PROGRAMA de Algoritmos

Carreras: Licenciatura en Informática - Licenciatura en Bioinformática

Asignatura: Algoritmos

Núcleo al que pertenece: Básico

Profesor: Pablo Factorovich y Germán Patricio Barletta

Asignaturas Correlativas: Programación Funcional

Objetivos:

- Que les alumnes incorporen la noción formal de orden de complejidad temporal y espacial de un algoritmo para el peor caso y el caso promedio.
- Que les alumnes elijan entre dos algoritmos en base a su propio cálculo de complejidad temporal y espacial en peor caso para los mismos.
- Que les alumnes pueden desarrollar algoritmos más eficientes que los triviales para distintos problemas, mediante el uso de las técnicas algorítmicas de Divide y Vencerás y Programación Dinámica.
- Que les alumnes encuentren un algoritmo a problemas de decisión, enumeración u optimización de cierta complejidad mediante el uso de Backtracking.
- Que les alumnes conozcan los algoritmos de Precondicionamiento Transformación de Dominio y los problemas de propagación de errores en los algoritmos numéricos.
- Que les alumnes conozcan los problemas clásicos de grafos, sus soluciones algorítmicas más importantes y sus aplicaciones más relevantes.
- Que les alumnes conozcan los problemas clásicos sobre cadenas, sus soluciones algorítmicas más importantes y sus aplicaciones más relevantes, en particular aquellas de la bioinformática.

Contenidos mínimos:

- Noción de algoritmo, ejemplos de algoritmos (criba de Eratóstenes, mcd, etc). Criterios de selección de un algoritmo.
- Notación O y W. Análisis teórico del tiempo de ejecución de un algoritmo Análisis práctico del tiempo de ejecución de un algoritmo.
- Algoritmos Divide y Vencerás. Análisis de procedimientos recursivos.
- Algoritmos Basados en Programación Dinámica.
- Algoritmos Greedy.
- Algoritmos de Precondicionamiento y Transformación del Dominio.
- Algoritmos de programación matemática, heurísticas. Algoritmos numéricos y propagación de errores.
- Casos: algoritmo de Huffman, encriptación, compresión, búsqueda, actualización, ordenamiento, estructuras de datos y algoritmos, árboles estrella, matrices.
- Algoritmos sobre grafos (DFS, BFD, Prim, Kruskal, Dijkstra, Floyd, sort topológico, etc).
- Algoritmos básicos sobre cadenas: matching, alineamiento, sufijos.

Carga horaria semanal: 6 hs

Programa analítico:

Unidad 1: Noción de algoritmo y complejidad.

Especificación de problema; definición de algoritmo; notación O y Omega; complejidad en peor caso, caso promedio, mejor caso; consideraciones prácticas. (1 semana)

Unidad 2: Técnicas algorítmicas.

Backtracking, (1 semana), programación dinámica (1 semana), divide y reinarás (1 semana), algoritmos golosos (1 semana).

Unidad 3: Problemas y algoritmos básicos de grafos.

Algoritmos de Prim y Kruskal para árbol generador mínimo (1 semana); algoritmo de Dijkstra para camino mínimo single source con pesos positivos y algoritmo para camino mínimo en DAGs, ordenamiento topológico, PERT (1 semana); algoritmos de Bellman-Ford y Floyd para camino mínimo (1 semana); recorrido euleriano, hamiltoniano y problema de viajante de comercio (1 semana).

Unidad 4: Heurísticas y metaheurísticas.

Algoritmos constructivos golosos y algoritmos de búsqueda local; metaheurísticas GRASP y tabú search. (1 semana).

Unidad 5: Modelado.

Modelado con grafos y modelos de programación lineal y lineal entera; método simplex como algoritmo de búsqueda local; definición de problema de optimización combinatoria y aplicación a los problemas de la unidad 3. (1 semana)

Unidad 6: Triangulación de matrices.

Método de Gauss; métodos iterativos; precondicionadores de Jacobi y Gauss-Seidel; número de condición; números fraccionarios y propagación de errores. (1 semana)

Unidad 7: algoritmos básicos sobre cadenas.

