

EX-2026-00088647- -UNC-ME#FAMAF

PROGRAMA DE ASIGNATURA	
ASIGNATURA: Algoritmos y Estructuras de Datos II	AÑO: 2026
CARÁCTER: Obligatoria	UBICACIÓN EN LA CARRERA: 2° año 1° cuatrimestre
CARRERA: Licenciatura en Ciencias de la Computación	
RÉGIMEN: Cuatrimestral	CARGA HORARIA: 180 horas

ASIGNATURA: Algoritmos y Estructuras de Datos	AÑO: 2026
CARÁCTER: Obligatoria	UBICACIÓN EN LA CARRERA: 3° año 1° cuatrimestre
CARRERA: Licenciatura en Matemática Aplicada	
RÉGIMEN: Cuatrimestral	CARGA HORARIA: 180 horas

FUNDAMENTOS Y OBJETIVOS

Se pretende que el/la estudiante adquiera: capacidad para comprender y describir el problema que resuelve un algoritmo (el “qué”) y diferenciarlo de la manera en que lo resuelve (el “cómo”); capacidad para analizar algoritmos, compararlos según su eficiencia en tiempo y en espacio; capacidad y hábito de identificar abstracciones relevantes al abordar un problema computacional; familiaridad con técnicas de diseño de algoritmos de uso frecuente; familiaridad con la programación (en el lenguaje c, entre otros) de algoritmos y estructuras de datos, familiaridad con la utilización de diversos niveles de abstracción y lenguajes de programación.

CONTENIDO

1. Análisis de Algoritmos

Motivación

Problema de Ordenación. Diferentes maneras de ordenar. Ordenación por selección. El ciclo for. Conteo de operaciones de un programa. Esquemas de conteo. Conteo de comparaciones de la ordenación por selección. Incidencia del crecimiento del tamaño de los datos en la performance del algoritmo. Introducción del término “del orden de”. Ordenación por inserción. Conteo. Peor caso, mejor caso y caso medio.

La notación O

Significado de peor caso y caso medio. Operaciones elementales. Análisis aproximado. La notación O. Ejemplos. Insignificancia de las constantes aditivas y multiplicativas. Reflexividad y transitividad. Igualdad entre los O's de funciones. Equivalencia entre logaritmos de diferente base. Regla del límite. Jerarquía: logaritmos, polinomios, exponenciales, factoriales. El O de la suma y el producto. El O de un polinomio. Terminología: funciones y algoritmos logarítmicos, cuadráticos,

cúbicos, polinomiales, exponenciales. Balance entre tiempo y espacio de los algoritmos.

Ejemplos

Búsqueda lineal. Análisis de mejor caso, peor caso y caso medio. Búsqueda lineal en un arreglo ordenado. Análisis de mejor caso, peor caso y caso medio. Búsqueda binaria. Análisis de mejor caso, peor caso y caso medio. Contraste entre el algoritmo lineal y el logarítmico cuando el tamaño de la entrada crece.

Motivación de la recurrencias

Transformación gradual de la ordenación por selección en la ordenación por intercalación. Versión funcional de la ordenación por intercalación. Versión imperativa. Análisis de la ordenación por intercalación. Resolución de la recurrencia.

Recurrencias

Recurrencias divide y vencerás. Formulación y resolución. Ejemplos. Demostración de la resolución de recurrencias divide y vencerás.

2. Estructuras de Datos

Introducción

Importancia de la elección de estructuras de datos adecuadas. Los tipos concretos como concepto relativo a un lenguaje de programación. Los tipos abstractos como concepto asociado a un problema que se quiere resolver. Tipos abstractos y sus diferentes representaciones.

Estructuras concretas

Estructuras concretas más comunes en los lenguajes de programación. Arreglos. Operaciones para manipularlos. Almacenamiento en memoria. Representación gráfica. Eficiencia de las operaciones. Diferentes tipos de índices. Tipos enumerados. Ciclo for generalizado. Listas como tipos concretos. Operaciones para manipularlos. Almacenamiento en memoria. Representación gráfica. Eficiencia de las operaciones. Registros. Operaciones para manipularlos. Almacenamiento en memoria. Representación gráfica. Problema de aliasing.

