==

🗺️ ROADMAP DETALLADO - MATEMÁTICAS PARA DESARROLLADOR SOBRESALIENTE

==

Por: Ernesto

Duración: 24 meses (2 años)

Inicio: Enero 2025

Meta: Dominar matemáticas aplicadas a la programación

==

==

📋 VISIÓN GENERAL

==

OBJETIVO PRINCIPAL:

Desarrollar fundamentos matemáticos sólidos aplicados a problemas reales
de programación para convertirme en un desarrollador sobresaliente.

PRINCIPIOS DEL ROADMAP:

✓ Aprender haciendo (70% práctica, 30% teoría)

✓ Cada concepto matemático se implementa en código

✓ Proyectos reales que demuestran dominio

✓ Iterativo: repasar y profundizar constantemente

✓ Consistencia diaria de 30-60 minutos

ESTRUCTURA:

- 24 meses divididos en 6 fases de 4 meses
- Cada mes tiene objetivos específicos medibles
- Proyectos integradores al final de cada fase
- Revisiones mensuales del progreso

==

🎯 AÑO 1: FUNDAMENTOS SÓLIDOS

==

FASE 1: MATEMÁTICA DISCRETA PROFUNDA

Meses 1-4 (Enero - Abril 2025)

Dedicación: 45 min/día

MES 1 - ENERO 2025: GRAFOS Y ÁRBOLES

SEMANA 1: Repaso de Árboles

├─ Teoría (3 días)

│ ├─ Árboles binarios: recorridos (preorden, inorden, postorden)

│ ├─ Árboles binarios de búsqueda (BST)

│ └─ Balanceo de árboles (conceptual)

│
├── Práctica Código (3 días)
│ ├── Implementar BST desde cero en Python
│ ├── Métodos: insert, search, delete
│ └── Recorridos iterativos y recursivos
│
└── Ejercicios (1 día)
└── 10 problemas de árboles de LeetCode Easy

SEMANA 2: Introducción a Grafos
├── Teoría (3 días)
│ ├── Conceptos: vértices, aristas, dirigidos/no dirigidos
│ ├── Representaciones: matriz de adyacencia, lista de adyacencia
│ └── Grafos ponderados
│
├── Práctica Código (3 días)
│ ├── Clase Graph en Python (lista de adyacencia)
│ ├── Métodos: add_vertex, add_edge, remove_edge
│ └── Representación visual con matplotlib
│
└── Ejercicios (1 día)
└── Implementar 3 formas de representar grafos

SEMANA 3: Algoritmos de Recorrido
├── Teoría (2 días)
│ ├── BFS (Breadth-First Search)
│ └── DFS (Depth-First Search)
│
├── Práctica Código (4 días)
│ ├── Implementar BFS iterativo (con cola)
│ ├── Implementar DFS recursivo e iterativo (con pila)
│ ├── Contar componentes conexas
│ └── Detección de ciclos
│
└── Ejercicios (1 día)
└── LeetCode: 5 problemas de BFS/DFS

SEMANA 4: Aplicaciones de Grafos
├── Práctica Código (5 días)
│ ├── Camino más corto sin pesos (BFS)
│ ├── Ordenamiento topológico
│ ├── Verificar si es árbol
│ └── PROYECTO: Visualizador de grafos interactivo
│ • Crear/eliminar nodos y aristas
│ • Visualizar BFS y DFS paso a paso
│ • Detectar ciclos visualmente
│
└── Revisión (2 días)
├── Repasar todos los algoritmos
└── Documentar código en GitHub

ENTREGABLES MES 1:

☐ Repositorio GitHub con todas las implementaciones

☐ 20 problemas de LeetCode resueltos (árboles y grafos básicos)

☐ Proyecto visualizador funcionando

☐ Notas personales de conceptos clave

MES 2 - FEBRERO 2025: ALGORITMOS DE GRAFOS AVANZADOS

SEMANA 1: Caminos Mínimos (Dijkstra)

```
├── Teoría (2 días)
│   ├── Algoritmo de Dijkstra
│   ├── Heap/Priority Queue
│   └── Complejidad: O(V log V + E log V)
│
├── Práctica Código (4 días)
│   ├── Implementar Min Heap desde cero
│   ├── Implementar Dijkstra con heap
│   ├── Reconstruir el camino más corto
│   └── Visualizar el proceso paso a paso
│
└── Ejercicios (1 día)
    └── 3 problemas de caminos mínimos
```