Problema de búsqueda, algoritmos de Knuth-Morris-Pratt y Boyer-Moore; árbol de prefijos (trie) y noción intuitiva del árbol de sufijos y sus aplicaciones a bioinformática (2 semanas)

Unidad 8: casos de algoritmos.

Repaso de estructuras de datos básicas y ejemplos de algoritmos concretos: hashing → encriptación , diccionarios → compresión, heaps → ordenamiento y algoritmos golosos, algoritmos golosos → huffman, análisis ajustado de complejidad → MCD, etc. (esta unidad es transversal al resto de la materia).

Bibliografía obligatoria:

- Thomas H. Cormen, Clifford Stein, Ronald L. Rivest y Charles E. Leiserson. 2009. Introduction to Algorithms (3rd ed.). McGraw-Hill Higher Education. Laurence A. Wolsey. 1998. Integer Programming. Wiley-Interscience
- Chvátal, V. s., Linear programming, W. H. Freeman and Company, New York, 1983.
- Faires, J. & Burden, R., Numerical Methods, 4th, Cengage Learning, 2012
- Gusfield, D.Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology Cambridge University Press, 1997
- Talbi, E.Metaheuristics: From Design to Implementation, Wiley, 2009
- Wolsey, L. A., Integer programming, John Wiley & Sons, Inc., New York, 1998

Bibliografía de consulta:

- Robert Sedgewick, Kevin Wayne. 2011. Algorithms (4th Edition). Addison-Wesley Professional

  Blum, C. & Roli, A., Metaheuristics in combinatorial optimization: Overview and conceptual comparison, ACM Computing Surveys (CSUR), ACM, 2003, 35, 268–308

  Brassard, G. & Bratley, P., Fundamentals of algorithmics, Prentice Hall, Inc., Englewood Cliffs, NJ, 1996.

Organización de las clases:

Cada semana se realiza un desarrollo teórico inicial y una clase práctica de fijación de contenidos que incluye ejercitación y consultas. Las clases siguen las unidades temáticas en orden, salvo pequeñas excepciones (e.g., método de la bisección se introduce primeramente como aplicación de la técnica divide y reinarás).

Guías prácticas

Objetivos de la Práctica 1 - Complejidad algorítmica: introducir la noción de complejidad algorítmica, tanto la temporal como la espacial. Introducir el concepto de comportamiento asintótico. Desarrollar una intuición sobre las clases de complejidades más frecuentes: constante, logarítmica, poli logarítmica, lineal, lineal - logarítmica, polinómica y super polinómica.

Objetivos de la Práctica 2 - Divide y vencerás: afianzar la noción de recursión Introducir la técnica de divide y vencerás. Identificar los pasos requeridos para resolver problemas con divide y vencerás. Calcular la complejidad algorítmica de algoritmos recursivos. Desarrollar optimizaciones para lograr una mayor eficiencia de los algoritmos.

Objetivos de la Práctica 3 - Programación dinámica: introducir la técnica de programación dinámica. Introducir la noción de optimización. Identificar características de problemas que pueden ser sujetos a soluciones de programación dinámica: subestructura óptima, superposición de subproblemas. Identificar los pasos para

alcanzar una solución canónica de programación dinámica. Utilizar soluciones *top-down* por *memoization* y *bottom-up*. Calcular complejidades de los algoritmos de programación dinámica. Evaluar posibles optimizaciones que aumenten la eficiencia de los algoritmos de programación dinámica.

Objetivos de la Práctica 4 - Backtracking: introducir la técnica de *backtracking* para el diseño de algoritmos que resuelvan problemas de enumeración; o encuentren una solución única o un conjunto de soluciones a un problema; o que resuelvan problemas de optimización. Identificar problemas que puedan ser resueltos por *backtracking*.

Objetivos de la Práctica 5 - Algoritmos golosos: introducir la técnica de diseño de algoritmos golosos para problemas de optimización. Identificar problemas que puedan ser resueltos óptimamente por esta técnica: subestructura óptima y optimización local que lleve a la optimización global. Comparar esta técnica con la de programación dinámica: los requisitos, los algoritmos resultantes y sus complejidades.

