



National University of Engineering (UNI)
School of Computer Science
Syllabus 2026-I

1. COURSE

CS211. Theory of Computation (Mandatory)

2. GENERAL INFORMATION

2.1 Course	: CS211. Theory of Computation
2.2 Semester	: 4 th Semester.
2.3 Credits	: 4
2.4 Hours	: 2 HT; 4 HP;
2.5 Duration of the period	: 16 weeks
2.6 Type of course	: Mandatory
2.7 Learning modality	: Face to face
2.8 Prerequisites	: CS1D1. Discrete Structures. (2 nd Sem)

3. PROFESSORS

Meetings after coordination with the professor

4. INTRODUCTION TO THE COURSE

Theory of Computation provides the mathematical foundations to understand what can be computed and how efficiently. It introduces formal languages, automata, and the limits of algorithmic solvability. This knowledge is fundamental for understanding compiler construction, formal verification, and the inherent complexity of computational problems.

5. GOALS

- Master the concepts of finite automata and regular expressions.
- Analyze context-free grammars and pushdown automata.
- Understand the Universal Turing Machine as a model of computation.
- Comprehend the boundaries of computability and the Halting Problem.
- Differentiate between complexity classes like P and NP.

6. COMPETENCES

AG-C08) Problem Analysis: Identifies, formulates, and analyzes complex computing problems. (Usage)

AG-C12) Uses systemic approaches to select, develop, apply, integrate, and manage secure computing technologies to achieve user objectives. (Usage)

7. TOPICS

Unit 1: Lenguajes Formales y Autómatas (18 hours)	
Competences Expected: AG-C08,AG-C12	
Topics	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]
Readings : [Sip12], [HMu13]	

Unit 2: Lenguajes Formales y Autómatas (16 hours)	
Competences Expected: AG-C08,AG-C12	
Topics	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) • Normal Forms: Chomsky and 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] • Construct CFGs for formal language descriptions [Evaluar]
Readings : [Sip12], [HMU13]	

Unit 3: Lenguajes Formales y Autómatas (16 hours)	
Competences Expected: AG-C08,AG-C12	
Topics	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) • Decidibilidad, (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]
Readings : [Sip12], [Koz06]	

Unit 4: Lenguajes Formales y Autómatas (12 hours)	
Competences Expected: AG-C08,AG-C12	
Topics	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 • Context-Sensitive Languages (Type-1) and applications	• 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]
Readings : [Sip12], [Koz06]	

Unit 5: Marco de Análisis de Complejidad (3 hours)	
Competences Expected: AG-C08,AG-C12	
Topics	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]
Readings : [Sip12], [Cor+22]	

Unit 6: Notación Asintótica y Clases de Complejidad (4 hours)	
Competences Expected: AG-C08,AG-C12	
Topics	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]
Readings : [Sip12], [Cor+22]	

Unit 7: Análisis de Complejidad II: Recursión, Amortización y Cotas Ajustadas (7 hours)	
Competences Expected: AG-C08,AG-C12	
Topics	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]
Readings : [Sip12], [Cor+22]	

Unit 8: Teoría de la Complejidad Computacional (6 hours)	
Competences Expected:	
Topics	Learning Outcomes
• Tractabilidad e intractabilidad: – Clases de Complejidad P, NP y NP-Completo – Problemas NP-Completo (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-Complejidad [Explicar] • Explicar cómo NP-Duro es una cota inferior y NP es una cota superior para la NP-Complejidad [Explicar] • Explicar ejemplos de problemas NP-completos [Explicar] • Explicar el Teorema de Cook-Levin y la NP-Complejidad 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]
Readings : [Sip12], [Cor+22]	

8. WORKPLAN

8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

9. EVALUATION SYSTEM

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

10. BASIC BIBLIOGRAPHY

- [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.