Programa de PROGRAMACIÓN 3

1. NOMBRE DE LA UNIDAD CURRICULAR

Programación 3

2. CRÉDITOS

15 créditos

3. OBJETIVOS DE LA UNIDAD CURRICULAR

Se espera que el estudiante obtenga conocimientos y desarrolle capacidades con los que pueda lograr los siguientes puntos.

1. Describir problemas computacionales clásicos y algoritmos que los resuelven.
2. Analizar un problema computacional y abstraerlo en términos algorítmicos.
3. Describir las técnicas estándar de diseños de algoritmos. Reconocer en un algoritmo dado cuál es la técnica en la que se basa su diseño.
4. Aplicar esas técnicas para diseñar algoritmos que resuelven problemas computacionales. Evaluar cuáles técnicas son adecuadas para cada problema.
5. Demostrar con rigurosidad matemática si un algoritmo resuelve correctamente un problema planteado.
6. Analizar los requerimientos de recursos de cómputo de un algoritmo.
7. Distinguir los conceptos de problema y algoritmo. Analizar rigurosamente problemas computacionales para identificar límites de complejidad de cómputo inherentes a esos problemas y reconocer la clase de complejidad a la que pertenecen.

4. METODOLOGÍA DE ENSEÑANZA

El contenido del temario se discutirá en clases teórico/prácticas, a razón de dos clases por semana de 2 hs. cada una, siguiendo el contenido de la bibliografía recomendada.

Se estiman 11 horas de dedicación semanal adicional de cada estudiante.

Se propondrán ejercicios de práctico, para resolución en domicilio, con el objetivo de afianzar los conocimientos teóricos y contribuir al cumplimiento de los objetivos de la unidad curricular. Asimismo se hará disponible a los estudiantes un cronograma sugerido de avance en la resolución de estos ejercicios.

El equipo docente ofrecerá un espacio de taller, de 2 horas semanales, disponible para todos los estudiantes, donde se podrá trabajar en la resolución de ejercicios, estudio del teórico y consultar dudas, tanto entre compañeros como con docentes.

Dependiendo de que la cantidad de inscriptos al curso lo permita, se ofrecerá una modalidad continua. En ese caso, al inicio del curso cada estudiante podrá elegir participar de una modalidad continua de enseñanza y evaluación. Al inicio del curso se informará el plazo y el mecanismo para registrar la decisión. La modalidad continua consiste en la participación en una instancia de monitoreo semanal, de una hora de duración en la que el docente asignado guiará al estudiante y hará un seguimiento del grado de avance. Algunas de las horas de monitoreo pueden ser destinadas a evaluación del estudiante. La modalidad continua no supone una dedicación horaria adicional, sino una forma más guiada de administrar el trabajo.

5. TEMARIO

1. **Análisis de algoritmos.**

   Análisis de corrección y de complejidad en peor caso y en media.

   Repaso de notación asintótica O, Ω, y Θ.

   Análisis de amortización de tiempo de ejecución.

2. **Problemas y algoritmos fundamentales sobre grafos (puede tener variaciones en cada edición).**

   Estructuras de datos para la representaciones de grafos.

   Exploración de grafos (recorridas BFS y DFS).

   Identificación de componentes conexas y fuertemente conexas.

   Identificación de ciclos.

   Verificación de condición de bipartito.

   Construcción de un orden topológico en grafos dirigidos acíclicos.

   Búsqueda del camino más corto (algoritmos de Dijkstra y Bellman).

   Construcción de árboles de cubrimiento mínimo (algoritmos de Prim y Kruskal). Propiedades de corte y de ciclo.

   Flujo máximo en una red (algoritmo de Ford Fulkerson).

   Independent set, vertex cover.

   Aplicaciones.

3. **Otros problemas y algoritmos clásicos (puede tener variaciones en cada edición).**

   Algoritmos de ordenamiento y selección (por ejemplo, ordenamiento mediante variantes de los algoritmos mergesort, quicksort, heapsort).

   Problemas sobre cadenas de texto (por ejemplo, alineamiento de secuencias).

Problemas geométricos (por ejemplo, par de puntos más cercanos).

Problemas aritméticos (por ejemplo, multiplicación de enteros).

Problemas combinatorios (por ejemplo, problema de la mochila).

Problemas sobre conjuntos (por ejemplo, Union-Find).

Problemas de decisión (por ejemplo, 3-SAT).

4. **Técnicas de diseño de algoritmos.**

   Búsqueda exhaustiva.

   Dividir y conquistar.

   Algoritmos ávidos (Greedy).

   Programación dinámica.

   Algoritmos de mejoramiento iterativo.

   Transformar y conquistar.

5. **Limitaciones inherentes a los problemas**

   Cota inferior asintótica para la complejidad de algoritmos de ordenamiento.

   Clases P y NP.

   La clase de problemas NP-completos.

   Reducción de problemas.

   Técnicas para atacar problemas intratables.

