ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA

GUIA DOCENTE

CURSO 2019/2020

Grado en Ingeniería Informática itinerario Computación

DATOS DE LA ASIGNATURA			
Nombre:			
Modelos Avanzados de Computación			
Denominación en inglés:			
Advanced Models of Computation			
Código:		**Carácter:**	
606010237		Obligatorio	
Horas:			
	Totales	**Presenciales**	**No presenciales**
Trabajo estimado:	150	60	90
Créditos:			

	Grupos reducidos			
Grupos grandes	**Aula estándar**	**Laboratorio**	**Prácticas de campo**	**Aula de informática**
3	0	0	0	3

Departamentos:	**Áreas de Conocimiento:**
Tecnologías de la Información	Ciencias de la Computación e Inteligencia Artificial
Curso:	**Cuatrimestre:**
4º - Cuarto	Primer cuatrimestre

DATOS DE LOS PROFESORES			
Nombre:	**E-Mail:**	**Teléfono:**	**Despacho:**
Carpio Cañada, Jose	jose.carpio@dti.uhu.es	959217658	ETSI 145
*Moreno Velo, Francisco José	francisco.moreno@dti.uhu.es	87659	Edificio ETSI, despacho 141

*Profesor coordinador de la asignatura

Consultar los horarios de la asignatura

DATOS ESPECÍFICOS DE LA ASIGNATURA

1. Descripción de contenidos

1.1. Breve descripción (en castellano):

- La máquina de Turing. Solución de problemas mediante máquinas de Turing.
- Los límites de la máquina de Turing. El teorema de Turing. Problemas no computables.
- Complejidad P, NP y PSPACE.
- Problemas NP-completos
- Modelos de computación alternativos. La máquina RAM.
- Funciones recursivas y parcialmente recursivas.
- Conjuntos recursivos y recursivamente enumerables.
- Equivalencia entre modelos de computación. La tesis de Church-Turing.
- Modelos de computación paralela: máquinas PRAM y circuitos booleanos
- Complejidad en tiempo paralelo.

1.2. Breve descripción (en inglés):

- The Turing machine. Troubleshooting by Turing machines.
- The limits of the Turing machine. Turing's theorem. Non-computable problems.
- Complexity P, NP and PSPACE.
- NP-complete problems
- Alternative computing models. RAM machine.
- Recursive and partially recursive functions.
- Recursive and recursively enumerable sets.
- Equivalence between computer models. The Church-Turing thesis.
- Models of parallel computation: PRAM machines and Boolean circuits
- Parallel time complexity.

2. Situación de la asignatura

2.1. Contexto dentro de la titulación:

Esta asignatura complementa la formación relativa a la complejidad y el análisis del algorimos. Los conceptos impartidos en esta asignatura son básicos para entender cuanto tardará un ordenador en resolver un determinado problema. En el contexto de la informática actual, con un creciente volumen de datos (Big Data) originados en redes sociales, sistemas de adquisición de datos o en sistemás de gestión empresarial, se hace si cabe más importante analizar la complejidad de los algoritmos, de forma que el ingeniero cuente con un soporte sólido a la hora de diseñar los sistemas de información actuales. Además, veremos que tipo de problemas son resolubles con un determinado modelo de computador (por ej. una máquina de Turing) en un tiempo razonable (que llamaremos P) y que problemas necesitán mucho más tiempo para su resolución (problemas NP) o incluso aquellos que no son resolubles con un determinado modelo de computador. Así mismo estudiaremos diferentes modelos de computadores distintos a los tradicionales como las máquinas de acceso aleatorio, las funciones recursivas, el cálculo lambda y los sistemas de producción.

2.2. Recomendaciones:

Serán recomendables conocimientos básicos de programación.

3. Objetivos (Expresados como resultados del aprendizaje):

• Conocer el modelo de la Máquina de Turing, su alcance y limitaciones.
• Conocer otros modelos de computación (máquinas RAM, lenguajes algorítmicos sencillos, modelos funcionales) y las relaciones existentes (tesis de Church-Turing).
• Conocer los conceptos de funciones recursivas y parcialmente recursivas.
• Conocer los conceptos de problemas decidibles y semidecidibles.
• Conocer las clases de complejidad computacional más importantes y las relaciones entre ellas.
• Comprender la NP-completitud. Ser capacer de comprobar si un problema es NP-completo.
• Conocer las clases de complejidad para aproximar problemas. Saber clasificar problemas concretos en dichas clases.
• Conocer la jerarquía polinómica. Saber ubicar problemas dentro de dicha jerarquía. Conocer problemas PESPACIO completos.
• Conocer y relacionar los modelos de computación paralela: máquinas PRAM y circuitos booleanos.
• Determinar problemas P-completos. Relacionar la complejidad en tiempo paralelo con la complejidad en espacio secuencial.

