Universidad Nacional de Rio Cuarto
Facultad de Ciencias Exactas, Físico-Químicas y Naturales

FORMULARIO PARA LA PRESENTACIÓN DE PROGRAMAS DE ASIGNATURAS

Año Lectivo: 2025

UNIVERSIDAD NACIONAL DE RÍO CUARTO

FACULTAD DE CIENCIAS EXACTAS, FÍSICO-QUÍMICAS Y NATURALES

DEPARTAMENTO DE COMPUTACIÓN

CARRERA/S: Analista en Computación y Licenciatura en Ciencias de la Computación

PLAN DE ESTUDIOS: 2024

ASIGNATURA: Análisis y Diseño de Algoritmos I **CÓDIGO:** 3383

MODALIDAD DE CURSADO: Presencial

DOCENTE RESPONSABLE: Dra. Valeria Susana Bengolea, Profesor Adjunto Dedicación Exclusiva.

EQUIPO DOCENTE: Dra**.** Valeria Susana Bengolea, Profesor Adjunto Dedicación Exclusiva. Jefe de Trabajos Prácticos: Lic. Sonia Permigiani . Ayudantes de Primera: Lic. Gastón Scilingo.

RÉGIMEN DE LA ASIGNATURA: Cuatrimestral

UBICACIÓN EN EL PLAN DE ESTUDIO: Segundo Cuatrimestre de segundo Año

RÉGIMEN DE CORRELATIVIDADES:

Carrera	Para Cursar		Para Rendir
	Regular	Aprobado	Aprobado
Lic. en Ciencias de la Computación	Algoritmos y Estructuras de Datos II (3380)	Algoritmos y Estructuras de Datos I (3378)	Algoritmos y Estructuras de Datos I (3378) Algoritmos y Estructuras de Datos II (3380)

Analista en Computación	**Estructura de Datos y Algoritmos (3412)**	**Introducción a la Computación y Programación II (3411)**	**Introducción a la Computación y Programación II (3411)** **Estructura de Datos y Algoritmos (3412)**

CARÁCTER DE LA ASIGNATURA: Obligatoria

CARGA HORARIA TOTAL: 112 horas

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

CARGA HORARIA SEMANAL: 8 horas (según el plan de estudio vigente)

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

1. CONTEXTUALIZACIÓN DE LA ASIGNATURA

Análisis y Diseño de Algoritmos I *es una asignatura obligatoria del plan de estudios 2024 de las carreras* ***Analista en Computación*** *y* ***Licenciatura en Ciencias de la Computación****. Se dicta en el segundo cuatrimestre del segundo año y constituye una materia central dentro del área de* ***Algoritmos y Lenguajes****.*

En el caso de la ***Licenciatura****, esta asignatura da continuidad al trayecto iniciado con:*

- ***Introducción a los Algoritmos (3375)***
- ***Algoritmos y Estructuras de Datos I (3378)***
- ***Algoritmos y Estructuras de Datos II (3380)***

En el caso de ***Analista en Computación****, completa el trayecto formado por:*

- ***Introducción a la Computación y Programación I (3410)***
- ***Introducción a la Computación y Programación II (3411)***

- ***Estructura de Datos y Algoritmos (3412)***

*De esta manera, la asignatura integra y profundiza los conocimientos fundamentales de la **algorítmica**, que incluyen:*

- *Los **elementos básicos de programación y algorítmica** (Introducción a los Algoritmos / Introducción a la Computación y Programación I).*
- ***Técnicas de programación más avanzadas** y herramientas para la construcción de programas correctos (Algoritmos y Estructuras de Datos I / Introducción a la Computación y Programación II).*
- *La **representación de datos en la programación y el análisis de Eficiencia de Algoritmos** (Algoritmos y Estructuras de Datos I y II / Introducción a la Computación y Programación II y Estructura de Datos y Algoritmos).*
- *Las **técnicas y estrategias de diseño de algoritmos** (Análisis y Diseño de Algoritmos I).*

OBJETIVOS PROPUESTOS

Con el desarrollo de esta asignatura se espera que el estudiante:

1. **Adquiera conocimientos, habilidades y aptitudes** necesarias para conocer y familiarizarse con las principales **técnicas y estrategias de diseño de algoritmos**, y que sea capaz de aplicarlas en la práctica, utilizando mecanismos de implementación en distintos lenguajes de programación.
    - El aprendizaje de estas técnicas se orientará de manera **independiente de lenguajes específicos**, favoreciendo una comprensión general y transferible.
2. **Aprenda a analizar la complejidad de algoritmos**, tanto en el **caso promedio** como en los **escenarios de peor y mejor caso**.
3. **Reconozca los límites de la algorítmica**, comprendiendo que existen problemas que no admiten soluciones mediante algoritmos.
4. **Conozca la clasificación estándar de problemas en clases de complejidad (P y NP)**, y desarrolle una comprensión inicial sobre la **NP-completitud** y su relevancia en la informática teórica y práctica.
5. **Integre los conocimientos previos adquiridos en las asignaturas del área de algoritmos**, consolidando una visión unificada.

