

**Planificación de la Asignatura:** Algoritmos y Estructuras de Datos - Bioinformática

**Fecha:** 23/10/2024 13:02

**Código:** L1316

**Carrera:** Licenciatura en Bioinformática

**Departamento Académico:** Informática

**Docente a cargo:**

**Correo del docente a cargo:** javier.diaz@uner.edu.ar

**Régimen de Dictado:** Cuatrimestral doble oferta

**Carga Horaria Semanal:** 4 horas semanales

**Carga Horaria Total:** 56 horas

---

**Contenidos Mínimos:**

Técnicas de Abstracción de Datos. Implementaciones Estáticas y Dinámicas. Estructuras lineales: Pilas. Colas. Listas. Otras estructuras secuenciales. Árboles binarios: Árboles binarios de búsqueda, árboles AVL, árboles parcialmente ordenados. Otros árboles binarios. Árboles generales: Árboles N-arios, árboles multcamino, árboles B. Otros árboles. Diccionarios, Conjuntos y Funciones. Dispersión. Grafos: Tipos y Algoritmos. Ficheros: Secuencial, Directo e Indexado. Diseño y análisis de algoritmos. Problemas P y NP.

**Competencias Genéricas:**

## TECNOLÓGICAS

-----

CT 1: Identificación, formulación y resolución de problemas de la disciplina Bioinformática. Nivel de Dominio 1.

## SOCIALES, POLÍTICAS Y ACTITUDINALES

-----

CS 1: Fundamentos para el desempeño en equipos de trabajo. Nivel de Dominio 1.

CS 2: Fundamentos para una comunicación efectiva. Nivel de Dominio 1.

CS 5: Fundamentos para el aprendizaje continuo y autónomo. Nivel de Dominio 1.

**Competencias Específicas:**

CE 3: Desarrollar estudios en metodologías estadísticas, matemáticas y computacionales para analizar el genoma y la expresión génica. Nivel de Dominio 1.

CE 6: Aplicar métodos computacionales y matemáticos en inmunología y virología. Nivel de Dominio 1.

**Argumentación de aportes marcados en la matriz de competencias:**

La asignatura de Algoritmos y Estructuras de Datos constituye un pilar fundamental en el desarrollo inicial de competencias clave de los estudiantes, proporcionando una base sólida para su crecimiento académico y preparación profesional. Aunque su contribución se sitúa en un nivel incipiente, nivel 1 de 3, sienta las bases para competencias más avanzadas y especializadas. Al abordar conceptos fundamentales como la identificación, formulación y resolución de problemas de la disciplina Bioinformática desde una visión algorítmica y del uso eficiente del recurso computacional, así como el fomento del trabajo en equipo y el aprendizaje continuo, esta asignatura prepara a los estudiantes para enfrentar desafíos de baja complejidad en la selección y aplicación de algoritmos y estructuras de datos en problemas de diversa complejidad. Su enfoque teórico-práctico, centrado en el desarrollo, depuración y testeo de algoritmos y estructuras de datos, brinda las herramientas necesarias para utilizar eficazmente el recurso computacional como una herramienta de trabajo integral. Además, al ubicarse en las etapas iniciales de la carrera, esta asignatura sienta las bases para cursos posteriores que seguirán fortaleciendo estas competencias y llevándolas a su nivel de dominio completo.

---

**Correlativas Regulares para cursar:**

Fundamentos de Programación

**Correlativas Aprobadas para cursar:**

No posee

**Correlativas Aprobadas para promocionar o rendir el examen final:**

Fundamentos de Programación

**Insercion de la Asignatura en el plan de Estudios:**

La asignatura Algoritmos y Estructuras de Datos, con un seguimiento docente muy cercano a los estudiantes, les aporta al desarrollo de la capacidad crítica y creativa, al desarrollo de habilidades para el eficiente uso de los recursos informáticos, tecnológicos y de la programación, habilidades para el aprendizaje continuo y autónomo, habilidades de comunicación efectiva y disposición al trabajo en equipo para la resolución de problemas de la ciencia y la ingeniería.

Algoritmos y Estructuras de Datos, junto a otras materias básicas de informática, son asignaturas que brindan a la formación de los futuros profesionales conocimientos y herramientas tecnológicas y metodológicas básicas de las ciencias de la computación y la ingeniería del software en la formación a los futuros profesionales. En el trayecto de los estudiantes por estos cursos desarrollan habilidades para el manejo de las herramientas computacionales, fomentando la tarea creativa como una actividad metódica y sistemática, desarrollando habilidades iniciales para el reconocimiento, planteo de problemas, aplicación de métodos de resolución e implementación de la solución e interpretación de los resultados.

