CIENCIA DE LA COMPUTACIÓN

PROGRAMACIÓN COMPETITIVA

4 CRÉDITOS

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

CE2A

REGLAS INTEGRIDAD ACADÉMICA

Todo estudiante matriculado en una asignatura de la Universidad de Ingeniería y Tecnología tiene la obligación de conocer y cumplir las reglas de integridad académica, cuya lista a continuación es de carácter enunciativo y no limitativo, ya que el/la docente podrá dar mayores indicaciones:

1. La copia y el plagio son dos infracciones de magnitud muy grave en la Universidad de Ingeniería y Tecnología (UTEC) conforme a lo establecido en el Reglamento de Disciplina de los Estudiantes. Tienen una sanción desde 2 semestres de suspensión hasta la expulsión.
2. Si se identifica la copia o plagio en evaluaciones individuales, el/la docente puede proceder a anular la evaluación.
3. Si la evaluación es personal o grupal-individual, la interacción entre equipos o compañeros se considera copia o plagio, según corresponda. Si la evaluación calificada no indica que es grupal, se presume que es individual.
4. La copia, plagio, el engaño y cualquier forma de colaboración no autorizada no serán tolerados y serán tratados de acuerdo con las políticas y reglamentos de la UTEC, implicando consecuencias académicas y sanciones disciplinarias.
5. Aunque se alienta a los estudiantes a discutir las tareas y trabajar juntos para desarrollar una comprensión más profunda de los temas presentados en este curso, no se permite la presentación del trabajo o las ideas de otros como propios. No se permite el plagio de archivos informáticos, códigos, documentos o dibujos.
6. Si el trabajo de dos o más estudiantes es sospechosamente similar, se puede aplicar una sanción académica a todos los estudiantes, sin importar si es el estudiante que proveyó la información o es quien recibió la ayuda indebida. En ese sentido, se recomienda no proveer el desarrollo de sus evaluaciones a otros compañeros ni por motivos de orientación, dado que ello será considerado participación en copia.
7. El uso de teléfonos celulares, aplicaciones que permitan la comunicación o cualquier otro tipo de medios de interacción entre estudiantes está prohibido durante las evaluaciones o exámenes, salvo que el/la docente indique lo contrario de manera expresa. Es irrelevante la razón del uso del dispositivo.
8. En caso exista algún problema de internet durante la evaluación, comunicarse con el/la docente utilizando el protocolo establecido. No comunicarse con los compañeros dado que eso generará una presunción de copia.
9. Se prohíbe tomar prestadas calculadoras o cualquier tipo de material de otro estudiante durante una evaluación, salvo que el/la docente indique lo contrario.
10. Si el/la docente encuentra indicios de obtención indebida de información, lo que también implica no cumplir con las reglas de la evaluación, tiene la potestad de anular la prueba, advertir al estudiante y citarlo con su Director de Carrera. Si el estudiante no asiste a la citación, podrá ser reportado para proceder con el respectivo procedimiento disciplinario. Una segunda advertencia será reportada para el inicio del procedimiento disciplinario correspondiente.
11. Se recomienda al estudiante estar atento/a a los datos de su evaluación. La consignación de datos que no correspondan a su evaluación será considerado indicio concluyente de copia.

ÍNDICE

UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA
SILABO 2023.1

1. ASIGNATURA

CS3101 - Programación Competitiva

2. DATOS GENERALES

2.1 Ciclo: 6°
2.2 **Créditos:** Cuatro (4) créditos
2.3 **Horas de teoría:** Dos (2) semanales
2.4 **Horas de práctica:** Cuatro (4) semanales
2.5 **Duración del período:** dieciséis (16) semanas
2.6 Condición:
- Obligatorio para Ciencia de la Computación

2.7 **Modalidad:** Presencial
2.8 **Requisitos**:
-CS2102-Análisis y Diseño de Algoritmos

3. 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.

4. OBJETIVOS

Sesión 1. Repasar sintaxis de C++, dominar técnicas recursivas de backtracking
Sesión 2. Conocer los conceptos básicos de teoría de números. Aplicar la criba de Eratóstenes en problemas específicos. Aplicar conceptos básicos de combinatoria en problemas específicos

Sesión 3. Reconocer cuándo un problema requiere de algoritmo voraz y saber implementarlo. Aplicar técnicas de sliding window y sweep line. Dominar búsqueda binaria y sus implementaciones. Saber aplicar búsqueda ternaria y sus ventajas respecto a búsqueda binaria.

Sesión 4. Dominar programación dinámica a nivel básico. Conocer y aplicar técnicas avanzadas de programación dinámica.