4. Competencias a adquirir por los estudiantes

4.1. Competencias específicas:

- **CE1-C:** 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.

4.2. Competencias básicas, generales o transversales:

- **CB5:** Que los estudiantes hayan desarrollado aquellas habilidades de aprendizaje necesarias para emprender estudios posteriores con un alto grado de autonomía
- **CG0:** Capacidad de análisis y síntesis: Encontrar, analizar, criticar (razonamiento crítico), relacionar, estructurar y sintetizar información proveniente de diversas fuentes, así como integrar ideas y conocimientos.
- **G03:** Capacidad para la resolución de problemas
- **G04:** Capacidad para tomar decisiones basadas en criterios objetivos (datos experimentales, científicos o de simulación disponibles) así como capacidad de argumentar y justificar lógicamente dichas decisiones, sabiendo aceptar otros puntos de vista
- **G07:** Motivación por la calidad y la mejora continua, actuando con rigor, responsabilidad y ética profesional.
- **G08:** Capacidad para adaptarse a las tecnologías y a los futuros entornos actualizando las competencias profesionales.
- **G09:** Capacidad para innovar y generar nuevas ideas.
- **CT2:** Desarrollo de una actitud crítica en relación con la capacidad de análisis y síntesis.
- **CT3:** Desarrollo de una actitud de indagación que permita la revisión y avance permanente del conocimiento.
- **CT4:** Capacidad de utilizar las Competencias Informáticas e Informacionales (CI2) en la práctica profesional.

5. Actividades Formativas y Metodologías Docentes

5.1. Actividades formativas:

- Sesiones de Teoría sobre los contenidos del Programa.
- Sesiones Prácticas en Laboratorios Especializados o en Aulas de Informática.
- Actividades Académicamente Dirigidas por el Profesorado: seminarios, conferencias, desarrollo de trabajos, debates, tutorías colectivas, actividades de evaluación y autoevaluación.

5.2. Metologías docentes:

- Clase Magistral Participativa.
- Desarrollo de Prácticas en Laboratorios Especializados o Aulas de Informática en grupos reducidos.
- Resolución de Problemas y Ejercicios Prácticos.
- Tutorías Individuales o Colectivas. Interacción directa profesorado-estudiantes.
- Planteamiento, Realización, Tutorización y Presentación de Trabajos.
- Evaluaciones y Exámenes.

5.3. Desarrollo y justificación:

• Clase Magistral Participativa
Las clases teóricas tendrán una duración de 2 horas. En ellas se expondrá y explicará, con ayuda del cañón de proyecciones y/o la pizarra los contenidos asociados a cada tema. Habrá bibliografía específica de cada tema disponibles en la web de la asignatura con antelación suficiente.
• Resolución de Problemas y Ejercicios Prácticos
Al finalizar las sesiones de teoría de cada tema se desarrollarán las sesiones de problemas correspondientes al tema desarrollado. Para cada tema de teoría se facilitará un boletín de problemas. En estas sesiones se resolverán los problemasmás representativos de cada boletín.
• Desarrollo de Prácticas en Laboratorios Especializados o Aulas de Informática en grupos reducidos
Las sesiones de prácticas se desarrollarán en aulas provistas de ordenadores y tendrán una duración de 2 horas. En estas prácticas se impartirán los contenidos de Programación Funcional utilizando para ello el lenguaje de programación Haskell.
• Planteamiento, Realización, Tutorización y Presentación de Trabajos
A lo largo del curso se planteará uno trabajo práctico a desarrollar por los alumnos de manera individual. El trabajo se referirá al desarrollo de un proyecto utilizando el lenguaje de proramación Haskell. Este trabajo se considera una actividad académica dirigida y su explicación se realizará en el horario de las sesiones de prácticas. El seguimiento de este trabajos se realizará en tutorías individualizadas.

6. Temario desarrollado:

Temario teórico
1. Introducción a los Modelos de Computación
2. Circuitos lógicos
3. Autómatas finitos y autómatas de pila
4. Máquinas de Turing
5. Problemas decidibles y no decidibles
6. Funciones recursivas
7. Complejidad temporal
8. Complejidad espacial
9. Modelos de computación paralela

Temario práctico
1. Introducción a Haskell
2. Tipos y funciones básicas
3. Definición de tipos
4. Programación de funciones
5. Entrada/Salida
6. Testado de programas
7. Mónadas
8. Manejo de errores
9. Programación paralela y concurrente

