Universidad Nacional del Altiplano (UNA)
Escuela Profesional de
Ciencia de la Computación
Sílabo 2024-II

1. CURSO

CS212. Análisis y Diseño de Algoritmos (Obligatorio)

2. INFORMACIÓN GENERAL

2.1 Curso	:	CS212. Análisis y Diseño de Algoritmos
2.2 Semestre	:	5^{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	:	

- CS210. Algoritmos y Estructuras de Datos. (4^{to} Sem)
- CS211. Teoría de la Computación. (4^{to} Sem)
- CS210. Algoritmos y Estructuras de Datos. (4^{to} Sem)
- CS211. Teoría de la Computación. (4^{to} Sem)

3. PROFESORES

Atención previa coordinación con el profesor

4. INTRODUCCIÓN AL CURSO

Un algoritmo es, esencialmente, un conjunto bien definido de reglas o instrucciones que permitan resolver un problema computacional. El estudio teórico del desempeño de los algoritmos y los recursos utilizados por estos, generalmente tiempo y espacio, nos permite evaluar si un algoritmo es adecuado para un resolver un problema específico, compararlo con otros algoritmos para el mismo problema o incluso delimitar la frontera entre lo viable y lo imposible.

Esta materia es tan importante que incluso Donald E. Knuth definió a Ciencia de la Computación como el estudio de algoritmos.

En este curso serán presentadas las técnicas más comunes utilizadas en el análisis y diseño de algoritmos eficientes, con el propósito de aprender los principios fundamentales del diseño, implementación y análisis de algoritmos para la solución de problemas computacionales.

5. OBJETIVOS

- Desarrollar la capacidad para evaluar la complejidad y calidad de algoritmos propuestos para un determinado problema.
- Estudiar los algoritmos más representativos, introductorios de las clases más importantes de problemas tratados en computación.
- Desarrollar la capacidad de resolución de problemas algorítmicos utilizando los principios fundamentales de diseño de algoritmos aprendidos.
- Ser capaz de responder a las siguientes preguntas cuando le sea presentado un nuevo algoritmo: ¿Cuán buen desempeño tiene?, ¿Existe una mejor forma de resolver el problema?

6. RESULTADOS DEL ESTUDIANTE

) ()

) ()

6) Aplicar la teoría de la computación y los fundamentos del desarrollo de software para producir soluciones basadas en computación. ()

7. TEMAS

Unidad 1: Análisis Básico (10)	
Resultados esperados:	
Temas	**Objetivos de Aprendizaje (*Learning Outcomes*)**
• Diferencias entre el mejor, el esperado y el peor caso de un algoritmo. • Análisis asintótico de complejidad de cotas superior y esperada. • Definición formal de la Notación Big O. • Clases de complejidad como constante, logarítmica, lineal, cuadrática y exponencial. • Uso de la notación Big O. • Relaciones recurrentes. • Análisis de algoritmos iterativos y recursivos. • Algunas versiones del Teorema Maestro.	• Explique a que se refiere con "mejor", "esperado" y "peor" caso de comportamiento de un algoritmo [Evaluar] • En el contexto de a algoritmos específicos, identifique las características de data y/o otras condiciones o suposiciones que lleven a diferentes comportamientos [Evaluar] • Determine informalmente el tiempo y el espacio de complejidad de simples algoritmos [Evaluar] • Indique la definición formal de Big O [Evaluar] • Lista y contraste de clases estándares de complejidad [Evaluar] • Use la notación formal de la Big O para dar límites superiores asintóticos en la complejidad de tiempo y espacio de los algoritmos [Evaluar] • Usar la notación formal Big O para dar límites de casos esperados en el tiempo de complejidad de los algoritmos [Evaluar] • Explicar el uso de la notación theta grande, omega grande y o pequeña para describir la cantidad de trabajo hecho por un algoritmo [Evaluar] • Usar relaciones recurrentes para determinar el tiempo de complejidad de algoritmos recursivamente definidos [Evaluar] • Resuelve relaciones de recurrencia básicas, por ejemplo. usando alguna forma del Teorema Maestro [Evaluar]
Lecturas : [**KT2005**], [**DPV2006**], [**CLRS2009**], [**S2013**], [**K1997**]	

