CIENCIA DE LA COMPUTACIÓN

PROGRAMACIÓN COMPETITIVA

4 CRÉDITOS

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

CE2A

ÍNDICE

UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA
SILABO 2020-2

1. ASIGNATURA

CS3101 - Programación Competitiva

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 (Eliminar si no aplica y ajustar al índice)

2.6 Modalidad: Virtual
2.7 Requisitos:

-CS2102-Análisis y Diseño de Algoritmos

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

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

4. INTRODUCCIÓN AL CURSO

La Programación Competitiva combina retos para solucionar problemas con el añadido de poder competir con otras personas. Enseña a los participantes a pensar más rápido y desarrollar habilidades para resolver problemas. Este curso enseñará la resolución de problemas algorítmicos de manera rápida combinando la teoría de algoritmos y estructuras de datos con la práctica la solución de los problemas.

Este curso enseña las técnicas y habilidades necesarias para resolver problemas de concursos de programación competitiva como los que aparecen en el ACM ICPC, Codeforces y Topcoder. También aprenderá a pensar en algoritmos y estructuras de datos de una manera más audaz, porque muchos de los problemas requieren idear un nuevo algoritmo, basado en los algoritmos

clásicos que se conocen. Estas habilidades serán de gran valor en sus entrevistas de trabajo y carrera profesional.

5. OBJETIVOS

Sesión 1. Aplicar técnicas algorítmicas para la resolución de problemas de programación competitiva.
Sesión 2. Conocer las principales estructuras de datos para problemas de programación competitiva
Sesión 3. Investigar la posibilidad de crear nuevas técnicas o algoritmos en problemas reales.

6. COMPETENCIAS

Las competencias que se van a trabajar en este curso son:

- a2: aplicar conocimientos de ciencias (nivel 2)
  El estudiante resuelve un set de problemas donde aplica lo aprendido de los fundamentos y principios básicos de algoritmos y estructura de datos para programación competitiva.

- b2: analizar información (nivel 2)
  El estudiante identifica los requerimientos computacionales apropiados para la solución de problemas de programación competitiva

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

7. RESULTADOS DE APRENDIZAJE

Al final del curso de Programación Competitiva se espera,

RA1. Que el estudiante sea capaz de resolver ejercicios de programación competitiva, demostrando dominio del tema, orden y coherencia en los resultados.

RA 2. Que el estudiante sea capaz de relacionar los conceptos de algoritmos y estructuras de datos, reconociendo su importancia en la construcción de algoritmos eficientes

RA 3. Que el estudiante sea capaz de realizar con fluidez la codificación de algoritmos en el lenguaje C++, así como capacidad para identificar bugs rápidamente

8. TEMAS

1. **Introducción**
   - 1.1. Características de C++: entrada y salida, tipos de datos, aritmética modular
   - 1.2. Algoritmos recursivos: subconjuntos, permutaciones y backtracking
   - 1.3. Manipulación de bits: operaciones básicas, representación de conjuntos

2. **Búsquedas y ordenamientos**
   - 2.1. Algoritmos de ordenamientos: Bubble sort, Merge sort, Counting sort.
   - 2.2. Sweep line, Sliding Window
   - 2.3. Búsqueda binaria, Búsqueda ternaria

3. **Cadenas**
   - 3.1. Árboles de sufijos: prefix doubling method, Finding patterns, LCP Arrays
   - 3.2. Algoritmo KMP
   - 3.3. Algoritmo Z
   - 3.4. Hashing

4. **Consultas de rango (Range Queries)**
   - 4.1. Consultas en arreglos estáticos: mínimo y suma
   - 4.2. Binary Indexed Tree (Fenwick tree),
   - 4.3. Segment trees, Lazy propagation
   - 4.4. SQRT Decomposition, Sparse Table

5. **Algoritmos en grafos**
   - 5.1. Conceptos básicos de grafos
   - 5.2. Búsquedas en grafos: BFS, DFS, DFS-Trees: puentes, articulaciones, biconectividad, subgrafos eulerianos
   - 5.3. Caminos mínimos: Algoritmo de Bellman-Ford, Algoritmo de Dijkstra, Algoritmo de Floyd-Warshall
   - 5.4. Grafos dirigidos acíclicos: ordenación topológica, programación dinámica
   - 5.5. Árboles generador mínimos: Algoritmo de Kruskal, Estructura Union-Find, Algoritmo de Prim
   - 5.6. Grafos dirigidos fuertemente conexos: Algoritmo de Kosaraju, 2SAT
   - 5.7. Fujo máximo en grafos: Algoritmo de Ford-Fulkerson, Caminos disjuntos, Emparejamiento máximo, Aplicaciones

6. **Temas avanzados**
   - 6.1. Optimizaciones en programación dinámica: convex hull trick, divide and conquer, Knuth Optimization
   - 6.2. Treaps
   - 6.3. Min-Queue & Min-Stack

9. PLAN DE TRABAJO

9.1 Metodología

Se aplicarán metodologías activas como aula clase invertida y aprendizaje práctico para fomentar la participación individual y en equipo, y además para exponer las ideas por parte de los estudiantes, motivándolos con puntos

adicionales en las diferentes etapas de la evaluación del curso. 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 teóricas serán desarrolladas bajo la estructura de clase invertida, lo que significa que el estudiante es responsable por su aprendizaje y preparación para la sesión de clase. Antes de cada clase, los estudiantes tendrán asignado un problema (indicada por el docente) y sobre dicho ejercicio se desarrollará el tema de la clase.

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 de programación competitiva usando jueces virtuales, participando en concursos virtuales los alumnos podrán medir sus conocimientos adquiridos en la teoría.

10. SISTEMA DE EVALUACIÓN

	TEORÍA	**PRÁCTICA Y/O LABORATORIO**
EVALUACIÓN	1. Evaluación continua (C1) (50%)	1. Evaluación continua (C2) (50%)
	50%	50%
	100%	

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

- Antti Laaksonen. Guide to Competitive Programming: Learning and Improving Algorithms Through Contests. Springer 2018
- Steven Halilm, Felix Halim. Competitive Programming 3: The New Lower Bound of Programming Contests, Volumen3. Lulu.com, 2013

- Steven S. Skiena, Miguel A. Revilla. Programming Challenges: The Programming Contest Training Manual. Springer Science & Business Media, 2006
- Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C. (2009). *Introduction to algorithms*. MIT press.