

Curso de Programación Algorítmica

Descripción general

Este curso será un reto motivador para quienes quieran introducirse en el fascinante mundo de la algoritmia y la programación competitiva. Aprenderás técnicas y algoritmos esenciales que te ayudarán a pensar de forma lógica y resolver problemas de manera eficiente.

"Una computadora puede hacer cualquier cosa que pueda ser descrita como un algoritmo."
— Alan Turing (*padre de la computación moderna*)

El curso está dirigido a estudiantes con un nivel medio de programación, ya que veremos temas avanzados que requieren una base sólida para no quedarse atrás.

- **Duración:** 25 clases
 - **Lenguajes:** Python y C++
 - **Plataformas:** [HackerRank](#) (principal), [Codeforces](#)
 - **Herramientas:** GitHub para las soluciones, grupo de Whatsapp para dudas y comunidad.
-

Metodología

- 15–30 minutos de teoría por clase
 - Resolución de ejercicios guiados relacionados con el tema
 - Tareas para profundizar (con corrección en la siguiente clase)
 - Cada 5 clases, una sesión de competición para aplicar lo aprendido
 - Entrega de libros y materiales recomendados para estudiar por cuenta propia
 - Se darán ejemplos resueltos y soluciones a los problemas tratados
-

Plan del curso (25 clases)

Módulo 1 – Matemáticas y fundamentos básicos (Clases 1–5)

- [Introducción y estructura del curso](#)
 - Complejidad computacional (Notación Big-O)
 - Complejidad espacial
 - Entrada y salida
- [Números primos](#)
 - Primalidad utilizando fuerza bruta $O(n)$
 - Primalidad optimizada con raíz cuadrada $O(\sqrt{n})$
 - Criba de Eratóstenes
- [Máximo común divisor \(MCD\) y mínimo común múltiplo \(MCM\)](#)
 - MCM: mínimo común múltiplo
 - MCD: máximo común divisor
 - Algoritmo de Euclides
- Combinatoria básica

- Factoriales y binomios
- nCk y aplicaciones simples
- **Competición 1:** Problemas matemáticos sencillos

Módulo 2 – Búsquedas y algoritmos voraces (Clases 6–10)

- Búsqueda binaria:
 - Teoría y ejemplos
 - Aplicaciones: encontrar el valor exacto, buscar límites
- Algoritmos voraces (greedy) 1:
 - Teoría básica
 - Problemas clásicos
- Algoritmos voraces (greedy) 2:
 - Otros ejemplos de problemas
- Técnicas útiles:
 - Sumas parciales, prefix max/min
 - Aplicaciones en intervalos
- **Competición 2:** Problemas de búsqueda y voraces

Módulo 3 – Grafos básicos (Clases 11–15)

- Representación de grafos:
 - Teoría
 - Grafos dirigidos y no dirigidos
 - Listas de adyacencia, matriz de adyacencia
- Recorrido en grafos: DFS (búsqueda en profundidad):
 - Explicación del algoritmo
 - Implementación recursiva
 - Implementación iterativa
- Recorrido en grafos: BFS (búsqueda en anchura):
 - Explicación del algoritmo
 - Implementación
 - BFS en cuadrículas
- Aplicaciones simples de DFS y BFS:
 - Contar componentes
 - Detección de ciclos
- **Competición 3:** Grafos y recorridos básicos

Módulo 4 – Programación dinámica básica (Clases 16–20)

- Introducción a la programación dinámica:
 - Conceptos clave
- DP en una dimensión:
 - Fibonacci
 - Tabla acumulativa
 - Subsecuencia creciente
- Problemas de la mochila 0-1:
 - Problema clásico y variantes

- Problema de contar cambio
- DP en dos dimensiones:
 - Tablas acumulativas
- **Competición 4:** Problemas de programación dinámica

Módulo 5 – Estructuras de datos básicas (Clases 21–24)

- Union-Find:
 - Teoría
 - Operaciones básicas y optimización
- Segment Tree I:
 - Concepto
 - Construcción
- Segment Tree II:
 - Construcción y consultas
- Trie (árbol de prefijos):
 - Ejemplos simples

Clase 25 – Competición final

- Problemas integradores con los temas vistos en el curso

Material complementario

- Libros para aprender C++
- Libros de Programación Competitiva (en Python y C++)
- Soluciones completas disponibles en GitHub
- Libros y materiales recomendados para profundizar
- Comunidad en Whatsapp para consultas y apoyo