Grado en Ingeniería Informática
30232 - Algoritmia para problemas difíciles

Guía docente para el curso 2013 - 2014

Curso: 4, Semestre: 1, Créditos: 6.0

Información básica

Profesores

- **Jorge Álvarez Jarreta** jorgeal@unizar.es
- **Francisco Javier Campos Laclaustra** jcampos@unizar.es
- **Fernando Tricas García** ftricas@unizar.es
- **Elvira Mayordomo Cámara** elvira@unizar.es

Recomendaciones para cursar esta asignatura

Interés y esfuerzo, además de los conocimientos adquiridos en las asignaturas previas de matemáticas, programación, estructuras de datos y algoritmos, teoría de la computación y algoritmia básica.

Actividades y fechas clave de la asignatura

El calendario de exámenes y las fechas de entrega de trabajos se anunciarán con suficiente antelación en la página web de la asignatura.

Inicio

Resultados de aprendizaje que definen la asignatura

El estudiante, para superar esta asignatura, deberá demostrar los siguientes resultados...

1:

Conoce los modelos de computación no exactos y es capaz de seleccionar el adecuado y utilizarlo para el modelado de dominios de aplicación heterogéneos.

2:

Es capaz de identificar los problemas NP-difíciles.

3:

Conoce técnicas algorítmicas probabilistas, aproximadas y heurísticas para los mismos.

4: Sabe identificar las componentes más relevantes de un problema y seleccionar la técnica aproximada más adecuada para el mismo, además de argumentar de forma razonada dicha elección.

5:
Sabe comparar problemas y utilizar dicha comparación para resolver un problema a partir de una solución eficiente de otro.

6:
Sabe razonar sobre la tasa de error y la eficiencia de los algoritmos aproximados y probabilistas.

7:
Sabe aplicar el análisis amortizado de eficiencia a estructuras de datos avanzadas.

8:
Habilidad para trabajar en grupo, identificar objetivos del grupo, trazar un plan de trabajo para alcanzarlo, reconocer los diferentes papeles dentro de un equipo y asume el compromiso de las tareas encomendadas.

9:
Gestión del autoaprendizaje y de desarrollo incluyendo el tiempo de gestión y de organización.

10:
Apreciar la necesidad del aprendizaje continuo.

Introducción

Breve presentación de la asignatura

Incluso con las técnicas algorítmicas avanzadas de asignaturas anteriores hay problemas que se resisten y para cuya solución eficiente se requieren ideas muy diferentes como permitir que el azar forme parte del algoritmo o utilizar sofisticadas heurísticas o ideas felices.

La URL de la web de la asignatura es http://webdiis.unizar.es/asignaturas/APD/

Contexto y competencias

Sentido, contexto, relevancia y objetivos generales de la asignatura

La asignatura y sus resultados previstos responden a los siguientes planteamientos y objetivos:

En esta asignatura el alumno mejorará su capacidad para diseñar y desarrollar algoritmos incidiendo en las técnicas probabilistas, aproximadas y heurísticas. El alumno aprenderá a reconocer los problemas en cuya solución se utilizan esas técnicas, a diseñar algoritmos que las utilicen y a justificar la corrección y eficiencia de los mismos.

Contexto y sentido de la asignatura en la titulación

La *Especialidad en Computación* abarca una amplia gama de conceptos, desde los fundamentos teóricos y algorítmicos hasta la vanguardia de los desarrollos en bioinformática, robótica, visión por computador, videojuegos, y otras áreas interesantes. *Algoritmia para problemas difíciles* es la segunda de las asignaturas de algoritmia de la Especialidad y pretende complementar la asignatura de *Algoritmia Básica* con los populares algoritmos probabilistas y el uso de heurísticas, entre otros.

Al superar la asignatura, el estudiante será más competente para...

1:
Combinar los conocimientos generalistas y los especializados de Ingeniería para generar propuestas

innovadoras y competitivas en la actividad profesional.

2:
Resolver problemas y tomar decisiones con iniciativa, creatividad y razonamiento crítico.

3:
Comunicar y transmitir conocimientos, habilidades y destrezas en castellano y en inglés.

4:
Usar las técnicas, habilidades y herramientas de la Ingeniería necesarias para la práctica de la misma

5:
Aprender de forma continuada y desarrollar estrategias de aprendizaje autónomo.

6:
Aplicar las tecnologías de la información y las comunicaciones en la Ingeniería.

7:
Capacidad para tener un conocimiento profundo de los principios fundamentales y modelos de la computación y saberlos aplicar para interpretar, seleccionar, valorar, modelar, y crear nuevos conceptos, teorías, usos y desarrollos tecnológicos relacionados con la informática.

