UNIVERSIDAD NACIONAL DE INGENIERÍA

SÍLABO

INFORMACIÓN GENERAL

ASIGNATURA	:TEORÍA DE LA COMPUTACIÓN
DOCENTE	:GERSON GARRIDO
HORAS POR SEMANA	:06 (TEORÍA: 04, LABORATORIO: 02)

SUMILLA

El curso es de naturaleza teórico-práctica y estudia qué es una computadora como objeto matemático, qué problemas no puede resolver ningún programa y cuáles exigen demasiado tiempo o memoria. El estudiante define cada modelo de cómputo con precisión, lo programa y demuestra sus límites. Se organiza en cuatro partes: **lenguajes regulares** (autómatas finitos, expresiones regulares, lema de bombeo y minimización); **lenguajes libres de contexto** (gramáticas, forma normal de Chomsky, algoritmo CYK y autómatas de pila); **computabilidad** (máquinas de Turing, problema de la parada, reducciones, teorema de Rice e incompletitud); y **complejidad** (P y NP sobre máquinas, teorema de Cook-Levin, PSPACE, teoremas de jerarquía y computación probabilística). El laboratorio se desarrolla en C++, con un juez automático, en un entorno común.

COMPETENCIAS

- Diseña un autómata finito, una expresión regular o una gramática para un lenguaje dado, y demuestra que reconoce exactamente ese lenguaje.
- Convierte entre autómatas deterministas, no deterministas y expresiones regulares, y minimiza un autómata, a mano y con un programa.
- Demuestra que un lenguaje no es regular o no es libre de contexto, con el lema de bombeo o con cadenas distinguibles.
- Decide si una cadena pertenece al lenguaje de una gramática con el algoritmo CYK, y relaciona gramáticas con autómatas de pila.
- Describe una máquina de Turing en sus tres niveles de detalle y explica qué afirma la tesis de Church-Turing.
- Demuestra que un problema es indecidible o no reconocible, por diagonalización o con una reducción en la dirección correcta.
- Ubica un problema en P, NP, coNP o PSPACE, y demuestra que es completo para su clase con una reducción en tiempo polinomial.
- Explica la demostración del teorema de Cook-Levin y lo que implican los teoremas de jerarquía.
- Implementa en C++ las construcciones del curso: simuladores de cada modelo, conversiones entre modelos y reducciones que se pueden ejecutar.
- Explica qué preguntas del área siguen abiertas y qué cambiaría con cada respuesta.

UNIDADES DE APRENDIZAJE

Unidad	Contenido	Semanas
I	Lenguajes regulares	1–4
II	Lenguajes libres de contexto	5–6
III	Computabilidad	7–11
IV	Complejidad	12–16

PROGRAMACIÓN SEMANAL

Cada semana de clases comprende dos sesiones de teoría de 2 horas cada una y una sesión de laboratorio de 2 horas. Cada capítulo de los apuntes abre con una ruta de lectura: qué leer antes de clase (15 a 20 minutos), qué se explica en clase y qué repasar después. Los temas marcados con (*) pueden pasar a lectura autónoma si se pierden clases por feriados.

Sem.	Teoría	Laboratorio
1	Presentación. Cadenas, lenguajes y problemas de decisión. Autómatas finitos deterministas. Construcción del producto	Un simulador de DFA
2	No determinismo: NFA y transiciones ε. Construcción de subconjuntos. Propiedades de cierre	De NFA a DFA
3	Expresiones regulares. De expresión regular a NFA. De DFA a expresión regular: el teorema de Kleene	Un `grep` pequeño
4	Lenguajes no regulares: el lema de bombeo. Myhill-Nerode y minimización de DFA. Una cota inferior de memoria en streaming (*)	*Práctica 1*
5	Gramáticas libres de contexto, derivaciones y ambigüedad. Forma normal de Chomsky y el algoritmo CYK	Un parser con CYK
6	Autómatas de pila y su equivalencia con las gramáticas. Lema de bombeo para lenguajes libres de contexto. Cómo trabaja un parser real (*)	*Práctica 2*
7	Máquinas de Turing: definición, configuraciones y ejemplos. Variantes de la máquina. Tesis de Church-Turing	Un simulador de máquinas de Turing
8	**Examen parcial**	
9	Lenguajes decidibles: problemas sobre autómatas y gramáticas. Máquina universal. Diagonalización. El problema de la parada	¿Son equivalentes dos DFA?
10	Reducciones. Teorema de Rice. Historias de cómputo. Problema de correspondencia de Post. Problemas indecidibles sobre gramáticas	Buscar una solución de PCP
11	Autorreferencia: el teorema de recursión y los quines. Incompletitud de Gödel. Complejidad de Kolmogorov (*)	*Práctica 3*
12	Complejidad en tiempo. La clase P sobre máquinas de Turing. NP: verificadores y máquinas no deterministas	Escribir verificadores
13	El teorema de Cook-Levin, demostrado. NP-completitud: de SAT a 3-SAT y más allá. coNP	Cook-Levin ejecutado: de una máquina a una fórmula
14	Complejidad en espacio. Teorema de Savitch. PSPACE. TQBF y juegos. L y NL	*Práctica 4*

