Universidad Nacional de Ingeniería (UNI)
Escuela Profesional de
Ciencia de la Computación
Sílabo 2026-I

1. CURSO

CS211. Teoría de la Computación (Obligatorio)

2. INFORMACIÓN GENERAL

2.1 Curso	:	CS211. Teoría de la Computación
2.2 Semestre	:	4^{to} Semestre.
2.3 Créditos	:	4
2.4 Horas	:	2 HT; 4 HP;
2.5 Duración del periodo	:	16 semanas
2.6 Condición	:	Obligatorio
2.7 Modalidad de aprendizaje	:	Presencial
2.8 Prerrequisitos	:	CS1D1. Estructuras Discretas. (2^{do} Sem)

3. PROFESORES

Atención previa coordinación con el profesor

4. INTRODUCCIÓN AL CURSO

La Teoría de la Computación proporciona los fundamentos matemáticos para entender qué puede ser computado y con qué eficiencia. Introduce lenguajes formales, autómatas y los límites de la resolubilidad algorítmica. Este conocimiento es fundamental para comprender la construcción de compiladores, la verificación formal y la complejidad inherente de los problemas computacionales.

5. OBJETIVOS

- Dominar los conceptos de autómatas finitos y expresiones regulares.
- Analizar gramáticas libres de contexto y autómatas de pila.
- Comprender la Máquina de Turing Universal como modelo de computación.
- Comprender los límites de la computabilidad y el Problema de la Parada.
- Diferenciar entre clases de complejidad como P y NP.

6. RESULTADOS DEL ESTUDIANTE

AG-C08) Análisis de Problemas: Identifica, formula y analiza problemas complejos de computación. (Usage)

AG-C12) Usa enfoques sistémicos para seleccionar, desarrollar, aplica, integrar y administrar tecnologías seguras de computación para lograr los objetivos del usuario. (Usage)

7. TEMAS

Unidad 1: Lenguajes Formales y Autómatas (18 horas)	
Resultados esperados: AG-C08,AG-C12	
Temas	**Aprendizaje esperado (*Learning Outcomes*)**
• Autómatas formales: – Estado Finito – Pila – Linealmente Acotado – Máquina de Turing • Lenguajes formales, gramáticas y Jerarquía de Chomsky: – Regulares (Tipo-3): ∗ Expresiones Regulares – Libres de Contexto (Tipo-2) – Sensibles al Contexto (Tipo-1) – Recursivamente Enumerables (Tipo-0) • Autómatas deterministas y no deterministas • Demostraciones del Lema de Bombeo: – Demostración de la limitación de Estado Finito/Lenguaje Regular – Limitación de Autómatas de Pila/Lenguaje Libre de Contexto	• Para cada autómata formal en esta unidad: – Explicar su definición comparando sus características con los otros autómatas de esta unidad – Usando un ejemplo, explicar paso a paso cómo el autómata opera sobre la entrada incluyendo si acepta la entrada asociada – Explicar un ejemplo de entradas que pueden y no pueden ser aceptadas por el autómata. [Explicar] • Dado un problema, desarrollar un autómata apropiado que aborde el problema [Crear] • Desarrollar una expresión regular para un lenguaje regular dado expresado en lenguaje natural [Crear] • Convertir entre notaciones equivalentemente poderosas para un lenguaje, incluyendo entre DFAs, NFAs y expresiones regulares, y entre PDAs y CFGs [Aplicar] • Aplicar lemas de bombeo, o medios alternativos, para demostrar las limitaciones de los autómatas de Estado Finito y Pila [Aplicar]
Lecturas : [Sip12], [HMU13]	

