FORMULARIO PARA LA PRESENTACIÓN DE PROGRAMAS DE ASIGNATURAS

Año Lectivo: 2026

UNIVERSIDAD NACIONAL DE RÍO CUARTO
FACULTAD DE CIENCIAS EXACTAS, FÍSICO-QUÍMICAS Y NATURALES
DEPARTAMENTO DE COMPUTACIÓN

CARRERA/S: Licenciatura en Ciencias de la Computación

PLAN DE ESTUDIOS: 2024.

ASIGNATURA: Algoritmos y Estructuras de Datos II

CÓDIGO: 3380.

MODALIDAD DE CURSADO: Presencial

DOCENTE RESPONSABLE: Dr. Pablo Ponzio, Profesor Adjunto, Dedicación Exclusiva.

EQUIPO DOCENTE: Lic. Sonia Permigiani, Lic. Gastón Scilingo, Lic. Claudio Dosantos.

RÉGIMEN DE LA ASIGNATURA: Cuatrimestral

UBICACIÓN EN EL PLAN DE ESTUDIO: 2do Año – 1er Cuatrimestre

RÉGIMEN DE CORRELATIVIDADES:

Para cursar:

- Asignaturas aprobadas: Introducción a los Algoritmos (3375).
- Asignaturas regulares: Algoritmos y Estructuras de Datos I (3378), Matemática Discreta (3379).

Para rendir:

- Asignaturas aprobadas: Introducción a los Algoritmos (3375). Algoritmos y Estructuras de Datos I (3378), Matemática Discreta (3379).

CARÁCTER DE LA ASIGNATURA: Obligatoria

CARGA HORARIA TOTAL: 112 horas horas

Teóricas:	**56 hs**	**Prácticas:**	**28hs**	**Teóricas -Prácticas:**	**…. hs**	**Laboratorio:**	**28 hs**

CARGA HORARIA SEMANAL: horas (según el plan de estudio vigente)

Teóricas:	**4 hs**	**Prácticas:**	**2 hs**	**Teóricas -Prácticas:**	**…. hs**	**Laboratorio:**	**2 hs**

1. CONTEXTUALIZACIÓN DE LA ASIGNATURA

La asignatura está en el segundo año de las carreras citadas, el material abordado en la materia provee las herramientas básicas para poder abordar los cursos más avanzados de computación. Los conceptos principales que se abordan en la materia son: el tiempo de ejecución de los programas, y la especificación e implementación eficiente de tipos abstractos de datos, usando estructuras de datos complejas (como por ejemplo árboles y grafos).

2. OBJETIVOS PROPUESTOS

El objetivo esencial de la materia es lograr que los alumnos se familiaricen con las técnicas más importantes de representación de datos, que comprendan los mecanismos de los lenguajes de programación modernos para su implementación, y sepan identificar los problemas prácticos en los cuales pueden emplearse cada una de las representaciones de datos estudiadas.

Otro objetivo relevante de la materia es que los estudiantes aprendan herramientas para analizar el tiempo de ejecución de los programas, y comprendan la importancia de implementar algoritmos eficientes en la práctica.

Finalmente, se espera que los alumnos profundicen sus conocimientos sobre conceptos fundamentales de la programación orientada a objetos (modularización, abstracción, encapsulamiento, etc.).

3. EJES TEMÁTICOS ESTRUCTURANTES DE LA ASIGNATURA Y ESPECIFICACIÓN DE CONTENIDOS

3.1. Contenidos mínimos

Contenidos mínimos: Medidas de tiempo de ejecución de programas. Análisis de tiempo de ejecución en el peor caso de programas iterativos y recursivos. Resolución de ecuaciones de recurrencias. Tasas de crecimiento de funciones de tiempo de ejecución. Representación de datos: funciones de abstracción e invariantes de representación. Especificación de tipos abstractos de datos. Formalización de tipos abstractos de datos como estructuras algebraicas. Encapsulamiento y abstracción. Estructuras de datos avanzadas: árboles binarios de búsqueda, árboles balanceados y tablas hash. Grafos: representación y algoritmos fundamentales.