6. Haga uso de las **estructuras de datos fundamentales** estudiadas en asignaturas anteriores, aplicándolas a problemas de mayor complejidad y con un análisis riguroso de su eficiencia.

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

3.1. Contenidos mínimos

Técnicas avanzadas de análisis de algoritmos. Análisis de algoritmos en caso promedio. Limitaciones del poder algorítmico. Problemas no computables. Problemas P y NP. Problemas NP-completos. Técnicas de diseño de algoritmos: Fuerza Bruta, *Divide & Conquer*, *Decrease & Conquer, Transform & Conquer*, Programación Dinámica, y Algoritmos *Greedy*. Soluciones aproximadas. Heurísticas.

3.2. Ejes temáticos o unidades

Revisión de conceptos previos.La importancia del Análisis de algoritmos. Concepto de tiempo de ejecución. Análisis Análitico y Empírico: Diferencias. Asintóticas de crecimiento. Reglas de análisis de programas imperativos y recursivos.

Conceptos básicos de probabilidades, Espacio Muestral, eventos, independencia, distribución de probabilidades discreta , distribución uniforme, Variable aleatoria discreta, Valor esperado de una variable aleatoria. Análisis de algoritmos en caso promedio, casos de estudios testigos. Análisis de algoritmos conocidos e importantes.

Introducción a la resolución algorítmica de problemas. Técnicas básicas de diseño de algoritmos. Fuerza bruta. Algoritmos basados directamente en definiciones de problemas. Búsqueda exhaustiva. Limitaciones de la técnica. *Divide & Conquer*. Relación entre *Divide & Conquer* y la recursión. Técnicas más específicas basadas en *Divide & Conquer: Decrease & Conquer* y *Transform & Conquer*. Análisis de algoritmos *Divide & Conquer*: Teorema Maestro, ejemplos y aplicaciones. Algunos algoritmos importantes basados en estas estrategias. Aplicación de la técnica en la práctica.

Técnicas avanzadas de diseño de algoritmos. Programación Dinámica. La programación dinámica como una optimización a soluciones *Divide & Conquer*. Programación Dinámica por *Memoization*. Algunos algoritmos importantes que siguen esta estrategia. Aplicación de la técnica en la práctica. La técnica *Greedy*. Algunos ejemplos de *Greedy* como una técnica completa. *Greedy* como una estrategia de aproximación a soluciones. Algunos algoritmos importantes que siguen esta estrategia. Aplicación de la técnica en la práctica. *Matroides* para la formulación de problemas de optimización y en la construcción de soluciones *Gready* óptimas.

Clases de problemas de acuerdo a su complejidad. La relación entre complejidad espacial y temporal. Máquinas de *Turing* deterministas y no deterministas. *P* y *NP*. *NP*-Completitud. Problemas tratables e intratables. Problemas de Decisión. El problema de la Satisfactibilidad Booleana. Relación entre *P* y *NP*. Problemas de grafos *NP*-completos: coloreo de grafos, grafos

hamiltonianos, recubrimiento mínimo de las aristas, corte máximo, etc. Problemas *NP-hard*. Ejemplos. Soluciones mediante aproximación para problemas *NP*-hard. Limitaciones del poder algorítmico. Problemas no computables. El Problema de la Parada.

3. ACTIVIDADES A DESARROLLAR

Como una forma de simplificar las tareas de los alumnos, se elegirán como lenguajes para la implementación de algoritmos, el lenguaje funcional *Haskell* y el lenguaje orientado a objetos *Java*. Con el fin de afianzar las técnicas de diseño de algoritmos estudiadas en la asignatura, se pedirá en algunos casos que se implementen soluciones en dos paradigmas diferentes, el funcional y el imperativo/orientado a objetos.