Unidad 2: Lenguajes Formales y Autómatas (16 horas)	
Resultados esperados: AG-C08,AG-C12	
Temas	**Aprendizaje esperado (*Learning Outcomes*)**
• Autómatas formales: – Estado Finito – Pila – Linealmente Acotado – Máquina de Turing • Lenguajes formales, gramáticas y Jerarquía de Chomsky: – Regulares (Tipo-3): ∗ Expresiones Regulares – Libres de Contexto (Tipo-2) – Sensibles al Contexto (Tipo-1) – Recursivamente Enumerables (Tipo-0) • Formas Normales: Chomsky y Greibach • Demostraciones del Lema de Bombeo: – Demostración de la limitación de Estado Finito/Lenguaje Regular – Limitación de Autómatas de Pila/Lenguaje Libre de Contexto • Relaciones entre autómatas formales, lenguajes y gramáticas	• Para cada autómata formal en esta unidad: – Explicar su definición comparando sus características con los otros autómatas de esta unidad – Usando un ejemplo, explicar paso a paso cómo el autómata opera sobre la entrada incluyendo si acepta la entrada asociada – Explicar un ejemplo de entradas que pueden y no pueden ser aceptadas por el autómata. [Explicar] • Dado un problema, desarrollar un autómata apropiado que aborde el problema [Crear] • Convertir entre notaciones equivalentemente poderosas para un lenguaje, incluyendo entre DFAs, NFAs y expresiones regulares, y entre PDAs y CFGs [Aplicar] • Aplicar lemas de bombeo, o medios alternativos, para demostrar las limitaciones de los autómatas de Estado Finito y Pila [Aplicar] • Construir GLCs para descripciones de lenguajes formales [Evaluar]
Lecturas : [Sip12], [HMU13]	

Unidad 3: Lenguajes Formales y Autómatas (16 horas)	
Resultados esperados: AG-C08,AG-C12	
Temas	**Aprendizaje esperado (*Learning Outcomes*)**
• Autómatas formales: – Estado Finito – Pila – Linealmente Acotado – Máquina de Turing • Lenguajes formales, gramáticas y Jerarquía de Chomsky: – Regulares (Tipo-3): ∗ Expresiones Regulares – Libres de Contexto (Tipo-2) – Sensibles al Contexto (Tipo-1) – Recursivamente Enumerables (Tipo-0) • Decidabilidad, (in)computabilidad y parada • La tesis de Church-Turing • Reducibilidad y reducciones • Modelos equivalentes de computación algorítmica: – Máquinas de Turing y Variaciones (por ejemplo, multicinta, no deterministas) – Cálculo Lambda – Funciones Mu-Recursivas	• Para cada autómata formal en esta unidad: – Explicar su definición comparando sus características con los otros autómatas de esta unidad – Usando un ejemplo, explicar paso a paso cómo el autómata opera sobre la entrada incluyendo si acepta la entrada asociada – Explicar un ejemplo de entradas que pueden y no pueden ser aceptadas por el autómata. [Explicar] • Explicar una Máquina de Turing universal y su operación [Explicar] • Presentar a una audiencia de compañeros de trabajo y gerentes la imposibilidad de proporcionarles un programa que verifique todos los otros programas, incluyendo algunos aparentemente simples, en busca de bucles infinitos incluyendo una explicación del problema de la parada, por qué no tiene solución algorítmica y su importancia para la computación algorítmica del mundo real [Explicar] • Explicar la Tesis de Church-Turing y su importancia para la computación algorítmica [Explicar] • Aplicar aritmetización y diagonalización para demostrar que el Problema de la Parada para Máquinas de Turing es Indecidible [Aplicar] • Dado un lenguaje indecidible conocido, aplicar una reducción por mapeo o historia computacional para demostrar que otro lenguaje es indecidible [Aplicar]
Lecturas : [Sip12], [Koz06]	