Objetivos de aprendizaje fundamentales: Con el desarrollo de esta asignatura se espera que el estudiante pueda apropiarse de los conocimientos, y desarrollar habilidades y aptitudes necesarias para: Explicar el uso de las notaciones O grande, omega y theta, para describir la cantidad de trabajo asociado a un algoritmo. Utilizar las notaciones O grande, omega y theta, para dar asintóticas superiores e inferiores, y cotas ajustadas, para el tiempo y el espacio correspondientes a la complejidad de los algoritmos. Determinar la complejidad en tiempo y espacio de algoritmos simples. Deducir relaciones de recurrencia que permitan describir la complejidad de tiempo de algoritmos recursivos. Resolver relaciones de recurrencia elementales. Razonar sobre la correcta implementación de tipos abstractos de datos. Utilizar adecuadamente estructuras de datos simples y avanzadas en la implementación de tipos de datos. Comprender las

características de diferentes alternativas en la implementación de tipos de datos, y poder elegir la que mejor se ajuste a cada instancia de aplicación.

3.2. Ejes temáticos o unidades

Programación orientada a objetos. Testing de Programas Orientados a Objetos. Clases y objetos. Modularización. Abstracción. Encapsulamiento. Diseño orientado a objetos. Cohesión y Acoplamiento. Principio abierto-cerrado. Herencia y polimorfismo. Genericidad.

La noción de tipos abstractos de datos. Representación de datos. Funciones de abstracción e invariantes de representación. Especificación e implementación de tipos abstractos de datos en lenguajes orientados a objetos. Especificación e implementación de tipos abstractos de datos en lenguajes funcionales. Corrección y completitud. Revisión de implementaciones clásicas de tipos abstractos de datos estructurados lineales. Listas, pilas y colas, y sus implementaciones clásicas. Comparación de las implementaciones clásicas de tipos abstractos de datos lineales.

Medidas de tiempo de ejecución de programas. Análisis de tiempo de ejecución en caso promedio y peor caso. Análisis de tiempo de ejecución de algoritmos iterativos y recursivos. Resolución de ecuaciones de recurrencias. Tasas de crecimiento de funciones de tiempo de ejecución. Análisis empírico del tiempo de ejecución de programas. Algoritmos avanzados de sorting: QuickSort y MergeSort.

Tipos abstractos de datos más avanzados. Los tipos abstractos de datos Conjunto y Diccionario. Diccionarios y su implementación. Árboles como estructuras de datos. Recorridos sobre árboles (inorder, preorder, postorder). Árboles binarios de búsqueda. AVLs y árboles 2-3. Comparación de las implementaciones clásicas de diccionarios de acuerdo a la eficiencia de sus operaciones. Estructuras de datos avanzadas. Heaps y sus aplicaciones.

El tipo abstracto de datos Grafo. Implementaciones clásicas de grafos (matrices de adyacencias y listas de adyacencias). Grafos dirigidos y no dirigidos. Grafos con costos. Aplicaciones del tipo abstracto de datos Grafo. Algoritmos de recorrido/visita de grafos: recorridos "primero en profundidad" y "primero a lo ancho". Recorridos en orden, preorden y postorden.

4. ACTIVIDADES A DESARROLLAR

CLASES TEÓRICAS: 4 horas semanales de clases teóricas, divididas en dos clases de dos horas cada una; en estas clases se les enseña a los estudiantes la teoría básica de la materia.

CLASES PRÁCTICAS: 2 horas semanales de clases prácticas y de laboratorio, estas clases permiten a los estudiantes afianzar los conocimientos teóricos mediante la aplicación de la teoría a problemas medianamente complejos.

CLASES DE TRABAJOS PRÁCTICOS DE LABORATORIO: 2 horas semanales de laboratorios, en estas clases los estudiantes implementan en lenguajes de programación los conceptos vistos en los teóricos y prácticos.