8:
Evaluar la complejidad computacional de un problema, conocer estrategias algorítmicas que puedan conducir a su resolución y recomendar, desarrollar e implementar aquella que garantice el mejor rendimiento de acuerdo con los requisitos establecidos.

9:
Capacidad para conocer los fundamentos, paradigmas y técnicas propias de los sistemas inteligentes y analizar, diseñar y construir sistemas, servicios y aplicaciones informáticas que utilicen dichas técnicas en cualquier ámbito de aplicación.

Importancia de los resultados de aprendizaje que se obtienen en la asignatura:

Los problemas difíciles a nivel informático se identifican habitualmente con los problemas NP-difíciles que se estudian en esta asignatura y abarcan problemas interesantes de todos los campos del saber. Se trata de problemas para los que las estrategias algorítmicas habituales resultan muy poco eficientes, o incluso no llegan a dar la solución buscada. Las técnicas algorítmicas no exactas son las adecuadas para muchos de estos problemas y permiten además reflejar intuiciones útiles en la construcción de algoritmos.

Evaluación

Actividades de evaluación

El estudiante deberá demostrar que ha alcanzado los resultados de aprendizaje previstos mediante las siguientes actividades de evaluacion

1:
A lo largo del cuatrimestre se plantearán enunciados prácticos de programación que deberán ser resueltos en el laboratorio. Para ello se formarán equipos integrados por un número determinado de alumnos que se fijará al principio del curso. Los trabajos presentados por los alumnos se calificarán con una nota cuantitativa de 0 a 10. Para obtener dichas notas se valorará el funcionamiento de los programas según especificaciones, la calidad de su diseño y su presentación, la adecuada aplicación de los métodos de resolución, el tiempo empleado, así como la capacidad de cada uno de los integrantes del equipo para explicar y justificar el diseño realizado.

Los alumnos que hayan cumplido con los plazos de entrega fijados para los trabajos prácticos de programación, y hayan demostrado en ellos un nivel de aprovechamiento y calidad de resultados adecuados, obteniendo en la valoración de su trabajo práctico una nota de 5.0 como mínimo, serán exentos de la

realización del examen práctico de programación en laboratorio. Para dichos alumnos, la calificación obtenida con sus prácticas se utilizará como nota de examen práctico de programación en laboratorio, ponderando un 20% de la nota final de la asignatura, salvo que el alumno decida presentarse a la prueba de examen práctico en laboratorio, en cuyo caso prevalecerá la nota obtenida en el examen práctico individual.

2:

Además, a lo largo del cuatrimestre se plantearán hojas de ejercicios para trabajo individual de entre las que los alumnos podrán entregar, no más tarde de la fecha especificada en cada hoja, un número determinado que se fijará al principio del curso. Los ejercicios presentados por los alumnos se calificarán con una nota cuantitativa de 0 a 10.

En la mitad aproximada del cuatrimestre tendrá lugar una prueba escrita intermedia que consistirá en la realización individual durante 50 minutos de algún ejercicio propuesto por el profesor. La prueba escrita intermedia se calificará con una nota cuantitativa de 0 a 10.

Los alumnos que hayan cumplido con los plazos de entrega fijados para los ejercicios de trabajo individual y que hayan realizado la prueba escrita intermedia, y hayan demostrado en ellos un nivel de aprovechamiento y calidad de resultados adecuados, obteniendo en la valoración de su trabajo una nota de 5.0 como mínimo, serán exentos de la realización de una parte del examen escrito. Para dichos alumnos, la calificación obtenida con sus ejercicios de trabajo individual y la obtenida con la prueba escrita intermedia, se utilizarán como parte de la nota final, con una ponderación de 20% para los ejercicios de trabajo individual y 20% para la prueba escrita intermedia, salvo que el alumno decida presentarse al examen final completo, en cuyo caso prevalecerá la nota obtenida en éste.

3:

Evaluación global

La prueba global de evaluación de la asignatura consta de dos partes:

- **Examen práctico** de programación en laboratorio e individual. En cada convocatoria se realizará un examen práctico de programación en laboratorio, en el que se le plantearán al alumno ejercicios de naturaleza similar a los realizados en las prácticas o vistos en clase. La calificación obtenida pondera un 20% de la nota final de la asignatura.

Los alumnos que hayan cumplido con los plazos de entrega fijados para las prácticas de laboratorio, y hayan demostrado en ellas un nivel de aprovechamiento y calidad de resultados adecuados obteniendo una valoración de su trabajo práctico de como mínimo 5.0, serán exentos de la realización del examen práctico de programación en laboratorio convirtiéndose automáticamente la nota numérica obtenida en la evaluación de sus prácticas de laboratorio en su nota final de examen práctico de programación. No obstante, para los alumnos exentos del examen práctico que se presenten al mismo, en cualquier convocatoria, prevalecerá la nota obtenida en el examen práctico individual de programación.

