueda con retroceso permiten ahondar en el concepto de búsqueda en un espacio de estados. Por otra parte, se presentan soluciones algorítmicas para problemas para los que no se conoce otro método resolutivo eficaz y también para otros que desarrollarán soluciones más eficientes en bloques posteriores. No obstante, tiene interés que el estudiante vea como un mismo problema puede abordarse con diferentes esquemas (con resultados dispares). La mayor parte de los problemas considerados son puzles combinatorios, pues son un excelente campo de aplicación de la búsqueda con teroceso y permiten cubrir numeros aspectos de la técnica. El tema finaliza con una exposición del algoritmo DLX para la resolución del problema de la cobertura y de otros por reducción a éste.

Estrategia voraz

La estrategia voraz es la aproximación más sencilla e inmediata. Se presentan algunos problemas en los que la estrategia es exitosa, como el del cambio de moneda en ciertos sistemas monetarios, el cálculo del código de Huffman para compresión de datos o el cálculo de la arborescencia mínima (tanto mediante el algoritmo de Prim como el de Kruskal). En ciertos problemas la estrategia voraz conduce a algoritmos correctos, pero en otros sólo permite diseñar soluciones aproximadas o heurísticas. No obstante, estas soluciones aproximadas o heurísticas son de interés por sí mismas y encuentran aplicación práctica en los algoritmos de ramificación y acotación que se presentan en el último bloque temático de la asignatura.

Programación dinámica

La programación dinámica es una de las estrategias con mayor aplicación práctica. Infinidad de problemas de optimización combinatoria encuentran solución con la programación dinámica, lo que justifica el interés y extensión de este bloque. De todas las estrategias, es en esta en la que el estudiante debe demostrar un mejor desempeño. La exposición de la programación dinámica se enmarca en el cálculo de ciertas funciones sobre secuencias de decisiones. Estas funciones se caracterizan por presentar cierta estructura con operadores de un semianillo. Así, la estrategia permite resolver una familia de cálculos entre las que se encuentran ciertos problemas de optimización. Los ejemplos que se detallan en este bloque cubren un amplio espectro de aplicaciones: carga óptima (problema de la mochila), problemas de optimización en redes, comparación de secuencias, reconocimiento de escritura, análisis sintáctico, etc.

Ramificación y acotación

La ramificación y acotación tiene interés doble: por una parte permite acelerar algoritmos de programación dinámica y, por otra, hace posible abordar problemas de elevada complejidad intrínseca en tallas pequeñas y medianas. El tema presenta estos dos objetivos de la ramificación y acotación y se centra especialmente en el diseño de cotas optimistas.

Metodología

La asignatura divide sus créditos en teoría/problemas y prácticas. Los créditos prácticos, concentrados en las primeras semanas, tienen por objeto repasar y profundizar en los conocimientos sobre el lenguaje de programación Python, en tanto que instrumentales para la asignatura, y a presentar un entorno de desarrollo con el que conviene estar familiarizado (Pydev sobre Eclipse). Los créditos de teoría/problemas se despliegan a lo largo de los dos semestres, con tres horas presenciales por semana y una carga media de unas cuatro horas semanales de trabajo grupal e individual.

Los alumnos se organizan en grupos de tres personas y firman un contrato en el que declaran el día de la semana y horario en que podrán estar reunidos un mínimo de tres horas dedicados a la asignatura. Las clases programadas en el horario dividen su tiempo en sesiones expositivas, actividades prácticas en aula y controles de bloque temático. Los estudiantes reciben una asignación de trabajo cada 15 días (en promedio) y cada asignación supone una dedicación de entre 6 y 8 horas para el grupo de trabajo. Se trata, por lo general, de problemas planteados en un marco realista y que obligan a interpretar los conocimientos adquiridos en un entorno aplicado y a diseñar e implementar soluciones algorítmicas. Cada trabajo supone la entrega de una memoria en la que los estudiantes detallan los pasos seguidos y dificultades experimentadas, a la vez que presentan una solución para el problema en cuestión. La memoria debe ir acompañada del acta de todas las reuniones mantenidas para realizar el trabajo. Las reuniones deben estar estructuradas formalmente, con orden del día fijado a priori, un presidente que organiza el tiempo de la sesión y un secretario que levanta acta (los roles son rotativos). En las actas se indica el tiempo invertido por el grupo en la reunión e individualmente si ha habido un reparto de trabajo individual. El grupo de trabajo debe contar con acceso a un sistema informático, ya que buena parte de las asignaciones comportan la escritura y prueba de programas. Todas las herramientas utilizadas son software libre (Python 3.1, Eclipse 3.5. Pydev 1.4.7), por lo que es gratuita su instalación en equipos personales. Se recomienda que los estudiantes utilicen un equipo portátil y mantengan las reuniones en la propia universidad para poder hacer uso de los servicios en red y recurrir al profesorado cuando surjan dudas.

