

Análisis y diseño de algoritmos

(3009430)

Objetivo del curso (oficial)

Cualquiera con una experiencia mínima en algoritmos y programación sabe que para un mismo problema pueden existir innumerables algoritmos de solución. Tales soluciones pueden variar en diversos aspectos, siendo de particular interés la eficiencia en términos de su costo computacional.

En el contexto de dicha eficiencia se puede hablar de algoritmos "mejores" que otros, siendo el objetivo principal de esta asignatura brindarle elementos al estudiante para analizar y diseñar algoritmos eficientes.

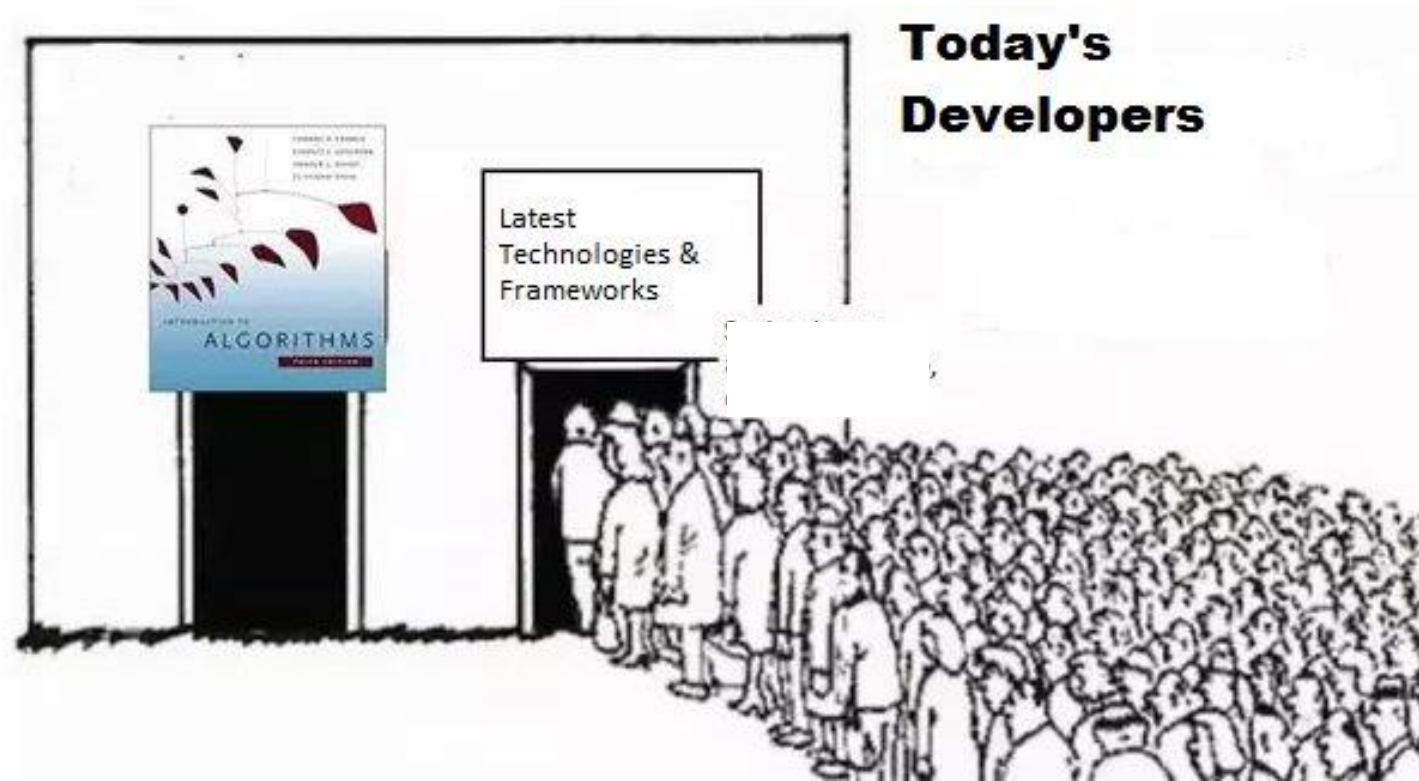
Objetivos del curso (extra-oficial)