- **Examen escrito** en el que se deberán resolver problemas de naturaleza similar a los ejercicios semanales propuestos durante el cuatrimestre, a los problemas planteados en la prueba escrita intermedia y, en su caso, responder preguntas conceptuales o resolver algún ejercicio en el que se demuestre haber logrado los resultados de aprendizaje requeridos en la asignatura. La calificación obtenida pondera un 80% de la nota final de la asignatura.

Los alumnos que hayan cumplido con los plazos de entrega fijados para los ejercicios semanales individuales y hayan realizado la prueba escrita intermedia, y hayan demostrado en ellos un nivel de aprovechamiento y calidad de resultados adecuados obteniendo una valoración de su trabajo de como mínimo 5.0, serán exentos de realizar una parte del examen escrito, debiendo realizar sólo una parte obligatoria del examen, que se determinará en ese mismo momento. En ese caso, la calificación obtenida en los ejercicios semanales individuales pondera un 20%, la de la prueba escrita intermedia un 20%, y la de la parte obligatoria del examen escrito final un 40%.

4:

En resumen, opción de liberación progresiva de exámenes finales:

1. Parte práctica:

- Prácticas de laboratorio (en grupo) durante el cuatrimestre: 20%.

2. Parte de teoría y problemas:

- Ejercicios de trabajo individual durante el cuatrimestre: 20%.
- Prueba escrita intermedia: 20%.
- Examen final (sólo una parte): 40%.

5:
Por el contrario, opción basada exclusivamente en exámenes finales:

1. Parte práctica:

- Examen práctico (individual) de programación en laboratorio: 20%.

2. Parte de teoría y problemas:

- Examen final (completo): 80%.

Actividades y recursos

Presentación metodológica general

El proceso de aprendizaje que se ha diseñado para esta asignatura se basa en lo siguiente:

El estudio y trabajo continuado desde el primer día de clase.

El aprendizaje de conceptos y metodologías para el diseño e implementación de algoritmos correctos y eficientes a través de las clases magistrales, en las que se favorecerá la participación de los alumnos.

La aplicación de tales conocimientos al diseño y análisis de algoritmos y programas en las clases de problemas. En estas clases los alumnos desempeñarán un papel activo en la discusión y resolución de los problemas.

El trabajo en equipo desarrollado para resolver las prácticas de la asignatura, cuyo resultado se plasma en la entrega de programas resultantes convenientemente diseñados y documentados, así como en la explicación y justificación del diseño realizado y decisiones adoptadas.

Un trabajo continuado en el que se conjugue la comprensión de conceptos, el análisis y la resolución de problemas de programación utilizando “lápiz y papel” y la puesta a punto en computador de algunos proyectos de programación.

Actividades de aprendizaje programadas (Se incluye programa)

El programa que se ofrece al estudiante para ayudarle a lograr los resultados previstos comprende las siguientes actividades...

1:
En las clases impartidas en el aula se desarrollará el temario de la asignatura.

2:
En las clases de problemas se resolverán problemas de aplicación de los conceptos y técnicas presentadas en el programa de la asignatura. Se propondrán problemas y ejercicios para ser resueltos antes de la clase de problemas en la que se presentarán y discutirán diferentes soluciones a dichos problemas. También se propondrán ejercicios durante la sesión de problemas para ser resueltos durante la misma, algunos de forma individual y otros para ser trabajados en grupo.

3:
El trabajo de prácticas se desarrolla en un laboratorio informático o bien en los computadores personales de los alumnos, en casa. En estas sesiones los alumnos deberán trabajar en equipo y realizar una serie de trabajos de programación directamente relacionados con los temas estudiados en la asignatura. Para ello se propondrán una serie de trabajos o ejercicios de programación para que los alumnos los resuelvan en grupo y los entreguen dentro de los plazos de tiempo que se fijen en cada caso.

Planificación y calendario

Calendario de sesiones presenciales y presentación de trabajos

La organización docente de la asignatura prevista es la siguiente:

- Clases teóricas (2 horas semanales)
- Clases de problemas (1 hora semanal)

Presentación de ejercicios y trabajos prácticos:

- Los ejercicios propuestos en clase y los trabajos de programación a desarrollar en laboratorio o en casa deberán ser realizados y presentados de acuerdo a lo especificado para cada uno de ellos, y dentro de las fechas límite que se anunciarán en el enunciado de cada uno de los trabajos propuestos o con suficiente antelación.

