

Análisis de Algoritmos

Luis Alfredo Alvarado Rodríguez

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
ESCUELA DE CIENCIAS FÍSICAS Y MATEMÁTICAS

15 de julio de 2025

Sumario

- 1 Presentación del curso
- 2 Herramientas de programación
- 3 Definición de algoritmo
- 4 Necesidad del análisis de algoritmos
- 5 Ámbitos clave del análisis de algoritmos

Presentación del curso (lectura del programa)

- Programa oficial del curso

Repositorio del curso

- Repositorio oficial en GitHub:
github.com/1u1s4/ECFM_M603_2S2025_

Herramientas de programación para el curso

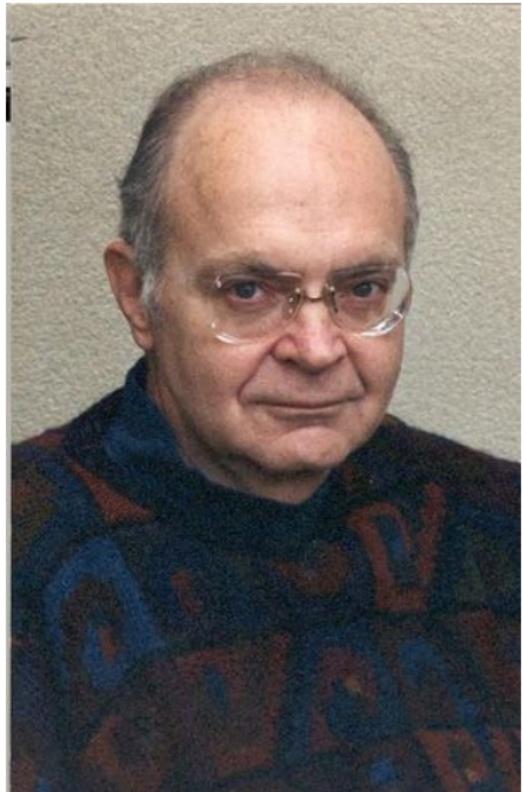
- Python
- VS Code
- Google Colab
- Proyecto Euler
- GitHub

Definición de algoritmo — Knuth (1968)

“Un algoritmo es un conjunto finito de reglas que proporciona una secuencia de operaciones para resolver un tipo específico de problema”.

—Donald,E. Knuth, *The Art of Computer Programming*, Vol. 1 (1968)

Knuth subraya cinco propiedades esenciales: finitud, definitud, entradas, salidas y efectividad; todas garantizan que el proceso termina y produce resultados precisos con recursos asumiblemente realizables.

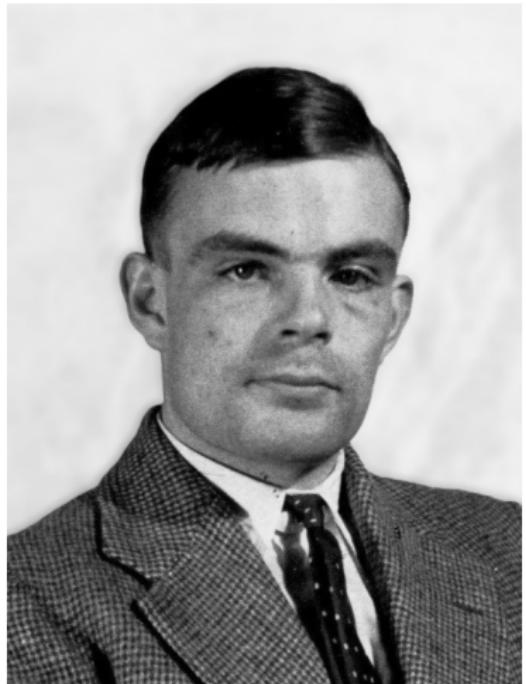


Definición de algoritmo — Turing (1936)

“Cualquier proceso que naturalmente se calificaría de procedimiento efectivo puede ser realizado por una máquina de Turing”.

—Alan M. Turing,
“On Computable Numbers” (1936)

Con su tesis Church-Turing, Turing equiparó el concepto intuitivo de algoritmo con lo que una máquina de Turing puede ejecutar paso a paso, sentando las bases de la computación teórica.

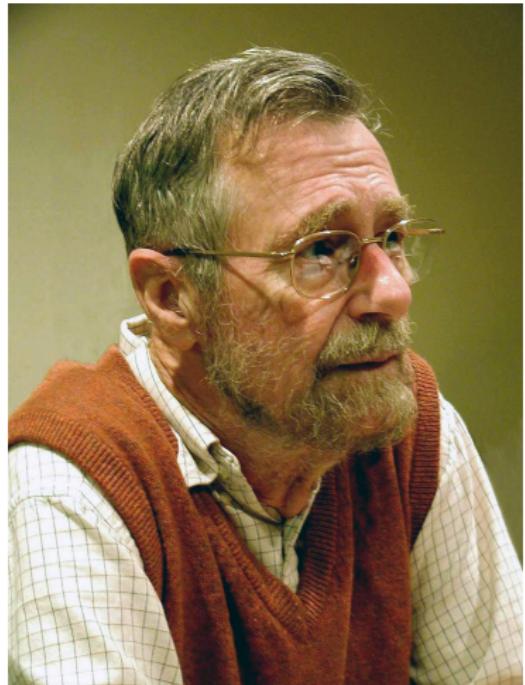


Definición de algoritmo — Dijkstra (1971)

“Un algoritmo es la descripción de un patrón de comportamiento, expresado en términos de un repertorio finito de acciones primitivas bien entendidas”.