El curso aporta al desarrollo inicial de competencias generales de los futuros profesionales y, también, al desarrollo de competencias específicas relacionadas al análisis y procesamiento de datos y la resolución de problemas de la ciencia y la ingeniería empleando computadoras y lenguajes de programación de alto nivel como herramientas de trabajo. Por lo tanto, se espera que el aporte en ambos tipos de competencia permita la continuidad de su desarrollo en los cursos curriculares posteriores de la carrera.

**Objetivo General:**

Lograr que el alumno integre el recurso informático a su proceso de formación básica, científica y técnica, favoreciendo el desarrollo de habilidades en el uso de la programación y los recursos informáticos, el pensamiento lógico y crítico dentro del contexto de trabajo grupal con el propósito de desarrollar habilidades y actitudes para el trabajo en equipos a fin de aplicarlos de manera efectiva en la actividad profesional.

**Objetivos Particulares:**

Que las y los estudiantes puedan:

- Dominar los conceptos de estructuras de datos y algoritmos computacionales y su importancia para resolver problemas computacionalmente.
- Identificar los parámetros que inciden en el desempeño de una estructura de datos y sus algoritmos asociados para poder formar un criterio de uso eficiente de los recursos informáticos.
- Reconocer los TAD elementales y su modo de utilización para construir estructuras de datos más complejas.
- Aplicar estructuras de datos y sus algoritmos asociados para resolver problemas en diversos casos de uso.
- Adquirir capacidades de incorporación de bibliotecas realizadas por terceros a programas propios para potenciar la capacidad de resolución de problemas de ingeniería.
- Adoptar estrategias de reutilización de desarrollos propios para desarrollar habilidades de organización y eficiencia del desarrollo de nuevas soluciones informáticas.

**Programa Analítico:**

- Unidad 1 - Estructuras de datos y tipos abstractos de datos.

Estructuras de datos. Tipos abstractos de datos (TAD): Especificación lógica. Operaciones de inserción, consulta, edición, borrado y copia. Niveles de abstracción. Clasificaciones. Implementación y prueba de TAD. TAD integrados o disponibles en bibliotecas.

- Unidad 2 - Diseño y análisis de algoritmos.

Introducción al diseño de algoritmos. Costo computacional. Análisis a priori y a posteriori. Tiempo de ejecución. Notación asintótica. Recursividad. Tipos de recursividad. Análisis de costo para algoritmos recursivos. Algoritmo recursivo versus iterativo. Problemas P y NP.

- Unidad 3 - Estructuras de datos lineales.

Arreglos, listas, listas con restricciones. Representación posicional y enlazada. Análisis de implementaciones disponibles. Tablas de dispersión. Colisiones. Tablas de dispersión abiertas y cerradas. Funciones de dispersión. Blockchain: aplicaciones.

- Unidad 4 - Algoritmos de búsqueda y ordenamiento.

Algoritmos de búsqueda lineal y dicotómica. Clasificaciones de algoritmos de ordenamiento. Algoritmos de ordenamiento.

- Unidad 5 - Árboles y algoritmos asociados.

Estructuras de datos jerárquicas. Árboles binarios. Recorridos en árboles binarios. Montículos binarios. Árbol binario de búsqueda. Árbol AVL. Árboles generales: Árboles N-arios, árboles multcamino, árboles B.

- Unidad 6 - Grafos y algoritmos asociados.

Definición formal de grafo. Representaciones computacionales de grafos. Clasificaciones de grafos. Algoritmos de grafos: búsqueda, recorrido, ordenamiento topológico, algoritmo de Prim, algoritmo de Dijkstra, algoritmo de Warshall.

**Metodología Didáctica:**

El dictado de la asignatura será de carácter teórico y práctico, estableciéndose en primer lugar los conceptos teóricos seguido de una aplicación inmediata por medio de estudio dirigido y trabajos prácticos sobre la computadora.

Las instancias para el desarrollo de la asignatura serán clases teóricas y prácticas. Para el desarrollo de cada una de las unidades en que se divide la asignatura, se partirá de elementos simples sobre los que se estructurará el conocimiento. El docente actuará como orientador y estimulador en la resolución de problemas.

La materia cuenta con una carga horaria semanal de 4 horas, las que se distribuirán de la siguiente forma: Una clase semanal de teoría de 1,5 hs que estará a cargo del docente titular y/o adjunto. Una clase de prácticas de 2,5 hs a cargo de Jefes de Trabajos Prácticos en colaboración con Auxiliares Docentes.

**Formación Práctica:**

En las prácticas se trabajará sobre dos Trabajos Prácticos, cuya complejidad requiere que los alumnos trabajen en forma grupal con división de tareas. Esto conlleva a que los integrantes del grupo realicen regularmente puestas en común sobre lo que están estudiando y desarrollando. La presentación de los TPs para obtener la regularidad se realiza en dos instancias y en forma grupal.