Programa

Programa de la asignatura

1. Introducción. Los problemas NP-difíciles. Los problemas de optimización.
2. Algoritmos aproximados. Concepto. Diseño de algoritmos. Garantías y límites.
3. Algoritmos probabilistas: Las Vegas y Montecarlo. Análisis. Generadores pseudoaleatorios.
4. Heurísticas. Algoritmos genéticos. Simulated annealing (templado simulado). Análisis amortizado. Estructuras de datos avanzadas.

Trabajo

Trabajo del estudiante

La dedicación del estudiante para alcanzar los resultados de aprendizaje en esta asignatura se estima en 156 horas distribuidas del siguiente modo:

- 45 horas, aproximadamente, de actividades presenciales (clases teóricas y de problemas);
- 45 horas de trabajo de programación en equipo para desarrollar los programas propuestos como trabajo de laboratorio;
- 60 horas de estudio personal efectivo (estudio de apuntes y textos, resolución de problemas, preparación de clases, desarrollo de programas);
- 6 horas de examen final de teoría escrito y de examen práctico en laboratorio.

Bibliografía

Bibliografía recomendada

Básica:

- J. Hromkovic. *Algorithms for Hard Problems*, Springer, 2001.
- T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein. *Introduction to Algorithms (3rd edition)*, The MIT Press, 2009.
- G. Brassard, P. Bratley. *Fundamentos de Algoritmia*, Prentice Hall, 1997.

Complementaria:

- J. Kleinberg, E. Tardos. *Algorithm Design*, Addison-Wesley, 2005.
- S. Dasgupta, C. Papadimitriou, U. Vazirani. *Algorithms*, McGraw-Hill, 2008.
- V. V. Vazirani. *Approximation Algorithms*, Springer, 2001.

Además: transparencias disponibles en la página web, fotocopias de problemas, ejercicios, etc.

Referencias bibliográficas de la bibliografía recomendada

Escuela Politécnica Superior

- 1. Hromkovic, Juraj : Algorithmics for hard problems : introduction to combinatorial optimization, randomization,

approximation, and heuristics / Juraj Hromkovic . - 2nd ed., corr. Berlin [etc.] : Springer-Verlag, 2004
- 2. Introduction to algorithms / Thomas H. Cormen ... [et al.] . - 3rd ed. Cambridge, Massachusetts ; London : MIT Press, cop. 2009
- 3. Brassard, Gilles : Fundamentos de algoritmia / G. Brassard, T. Bratley ; traducción, Rafael García-Bermejo ; revisión técnica, Narciso Martí, Ricardo Peña, Luis Joyanes Aguilar . - 1ª ed. en español, reimp. Madrid [etc.] : Prentice Hall, 2008
- 4. Kleinberg, Jon. Algorithm design / Jon Kleinberg, Éva Tardos . Boston : Pearson/Addison-Wesley, cop. 2006
- 5. Dasgupta, Sanjoy. Algorithms / Sanjoy Dasgupta, Christos Papadimitriou, Umesh Vazirani . Boston [etc.] : McGraw Hill Higher Education, cop. 2008
- 6. Vazirani, Vijay V. : Approximation algorithms / Vijay V. Vazirani . - [2ª ed.], reimp. Berlin : Springer, cop. 2010

Escuela Universitaria Politécnica

- Brassard, Gilles. Fundamentos de algoritmia / G. Brassard, T. Bratley ; traducción, Rafael García-Bermejo ; revisión técnica, Narciso Martí, Ricardo Peña, Luis Joyanes Aguilar . 1ª ed. en español, reimp. Madrid [etc.] : Prentice Hall, 2008
- Dasgupta, Sanjoy. Algorithms / Sanjoy Dasgupta, Christos Papadimitriou, Umesh Vazirani . Boston [etc.] : McGraw Hill Higher Education, cop. 2008
- Hromkovic, J. Algorithmics for Hard Problems: Introduction to Combinatorial Optimization, Randomization, Approximation, and Heuristics / Juraj Hromkovic. Heidelberg (Germany) : Springer, 2001
- Introduction to algorithms / Thomas H. Cormen ... [et al.] . 3rd ed. Cambridge, Massachusetts ; London : MIT Press, cop. 2009
- Kleinberg, Jon. Algorithm design / Jon Kleinberg, Éva Tardos . Boston : Pearson/Addison-Wesley, cop. 2006
- Vazirani, Vijay V.. Approximation algorithms / Vijay V. Vazirani . [2ª ed.], reimp. Berlin : Springer, cop. 2010