Unidad 4: Lenguajes Formales y Autómatas (12 horas)	
Resultados esperados: AG-C08,AG-C12	
Temas	**Aprendizaje esperado (*Learning Outcomes*)**
• Lenguajes formales, gramáticas y Jerarquía de Chomsky: – Regulares (Tipo-3): ∗ Expresiones Regulares – Libres de Contexto (Tipo-2) – Sensibles al Contexto (Tipo-1) – Recursivamente Enumerables (Tipo-0) • Relaciones entre autómatas formales, lenguajes y gramáticas • Modelos equivalentes de computación algorítmica: – Máquinas de Turing y Variaciones (por ejemplo, multicinta, no deterministas) – Cálculo Lambda – Funciones Mu-Recursivas • Lenguajes Sensibles al Contexto (Tipo-1) y aplicaciones	• Para cada modelo formal en esta unidad: – Explicar su definición comparando sus características con las otras en esta unidad – Explicar ejemplos de entradas que son y no pueden ser aceptadas por el lenguaje/gramática. [Explicar] • Explicar las funciones Recursivas Primitivas y Generales (cero, sucesor, selección, recursión primitiva, composición y Mu), su importancia e implementaciones en Máquina de Turing [Explicar] • Explicar cómo se realiza la computación en el Cálculo Lambda (por ejemplo, conversión alfa y reducción beta) [Explicar] • Explicar el teorema de Rice y su importancia [Explicar]
Lecturas : [Sip12], [Koz06]	

Unidad 5: Marco de Análisis de Complejidad (3 horas)	
Resultados esperados: AG-C08,AG-C12	
Temas	**Aprendizaje esperado (*Learning Outcomes*)**
• Marco de Análisis de Complejidad: – Rendimiento de un algoritmo en el mejor caso, caso promedio y peor caso – Mediciones empíricas y relativas (Orden de Crecimiento) – Tamaño de entrada y operaciones primitivas – Eficiencia de tiempo y espacio • Mediciones empíricas de rendimiento • Compromisos tiempo-espacio en algoritmos	• Preparar una presentación que explique a estudiantes de primer año los conceptos básicos de complejidad algorítmica incluyendo comportamiento de algoritmo en mejor caso, caso promedio y peor caso, notaciones Big-O, Omega y Theta, clases de complejidad, compromisos tiempo-espacio, medición empírica e impacto en problemas prácticos [Explicar] • Para cada algoritmo en la unidad Fundamentos Algorítmicos (AL)-FoundationalDataStructuresAlgorithms, explicar su clase de complejidad de tiempo de ejecución y por qué pertenece a esta clase [Explicar] • Evaluar informalmente la clase de complejidad fundamental de algoritmos simples [Evaluar] • Desarrollar estudios empíricos para determinar y validar hipótesis sobre la complejidad de tiempo de ejecución de varios algoritmos ejecutando algoritmos con entradas de varios tamaños y comparando el rendimiento real con el análisis teórico [Crear] • Explicar ejemplos que ilustren los compromisos tiempo-espacio de algoritmos [Explicar] • Explicar cómo el balance del árbol afecta la eficiencia de las operaciones de árbol de búsqueda binaria [Explicar]
Lecturas : [Sip12], [Cor+22]	

Unidad 6: Notación Asintótica y Clases de Complejidad (4 horas)	
Resultados esperados: AG-C08,AG-C12	
Temas	**Aprendizaje esperado (*Learning Outcomes*)**
• Análisis de complejidad asintótica (cotas promedio y peor caso): – Notaciones formales Big-O, Big-Omega y Big-Theta – Clases de Complejidad Fundamentales y Ejemplos/Problemas Representativos: ∗ $O(1)$ Constante (por ejemplo, acceso a arreglo) ∗ $O(\log_2 n)$ Logarítmica (por ejemplo, búsqueda binaria) ∗ $O(n)$ Lineal (por ejemplo, búsqueda lineal) ∗ $O(n \log_2 n)$ Log Lineal (por ejemplo, mergesort) ∗ $O(n^2)$ Cuadrática (por ejemplo, ordenamiento por selección) ∗ $O(n^c)$ Polinomial (por ejemplo, $O(n^3)$ eliminación gaussiana) ∗ $O(2^n)$ Exponencial (por ejemplo, Mochila, Satisfactibilidad (SAT), Viajante de Comercio (TSP), todos los subconjuntos) ∗ $O(n!)$ Factorial (por ejemplo, circuito hamiltoniano, todas las permutaciones)	• Usando ejemplos, explicar cada una de las clases de complejidad fundamentales en esta unidad [Explicar] • Para cada clase de complejidad fundamental en esta unidad, explicar un algoritmo que demuestre la complejidad de tiempo de ejecución asociada [Explicar] • Explicar a una audiencia no técnica la importancia de los algoritmos tratables versus intratables usando una explicación intuitiva de la complejidad Big-O [Explicar] • Aplicar la notación Big-O para dar cotas superiores en la complejidad tiempo/espacio de algoritmos [Aplicar]
Lecturas : [Sip12], [Cor+22]	