HORARIOS DE CONSULTA: Cada docente provee horas de consulta adicionales para los prácticos, teóricos y laboratorios, estas son dos horas semanales adicionales.

5. PROGRAMAS Y/O PROYECTOS PEDAGÓGICOS INNOVADORES E INCLUSIVOS

6. CRONOGRAMA TENTATIVO DE CLASES E INSTANCIAS EVALUATIVAS

Semana	Actividad: tipo y descripción*	Exámenes	Dia	Mes
1	Teórico: Especificación e implementación de tipos de datos en Java		10	3
1	Teórico: Testing y debugging.		11	3
1	Práctico: Especificación e implementación de tipos de datos en Java. Testing y debugging		12	3
1	Práctico: Especificación e implementación de tipos de datos en Java. Testing y debugging		13	3
2	Teórico: Abstracción y Modularización.		17	3
2	Teórico: Tipos Abstractos de Datos - Implementaciones de secuencias, sets y maps predefinidos en Java.		18	3
2	Práctico: Abstracción y Modularización - Tipos Abstractos de Datos - Implementaciones de secuencias, sets y maps predefinidos en Java.		19	3
2	Práctico: Abstracción y Modularización - Tipos Abstractos de Datos - Implementaciones de secuencias, sets y maps predefinidos en Java.		20	3
3	Feriado		24	3
3	Teórico: Herencia y polimorfismo		25	3
3	Práctico: Herencia y manejo de errores		26	3
3	Práctico: Herencia y manejo de errores		27	3
4	Teórico: Herencia y polimorfismo - Manejo de errores		31	3
4	Teórico: Implementaciones de TADs básicos en Java (pilas, colas, listas) - Nociones de corrección - Función de abstracción - Invariante de representación - Testing de TADs		1	4
4	Feriado		2	4
4	Feriado		3	4
5	Teórico: Implementaciones de TADs básicos en Java (pilas, colas, listas) - Nociones de corrección - Función de abstracción - Invariante de representación - Testing de TADs		7	4
5	Teórico: Programación funcional - Haskell		8	4
5	Práctico: Implementaciones de TADs básicos en Java (pilas, colas, listas), y nociones de corrección		9	4
5	Práctico: Implementaciones de TADs básicos en Java (pilas, colas, listas), y nociones de corrección		10	4

6	Teórico: Implementaciones de TADs en programación funcional		14	4
6	Teórico: Teoría de corrección de TADs: Álgebras - Función de abstracción - Invariante de representación		15	4
6	Práctico: Haskell - TADs en Haskell		16	4
6	Práctico: Haskell - TADs en Haskell		17	4
7	Teórico: Teoría de corrección de TADs: Álgebras - Función de abstracción - Invariante de representación		21	4
7	Teórico: Tiempo de Ejecución para programas imperativos		22	4
7	Práctico: Álgebras - Función de abstracción - Invariante de representación		23	4
7	Práctico: Álgebras - Función de abstracción - Invariante de representación		24	4
8	Teórico: Tiempo de Ejecución para programas imperativos		28	4
8	Teórico: Tiempo de Ejecución para Programas recursivos		29	4
8	Práctico: Álgebras - Función de abstracción - Invariante de representación		30	4
8	Feriado		1	5
9	Teórico: Algoritmos avanzados de ordenamiento y búsqueda		5	5
9	Teórico: Árboles		6	5
9	Práctico: Ordenamiento, búsqueda y tiempo de ejecución		7	5
9	Práctico: Ordenamiento, búsqueda y tiempo de ejecución		8	5
10	Teórico: Árboles Binarios de Búsqueda		12	5
10	Teórico: Árboles balanceados: AVLs		13	5
10	Práctico: Árboles, Árboles Binarios de Búsqueda		14	5
10	Práctico: Árboles, Árboles Binarios de Búsqueda	Primer parcial	15	5
11	Teórico: Árboles balanceados: Heaps		19	5
11	Teórico: Grafos: Implementaciones Básicas		20	5
11	Práctico: Árboles balanceados: AVLs, Heaps		21	5
11	Práctico: Árboles balanceados: AVLs, Heaps	Recup. primer parcial	22	5
12	Teórico: Grafos: Algoritmos de recorridos	Presentación TP	26	5
12	Teórico: Grafos: Algoritmos de recorridos		27	5
12	Práctico: Grafos		28	5
12	Práctico: Grafos		29	5
13	Teórico: Hashing		2	6
13	Teórico: Disjoint Sets		3	6
13	Práctico: Grafos		4	6
13	Práctico: Grafos	Segundo parcial	5	6
14	Consulta parciales y TP		9	6

