CIENCIA DE LA COMPUTACIÓN

ANÁLISIS Y DISEÑO DE ALGORITMOS

4 CRÉDITOS

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

CE2A

ÍNDICE

UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA
SILABO 2020-2

1. ASIGNATURA

C22102 - Análisis y Diseño de Algoritmos

2. DATOS GENERALES

2.1 Créditos: cuatro (4) créditos
2.2 Horas de teoría: dos (2) semanales
2.3 Horas de práctica: cuatro (4) semanales
2.4 Duración del período: dieciséis (16) semanas
2.5. Condición:
- Obligatorio para Ciencia de la Computación Modalidad: Virtual

2.6. Modalidad: Virtual
2.7. Requisitos:
- CS2100- Algoritmos y Estructura de Datos

3. PROFESORES

3.1. Profesor coordinador del curso

Juan Gutiérrez (jgutierreza@utec.edu.pe)
Horario de atención: previa coordinación con el profesor TP

3.2. Profesor(es) instructor(es) del curso

Ronald Ubel Adolfo Gonzales Vega, (rgonzalesveg@utec.edu.pe)
Horario de atención: previa coordinación con el profesor

4. INTRODUCCIÓN AL CURSO

Un algoritmo es un conjunto de reglas o instrucciones que permiten resolver un problema, el estudio teórico de algunos aspectos como su rendimiento, su tiempo de ejecución o su espacio utilizado nos permite analizar y definir qué tan apropiado es un determinado algoritmo para resolver un problema en específico. Estos algoritmos detrás de las soluciones a diversos problemas han promovido el desarrollo de las tecnologías que se usan día a día en diversos campos como economía, geología, exploración espacial, medicina, biología, entre otros.

Dada la importancia del estudio teórico podemos definir la Ciencia de la Computación como el estudio de algoritmos. En este curso se presenta las técnicas usadas en el análisis y diseño de algoritmos con el propósito de

aprender y aplicar los principios fundamentales para el diseño e implementación de métodos computacionales en la resolución de problemas.

Este curso es de vital importancia para el desarrollo profesional del estudiante, ya que proporciona conceptos clave para la Ciencia de la Computación, los cuales son tratados de manera mucho más formal que en otros cursos. Con estos conceptos el estudiante será capaz de desarrollar software mucho más sólido en la industria, ya que conocerá a fondo los principios sobre el cual descansa la Ciencia de la Computación.

5. OBJETIVOS

Sesión 1. Desarrollar la capacidad para evaluar la complejidad y calidad de algoritmos propuestos para un determinado problema.
Sesión 2. Conocer los tipos de algoritmos más representativos para resolver distintos tipos de problemas tratados en computación.
Sesión 3. Desarrollar la habilidad para resolver problemas usando los principios de diseño de algoritmos aprendidos.
Sesión 4. Responder las siguientes preguntas cuando un nuevo algoritmo es presentado: ¿Qué tan eficiente es su rendimiento?, ¿Hay alguna mejor forma de solucionar el problema?

6. COMPETENCIAS

- a2: aplicar conocimientos de ciencias (nivel 2)
  El estudiante resuelve un set de problemas donde aplica lo aprendido de manera teórica: diseñar un algoritmo, analizar su tiempo de ejecución, demostrar mediante recurrencias o teorema maestro, analizar la correctitud del algoritmo diseñado.

- b2: analizar información (nivel 2)
  El estudiante identifica los requerimientos computacionales apropiados para el diseño de un algoritmo, con lo cual consigue diseñar un algoritmo eficiente para el problema y demostrar su correctitud

- d1: trabajar en equipo (nivel 2)
  El estudiante participa y se comunica grupalmente en la resolución de problemas

7. RESULTADOS DE APRENDIZAJE

Al final del curso Análisis y diseño de algoritmos se espera:

RA1. Que el estudiante sea capaz de aplicar conocimiento matemático para resolver problemas computacionales.

RA2. Que el estudiante sea capaz de diseñar y desarrollar experimentos, así como analizar e interpretar datos para un determinado problema.

8. TEMAS

1. **Introducción**
   - 1.1. Algoritmos, problema de ordenación, comparación intuitiva del tiempo de ejecución entre dos algoritmos
   - 1.2. Sumatorias, fórmulas y propiedades básicas: linearidad, series aritméticas, suma de cuadrados y cubos, serie geométrica, serie armónica, series telescópicas
   - 1.3. Acotando sumatorias: inducción, acotando términos, dividiendo sumatorias
   - 1.4. Conceptos básicos de grafos y árboles
   - 1.5. Insertion sort: correctitud y análisis
   - 1.6. Crecimiento de funciones: notación Teta, notación O-grande, notación Gamma, notación Omega, notación o-pequeño

2. **División y conquista**
   - **2.1.** Ingredientes básicos del método de divisón y conquista
   - **2.2.** Análisis del Mergesort: correctitud y análisis del tiempo de ejecución
   - **2.3.** Recurrencias: resolución por expansión, verificación por inducción, teorema maestro
   - **2.4.** Problema del subarreglo máximo
   - **2.5.** Multiplicación de números naturales (algoritmo de Karatusuba)
   - **2.6.** Conteo de inversiones