Unidad 7: Análisis de Complejidad II: Recursión, Amortización y Cotas Ajustadas (7 horas)	
Resultados esperados: AG-C08,AG-C12	
Temas	**Aprendizaje esperado (*Learning Outcomes*)**
• Notaciones Little-o, Little-Omega y Little Theta • Análisis recursivo formal • Análisis amortizado	• Dado un problema para programar para el cual puede haber varios enfoques algorítmicos, evaluarlos y determinar cuáles son factibles, y seleccionar uno que sea óptimo en implementación y comportamiento de tiempo de ejecución [Evaluar] • Usar relaciones de recurrencia para evaluar la complejidad de tiempo de algoritmos definidos recursivamente [Aplicar] • Aplicar relaciones de recurrencia elementales usando una forma del Teorema Maestro [Aplicar]
Lecturas : [Sip12], [Cor+22]	

Unidad 8: Teoría de la Complejidad Computacional (6 horas)	
Resultados esperados:	
Temas	**Aprendizaje esperado (*Learning Outcomes*)**
• Tractabilidad e intractabilidad: – Clases de Complejidad P, NP y NP-Completo – Problemas NP-Completos (por ejemplo, SAT, Mochila, TSP) – Reducciones • Modelos de complejidad basados en Máquinas de Turing: – Complejidad de tiempo: ∗ Clases P, NP, NP-C y EXP ∗ Teorema de Cook-Levin – Complejidad de espacio: ∗ NSpace y PSpace ∗ Teorema de Savitch	• Explicar la importancia de la NP-Completitud [Explicar] • Explicar cómo NP-Duro es una cota inferior y NP es una cota superior para la NP-Completitud [Explicar] • Explicar ejemplos de problemas NP-completos [Explicar] • Explicar el Teorema de Cook-Levin y la NP-Completitud de SAT [Explicar] • Explicar las clases P y NP [Explicar] • Demostrar que un problema es NP-Completo reduciendo un problema NP-C clásico conocido a él (por ejemplo, 3SAT y Clique) [Crear] • Explicar la clase P-espacio y su relación con la clase EXP [Explicar]
Lecturas : [Sip12], [Cor+22]	

8. PLAN DE TRABAJO

8.1 Metodología

Se fomenta la participación individual y en equipo para exponer sus ideas, motivándolos con puntos adicionales en las diferentes etapas de la evaluación del curso.

8.2 Sesiones Teóricas

Las sesiones de teoría se llevan a cabo en clases magistrales donde se realizarán actividades que propicien un aprendizaje activo, con dinámicas que permitan a los estudiantes interiorizar los conceptos.

8.3 Sesiones Prácticas

Las sesiones prácticas se llevan en clase donde se desarrollan una serie de ejercicios y/o conceptos prácticos mediante planteamiento de problemas, la resolución de problemas, ejercicios puntuales y/o en contextos aplicativos.

9. SISTEMA DE EVALUACIÓN

********* EVALUATION MISSING ********

10. BIBLIOGRAFÍA BÁSICA

[Koz06] Dexter C. Kozen. *Theory of Computation*. Springer, 2006.

[Sip12] Michael Sipser. *Introduction to the Theory of Computation*. 3rd. Cengage Learning, 2012.

[HMU13] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. 3rd. Pearson, 2013.

[Cor+22] Thomas H. Cormen et al. *Introduction to Algorithms*. 4th. MIT Press, 2022.