El trabajo práctico obligatorio, cuya aprobación es requisito para la regularidad, tienen por objetivo lograr que los alumnos puedan aplicar la teoría aprendida en la resolución algorítmica de problemas mediante la aplicación de las distintas técnicas de diseño de algoritmos estudiadas. Además, permiten integrar los contenidos de esta asignatura a muchos otros aprendidos en otras materias, principalmente de programación y lenguajes (evaluando las diferencias entre los distintos paradigmas de programación, por ejemplo), y afianzar las capacidades adquiridas en las materias de Lógica y Resolución de Problemas y Matemática Discreta. Se intentará utilizar ejemplos y problemas interesantes, en los cuales las soluciones más adecuadas sean difíciles de reconocer y desafiantes, intentando de esta manera, estimular al alumnado.

Se fomentará la lectura de material adicional y la auto organización de los alumnos en sus actividades. Además, se dejará en manos de los alumnos la instalación y manejo de las herramientas de software utilizadas en la asignatura, como una manera de estimular la práctica en cuestiones más técnicas (no necesariamente ligadas a los tópicos que la asignatura abarca), y la experiencia en la utilización de herramientas nuevas.

CLASES TEÓRICAS: 4 horas semanales

CLASES PRÁCTICAS: 4 horas semanales

CLASES DE TRABAJOS PRÁCTICOS DE LABORATORIO: Todas las clases prácticas serán de laboratorio.

4. PROGRAMAS Y/O PROYECTOS PEDAGÓGICOS INNOVADORES E INCLUSIVOS

No existen actividades de la asignatura enmarcadas en programas y/o proyectos pedagógicos.

5. CRONOGRAMA TENTATIVO DE CLASES E INSTANCIAS EVALUATIVAS

Semana	Clases Teóricas	Clases Prácticas
1	Introducción al diseño de algoritmos. Revisión de conceptos previos de algoritmos y representación de datos. conceptos de análisis de programas.	Repaso de conceptos previos de algoritmos y representación de datos. Análisis de Eficiencia de Programas Analítica y Empírica
2	Concepto de tiempo de ejecución. . Conceptos básicos de probabilidades Análisis de algoritmos en caso promedio, casos de estudios testigos. Análisis de algoritmos conocidos e importantes.	Repaso de conceptos previos de algoritmos Análisis de tiempo de ejecución: *Caso Promedio*
3	Técnica de diseño: Fuerza Bruta. Búsqueda Exhaustiva Análisis de tiempo de ejecución.	Fuerza Bruta en la resolución de problemas computacionales. Implementación en diferentes paradigmas: Funcional/Imperativo. Análisis de tiempo de ejecución.
4	Búsqueda exhaustiva. Recorridos sobre Grafos*: Depth First Search*/ *Breadth First Search*. Problema de la mochila Análisis de tiempo de ejecución.	Fuerza Bruta en la resolución de problemas computacionales: **Búsqueda Exhaustiva.** Análisis de tiempo de ejecución.
5	Técnica de Diseño *Divide & Conquer*. Análisis de tiempo de ejecución.	*Divide & Conquer* en la resolución de problemas computacionales. Análisis de tiempo de ejecución. Implementación en diferentes paradigmas: Funcional/Imperativo
6	Técnica de Diseño *Divide & Conquer*. Análisis de tiempo de ejecución. Técnicas específicas basadas en *D&C: Decrease and Conquer. Transform and Conquer*. Teorema Maestro.	Técnicas basadas en *Divide & Conquer*. Implementación en diferentes paradigmas: Funcional/Imperativo Análisis de tiempo de ejecución.
7	Técnicas específicas basadas en *D&C: Decrease and Conquer. Transform and Conquer*. Teorema Maestro. Revisión General para el Examen Parcial 1.	Técnicas basadas en *Divide & Conquer*. Implementación en diferentes paradigmas: Funcional/Imperativo Análisis de tiempo de ejecución. Aplicación del Teorema Maestro. Práctico de repaso.
8	Técnica de Diseño Programación Dinámica. **Primer Parcial (viernes 3/10)**	Programación Dinámica como una optimización a soluciones *Divide & Conquer*. Análisis de algoritmos en relación a su complejidad temporal y

		espacial. Implementación en diferentes paradigmas: Funcional/Imperativo
9	Programación Dinámica mediante *Memoization*	*Memoization* como técnica de Programación Dinámica. Análisis de algoritmos en relación a su complejidad temporal y espacial. Implementación en diferentes paradigmas: Funcional/Imperativo
10	Técnica de Diseño *Greedy*. Matroides **Recuperatorio Examen Parcial 1 (viernes 17/10)**	*Greedy* en la resolución de problemas computacionales. Análisis de algoritmos en relación a su complejidad temporal y espacial. Implementación en diferentes paradigmas: Funcional/Imperativo. Soluciones aproximadas vs Soluciones óptimas
11	Técnica de Diseño *Greedy* (continuación) Matroides	**Presentación Trabajo Práctico 1 Obligatorio.** *Greedy* en la resolución de problemas computacionales. *Matroides*
12	Clases de problemas de acuerdo con su complejidad temporal y espacial	Nociones de Complejidad y Computabilidad
13	NP Completitud: El problema de la Satisfactibilidad Booleana. Otros Computabilidad: Problemas no Computables **Examen Parcial 2 (lunes 3/11)**	Nociones de Complejidad y Computabilidad
14	Defensas de Trabajo Obligatorio **Recuperatorio Examen 2(1viernes 4/11)**	Defensas de Trabajo Obligatorio