Objetivos de la Práctica 6 - Introducción a Grafos: introducir las definiciones y nomenclatura básica de grafos. Ejemplificar los distintos tipos de grafos y como sus características afectan los algoritmos a utilizar. Introducir algoritmos *Depth First Search* y *Breadth First Search.*

Objetivos de la Práctica 7 - AGM y camino mínimo: comprender el problema de Árbol Generador Mínimo y su diferencia con los de camino mínimo; distinguir 3 tipos de problemas de camino mínimo. Introducir y ejercitar algoritmos de camino mínimo: Prim, Kruskal, Bellman-Ford, Floyd–Warshall, Dijkstra. Identificar las técnicas algorítmicas utilizadas en su diseño y sus complejidades

Objetivos de la Práctica 8 - Recorridos eulerianos, hamiltonianos y TSP: introducir los conceptos de recorridos eulerianos y hamiltonianos y sus requisitos. Desarrollar algoritmos que determinen la satisfacción ---o la no satisfacción--- de sus requisitos y algoritmos que encuentren estos caminos. Introducir el problema del viajante de comercio. Aplicar técnicas de *backtracking* y golosas para resolver éste problema u obtener una solución heurística.

Objetivos de la Práctica 9 - Modelado a través de problemas de grafos: ejercitar una noción general del modelado por grafos. Aplicarla a distintos problemas clásicos como ejemplos del proceso que el modelado por grafos implica. Luego resolver estos problemas con los métodos ya aprendidos en la materia.

Objetivos de la Práctica 10 – Triangulación de matrices: introducir la solución directa con reducción de Gauss y su complejidad. Trabajar la noción de pivoteo para reducir la propagación de errores en matrices con alto número de condición. Introducir métodos iterativos, su complejidad y condiciones de convergencia. Ejemplificar precondicionadores de Jacobi y Gauss-Seidel; evaluar la reducción en el número de iteraciones en ciertas condiciones de convergencia.

Objetivos de la Práctica 11 - Problemas de cadenas: introducir algoritmos clásicos de búsqueda exacta de subcadenas de caracteres: Rabin-Karp, algoritmo del autómata finito, Knuth-Morris-Pratt y Boyer-Moore. Evaluar las técnicas utilizadas en su diseño y sus complejidades. Introducir variaciones a los algoritmos para la resolución de problemas específicos de la bioinformática.

Modalidad de evaluación:

Los mecanismos de evaluación en modalidades libre y presencial de esta asignatura están reglamentados según los siguientes artículos del Régimen de estudios de la UNQ (Res. CS 201/18)
En la modalidad de libre, se evaluarán los contenidos de la asignatura con un examen escrito, un examen oral e instancias de evaluación similares a las realizadas en la modalidad presencial.

CRONOGRAMA TENTATIVO

Semana	Tema/unidad	Actividad*				Evaluación
		Teórico	Práctico			
			Res Prob.	Lab.	Otros Especificar	
1	Complejidad	x	x	X		
2	Divide y vencerás	x	x	X		
3	Programación dinámica	x	x	X		
4	Backtracking	x	x	X		
5	Algoritmos golosos	x	x	X		
6	Parcial		x	X		Parcial
7	Grafos y Árbol generador mínimo	x	x	X		
8	Camino mínimo I	x	x	X		
9	Camino mínimo II	x	x	X		
10	Recorridos eulerianos, hamiltonianos y TSP	x	x	X		
11	Modelado a través de problemas de grafos	x	x	X		
12	Problemas de cadenas	x	x	X		
13	Heurísiticas y metahuerísticas	x	x	X		
14	Otros problemas de grafos	x	x	X		
15	Presentación de evaluación y desarrollo		x	X		TP
16	Consultas y entrega de TP		x	X		TP

*INDIQUE CON UNA CRUZ LA MODALIDAD

Firmado digitalmente por: SUAREZ Mariana Alejandra
Directora del Departamento de Ciencia y Tecnologla
UNIVERSIDAD NACIONAL DE QUILMES