Universidad Católica San Pablo (UCSP)
Escuela Profesional de
Ciencia de la Computación
SILABO

CS211. Teoría de la Computación (Obligatorio)

2020-I

1. Información general		
1.1 Escuela	:	Ciencia de la Computación
1.2 Curso	:	CS211. Teoría de la Computación
1.3 Semestre	:	4^{to} Semestre.
1.4 Prerrequisitos	:	CS1D2. Estructuras Discretas II. (2^{do} Sem)
1.5 Condición	:	Obligatorio
1.6 Modalidad de aprendizaje	:	Virtual
1.7 horas	:	2 HT; 2 HP; 2 HL;
1.8 Créditos	:	4

2. Profesores

3. Fundamentación del curso

Este curso hace enfasis en los lenguajes formales, modelos de computación y computabilidad, además de incluir fundamentos de la complejidad computacional y de los problemas NP completos.

4. Resumen

1. Computabilidad y complejidad básica de autómatas 2. Complejidad Computacional Avanzada 3. Teoría y Computabilidad Avanzada de Autómatas

5. Objetivos Generales

- Que el alumno aprenda los conceptos fundamentales de la teoría de lenguajes formales.

6. Contribución a los resultados (*Outcomes*)

Esta disciplina contribuye al logro de los siguientes resultados de la carrera:

1) Analizar un problema computacional complejo y aplicar los principios computacionales y otras disciplinas relevantes para identificar soluciones. (**Evaluar**)

6) Aplicar fundamentos de teoria de ciencias de la computación y desarrollo de software para producir soluciones basados en computación. (**Evaluar**)

7. Contenido

UNIDAD 1: Computabilidad y complejidad básica de autómatas (20)	
Competencias:	
Contenido	**Objetivos Generales**
• Máquinas de estado finito. • Expresiones regulares. • Problema de la parada. • Gramáticas libres de contexto. • Introducción a las clases P y NP y al problema P vs. NP. • Introducción y ejemplos de problemas NP- Completos y a clases NP-Completos. • Máquinas de Turing, o un modelo formal equivalente de computación universal. • Máquinas de Turing no determinísticas. • Jerarquía de Chomsky. • La tesis de Church-Turing. • Computabilidad. • Teorema de Rice. • Ejemplos de funciones no computables. • Implicaciones de la no-computabilidad.	• Discute el concepto de máquina de estado finito [Evaluar] • Diseñe una máquina de estado finito determinista para aceptar un determinado lenguaje [Evaluar] • Genere una expresión regular para representar un lenguaje específico [Evaluar] • Explique porque el problema de la parada no tiene solucion algorítmica [Evaluar] • Diseñe una gramática libre de contexto para representar un lenguaje especificado [Evaluar] • Define las clases P y NP [Evaluar] • Explique el significado de NP-Completitud [Evaluar] • Explica la tesis de Church-Turing y su importancia [Familiarizarse] • Explica el teorema de Rice y su importancia [Familiarizarse] • Da ejemplos de funciones no computables [Familiarizarse] • Demuestra que un problema es no computable al reducir un problema clásico no computable en base a él [Familiarizarse]
Lecturas: Martin (2010), Linz (2011), Sipser (2012)	

UNIDAD 2: Complejidad Computacional Avanzada (20)	
Competencias:	
Contenido	**Objetivos Generales**
• Revisión de las clases P y NP; introducir spacio P y EXP. • Jerarquía polimonial. • NP completitud (Teorema de Cook). • Problemas NP completos clásicos. • Técnicas de reducción.	• Define las clases P y NP (También aparece en AL / Automata Básico, Computalidad y Complejidad) [Evaluar] • Define la clase P-Space y su relación con la clase EXP [Evaluar] • Explique el significado de NP-Completo (También aparece en AL / Automata Básico, Computalidad y Complejidad) [Evaluar] • Muestre ejemplos de problemas clásicos en NP - Completo [Evaluar] • Pruebe que un problema es NP- Completo reduciendo un problema conocido como NP-Completo [Evaluar]
Lecturas: Martin (2010), Linz (2011), Sipser (2012), Hopcroft and Ullman (2013)	

UNIDAD 3: Teoría y Computabilidad Avanzada de Autómatas (20)	
Competencias:	
Contenido	**Objetivos Generales**
• Conjuntos y Lenguajes: – Lenguajes Regulares. – Revisión de autómatas finitos determinísticos (Deterministic Finite Automata DFAs) – Autómata finito no determinístico (Nondeterministic Finite Automata NFAs) – Equivalencia de DFAs y NFAs. – Revisión de expresiones regulares; su equivalencia con autómatas finitos. – Propiedades de cierre. – Probando no-regularidad de lenguajes, a través del lema de bombeo (Pumping Lemma) o medios alternativos. • Lenguajes libres de contexto: – Autómatas de pila (Push-down automata (PDAs) – Relación entre PDA y gramáticas libres de contexto. – Propiedades de los lenguajes libres de contexto.	• Determina la ubicación de un lenguaje en la jerarquía de Chomsky (regular, libre de contexto, enumerable recursivamente) [Evaluar] • Convierte entre notaciones igualmente poderosas para un lenguaje, incluyendo entre estas AFDs, AFNDs, expresiones regulares, y entre AP y GLCs [Evaluar]
Lecturas: Hopcroft and Ullman (2013), Brookshear (1993)	

8. Metodología

El profesor del curso presentará clases teóricas de los temas señalados en el programa propiciando la intervención de los alumnos.

El profesor del curso presentará demostraciones para fundamentar clases teóricas.

El profesor y los alumnos realizarán prácticas

Los alumnos deberán asistir a clase habiendo leído lo que el profesor va a presentar. De esta manera se facilitará la comprensión y los estudiantes estarán en mejores condiciones de hacer consultas en clase.

9. Evaluar

Evaluación Continua 1 : 20 %

Examen parcial : 30 %

Evaluación Continua 2 : 20 %

Examen final : 30 %

References

Brookshear, J. Glenn (1993). *Teoría de la Computación*. Addison Wesley Iberoamericana.

Hopcroft, John E. and Jeffrey D. Ullman (2013). *Introducción a la Teoría de Autómatas, Lenguajes y Computación*. Pearson Education.

Linz, Peter (2011). *An Introduction to Formal Languages and Automata*. 5th. Jones and Bartlett Learning.

Martin, John (2010). *Introduction to Languages and the Theory of Computation*. 4th. McGraw-Hill.
Sipser, Michael (2012). *Introduction to the Theory of Computation*. 3rd. Cengage Learning.