**Listado de Actividades de Formación Práctica:**

Trabajo Práctico N° 1: Aplicaciones de estructuras lineales y algoritmos de búsqueda y ordenamiento.

Trabajo Práctico N° 2: Aplicaciones de árboles, grafos y algoritmos asociados.



**Intensidad de la formación práctica**

Detalle de la carga horaria total prevista para cada una de las siguientes actividades:

Actividades prácticas que aportan a las competencias específicas en el Nivel de dominio 1: 35 horas

Actividades prácticas que aportan a las competencias específicas en el Nivel de dominio 2: 0 horas

Actividades prácticas que aportan a las competencias específicas en el Nivel de dominio 3: 0 horas

Horas totales de actividades de formación práctica: 35 horas

**Metodología de Evaluación Durante el cursado:**

La evaluación se realizará en dos partes, las cuales en conjunto permiten indagar las capacidades adquiridas en resolución de problemas, habilidades de programación y manejo de conceptos de la asignatura. Para lograr esto, por un lado, los alumnos deberán desarrollar, entregar y defender trabajos prácticos, y por otro, completar una evaluación de conceptos.

-----  
Regularidad:

-----  
**Evaluación de Trabajos Prácticos**

Los alumnos deberán resolver y entregar en forma grupal dos trabajos prácticos con la guía y supervisión del JTP. El JTP revisará las entregas y hará las retroalimentaciones correspondientes. La evaluación de los trabajos prácticos contempla la habilidad lograda para resolver problemas aplicando los conceptos de la materia y las habilidades logradas en programación. El resultado de la evaluación surge de la producción entregada (código, pruebas e informe). Los trabajos prácticos se aprueban con una nota mayor o igual al 60% y todos pueden recuperarse.

**Evaluación de conceptos**

Esta evaluación está dividida en dos exámenes parciales basados en cuestionarios de carácter teórico-práctico de resolución individual, los cuales deben ser aprobados con una nota mayor o igual al 60%. En caso de no aprobar, podrán recuperarse en instancias evaluativas semejantes al finalizar el período de cursado.

-----  
Promoción:

-----  
La promoción se alcanza realizando la defensa teórico-práctica del segundo trabajo práctico. Durante la defensa el alumno deberá exponer individualmente de forma oral el trabajo práctico, justificando su desarrollo, implementación y pruebas, respondiendo correctamente a las preguntas que formulen los docentes basándose en la documentación presentada.

Se tendrá en cuenta durante la defensa:

- La correcta explicación de los problemas planteados en el trabajo práctico mientras muestra el

funcionamiento del programa.

- Un adecuado nivel de cumplimiento de los requisitos.
- Un adecuado dominio general de los conceptos centrales de la asignatura.

Para el acceso a la defensa del trabajo práctico se requerirá al alumno:

- Haber cumplido los requisitos de regularidad.
- Entregar y presentar el código fuente, pruebas e informe del trabajo práctico.

### **Metodología de Evaluación en Exámenes Finales:**

-----

Regulares

-----

El examen final consta de las siguientes etapas eliminatorias:

- 1 - Aprobar un cuestionario de evaluación de carácter teórico-práctica con un puntaje mayor o igual al 60%.
- 2 - Aprobar la defensa teórico-práctica de uno de los trabajos prácticos de su correspondiente cursado con un puntaje mayor o igual al 60%. El docente evaluador selecciona el trabajo práctico a defender.

Durante la defensa del trabajo práctico el alumno deberá exponer individualmente de forma oral el trabajo práctico, justificando su implementación y pruebas, respondiendo correctamente a las preguntas que formulen los docentes basándose en el material presentado. Se tendrá en cuenta durante la defensa el correcto planteo de los problemas y sus soluciones demostrando un adecuado dominio teórico-práctico del código producido.

-----

Libres

-----

El examen final consta de las siguientes etapas eliminatorias:

- 1 - Aprobar la defensa de la entrega de los trabajos prácticos de la presente planificación.
- 2 - Continúa con etapas eliminatorias de examen final para regulares.

Los alumnos que rindan libres deberán presentar y defender los Trabajos Prácticos implementados en el cuatrimestre inmediato anterior. En este caso, los alumnos deben coordinar con antelación con el responsable de la asignatura, o el profesor que este disponga, la entrega de los TPs.

La defensa de la entrega de los trabajos prácticos consiste en dar cuenta del correcto planteo de las soluciones a los programas de los trabajos prácticos y de su correcta ejecución.