6. BIBLIOGRAFÍA

Tema	Básica	Complementaria
Todos los temas	(1,2,3)	

6.1 Básica

1. Kleinberg, J., & Tardos, E. (2006). Algorithm design. Boston: Pearson/Addison-Wesley. ISBN 0-321-29535-8.
2. Cormen, T. H., & Leiserson, C. E., Rivest, R. L., Stein, C. . (2022). Introduction to algorithms. Cambridge, Mass: MIT Press. ISBN 9780262367509.
3. Brassard, G. & Bratley, P.. (1998). Fundamentos de Algoritmia. Madrid: Prentice Hall. ISBN 84-89660-00-X.

7. CONOCIMIENTOS PREVIOS EXIGIDOS Y RECOMENDADOS

7.1 Conocimientos Previos Exigidos: Conocimientos de programación imperativa. Conocimiento de tipos abstractos de datos (TAD) y estructuras de datos: conjunto, diccionario, cola, pila, cola de prioridad, lista, tabla de dispersión y árbol. Conocimientos de matemática discreta (incluyendo técnicas de demostración matemática, y teoría de grafos).

7.2 Conocimientos Previos Recomendados: Lógica de primer orden.

ANEXO A
Para todas las Carreras

A1) INSTITUTO

Instituto de Computación

A2) CRONOGRAMA TENTATIVO

A2) CRONOGRAMA TENTATIVO

Semana 1	Introducción. Repaso de análisis de algoritmos, TAD, estructuras de datos y fundamentos matemáticos. Problemas representativos.
Semana 2	Análisis de corrección de algoritmos sencillos.
Semana 3	Grafos: conceptos fundamentales, representación, recorridas, identificación de ciclos.
Semana 4	Grafos: algoritmos para reconocer la bipartición, y para obtener ordenamiento topológico.
Semana 5	Algoritmos ávidos. Ejemplos sencillos.
Semana 6	Problemas de camino más corto y árboles de cubrimiento de costo mínimo. Algoritmos de Dijkstra, Prim y Kruskal.
Semana 7	Dividir y conquistar. Variante del Master theorem. Ejemplos: algoritmos para contar inversiones, encontrar puntos más cercanos y multiplicación de enteros (puede variar con cada edición).
Semana 8	Programación dinámica.
Semana 9	Programación dinámica.
Semana 10	Algoritmos de mejoramiento iterativo. Problemas de flujo máximo en redes y de corte de capacidad mínima. Algoritmo de Ford Fulkerson.
Semana 11	Transformar y conquistar.
Semana 12	Cota inferior de algoritmos de ordenamiento. Clases P y NP.
Semana 13	Clases P y NP. Clase de problemas NP-Completos. Reducciones.
Semana 14	Manejo de problemas intratables.
Semana 15	Repaso general.

A3) MODALIDAD DEL CURSO Y PROCEDIMIENTO DE EVALUACIÓN

La unidad curricular se evalúa por medio de dos parciales y, además, para aquellos estudiantes que optan por la modalidad continua, con instancias de evaluación vinculadas a los monitoreos. El formato y la cantidad de estas instancias se comunicará al inicio de cada edición del curso. A modo de ejemplo, las evaluaciones pueden consistir en pruebas escritas o en defensas orales de trabajos realizados en domicilio. La cantidad estará en el entorno de 4.

Los parciales son instancias de evaluación teórico-práctica que se realizan con el objetivo de evaluar los conocimientos globales adquiridos por el estudiante y su capacidad de incorporarlos en la solución de problemas.

A diferencia de los parciales, en la modalidad continua se evalúan el cumplimiento de actividades concretas sugeridas para llegar a esos objetivos, como por ejemplo la resolución de ejercicios de práctico pautados y el estudio de temas teóricos.

Las pruebas parciales representan en conjunto un total de 100 puntos. Las instancias de evaluación de la modalidad continua representan en conjunto un total de 12 puntos.

La cantidad total de puntos acumulados determina alguno de los siguientes resultados:

- Exoneración del examen final: se aprueba la unidad curricular.
- Suficiencia en el curso: habilita a rendir el examen.
- Insuficiencia en el curso: se reprueba el curso.

Se presentan a continuación las condiciones que deben alcanzarse para obtener la exoneración y para obtener la suficiencia en el curso; si no se cumplen estos últimos requisitos el resultado será el de insuficiencia en el curso.

Modalidad continua:

- Requisitos para exoneración:
  - reunir al menos 60 puntos en total (incluyendo evaluaciones en modalidad continua) de un total de 112.
  - reunir al menos 6 puntos en las evaluaciones de la modalidad continua.

- Requisitos para suficiencia en el curso:
  - reunir al menos 25 puntos en total (incluyendo evaluaciones en modalidad continua) de un total de 112.
  - reunir al menos 4 puntos en las evaluaciones de la modalidad continua.

Modalidad en base solo a parciales:

- Requisitos para exoneración:
    - reunir al menos 60 puntos en total de un total de 100.

- Requisitos para suficiencia en el curso:
    - reunir al menos 25 puntos en total de un total de 100.

La nota se determina en función del puntaje obtenido, sobre un total de 112 puntos para los estudiantes que optaron por la modalidad continua y sobre un total de 100 para los que no lo hayan hecho.

A4) CALIDAD DE LIBRE

Esta unidad curricular adhiere a la resolución del consejo sobre condición de libre.

A5) CUPOS DE LA UNIDAD CURRICULAR

No tiene cupos.