Sesión 5. Saber construir un suffix array para responder consultas en cadenas. Dominar técnicas de strings para problemas específicos

Sesión 6: Saber distinguir entre queries estáticas y dinámicas. Conocer cómo se construye un fenwick tree y un segment tree. Saber aplicar dichas estructuras a problemas específicos.

Sesión 7: Dominar BFS y DFS, así como su aplicación a problemas específicos. Aplicar técnicas de programación dinámica en grafos dirigidos acíclicos. Aplicar algoritmos en componentes fuertemente conexos en casos específicos.

5. COMPETENCIAS

Las competencias para los programas de Computación que se van a trabajar en este curso:

- COM1 Analizar un problema computacional complejo y aplicar principios de computación y otras disciplinas relevantes para identificar soluciones.

- COM2 Diseñar, implementar y evaluar una solución computacional para satisfacer un conjunto determinado de requerimientos computacionales en el contexto de la disciplina del programa.

- CCS6 Aplicar la teoría de la ciencia de la computación y los fundamentos de desarrollo de software para producir soluciones basadas en computación. [CS]

6. RESULTADOS DE APRENDIZAJE

Los resultados de aprendizaje (RA) para los estudiantes de Computación son:

- RA1 Argumentar la complejidad algorítmica de un algoritmo eficiente usando las propiedades y técnicas conocidas de análisis de algoritmos.
- RA2 Construir programas que satisfacen las restricciones de tiempo y memoria de un problema dado.
- RA3 Construir algoritmos eficientes para problemas de programación competitiva usando las técnicas algorítmicas más adecuadas al problema.
- RA4 Desarrollar habilidades de adaptación a nuevas técnicas y nuevos desafíos de programación que ocurren en la construcción de software

7. TEMAS

1. **Introducción**
   - **1.1.** Entrada y salida, tipos de datos, aritmética modular
   - **1.2.** Búsqueda completa iterativa
   - **1.3.** Búsqueda completa recursiva: subconjuntos, permutaciones y backtracking

2. **Matemática**
   - 1.1. Teoría de números: números primos, Criba de Eratóstenes, algoritmo de Euclides, exponenciación modular
   - 1.2. Combinatoria: coeficientes binomiales, números de Catalán, lema de Burnside

3. **Búsquedas y ordenamientos**
   - **3.1.** Algoritmos voraces
   - **3.2.** Búsqueda binaria
   - **3.3.** Búsqueda binaria de la respuesta (Binary search the answer)
   - **3.4.** Búsqueda ternaria

4. **Programación dinámica**
   - **4.1.** Problemas clásicos: Monedas, LIS, LCS, Edit Distance, Mochila
   - **4.2.** Programación dinámica usando bitmask (TSP)
   - **4.3.** Optimizaciones en programación dinámica.
   - **4.4.** Uso de recurrencias y matrices

5. **Cadenas**
   - **5.1.** Función prefijo y algoritmo KMP
   - **5.2.** Función Z
   - **5.3.** Hashing
   - **5.4.** Árbol de sufijos (suffix array)

6. **Consultas de rango (Range Queries)**
   - **6.1.** Consultas en arreglos estáticos
   - **6.2.** Sparse Table
   - **6.3.** Binary Indexed Tree (Fenwick tree),
   - **6.4.** Segment tree.

7. **Algoritmos en grafos**
   - **7.1.** Búsquedas en grafos: DFS, BFS.
   - **7.2.** Puentes y articulaciones
   - **7.3.** Grafos dirigidos acíclicos: ordenación topológica y programación dinámica
   - **7.4.** Componentes fuertemente conexas: Algoritmo de Kosaraju y Tarjan
   - **7.5.** Caminos mínimos en grafos: algoritmos de Dijkstra, Bellman Ford y Floyd-Warshall
   - **7.6.** Disjoint set union y spanning trees

8. PLAN DE TRABAJO

8.1 Metodología

En el curso se aplicarán las metodologías de clase invertida, aprendizaje cooperativo, aprendizaje basado en problemas, activa y expositiva. Estas metodologías aumentan el interés del estudiante y promueven su compromiso en el aprendizaje.

8.2 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 (indicado por el docente) y sobre dicho ejercicio se desarrollará el tema de la clase.

8.3 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.

9. SISTEMA DE EVALUACIÓN

	TEORÍA	**PRÁCTICA Y/O LABORATORIO**
EVALUACIÓN	Examen 1 **E1 (10%)** Examen 2 **E2 (10%)**	Evaluación continua **C1** (40%) Evaluación continua **C2** (40%)
	20%	80%
	100%	

Contest – rúbrica

10. 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.