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

CS212. Análisis y Diseño de Algoritmos (Obligatorio)

2023-I

1. Información general		
1.1 Escuela	:	Ciencia de la Computación
1.2 Curso	:	CS212. Análisis y Diseño de Algoritmos
1.3 Semestre	:	5^{to} Semestre.
1.4 Prerrequisitos	:	CS210. Algoritmos y Estructuras de Datos. (4^{to} Sem)
1.5 Condición	:	Obligatorio
1.6 Modalidad de aprendizaje	:	Presencial
1.7 horas	:	2 HT; 4 HP;
1.8 Créditos	:	4
1.9 Plan	:	Plan Curricular 2016

2. Profesores

Titular

- Juan Carlos Gutiérrez Cáceres <jcgutierrezc@ucsp.edu.pe>
  - Doctor en Ciencia de la Computación, Universidad Nacional de San Agustín, Perú, 2013.
  - Master en Ciencia de la Computación, ICMC-USP, Brasil, 2003.

3. Fundamentación del 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.

4. Resumen

1. Análisis Básico 2. Estrategias Algorítmicas 3. Algoritmos y Estructuras de Datos fundamentales 4. Computabilidad y complejidad básica de autómatas 5. Estructuras de Datos Avanzadas y Análisis de Algoritmos

5. Objetivos Generales

- 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. Contribución a los resultados (*Outcomes*)

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

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

5) S.O. Funcionar efectivamente como miembro o líder de un equipo involucrado en actividades apropiadas a la disciplina del programa. (**Usar**)

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

7. Contenido

UNIDAD 1: Análisis Básico (10)

Competencias:

Contenido	**Objetivos Generales**
• 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. • Teorema Maestro y Árboles Recursivos.	• 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 diferentes 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: Kleinberg and Tardos (2005), Dasgupta, Papadimitriou, and Vazirani (2006), Cormen et al. (2009), Sedgewick and Flajolet (2013), Knuth (1997)

UNIDAD 2: Estrategias Algorítmicas (30)	
Competencias:	
Contenido	**Objetivos Generales**
• 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] • Usa un algoritmo de divide-y-vencerás para resolver un determinado problema [Evaluar] • Usa programación dinámica para resolver un problema determinado [Evaluar] • Determina el enfoque algorítmico adecuado para un problema [Evaluar]
Lecturas: Kleinberg and Tardos (2005), Dasgupta, Papadimitriou, and Vazirani (2006), Cormen et al. (2009), Alsuwaiyel (1999)	

UNIDAD 3: Algoritmos y Estructuras de Datos fundamentales (10)

Competencias:

Contenido	Objetivos Generales

• 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:
  – Problema de corte máximo y mínimo
  – Busqueda local
	• 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 propiedad del heap y el uso de heaps como una implementación de colas de prioridad [Evaluar]
	• Resolver problemas usando algoritmos de grafos, incluyendo camino más corto de una sola fuente y camino más corto de todos los pares, y como mínimo un algoritmo de arbol de expansion minima [Evaluar]

Lecturas: Kleinberg and Tardos (2005), Dasgupta, Papadimitriou, and Vazirani (2006), Cormen et al. (2009), Sedgewick and Wayne (2011), Goodrich and Tamassia (2009)

UNIDAD 4: Computabilidad y complejidad básica de autómatas (2)

Competencias:

Contenido	Objetivos Generales
• 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: Kleinberg and Tardos (2005), Dasgupta, Papadimitriou, and Vazirani (2006), Cormen et al. (2009)

UNIDAD 5: Estructuras de Datos Avanzadas y Análisis de Algoritmos (8)	
Competencias:	
Contenido	**Objetivos Generales**
• 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: Kleinberg and Tardos (2005), Dasgupta, Papadimitriou, and Vazirani (2006), Cormen et al. (2009), Tarjan (1983), Rawlins (1992)	

8. Metodología

1. El profesor del curso presentará clases teóricas de los temas señalados en el programa propiciando la intervención de los alumnos.
2. El profesor del curso presentará demostraciones para fundamentar clases teóricas.
3. El profesor y los alumnos realizarán prácticas
4. 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

Alsuwaiyel, H. (1999). *Algorithms: Design Techniques and Analysis*. World Scientific. ISBN: 9789810237400.

Cormen, Thomas H. et al. (2009). *Introduction to Algorithms, Third Edition*. 3rd. The MIT Press. ISBN: 0262033844.

Dasgupta, S., C. Papadimitriou, and U. Vazirani (2006). *Algorithms*. McGraw-Hill Education. ISBN: 9780073523408.

Goodrich, Michael T. and Roberto Tamassia (2009). *Algorithm Design: Foundations, Analysis and Internet Examples*. 2nd. John Wiley & Sons, Inc. ISBN: 0470088540, 9780470088548.

Kleinberg, Jon and Eva Tardos (2005). *Algorithm Design*. Addison-Wesley Longman Publishing Co., Inc. ISBN: 0321295358.

Knuth, D.E. (1997). *The Art of Computer Programming: Fundamental algorithms*. v. 1. Addison-Wesley. ISBN: 9780201896831.

Rawlins, G.J.E. (1992). *Compared to What?: An Introduction to the Analysis of Algorithms*. Computer Science Press. ISBN: 9780716782438.

Sedgewick, R. and P. Flajolet (2013). *An Introduction to the Analysis of Algorithms*. Pearson Education. ISBN: 9780133373486.

Sedgewick, R. and K. Wayne (2011). *Algorithms*. Pearson Education. ISBN: 9780132762564.

Tarjan, Robert Endre (1983). *Data Structures and Network Algorithms*. Society for Industrial and Applied Mathematics. ISBN: 0-89871-187-8.