Al finalizar cada bloque temático, los estudiantes realizan una prueba en aula. La prueba consta de dos partes: una que debe realizarse individualmente y otra de carácter colectivo. La parte individual se centra en el nivel mínimo de conocimientos exigible al estudiante y la parte colectiva es la resolución de un problema completo, entrando en juego los aspectos teóricos y prácticos del tema.

Al finalizar cada semestre hay una prueba parcial que, junto a los trabajos realizados durante el curso, permite liberar materia. La prueba parcial es individual.

Se procura que la actividad presencial en aula haga mínimo el tiempo de exposición oral de contenido teórico, pues los estudiantes disponen de apuntes completos de la asignatura. El profesor puede resolver problemas en pizarra o con ayuda de soporte multimedia. Los alumnos han de realizar ejercicios en aula para poner en práctica los conocimientos recién adquiridos.

Una o dos veces a lo largo del curso se reservan tres sesiones para realiza un puzle de Aronson. El profesor divide los grupos en comités de expertos que estudian diferentes problemas ilustrativos de un bloque temático extraídos de aplicaciones realistas. La primera sesión es de estudio del problema y búsqueda de la solución. La segunda sesión es una puesta en común de la solución o soluciones encontradas y la redacción de una decena de transparencias. Este material, realizado por cada comité de expertos, se pone a disposición de todos ellos para la tercera sesión. En esta última sesión los grupos de tres personas habituales se reúnen y ponen en común lo aprendido haciendo uso del material confeccionado. Los últimos 20 minutos de la sesión se dedican a responder un cuestionario, con preguntas sencillas para los temas de los que no se es experto y avanzadas por el tema estudiado en profundidad.

Es importante que los estudiantes sigan el curso a través del aula virtual, pues allí hay un repositorio del material de la asignatura (apuntes de teoría, problemas propuestos, diferente material de consulta) y foros en los que se pueden plantear y responder dudas.

Materiales y bibliografía

La asignatura sigue unos apuntes redactados por el profesorado y que se ponen a disposición de los alumnos en reprografía y en el aula virtual.

Es posible ampliar conocimientos en aspectos puntuales con estos otros tratados:

* Introduction to Algorithms, de T.H.Cormen, C.E.Leiserson, R.L.Rivest y C.Stein. The MIT Press, segunda edición, 2001.

* Introduction to teh Design and Analysis of Algorithms, de A.V.Levitin. Addison Wesley, segunda edición, 2006.

* The Algorithm Design Manual, de S.S.Skiena. Springer, segunda edición, 2008.

Planificación de actividades

D'acord amb els **7,5 crèdits ECTS**, la dedicació requerida a l'alumnat es de **200 hores**, que es distribuiràn aproximadament de la següent forma:

- Asistencia a clases lectivas teóricas (Exposición de aspectos teóricos):30 hores presencials

- Asistencia a clases lectivas prácticas: Aula (Resolución de problemas en aula):45 hores presencials
- Asistencia clases lectivss prácticas: Lab. (Prácticas de laboratorio):15 hores presencials
- Estudio personal (Estudio, consultas bibliográficas):20 hores no presencials
- Elaboración Informes de Prácticas (Resolución de problemas en equipo):90 hores no presencials

Sistema de evaluación

Evaluación

Se propone un único itinerario de evaluación.

		A
Ejercicios entregables quincenalmente	• Describir algunos algoritmos fundamentales sobre grafos y estructuras de datos avanzadas indicando los contextos en los que son aplicables y su complejidad computacional. • Implementar algoritmos atendiendo a su eficiencia. • Resolver problemas de diseño, análisis e implementación de algoritmos en equipos de trabajo, dejando constancia del procedimiento seguido. • Seleccionar y combinar los algoritmos apropiados de un catálogo para resolver efectivamente un problema.	40%
Controles al final de cada bloque temático	• Comparar diferentes algoritmos en términos de su complejidad computacional y coste practico´y seleccionar el más apropiado y eficiente para la resolución de un problema. • Diseñar algoritmos siguiendo la metodología basada en esquemas algorítmicos.	20%
Exámenes parciales (o final)	• Comparar diferentes algoritmos en términos de su complejidad computacional y coste practico´y seleccionar el más apropiado y eficiente para la resolución de un problema. • Describir algunos algoritmos fundamentales sobre grafos y estructuras de datos avanzadas indicando los contextos en los que son aplicables y su complejidad computacional. • Diseñar algoritmos siguiendo la metodología basada en esquemas algorítmicos. • Resolver problemas de diseño, análisis e implementación de algoritmos en equipos de trabajo, dejando constancia del procedimiento seguido. • Seleccionar y combinar los algoritmos apropiados de un catálogo para resolver efectivamente un problema.	40%
	Total acumulado:	100%