Tipos abstractos de datos (TAD's)

Tipos abstractos más usuales. Tipos abstractos como concepto que surge de un problema a resolver. Chequeo de paréntesis balanceados: TAD Contador, operaciones, ecuaciones. Chequeo de delimitadores balanceados: TAD Pila, operaciones, ecuaciones. Representaciones posibles de contadores. Ejemplo: versión iterativa de la ordenación por intercalación usando una pila. Ejemplo: evaluación de expresiones en notación polaca inversa usando una pila. TAD Lista. Operaciones. Ecuaciones. Representaciones usando arreglos. Representaciones de pilas usando arreglos y listas. Transmisión de datos: TAD cola, operaciones, ecuaciones. Representaciones usando arreglos y listas. Listas enlazadas. Representación gráfica. Representaciones de listas, pilas y colas usando listas enlazadas, listas enlazadas con puntero al último y listas circulares. Aliasing y errores usuales al programar con punteros. Manejo de memoria en ejecución. Diccionarios: TAD árbol binario. Representación gráfica. Operaciones. Ecuaciones. Terminología botánica y genealógica. Posiciones. Subárbol correspondiente a una posición. Posiciones de un árbol. Elemento alojado en una posición de un árbol. Representación usando punteros. Árboles binario de búsqueda (ABB). Operaciones: versiones recursiva e iterativa. Eficiencia. TAD cola de prioridades. Operaciones. Ecuaciones. Heap. Implementación de cola de prioridades usando un heap.

Eficiencia de las operaciones. Heap usando arreglos. Eficiencia. Ordenación con heap. Eficiencia. Ordenación con heap sin arreglo auxiliar.

Otras estructuras

Problema unión-find. Inicialización virtual.

3. Estrategias conocidas de resolución de problemas

Uso de heurísticas en algoritmos. Estrategias de diseño de algoritmos.

Algoritmos voraces

Propiedades generales de los algoritmos voraces (o greedy o glotones o golosos). Esquema general. Problema de la moneda simplificado. Problema de la mochila simplificado. Problema del camino de costo mínimo. Algoritmo de Dijkstra. Problema del árbol generador de costo mínimo. Algoritmos de Prim y de Kruskal.

Divide y vencerás

Propiedades generales de la técnica divide y vencerás. Esquema general. Búsqueda binaria. Ordenación por intercalación. Ordenación rápida (quicksort). Cálculo eficiente de la potencia n-ésima de un número. Multiplicación de grandes números.

Backtracking

Motivación: algoritmo para salir de un laberinto. Problema de la moneda. Problema de la mochila. Problema de los caminos de costo mínimo.

Programación dinámica

Funciones recursivas potencialmente exponenciales. Confección de tablas. Fibonacci. Problema de la moneda. Problema de la mochila. Funciones con memoria. Revisión de los problemas de la moneda y de la mochila. Problema de los caminos de costo mínimo. Algoritmo de Floyd. Cómputo de números combinatorios. Reducción del espacio necesario para las tablas.

Recorrida de grafos y más backtracking

Recorrida de árboles binarios. Pre-orden, in-orden y pos-orden de izquierda a derecha y de derecha a izquierda. In-orden para listar ordenadamente un ABB. Recorrida de árboles finitarios. Precondicionamiento. Pre-orden y pos-orden para resolver el problema del ancestro. Recorrida de árboles dirigidos o no. DFS recursivo e iterativo con pila. BFS con cola. Grafos implícitos. Problema de las ocho reinas. Podas graduales al grafo de búsqueda

BIBLIOGRAFÍA

BIBLIOGRAFÍA BÁSICA

- Fundamentos de Algoritmia, Gilles Brassard, Paul Bratley. Prentice-Hall, 1997.
- Fundamentals of Algorithmics. Gilles Brassard, Paul Bratley. Prentice-Hall, 1995.
- Introduction to Algorithms. Thomas Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein. Cambridge, 2009.
- Introduction to Algorithms: A Creative Approach. Udi Manber. Addison-Wesley, 1989.