14	Consulta parciales y TP	Entrega TP	10	6
14	Práctico: Hashing y Disjoint Sets		11	6
14	Práctico: Hashing y Disjoint Sets	Recup. segundo parcial	12	6

*Teóricos, teóricos-prácticos, trabajos de laboratorios, salidas a campo, seminarios, talleres, coloquios, instancias evaluativas, consultas grupales y/o individuales, otras.

7. BIBLIOGRAFÍA

7.1. Bibliografía obligatoria y de consulta

B. Liskov, J. Guttag. "Program Development in Java - Abstraction, Specification, and Object-Oriented Design". Addison-Wesley. 2001.

D. Barnes, M. Kölling. "Objects First with Java A Practical Introduction using BlueJ". Sixth Edition. Pearson. 2016.

Robert Sedgewick, Kevin Wayne. "Algorithms (Fourth edition)". Addison-Wesley 2016.

T.Cormen, C. Leiserson, R.Rivest. C.Stein, “Introduction to Algorithms, Fourth Edition”, The MIT Press, 2022.

Mark Allen Weiss, " Data Structures and Algorithm Analysis in Java", Addison-Wesley, 2010.

7.2. Plataformas/herramientas virtuales; materiales audiovisuales, enlaces, otros.

Se utilizará Google Classroom como repositorio de materiales, apuntes, presentaciones, videos, etc. Se utilizará la aplicación de mensajería instantánea Slack, y los repositorios para código de programación GitHub. Por el cual, se dejará disponible código de programación para que los alumnos puedan acceder al mismo, leer el código existente y extenderlo/modificarlo.

8. DÍA Y HORARIOS DE CLASES

Clases Teóricas: Martes y Miércoles de 16 a 18 hs.

Prácticas/Laboratorio:

Comisión 1: Martes de 10 a 12 hs. y Jueves de 8 a 10 hs.

Comisión 2: Miércoles de 12 a 14 hs. y Jueves de 14 a 16 hs.

Comisión 3: Martes de 8 a 10 hs. y Jueves de 12 a 14 hs.

9. DÍA Y HORARIO DE CLASES DE CONSULTAS

Serán acordadas con los estudiantes.

10. REQUISITOS PARA OBTENER LA REGULARIDAD Y LA PROMOCIÓN

Para regularizar la materia se deben aprobar los dos parciales o sus respectivos recuperatorios. Además, los estudiantes deberán aprobar un trabajo práctico integrador de programación, que se deberá resolver en un plazo de 15 días. La aprobación del trabajo práctico es fundamental, ya que evalúa la capacidad de los estudiantes de aplicar los conocimientos adquiridos en la materia a un problema real de programación de complejidad media.

La materia no es promocionable.

11. CARACTERÍSTICAS, MODALIDAD Y CRITERIOS DE LAS INSTANCIAS EVALUATIVAS

Los parciales se desarrollarán de manera presencial. De la misma forma se desarrollarán los recuperatorios. El trabajo práctico de programación se deberá entregar en los plazos establecidos (15 días).

El examen final para alumnos regulares se llevará a cabo de manera escrita y presencial. El examen final abarca la totalidad de los contenidos de la asignatura.

Debido a la alta carga de programación de la materia, que incluye el requisito de la aprobación de un trabajo práctico para la regularidad, el examen final no se podrá rendir en condición de libre.

Firma Profesor/a Responsable **Firma Secretario/a Académico/a**