**Condiciones de Regularidad :**

Serán reconocidos como alumnos regulares aquellos que hayan aprobado la evaluación de la entrega del total de los trabajos prácticos y aprobado cada una de las etapas de evaluación parcial con calificación mayor o igual a 60%.

Serán reconocidos como alumnos promocionados quienes hayan alcanzado la condición de regularidad y aprobado la defensa del trabajo práctico N° 2 con una nota mayor o igual a 60%.

Todas las instancias de evaluación se pueden recuperar para lograr la regularidad o la promoción.

**Cronograma de parciales durante el primer Cuatrimestre:**

**Primer Examen Parcial:** 22 de Abril de 2024

**Segundo Examen Parcial:** 11 de Junio de 2024

**Recuperatorio 01:** 17 de Junio de 2024

**Recuperatorio 02:** 20 de Junio de 2024

---

**Cronograma de parciales durante el segundo Cuatrimestre:**

**Primer Examen Parcial:** 16 de Septiembre de 2024

**Segundo Examen Parcial:** 05 de Noviembre de 2024

**Recuperatorio 01:** 11 de Noviembre de 2024

**Recuperatorio 02:** 14 de Noviembre de 2024

**Bibliografía Principal:**

- Aho, A. V., Hopcroft, J. E., & Ullman, J. D. (1998). Estructuras de datos y algoritmos (A. Vargas Villazón & J. Lozano Moreno, Trads.; 1.a ed.). Pearson Educación.
- Bhargava, A. (2016). Grokking Algorithms: An Illustrated Guide for Programmers and Other Curious People.
- Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). Introduction to algorithms (Fourth edition). The MIT Press.
- Kok, A. S. (2019). Hands-On Blockchain for Python Developers: Gain blockchain programming skills to build decentralized applications using Python. Packt Publishing Ltd.
- Miller, B., & Ranum, D. (2013). Solución de problemas con algoritmos y estructuras de datos usando Python (M. Orozco-Alzate, Trad.; 2.a ed.).

**Bibliografía Complementaria:**

- Downey, A., Elkner, J., & Meyers, C. (2002). Aprenda a Pensar Como un Programador con Python (1.a ed.). Green Tea Press. <https://openlibra.com/es/book/aprenda-a-pensar-como-un-programador-con-python>
- Jones, N. C., & Pevzner, P. A. (2004). An Introduction to Bioinformatics Algorithms.
- Kok, A. S. (2019). Hands-On Blockchain for Python Developers: Gain blockchain programming skills to build decentralized applications using Python. Packt Publishing Ltd.
- al Varó, A., Gracia Luengo, I., & García-Sevilla, P. (2014). Introducción a la programación con Python 3 (1.a ed.). Universitat Jaume I. <https://doi.org/10.6035/Sapientia93>
- Mertz, D. (2015). Functional Programming in Python (1.a ed.). O'Reilly Media, Inc. <https://openlibra.com/es/book/functional-programming-in-python>
- Nyhoff, L. R. (2006). TADs, Estructuras de datos y resolución de problemas con C++ (M. P. Tarjuelo, C. Segura Díaz, & J. A. Verdejo Díaz, Trads.; 2.a ed.). Pearson Educación.
- Roughgarden, T. (2017). Algorithms Illuminated (Part 1): The Basics.
- Roughgarden, T. (2018). Algorithms Illuminated (Part 2): Graph Algorithms and Data Structures.
- Roughgarden, T. (2019). Algorithms Illuminated (Part 3): Greedy Algorithms and Dynamic Programming.
- Sedgewick, R., & Wayne, K. (2011). Algorithms.
- Wengrow, J. (2020). A Common-Sense Guide to Data Structures and Algorithms, Second Edition: Level Up Your Core Programming Skills (2.a ed.).
- Zelle, J. M. (2017). Python programming: An introduction to computer science (Third edition). Franklin, Beedle & Associates Inc.

**Equipo de Cátedra:**

Profesor Titular: Javier Eduardo Diaz Zamboni

Profesor Adjunto: Jordán Francisco Insfrán

Jefes de Trabajos Prácticos: Diana Vertiz del Valle

Jefes de Trabajos Prácticos: Jonathan José Carlos Nicolet

Auxiliar de Primera: Bruno Breggia

Auxiliar Alumna: Giovanna Viñoli Miotti

Auxiliar Alumno: Santiago Salinas Sosa



**Actividades de Investigación Gestión y Extensión:**

---

**Requisitos de admisión para alumnos oyentes:**

Tener conocimientos de programación.

---

**Infraestructura, equipamiento y recursos necesarios:**

Para instancias de clases prácticas se precisa de laboratorios de computación.

Para todas las instancias de clases se requieren proyectores (cañones) y marcadores de pizarra.

Instalación de Anaconda.

**Otros:**