Unidad 2: Estrategias Algorítmicas (30)	
Resultados esperados:	
Temas	**Objetivos de Aprendizaje (*Learning Outcomes*)**
• Algoritmos de fuerza bruta. • Algoritmos voraces. • Divide y vencerás. • Programación Dinámica.	• Para cada una de las estrategias (fuerza bruta, algoritmo goloso, divide y vencerás, recursividad en reversa y programación dinámica), identifica un ejemplo práctico en el cual se pueda aplicar [Evaluar] • Utiliza un enfoque voraz para resolver un problema específico y determina si la regla escogida lo guía a una solución óptima [Evaluar] • Utiliza un enfoque voraz para resolver un problema específico y determina si la regla escogida lo guía a una solución óptima [Evaluar] • Usa programación dinámica para resolver un problema determinado [Evaluar] • Determina el enfoque algorítmico adecuado para un problema [Evaluar]
Lecturas : [**KT2005**], [**DPV2006**], [**CLRS2009**], [**A1999**]	

Unidad 3: Algoritmos y Estructuras de Datos fundamentales (10)	
Resultados esperados:	
Temas	**Objetivos de Aprendizaje (*Learning Outcomes*)**
• Algoritmos numéricos simples, tales como el cálculo de la media de una lista de números, encontrar el mínimo y máximo. • Algoritmos de búsqueda secuencial y binaria. • Algoritmos de ordenamiento de peor caso cuadrático (selección, inserción) • Algoritmos de ordenamiento con peor caso o caso promedio en O(N lg N) (Quicksort, Heapsort, Mergesort) • Grafos y algoritmos en grafos: – Representación de grafos (ej., lista de adyacencia, matriz de adyacencia) – Recorrido en profundidad y amplitud • Montículos (Heaps) • Grafos y algoritmos en grafos: – Algoritmos de la ruta más corta (algoritmos de Dijkstra y Floyd) – Árbol de expansión mínima (algoritmos de Prim y Kruskal)	• Implementar algoritmos numéricos básicos [Evaluar] • Implementar algoritmos de busqueda simple y explicar las diferencias en sus tiempos de complejidad [Evaluar] • Ser capaz de implementar algoritmos de ordenamiento comunes cuádraticos y O(N log N) [Evaluar] • Discutir el tiempo de ejecución y eficiencia de memoria de los principales algoritmos de ordenamiento, busqueda y hashing [Usar] • Discutir factores otros que no sean eficiencia computacional que influyan en la elección de algoritmos, tales como tiempo de programación, mantenibilidad, y el uso de patrones específicos de la aplicación en los datos de entrada [Familiarizarse] • Resolver problemas usando algoritmos básicos de grafos, incluyendo busqueda por profundidad y busqueda por amplitud [Evaluar] • Demostrar habilidad para evaluar algoritmos, para seleccionar de un rango de posibles opciones, para proveer una justificación por esa selección,y para implementar el algoritmo en un contexto en específico [Evaluar] • Describir la implementación de tablas hash, incluyendo resolución y el evitamiento de colisiones [Evaluar] • Resolver problemas usando algoritmos básicos de grafos, incluyendo busqueda por profundidad y busqueda por amplitud [Evaluar]
Lecturas : [**KT2005**], [**DPV2006**], [**CLRS2009**], [**S2011**], [**GT2009**]	

Unidad 4: Computabilidad y complejidad básica de autómatas (2)	
Resultados esperados:	
Temas	**Objetivos de Aprendizaje (*Learning Outcomes*)**
• 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.	• Define las clases P y NP [Familiarizarse] • Explique el significado de NP-Completitud [Familiarizarse]
Lecturas : [**KT2005**], [**DPV2006**], [**CLRS2009**]	

Unidad 5: Estructuras de Datos Avanzadas y Análisis de Algoritmos (8)	
Resultados esperados:	
Temas	**Objetivos de Aprendizaje (*Learning Outcomes*)**
• Grafos (ej. Ordenamiento Topológico, encontrando componentes puertemente conectados) • Algoritmos Teórico-Numéricos (Aritmética Modular, Prueba del Número Primo, Factorización Entera) • Algoritmos aleatorios. • Análisis amortizado. • Análisis Probabilístico.	• Entender el mapeamento de problemas del mundo real a soluciones algorítmicas (ejemplo, problemas de grafos, programas lineares,etc) [Familiarizarse] • Seleccionar y aplicar técnicas de algoritmos avanzadas (ejemplo, randonmización, aproximación) para resolver problemas reales [Usar] • Seleccionar y aplicar técnicas avanzadas de análisis (ejemplo, amortizado, probabilistico,etc) para algoritmos [Usar]
Lecturas : [**KT2005**], [**DPV2006**], [**CLRS2009**], [**T1983**], [**R1992**]	

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