UNIVERSIDAD NACIONAL DE ASUNCIÓN
FACULTAD POLITÉCNICA
LICENCIATURA EN CIENCIAS INFORMÁTICAS
ÉNFASIS EN ANÁLISIS DE SISTEMAS INFORMÁTICOS
PLAN 2009
PROGRAMA DE ESTUDIOS
Resolución N° 17/16/30-00 Acta N° 1004/31/07/2017-ANEXO 01

I. IDENTIFICACIÓN

1.	Asignatura	: Electiva III – Programación Competitiva
2.	Código	: 7.2. B.
3.	Horas semanales	: 5 horas
	3.1. Clases teóricas	: 5 horas
	3.2. Clases prácticas	: ---
4.	Total real de horas disponibles	: 80 horas
	4.1. Clases teóricas	: 80 horas
	4.2. Clases prácticas	: ---

II. JUSTIFICACIÓN

La directiva en programación competitiva es “dado un problema de computación bien conocido, resuélvelo lo más rápido posible”. La programación competitiva se trata de trabajar con problemas que ya fueron resueltos, y por lo tanto, con resolver un problema en este contexto se refiere a empujar todos nuestros conocimientos de computación hasta cierto nivel de manera a producir un programa con código que funcione en un lenguaje de programación. El elemento competitivo aparece en la rapidez requerida para resolver problemas.

La programación competitiva se utiliza para formar programadores altamente entrenados para producir un mejor software y enfrentar problemas de innovación e investigación en el futuro. Este curso servirá para entrenar capacidades avanzadas de programación y resolución de problemas con miras a las competencias de programación de la ACM/ICPC (Association for Computing Machinery International Collegiate Programming Constest). De esta manera se espera mejorar las habilidades de programación integral que toda persona dedicada a las ciencias de la computación, la informática, o cualquiera que utiliza la programación como elemento de trabajo.

III. OBJETIVOS GENERALES

1. Conocer en profundidad las competencias internacionales de programación, y en especial, la ACM/ICPC.
2. Desarrollar habilidades de codificación avanzada en lenguaje C/C++ y Java.
3. Aplicar las técnicas de diseño de algoritmos a problemas de competencia.
4. Preparar al estudiante para competencias internacionales de programación.

IV. OBJETIVOS ESPECÍFICOS

A. Conocimientos

1. Conocer la estructura de la competencia ACM/ICPC.
2. Implementar con solvencia los algoritmos más importantes de las ciencias de la computación.
3. Deducir las diferentes técnicas de diseño de algoritmos.
4. Identificar las formas de generación de código eficiente.

B. Habilidades

1. Adquirir velocidad en la escritura de código C/C++ y Java.
2. Crear tipos de datos abstractos.
3. Desarrollar actitudes para el trabajo en equipo.
4. Desarrollar espíritu de competencia en equipos.

C. Competencias

1. Manejo eficiente de lenguajes de programación C/C++ y Java.
2. Capacidad de creación de equipos de programación competitiva.
3. Organizar competencias de programación.

V. PRE – REQUISITO

- Ingeniería de Software I
- Base de datos III

VI. CONTENIDO

6.1. Unidades programáticas

1. Introducción a la programación competitiva.
2. Estructuras de Datos y librerías.
3. Paradigmas de resolución de problemas.
4. Grafos.

6.2. Desarrollo de las unidades programáticas

1. Introducción a la programación competitiva
- 1.1. Competencias de programación.
- 1.2. Consejos para ser competitivo.
  - 1.2.1. Técnicas de codificación rápida.
  - 1.2.2. Identificación rápida de problemas.
  - 1.2.3. Lenguajes de programación.
  - 1.2.4. Testeo de código.
  - 1.2.5. Trabajo en equipo.
- 1.3. Problemas de competición.
  - 1.3.1. Estructura de una competencia de programación.
  - 1.3.2. Diseño de la Entrada/Salida de los problemas.
- 1.4. Problemas Ad Hoc.

2. Estructuras de Datos y Librerías
- 2.1. Estructuras de datos lineales y librerías.
- 2.2. Estructuras de datos no lineales y librerías.
- 2.3. Estructuras de datos construidas a medida.
  - 2.3.1. Grafos.
  - 2.3.2. Union-Find.
  - 2.3.3. Segment Tree.
  - 2.3.4. Árboles de Fenwick

3. Paradigmas de resolución de problemas.
- 3.1. Búsqueda completa
  - 3.1.1. Búsqueda iterativa.
  - 3.1.2. Búsqueda recursiva.
- 3.2. Divide y Conquistarás.
  - 3.2.1. Aplicaciones de búsqueda binaria.
- 3.3. Algoritmos Greedy.
- 3.4. Programación Dinámica (DP).
  - 3.4.1. Ilustración de DP.
  - 3.4.2. Ejemplos clásicos.
  - 3.4.3. Ejemplos no clásicos.