7. Bibliografía

7.1. Bibliografía básica:

- M. Fernández. Models of Computation: An Introduction to Computability Theory. Undergraduate Topics in Computer Science. Springer (2009). ISBN 978-1-84882-433-1.
- J. E. Savage. Models Of Computation: Exploring the Power of Computing. (1998) (http://cs.brown.edu/~jes/book/home.html).
- M.D. Davis, R. Sigal, E.J. Weyujer. Computability, Complexity, and Languages (2nd. Ed.): Fundamentals of theoretical Computer Science. Academic Press (1994).
- M. Sipser. Introduction to the Theory of Computation (2nd Edition). Thompson (2005).
- B. O'Sullivan, D. Stewart, J. Goerzen. Real World Haskell. O'Really Media (2009). (http://book.realworldhaskell.org/read/index.html)

7.2. Bibliografía complementaria:

- S. Arora, B. Barak. Computational Complexity: A Modern Approach.Cambridge University Press (2009)
- G. Ausiello, P. Creszendi et al. Complexity and Approximation. Springer-Verlag, Berlin (1999)
- R. Greenlaw, H.J. Hoover, W.L. Ruzzo. Limits to Parallel Computation: P-Completeness Theory (1995) Oxford University Press.
- J.E. Hopcroft, R. Motwani, J.D. Ullman. Introducción a la Teoría de Autómatas, Lenguajes y Programación, 2ª Ed. Addison Wesley (2002)

8. Sistemas y criterios de evaluación.

8.1. Sistemas de evaluación:

- Examen de teoría/problemas
- Defensa de Trabajos e Informes Escritos
- Examen de prácticas

8.2. Criterios de evaluación y calificación:

La asistencia tanto a las clases teóricas como prácticas no será obligatoria. La evaluación de la asignatura consta de las siguientes partes:

- Examen de teoría/problemas: 50%
- Examen de prácticas: 30%
- Defensa de trabajos e Informes Escritos: 20%

La evaluación de la asignatura consta de una parte teórica y una parte práctica.
La parte teórica (50% de la nota final) se evalúa por medio de las convocatorias oficiales de exámenes.
La parte práctica (50% de la nota final) se evalúa por medio de un examen de prácticas (30% de la nota final) y de un trabajo individual (20% de la nota final).
La calificación global final será la media de la calificación teórica final y la calificación práctica final, siendo necesaria una calificación mínima de 4.0 puntos (sobre 10.0) en el examen teórico y de 4.0 puntos (sobre 10.0) en la parte práctica.
Mediante la parte teórica se evaluarán las competencias CE1-C, CG0, G03, G09 y T02, mientras que por medio de la parte práctica se evaluarán las competencias CB5, G03, G04, G07 y G08.
Aquellos estudiantes que así lo consideren pueden acogerse a la realización de una evaluación única final. En este caso deberá presentar una solicitud en el REGISTRO GENERAL de la Universidad, en cualquiera de sus REGISTROS AUXILIARES o en el REGISTRO TELEMÁTICO, dirigida a la dirección del departamento y al coordinador de la asignatura.
La evaluación única final consistirá, para todas las convocatorias, en un solo acto académico que estará formado por lassiguientes pruebas:

- Prueba 1: Examen escrito sobre los contenidos explicados en las sesiones de teoría y problemas. Tendrá un carácter presencial e individual, con una duración máximade hasta 3 horas.
- Prueba 2: Examen práctico en el que se planteará un problema a resolver mediante programación funcional. El examen se desarrollará en un aula de ordenadores y su duración máxima será de 1 hora.

Para aprobar la asignatura se tienen que superar con mas de un 5.0 independientemente ambas pruebas.
En el caso de haber más candidatos que posibilidades de matrículas de honor por número de estudiantes en la asignatura,ycon el objetivo de discriminar situaciones de equidad en la calificación final, se seguirá como criterio la mejor nota en elexamen teórico. En caso de seguir el empate no se dará Matricula de honor a ninguno de los alumnos implicados.

9. Organización docente semanal orientativa:

Semanas	Grupos Grandes	Grupos Reducidos Aula Estándar	Grupos Reducidos Aula de Informática	Grupos Reducidos Laboratorio	Grupos Reducidos prácticas de campo	Pruebas y/o actividades evaluables	Contenido desarrollado
#1	2	0	2	0	0		Presentación / Tema1
#2	2	0	2	0	0		Tema 1
#3	2	0	2	0	0		Tema 2
#4	2	0	2	0	0		Tema 3
#5	2	0	2	0	0		Tema 4
#6	2	0	2	0	0		Tema 4
#7	2	0	2	0	0		Tema 5
#8	2	0	2	0	0		Tema 6
#9	2	0	2	0	0		Tema 6
#10	2	0	2	0	0		Tema 7
#11	2	0	2	0	0		Tema 7
#12	2	0	2	0	0		Tema 8
#13	2	0	2	0	0		Tema 8
#14	2	0	2	0	0		Tema 9
#15	2	0	2	0	0		Tema 9
	30	0	30	0	0		