—Edsger, W. Dijkstra, A Short Introduction to the Art of Programming (1971)

Dijkstra enfatiza la separación entre la estructura lógica del algoritmo y su implementación concreta, destacando la claridad y corrección formal del diseño.

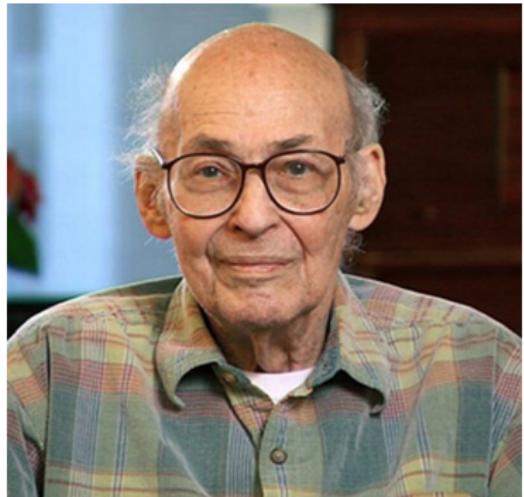


Definición de algoritmo — Minsky (1967)

“Un algoritmo es un procedimiento efectivo: un conjunto de reglas que nos dice, de un momento a otro, con precisión cómo actuar”.

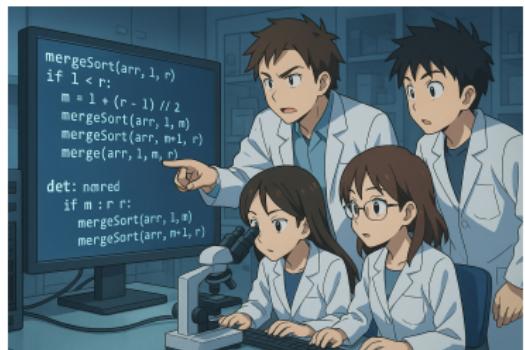
—Marvin Minsky, *Computation: Finite and Infinite Machines* (1967)

Para Minsky, lo esencial es la naturaleza determinista y precisa de las instrucciones que un agente (humano o máquina) puede seguir paso a paso hasta obtener un resultado.



Necesidad del análisis de algoritmos

El análisis de algoritmos es necesario porque nos permite estimar de forma rigurosa los recursos —principalmente tiempo de ejecución y memoria— que un programa consumirá antes de implementarlo o desplegarlo; esto ayuda a elegir entre distintas soluciones, garantiza que el software escalará adecuadamente al crecer los datos, identifica cuellos de botella potenciales y ofrece criterios cuantitativos para optimizar, planificar costos y asegurar la viabilidad de sistemas críticos.



```
mergeSort(arr, l, r)
if l < r:
    m = l + (r - 1) // 2
    mergeSort(arr, l, m)
    mergeSort(arr, m+1, r)
    merge(arr, l, m, r)

det: named
if m := r:
    mergeSort(arr, l, n)
    mergeSort(arr, n+1, r)
    merge(arr, l, n, r)
```

Ámbitos clave del análisis de algoritmos

Procesamiento de grandes volúmenes de datos (Big,Data)

Elegir algoritmos con eficiencias conocidas —por ejemplo, Map-Reduce optimizados o estructuras de índices— hace viable analizar petabytes sin que los tiempos de respuesta se disparen.

Ámbitos clave del análisis de algoritmos

Procesamiento de grandes volúmenes de datos (Big,Data)

Elegir algoritmos con eficiencias conocidas —por ejemplo, Map-Reduce optimizados o estructuras de índices— hace viable analizar petabytes sin que los tiempos de respuesta se disparen.

Criptografía y ciberseguridad

La solidez de un sistema criptográfico depende de que no existan algoritmos *atajo* más rápidos que la fuerza bruta; medir la complejidad permite dimensionar su resistencia real.

Ámbitos clave del análisis de algoritmos

Procesamiento de grandes volúmenes de datos (Big,Data)

Elegir algoritmos con eficiencias conocidas —por ejemplo, Map-Reduce optimizados o estructuras de índices— hace viable analizar petabytes sin que los tiempos de respuesta se disparen.

Criptografía y ciberseguridad

La solidez de un sistema criptográfico depende de que no existan algoritmos *atajo* más rápidos que la fuerza bruta; medir la complejidad permite dimensionar su resistencia real.

Inteligencia artificial y aprendizaje automático

Entrenar modelos implica procesar enormes matrices o grafos; prever el costo temporal y de memoria de algoritmos de optimización (SGD, back-prop, clustering, etc.) evita sobrepasar presupuestos y acelera iteraciones.

Ámbitos clave del análisis de algoritmos

Sistemas en tiempo real y de misión crítica

Desde control de vuelos hasta *trading* algorítmico, es imprescindible garantizar cotas superiores de ejecución (*worst-case*) para que cada tarea se complete dentro del plazo seguro.

Ámbitos clave del análisis de algoritmos

Sistemas en tiempo real y de misión crítica

Desde control de vuelos hasta *trading* algorítmico, es imprescindible garantizar cotas superiores de ejecución (*worst-case*) para que cada tarea se complete dentro del plazo seguro.

Bases de datos y motores de búsqueda

Consultas, ordenamientos e índices dependen de algoritmos con complejidad bien analizada; así se asegura que, al crecer los datos, las operaciones sigan respondiendo en milisegundos y no en minutos.