Sem.	Teoría	Laboratorio
15	Teoremas de jerarquía. Oráculos y los límites de la diagonalización (*). Computación probabilística: BPP. Qué sigue abierto. Repaso	Verificar un producto de matrices al azar
16	**Examen final**	

METODOLOGÍA

- **Teoría:** clases expositivas con máquinas seguidas paso a paso y demostraciones construidas en clase. Antes de cada semana, el estudiante lee la parte del capítulo que su ruta de lectura marca como previa.
- **Laboratorio:** una sesión semanal en un entorno común para todos los estudiantes, donde programan en C++ una de las construcciones demostradas en clase y la comprueban con un juez automático, que da la respuesta de inmediato.
- **Prácticas:** cuatro prácticas, en las semanas 4, 6, 11 y 14, con una construcción o una demostración escrita y un problema en C++. Ocupan la sesión de laboratorio de su semana, y el problema de esa semana queda como ejercicio opcional.
- Los términos técnicos se presentan en español cuando así se dicen (autómata, gramática, reducción) y en inglés cuando la traducción suena forzada (parser, streaming). Las siglas de los modelos se usan en inglés: DFA, NFA, PDA y CFG.

NORMAS DEL CURSO

- **Trabajo:** Individual. Se puede discutir ideas, pero cada estudiante escribe su propio código y sus demostraciones. Copiar soluciones de otras personas o de internet cuenta como plagio.

Herramientas permitidas

En este curso no se usa IA generativa en ningún trabajo: laboratorios, prácticas ni exámenes. Las demás herramientas se permiten en laboratorios y prácticas, dentro del entorno del curso.

Herramienta	¿Se permite?
Mensajes del compilador, `gdb`, valgrind y sanitizers	Sí
Documentación oficial (cppreference, manuales del entorno) y búsqueda en la web	Sí, para consultar el lenguaje y la biblioteca. Copiar una solución ajena es plagio
Autocompletado del editor que no genera soluciones enteras (nombres, firmas, paréntesis)	Sí
Correctores de ortografía, traductores y herramientas de accesibilidad	Sí
Formateadores de código y linters	Sí
Simuladores de autómatas y de máquinas de Turing, para comprobar una construcción propia	Sí. Diseñar la construcción con un generador automático no cuenta como propia
IA generativa para pedir explicaciones, sin que escriba código ni demostraciones	No
IA generativa que escribe código o demostraciones	No

Cómo se declara lo usado: al final de la entrega, una nota breve con la documentación, las páginas consultadas y el simulador usado, si lo hubo, salvo las del compilador y el editor. Cualquier

entrega puede llevar una explicación oral: quien entrega debe poder explicar cualquier parte de su código y de su demostración.

BIBLIOGRAFÍA

- Apuntes del curso, un capítulo por semana, en el sitio del curso. No se requiere comprar ningún libro.
- Sipser, M. *Introduction to the Theory of Computation*, 3.ª edición. Cengage, 2013. Texto de referencia.
- Erickson, J. *Models of Computation*, notas de lectura gratuitas. `https://jeffe.cs.illinois.edu/teaching/algorithms/`
- Barak, B. *Introduction to Theoretical Computer Science*, gratuito en línea. `https://introtcs.org/public/index.html`
- Sitio del curso: guías semanales, apuntes, diapositivas, código y enunciados.