- Ofertar una asignatura disciplinar optativa para Ingeniería de Sistemas e Informática. De igual manera que sirva en el componente electivo de otros programas de la Universidad Nacional de Colombia.
- Brindar un espacio para que los estudiantes a quienes les guste la programación mejoren sus habilidades
- Preparar a los estudiantes para participar en el circuito colombiano de maratones de programación y demás competencias, y llevar a la Sede Medellín a los primeros lugares
- Mejorar las probabilidades de vinculación laboral en empresas de tecnología importantes, o por qué no, de creación de nuevas empresas

¿Por qué estudiar algoritmos?

- ✓ Es importante para comprender conceptos y problemas de muchas otras áreas de computación y de ingeniería en general: redes y telecomunicaciones, criptografía, optimización, etc.
- ✓ Da “super poderes” en programación
- ✓ Es desafiante y satisfactorio

Ejemplo: ¿Cómo podemos contar un número muy grande de asistentes a un evento?



Pre-requisitos

- ✓ Excelentes bases de programación y estructuras de datos:
 - Recursividad
 - Arreglos y listas, Pilas y colas, Grafos, Árboles binarios, Montículos binarios
- ✓ Conocimientos mínimos de POO y buen manejo de al menos un lenguaje imperativo objetual (Python, Java, o C++)
- ✓ Conocimientos de cálculo: función logaritmo, series, integrales (básico)
- ✓ Conocimientos de estadística: valor esperado, variables aleatorias discretas (básico)

Metodología

Clases “magistrales”, no hay práctica, y este no es un curso de programación.

Evaluación:

- 4 Talleres prácticos, individuales y continuos con fechas de corte, de 20% cada uno

$$\text{Calificación de cada taller} = 5.0 * \frac{\text{ejercicios realizados}}{\text{ejercicios propuestos}}$$

Cantidad de ejercicios por taller: 5

Cantidad total de ejercicios en el semestre: 60

- 1 Trabajo final con exposición oral de 20%

Programa calendario

Sesión	Módulo	Contenido
1	Complejidad de algoritmos	Introducción Notación Big O Comparación de ordenes de complejidad
2	Algoritmos ad-hoc y clasificación de problemas	Cantidad de divisores Algoritmo de Euclides Criba de Eratóstenes Problemas P y NP
3 ¹	Búsqueda exhaustiva	Cuadrados mágicos Ocho reinas Travel Salesman Problem Suma de subconjunto Coloreado de grafos
4	Algoritmos voraces	Cambio de monedas Planificación de tareas Programación de procesos unitarios Círculos de Malfatti
5		Selección de actividades Código de Huffman
6 ²		Algoritmo de Dijkstra Algoritmo de Prim Algoritmo de Edmonds–Karp
7	Programación dinámica	Máximo conjunto independiente Backtracking Corte de cable Knapsack
8		Cambio de monedas con programación dinámica Mayor subsecuencia común Alineación de secuencias Algoritmo de Floyd-Warshall
9 ³		Multiplicación en cadena de matrices Algoritmo de Prim
10	Divide y vencerás	MergeSort, Método maestro Búsqueda binaria
11		Cantidad de inversiones de un arreglo QuickSort Estadístico de orden k
12 ⁴		Potenciación recursiva Fibonacci matricial Multiplicación de Karatusuba Multiplicación de matrices de Strassen Pares más cercanos
13	Aplicaciones	
14	Trabajo final	

Temas “gruesos” en algoritmia

Introduction to
algorithms, Cormen et
al.

+	I Foundations
+	II Sorting and Order Statistics
+	III Data Structures
+	IV Advanced Design and Analysis Techniques
	15 Dynamic Programming
	16 Greedy Algorithms
	17 Amortized Analysis
+	V Advanced Data Structures
+	VI Graph Algorithms
+	VII Selected Topics
	27 Multithreaded Algorithms
	28 Matrix Operations
	29 Linear Programming
	30 Polynomials and the FFT
	31 Number-Theoretic Algorithms
	32 String Matching
	33 Computational Geometry
	34 NP-Completeness
	35 Approximation Algorithms

Algorithms,
Dasgupta et al.

