

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.

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

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.

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.

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.

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.