6. BIBLIOGRAFÍA

7.1. Bibliografía obligatoria y de consulta

Obligatoria

A. Levitin, *"Introduction to the Design and Analysis of Algorithms"*, 3rd. Edition, Addison-Wesley, 2014.

T. Cormen, C. Leiserson, R. Rivest y C. Stein, *"Introduction to Algorithms"*, Third Edition, MIT Press, 2009.

De consulta

A. Aho, J. Hopcroft y J. Ullman, *"Data Structures and Algorithms"*, Addison-Wesley, 1983.

Ken Bogart y Cliff Stein, "*Discrete Math in Computer Science*", Key College Publishing, 2005

Miran Lipovaca, *Learn You a Haskell for Great Good! A Beginner's Guide,* No Starch Press, *2011*

U. Manber, *"Introduction to Algorithms, A Creative Approach"*, Addison-Wesley, 1989.

S. Baase, *"Computer Algorithms, Introduction to Design and Analysis"*, second edition, Addison-Wesley, 1988.

B. Meyer, *"Object Oriented Software Construction"* second edition, Addison-Wesley, 2000.

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

7.2. Otros: materiales audiovisuales, enlaces, otros.

Slack (comunicaciones, material, etc.)

Google Classroom (comunicaciones, material adicional, Notas Teóricas)

Github (Templates de programación con tests para autocorrección, repositorio de código para interacción entre estudiantes y docentes)

7. DÍA Y HORARIOS DE CLASES

Horarios de Clases Prácticas Tentativos:

Comisión 1: martes de 8 a 10 y viernes de 8 a 10

Comisión 2: martes de 18 a 20 y viernes de 12 a 14

Horarios de Clases Teóricas: jueves de 16 a 18 y viernes de 14 a 16

8. DÍA Y HORARIO DE CLASES DE CONSULTAS

La asignatura fijará los horarios de consultas en la primera clase. Los mismos serán acordados con los alumnos. El alumno podrá consultar por Slack, Classroom o email cuando sea necesario.

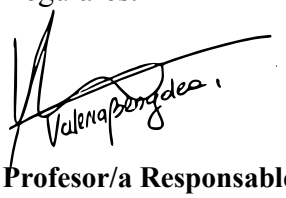
9. REQUISITOS PARA OBTENER LA REGULARIDAD Y LA PROMOCIÓN

CONDICIONES DE REGULARIDAD: Las condiciones para la regularidad de la asignatura son las siguientes: Aprobación de los dos exámenes parciales y el trabajo práctico obligatorio con nota mayor o igual a 5.

CONDICIONES DE PROMOCIÓN: Las condiciones son las mismas que para la regularidad, con 7 como nota promedio entre los dos exámenes parciales y nota mayor igual a 7 en el trabajo práctico obligatorio. La nota de promoción se calculará como un promedio ponderado de las notas de os dos parciales y el trabajo práctico.

10. CARACTERÍSTICAS, MODALIDAD Y CRITERIOS DE LAS INSTANCIAS EVALUATIVAS

- **Evaluaciones Parciales:** La materia tendrá dos exámenes parciales, cada uno con una recuperación. Los exámenes parciales serán teórico-prácticos, y abarcarán la primera y la segunda mitad de los contenidos de la asignatura, respectivamente. La evaluación del alumnado será complementada con un trabajo práctico obligatorio (ver cronograma).
- **Evaluación Final:** **El examen final para alumnos regulares** se llevará a cabo mediante evaluación oral, si el número de alumnos evaluados así lo permite. Abarcará la totalidad de los contenidos de la asignatura, verificando que los alumnos hayan adquirido los conocimientos teóricos y puedan aplicarlos en casos concretos en la práctica. Por otra parte, **el examen final para alumnos libres** estará compuesto por una primera instancia escrita, en la cual se evaluarán las capacidades de los alumnos para diseñar algoritmos utilizando las técnicas de diseño de algoritmos aprendidas en la asignatura. La segunda instancia del examen final para alumnos libres será similar al examen final para alumnos regulares.



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