3. **Heap**
   - 3.1. Definición de heap, propiedades básicas de heap
   - 3.2. Construcción de max-heap: max-heapify y build-max-heap
   - 3.3. El algoritmo heapsort
   - 3.4. Filas de prioridades

4. **Análisis Probabilístico y Quicksort**
   - 4.1. Repaso de probabilidades:variable aleatoria, función de probabilidad, valor esperado
   - 4.2. El problema de contratación (Hiring problem)
   - 4.3. El algoritmo Quicksort. Análisis de tiempo del Quicksort: peor caso, mejor caso y caso promedio

5. **Programación Dinámica**
   - 5.1. Conceptos básicos de Programación dinámica y memoización
   - 5.2. Números de Fibonacci
   - 5.3. Coeficientes binomiales
   - 5.4. Intervalos disjuntos
   - 5.5. Subsecuencia creciente máxima
   - 5.6. Subset sum y mochila
   - 5.7. Partición lineal justa

6. **Algoritmos voraces (Greedy)**
   - 6.1. Conceptos básicos de la estrategia voraz
   - 6.2. Técnicas para demostrar la correctitud de un algoritmo voraz

6.3. Intervalos disjuntos
6.4. Planificación de tareas
6.5. Problema de la mochila fraccionaria
6.6. Código de Huffman: implementación recursiva e implementación con fila de prioridades

7. Análisis amortizado
7.1. Problema de operaciones en pilas. Problema del contador binario
7.2. Análisis agregado
7.3. Método de recargas

8. Algoritmos en grafos
8.1. Representaciones: matrices de adyacencia y listas de adyacencia
8.2. Búsqueda en profundidad
8.3. Búsqueda en largura
8.4. Caminos en grafos: algoritmo de Dijkstra, algoritmo de Bellman-Ford, algoritmo de Floyd-Warshall
8.5. Flujos en grafos

9. Complejidad computacional
9.1. Reducciones polinomiales. Definición y ejemplos.
9.2. La clase NP. La clase NP-completo. Teorema de Cook.
9.3. Estrategia para demostrar que un problema es NP-Completo

9. PLAN DE TRABAJO

9.1. Metodología

Se fomenta la participación individual y en equipo para exponer las ideas de los estudiantes, motivándolos con puntos adicionales en las diferentes etapas de la evaluación del curso. A lo largo del curso se proporcionan diferentes lecturas, las cuales serán evaluadas. El uso de herramientas online permitirá al alumno acceder a la información del curso, interactuar fuera del aula con el profesor y con los otros estudiantes.

9.2. Sesiones de teoría

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

9.3. Sesiones de práctica (laboratorio o taller)

Las sesiones prácticas/laboratorio se desarrollarán a través de una metodología activa generando el aprendizaje práctico por parte del estudiante.
Las sesiones de práctica se caracterizan por el desarrollo de problemas tanto de demostración de propiedades como de diseño de nuevos algoritmos.

Para verificar que los alumnos hayan alcanzado el logro planteado para cada una de las unidades de aprendizaje, realizarán actividades que les permita aplicar los conocimientos adquiridos durante las sesiones de teoría y se les propondrá retos para que permitan evaluar el desempeño de los alumnos.

10. SISTEMA DE EVALUACIÓN

	TEORÍA	**PRÁCTICA Y/O LABORATORIO**
EVALUACIÓN *La ponderación de la evaluación se hará si ambas partes están aprobadas	0.15 (E_1) + 0.15 (E_2) 4 Exámenes (uno por módulo) (68%)	0.15 (C_1) + 0.15 (C_2) + 0.20 (P_1) + 0.20 (P_2) 1 Evaluación continua(12%) 1 Proyecto (20%)
	30%	70%
	100%	

Donde:

P: Proyecto de Laboratorio (2)
- P_1 (Semanas 1 - 7)
- P_2 (Semanas 8 - 15)

E: Examen Teórico (2)
- E_1 (Semanas 1 - 7)
- E_2 (Semanas 8 - 15)

C: Evaluación Continua (2)
- C_1 (Semanas 1 - 7)
- C_2 (Semanas 8 - 15)

Las rúbricas que permitirán medir las actividades más significativas del curso y que, además se relacionan con la evaluación de las competencias del estudiante son: enlace

11. REFERENCIAS BIBLIOGRÁFICAS

- Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C. (2009). *Introduction to algorithms*. MIT press.
- Graham, R. L., Knuth, D. E., & Patashnik, O. (1994). *Concrete mathematics: A foundation for computer science*. Reading, Mass: Addison-Wesley.
- Sedgewick, R., & Flajolet, P. (2013). *An introduction to the analysis of algorithms*. Pearson Education India.

- Sedgewick, R., & Wayne, K. (2011). *Algorithms*. Addison-wesley professional.
- Knuth, D. E., & Knuth, D. F. (1973). *The art of computer programming: Fundamental algorithms* (Vol. 3). Addison Wesley Publishing Company.