BIBLIOGRAFÍA COMPLEMENTARIA

- Programación Metódica. José Luis Balcázar. McGraw-Hill, 1993.
- Matemática Discreta. Norman L. Biggs. Vives V., 1998
- Cálculo de Programas. Javier Blanco, Silvina Smith, Damián Barsotti. Universidad Nacional de Córdoba, 2008.
- Programming: the Derivation of Algorithms. Anne Kaldewaij. Prentice-Hall, 1990.

METODOLOGÍA DE ENSEÑANZA

Los contenidos del programa se organizan en tres partes: análisis de algoritmos, estructuras de datos y estrategias de diseño algorítmico. Cada una de esas partes presenta coherencia temática y brinda herramientas necesarias para abordar las siguientes.

Para el desarrollo de la asignatura se combinan clases teóricas, prácticas y de laboratorio. En las clases teóricas se formulan preguntas y/o problemas disparadores, estimulando la participación a través de preguntas, comprensión del problema y propuestas de solución. Las clases prácticas se desarrollan en aulas comunes, donde estudiantes de manera individual o en grupos resuelven una guía de ejercicios y problemas, realizando consultas a sus docentes cuando lo requieren. Ejercicios y problemas seleccionados se discuten y resuelven de manera colectiva. Las clases de laboratorio se desarrollan en aulas con computadoras, agrupando a los/as estudiantes en diferentes comisiones. Se resuelven en la computadora problemas modélicos que ya se han estudiado en clases teóricas y prácticas. También se proponen desarrollos más complejos que dan lugar a proyectos de varias semanas de duración realizados de manera grupal y colaborativa bajo la orientación de sus docentes. Los programas deben implementarse en el lenguaje de programación C. En las clases de laboratorios se dictan contenidos necesarios sobre el lenguaje de programación. Las aulas de laboratorio cuentan con suficiente número de computadoras, con todas las herramientas necesarias bajo el sistema operativo Linux Ubuntu). Estudiantes que deseen traer su propia notebook pueden hacerlo y realizar los desarrollos en el lenguaje C en cualquier plataforma (Linux, Windows o MacOS).

Entre las tareas que se propone a los/as estudiantes podemos mencionar: lectura del material empleado en las clases teóricas, resolución en grupo o individual de los ejercicios y problemas de las guías, implementación en computadoras de soluciones a los problemas planteados en el laboratorio sujeto a las pautas formuladas en los enunciados y con el acompañamiento y orientación docente. Se promueve la participación en foros del Aula Virtual, la interacción con docentes y compañeros/as. Además del material bibliográfico y las guías de ejercicios y problemas diseñadas para la materia, se brinda acceso a los apuntes del/ de la docente de teóricos y a las presentaciones multimedia empleadas en el dictado. Se ponen a disposición de estudiantes simuladores de acceso libre existentes en Internet que permiten visualizar el funcionamiento de los algoritmos y comprender de manera visual las estructuras de datos y los algoritmos que se estudian. También se brinda acceso a manuales del lenguaje de programación y del sistema operativo. Todo el material es accesible desde el Aula Virtual de la materia, que cuenta asimismo con la posibilidad de la interacción con docentes y pares a través de los foros.

EVALUACIÓN

FORMAS DE EVALUACIÓN

Se toman dos evaluaciones parciales escritas evaluando los contenidos teóricos y prácticos, a través de ejercicios similares a los de las guías de práctico. Se evalúan

dos trabajos prácticos de laboratorio mediante presentaciones breves de lo realizado.

Se puede recuperar un examen parcial teórico/práctico y un examen parcial de laboratorio.

El examen final para regulares consiste de una evaluación escrita teórico-práctica.

El examen final para libres consiste de una evaluación escrita teórico-práctica. y una evaluación de laboratorio, es necesario aprobar ambas.

REGULARIDAD

- Aprobar las dos evaluaciones parciales de laboratorio, o sus correspondientes recuperatorios (podrá recuperar una sola instancia).

PROMOCIÓN

- Aprobar las dos evaluaciones parciales de laboratorio, o sus correspondientes recuperatorios (podrá recuperar una sola instancia).

- Aprobar las dos evaluaciones parciales con una nota no menor a 6 (seis), y obteniendo un promedio no menor a 7 (siete).