4. Grafos
- 4.1. Recorridos de grafos.
  - 4.1.1. DFS.
  - 4.1.2. BFS.
  - 4.1.3. Componentes conectados (grafos no dirigidos).
  - 4.1.4. Coloreamiento de componentes conectados.
  - 4.1.5. Ordenamiento topológico (grafos dirigidos acíclicos).
  - 4.1.6. Grafos bipartitos.
  - 4.1.7. Chequeos de propiedades de aristas.
  - 4.1.8. Búsqueda de puntos de articulación y puentes (grafos no dirigidos).
  - 4.1.9. Búsqueda de componentes fuertemente conectados (grafos dirigidos).
- 4.2. Árboles de expansión mínimo.
  - 4.2.1. Algoritmo de Kruskal.
  - 4.2.2. Algoritmo de Prim.
  - 4.2.3. Otras aplicaciones.
- 4.3. Caminos cortos con una sola fuente (SSSP).
  - 4.3.1. SSSP en grafos no ponderados.
  - 4.3.2. SSSP en grafos ponderados.
  - 4.3.3. SSSP en grafos con ciclos negativos.
- 4.4. Todos los caminos cortos de a pares.
  - 4.4.1. Algoritmo de Floyd-Warshall.
  - 4.4.2. Otras aplicaciones.
- 4.5. Flujos.
  - 4.5.1. Algoritmo de Ford-Fulkerson.
  - 4.5.2. Algoritmo de Edmonds-Karp.
  - 4.5.3. Modelamiento con flujos.
  - 4.5.4. Grafos especiales.
    - 4.5.4.1. Grafos dirigidos acíclicos.
    - 4.5.4.2. Árboles.
    - 4.5.4.3. Grafos Eulerianos.
    - 4.5.4.4. Grafos bipartitos.

5. Procesamiento de cadenas.
- 5.1. Procesamiento básico de cadenas.
- 5.2. String Matching.
  - 5.2.1. Algoritmo de Knuth-Morris-Pratt.
  - 5.2.2. Grillas 2D.
- 5.3. Procesamiento de cadenas con programación dinámica.
  - 5.3.1. Alineamiento de cadenas.
  - 5.3.2. Subsecuencia común más larga.
  - 5.3.3. Otras algoritmos basados en DP.
- 5.4. Suffix Trie/Tree/Array
  - 5.4.1. Suffix Trie.
  - 5.4.2. Sufffix Tree.
  - 5.4.3. Suffix Array

6. Geometría Computacional
- 6.1. Objetos básicos con librerías.

6.1.1. 0D: Puntos.
6.1.2. 1D: Líneas.
6.1.3. 2D: Círculos.
6.1.4. 2D: Triángulos.
6.1.5. 2D: Cuadriláteros.
6.2. Algoritmos para polígonos con librerías.
6.2.1. Representación de polígonos.
6.2.2. Perímetro de un polígono.
6.2.3. Área de un polígono.
6.2.4. Convexidad de un polígono.
6.2.5. Verificación de puntos dentro de un polígono.
6.2.6. Envolvente convexa de un conjunto de puntos.

VII. ESTRATEGIAS METODOLÓGICAS

1. Clases magistrales.
2. Prácticas en laboratorio.
3. Utilización de la plataforma virtual de la Facultad para: foros de discusión, tareas individuales y grupales, video con tutoriales, entregas de memorias, etc.

VIII. MEDIOS AUXILIARES

1. Pizarras acrílicas.
2. Marcadores.
3. Borrador de pizarra acrílica.
4. Equipo multimedia.
5. Plataforma virtual "EDUCA".
6. Sala de laboratorio equipada para las prácticas.
   6.1. Computadoras en red.
   6.2. Sistemas operativos Linux, Windows.
   6.3. Acceso a internet.

IX. EVALUACIÓN

Para evaluar la asignatura cada examen consistirá de una competencia de programación entre los estudiantes quienes deberán entregar el código generado a través de la plataforma EDUCA. La calificación estará basada en la solución de un problema y la calidad de la solución.

X. BIBLIOGRAFÍA

- Competitive Programming 3. Steven Halim & Felix Halim. Handbook for ACM ICPC and IOI Contestants.
- Introduction to Algorithms, 3rd Edition. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein.

MATERIALES BIBLIOGRÁFICOS DISPONIBLES EN LA BIBLIOTECA DE LA FACULTAD POLITÉCNICA

- Bonanata, M. (2003). *Programación y algoritmos: aprenda a programar con los lenguajes C y Pascal.* Buenos Aires: MP Ediciones.
- Cairó Battistutti, O. (2003). *Metodología de la programación: algoritmos, diagramas de flujo y programas* (2° ed.). México : Alfaomega.
- Joyanes Aguilar, L. (2003). *Fundamentos de programación: algoritmos, estructuras de datos y objetos* (3° ed.). Madrid : McGraw-Hill.
- Joyanes Aguilar, L. (2008). *Fundamentos de programación: algoritmos, estructura de datos y objetos* (4° ed.). Madrid : McGraw-Hill.

RECURSOS DISPONIBLES A TRAVÉS DE CICCO

- César Liza, Á. (2016). *Algoritmos y su codificación en C++*. Lima: Fondo Editorial UPN.
- Llorens Largo, F. (2002). *Programación : formalización, análisis y reutilización de algoritmos matemáticos*. [Alicante]: Digitalia.
- Rodríguez-Puente, R., & Lazo-Cortés, M. (2017). Búsquedas de caminos mínimos haciendo uso de grafos reducidos. *Ingenieria Industrial*, *38*(1), 32-42.

RECURSOS DISPONIBLES A TRAVÉS DE COLECCIONES MHE

- Corona, N. M. A., & Ancona, V. M. D. L. Á. (2011). *Diseño de algoritmos y su codificación en lenguaje C.* México, D.F., MX: McGraw-Hill Interamericana.