Preface	
0 Prolog	4 Paths in graphs
0.1	4.1 Distances
0.2	4.2 Breadth-first search
0.3	4.3 Lengths on edges
Exercises	4.4 Dijkstra's algorithm
	4.5 Priority queue implementations
1 Algorithms	4.6 Shortest paths in the presence of negative edges
1.1	4.7 Shortest paths in dags
1.2	Exercises
1.3	
1.4	5 Greedy algorithms
1.5	5.1 Minimum spanning trees
Exercises	5.2 Huffman encoding
	5.3 Horn formulas
	5.4 Set cover
	Exercises
Random	
2 Divide and Conquer	6 Dynamic programming
2.1	6.1 Shortest paths in dags, revisited
2.2	6.2 Longest increasing subsequences
2.3	6.3 Edit distance
2.4	6.4 Knapsack
2.5	6.5 Chain matrix multiplication
2.6	6.6 Shortest paths
Exercises	6.7 Independent sets in trees
	Exercises
3 Decision	7 Linear programming and reductions
3.1	7.1 An introduction to linear programming
3.2	7.2 Flows in networks
3.3	7.3 Bipartite matching
3.4	7.4 Duality
Exercises	7.5 Zero-sum games
	7.6 The simplex algorithm
	7.7 Postscript: circuit evaluation
	Exercises
	8 NP-complete problems
	8.1 Search problems
	8.2 NP-complete problems
	8.3 The reductions
	Exercises

Algorithm design,
Kleinberg and Tardos

1 Introduction: Some Representative Problems	1
1.1 A First Problem: Stable Matching	1
4 Greedy Algorithms	115
4.1 Interval Scheduling: The Greedy Algorithm Stays Ahead	116
4.2 Scheduling to Minimize Lateness: An Exchange Argument	125
4.3 Optimal Caching: A More Complex Exchange Argument	131
4.4 Shortest Paths in a Graph	137
4.5 The Minimum Spanning Tree Problem	142
4.6 Implementing Kruskal's Algorithm: The Union-Find Data Structure	151
4.7 Clustering	157
4.8 Huffman Codes and Data Compression	161
* 4.9 Minimum-Cost Arborescences: A Multi-Phase Greedy Algorithm	177
Solved Exercises	183
Exercises	188
Notes and Further Reading	205
5 Divide and Conquer	209
5.1 A First Recurrence: The Mergesort Algorithm	210
5.2 Further Recurrence Relations	214
5.3 Counting Inversions	221
5.4 Finding the Closest Pair of Points	225
5.5 Integer Multiplication	231
5.6 Convolutions and the Fast Fourier Transform	234
Solved Exercises	242
Exercises	246
Notes and Further Reading	249
6 Dynamic Programming	251
6.1 Weighted Interval Scheduling: A Recursive Procedure	252
6.2 Principles of Dynamic Programming: Memoization or Iteration over Subproblems	258
6.3 Segmented Least Squares: Multi-way Choices	261

Consideraciones importantes

- La motivación para tomar este curso debería ser una sola: aprender
- Relación de confianza profesor-estudiante y queda terminantemente prohibido copiar códigos de colegas en los ejercicios (se puede discutir y trabajar en grupos pero cada quien debe escribir su propio código)
- El profesor no se las sabe todas
- Casi nada del contenido del curso es original
- Las ayudas del curso son para aprovecharlas: grupo, monitor, asesorías, etc.
- Haremos uso de conceptos, teoremas, y métodos probados, [más no nos vamos a concentrar necesariamente en su demostración](#)
- Este curso es un punto de partida, el perfeccionamiento de las habilidades de programación depende de cada uno, de hecho, aún quedarían muchos temas por cubrir

Asesorías

Profesor

Julián Moreno, jmoreno1@unal.edu.co

Lunes 14:00 – 16:00, M8A-311

Monitores

David Antonio Aristizabal, daaristizabalg@unal.edu.co

Día, hora y modalidad por definir

Grupo de WhatsApp

<https://chat.whatsapp.com/KDoqRMlcixN6tNMTUU2xmf>

