UTN.BA
DEPARTAMENTO
INGENIERIA EN SISTEMAS DE INFORMACION

ASIGNATURA	PATRONES ALGORÍTMICOS PARA ESTRUCTURAS AVANZADAS
DEPARTAMENTO	ING. EN SIST. DE INFORMACIÓN
ÁREA	ELECTIVA
BLOQUE	TECNOLOGÍAS APLICADAS

MODALIDAD	CUATRIMESTRAL
HORAS SEMANALES	4 horas
HORAS / AÑO	64 horas
HORAS RELOJ / AÑO	48
NIVEL	3º AÑO
AÑO DE DICTADO	2018

OBJETIVOS

- El objetivo de la materia es lograr que los alumnos integren y profundicen los conocimientos que han adquirido durante el curso de las materias del área de programación (Algoritmos y Estructura de Datos, Matemática Discreta, Sintaxis y Semántica de los Lenguajes, Paradigmas de Programación). Complementándolos con la incorporación de nuevos, y más complejos, patrones algorítmicos.
- Este objetivo principal guía un conjunto de objetivos específicos:
- Abordar diversas clases de problemas de modo de identificarlos y categorizarlos de acuerdo a su complejidad.
- Profundizar el concepto de abstracción, separando especialmente interfaces e implementación.
- Aplicar estrategias de solución de problemas complejos, definidas en forma abstracta, decidiendo y analizando en función de variables tales como: costos de programación, de hardware, tiempos y rendimiento.
- Observar los conceptos de Complejidad Computacional para garantizar la eficiencia del producto final.

Contenidos Mínimos (Programa Sintético)

- Implementación de algoritmos recursivos. Análisis de su eficiencia y comparación de los mismos respecto de sus respectivas versiones iterativas.
- Estructuras arbóreas: Análisis de algoritmos que recorren árboles, contrastando sus implementaciones recursivas e iterativas. Árbol Binario de Búsqueda, Árbol AVL, Algoritmos de balanceo, Árbol B.

- Complejidad algorítmica: Entender la Complejidad Algorítmica o Complejidad Computacional como una herramienta analítica que permite medir y comparar la eficiencia de los algoritmos equivalentes.
- Algoritmos de ordenamiento: Estudio de los diferentes métodos de ordenamiento, comparando rendimientos en base a sus funciones de Complejidad Algorítmica.
- Patrones algorítmicos: Analizar cualidades y características de diferentes algoritmos modelo, para clasificarlos según los diversos patrones conocidos. Estrategias de diseño: *Divide & Conquer*, *Greedy*, Programación dinámica, *Backtracking*.
- Grafos: Algoritmos para visitar todos los nodos de un grafo y dar solución a los problemas típicos que, usualmente, se representan con esta estructura de datos. Estudio de las diferentes implementaciones, en función de los patrones algorítmicos antes mencionados. Algoritmos de Dijkstra, Floyd-Warshall, Kruskal, Prim,

Contenidos Analíticos

Unidad 1. Algoritmia y programación

Revisión de los conceptos previos (lenguajes, algoritmia, paradigmas). La importancia de las especificaciones de Pre y Post-condición. Invariantes de ciclos. Pruebas de corrección, pruebas de terminación y técnicas de transformación de programas. Estructuras de datos fundamentales. Abstracción mediante Tipos Abstractos de Datos (TAD) y a través de Clases y Objetos. Interfaces e Implementaciones. Análisis de algoritmos. Concepto de tiempo de ejecución. Reglas de análisis de programas imperativos y recursivos.

Unidad 2. Principios de recursividad

Principios de la recursividad. Condición de corte. Árbol de llamadas. Diseño de algoritmos recursivos. “Dividir y Conquistar”. Recursividad de Cola, Métodos para la eliminación de la recursión. Comparación de algoritmos recursivos respecto de sus implementaciones iterativas.

Unidad 3. Estructuras arbóreas

Árbol Binario, árboles completos y semicompletos. Representación de *arrays* en árboles binarios semicompletos. Montículos (*Heap*), Árbol Binario de Búsqueda (ABB). Operaciones básicas e implementaciones. Balanceo de árboles. Árboles de expresiones. Árboles AVL. Árboles *n*-arios. Árboles B. Aplicación y uso de la estructura de árbol.

Unidad 4. **Complejidad Algorítmica**

La complejidad algorítmica como una herramienta de comparación de algoritmos equivalentes. Funciones de complejidad. Cota Asintótica Superior, Media y Ajustada. Peor caso: Notación O, caso promedio: Notación Θ, mejor caso: Notación Ω. Relación entre complejidad espacial y temporal. Clasificación de algoritmos en función de su complejidad.

Unidad 5. **Algoritmos de ordenamiento**

Principales algoritmos de ordenamiento: *Bubble sort*, *Quicksort*, *Heapsort*. Ordenamiento por inserción, ordenamiento de raíz y otras técnicas conocidas. Clasificación de los algoritmos estudiados según sus funciones de complejidad. Vinculación de las implementaciones de estos algoritmos con los temas analizados anteriormente.

Unidad 6. **Patrones algorítmicos**

Resolución algorítmica de problemas. Técnicas básicas de diseño de algoritmos: Fuerza bruta. Algoritmos basados directamente en definiciones de problemas. Búsqueda exhaustiva. Relación entre Divide y Conquista y la recursividad. Algoritmos voraces (*Greedy*). Programación Dinámica: almacenamiento de resultados previamente calculados y su aplicación para optimizar soluciones recursivas. Aplicaciones prácticas. Clasificación de algoritmos según los diferentes tipos de patrones analizados.

Unidad 7. **Grafos**

Estructura Grafo. Grafos dirigidos y no dirigidos. Implementaciones diversas. Recorridos en profundidad y anchura. Aciclidad, recorridos topológicos. Algoritmo de los caminos mínimos de Dijkstra, algoritmo de Warshall-Floyd. Árbol de expansión de coste mínimo. Algoritmos Kruskal y Prim. Estudio de la complejidad. Análisis estratégico de sus implementaciones: *Greedy* vs. Programación dinámica.

BIBLIOGRAFÍA

- Diseño de Algoritmos, Nivio Siviani. (Ed. Tomson, 2007).
- Desafíos de Programación, Steven S. Skiena, Miguel A. Revilla. (Ed. Lulu.com, 2002).
- Fundamentos de Programación y Bases de Datos, Ángel Arias. (Ed. CreateSpace, 2014).

CORRELATIVAS

Para Cursar:

Cursadas:

- Análisis de Sistemas
- Sintaxis y Semántica del Lenguaje
- Paradigmas de Programación

Para Rendir:

Aprobadas:

- Análisis de Sistemas
- Sintaxis y Semántica del Lenguaje
- Paradigmas de Programación