SEMANA 2: Más Algoritmos de Caminos

```
├── Teoría (2 días)
│   ├── Bellman-Ford (grafos con pesos negativos)
│   ├── Floyd-Warshall (todos los pares)
│   └── A* (heurística para pathfinding)
│
├── Práctica Código (4 días)
│   ├── Implementar Bellman-Ford
│   ├── Implementar Floyd-Warshall
│   ├── Implementar A* básico
│   └── Comparar rendimiento de algoritmos
│
└── Ejercicios (1 día)
    └── Casos de uso de cada algoritmo
```

SEMANA 3: Árboles de Expansión Mínima

```
├── Teoría (2 días)
│   ├── Minimum Spanning Tree (MST)
│   ├── Algoritmo de Kruskal
│   ├── Algoritmo de Prim
│   └── Union-Find (Disjoint Set)
│
├── Práctica Código (4 días)
│   ├── Implementar Union-Find con path compression
```

│ ├─ Implementar Kruskal's MST
│ ├─ Implementar Prim's MST
│ └─ Visualizar MST paso a paso
│
└─ Ejercicios (1 día)
   └─ Problemas de MST en LeetCode

SEMANA 4: Proyecto Integrador - Sistema de Rutas
├─ PROYECTO COMPLETO (7 días)
│ DESCRIPCIÓN: Sistema de navegación tipo Google Maps simplificado
│
│ FEATURES:
│ ├─ Cargar mapa de ciudad (nodos = intersecciones, aristas = calles)
│ ├─ Camino más corto entre dos puntos (Dijkstra)
│ ├─ Ruta alternativa (A*)
│ ├─ Visualización del mapa
│ ├─ Tiempos estimados de viaje
│ └─ Detectar calles de un solo sentido (grafo dirigido)
│
│ TECNOLOGÍAS:
│ ├─ Python
│ ├─ Matplotlib para visualización
│ ├─ Clases propias (no usar NetworkX todavía)
│ └─ Datos: crear dataset ficticio o usar OpenStreetMap
│
└─ Presentación (1 día)
   └─ README completo, screenshots, documentación

ENTREGABLES MES 2:
☐ Todos los algoritmos implementados y testeados
☐ 15 problemas más de LeetCode (Medium)
☐ Proyecto de navegación funcionando
☐ Comparación de rendimiento de algoritmos

MES 3 - MARZO 2025: COMBINATORIA Y ANÁLISIS DE COMPLEJIDAD

SEMANA 1: Combinatoria Básica
├─ Teoría (3 días)
│ ├─ Principio de la multiplicación y adición
│ ├─ Permutaciones: P(n,r) = n!/(n-r)!
│ ├─ Combinaciones: C(n,r) = n!/(r!(n-r)!)
│ ├─ Principio del palomar
│ └─ Inclusión-Exclusión (repasar)
│
├─ Práctica Código (3 días)
│ ├─ Generar todas las permutaciones
│ ├─ Generar todas las combinaciones
│ ├─ Subconjuntos de un conjunto (2^n)

│ └─ Project Euler problemas 15-25
│
└─ Ejercicios (1 día)
  └─ 20 ejercicios de combinatoria manual

SEMANA 2: Recursión y Relaciones de Recurrencia
├─ Teoría (2 días)
│ ├─ Fibonacci y variantes
│ ├─ Torres de Hanoi
│ ├─ Relaciones de recurrencia
│ └─ Método maestro
│
├─ Práctica Código (4 días)
│ ├─ Fibonacci: recursivo, memo, dp, matriz
│ ├─ Torres de Hanoi visualizado
│ ├─ Problemas clásicos de backtracking
│ └─ N-Queens problem
│
└─ Ejercicios (1 día)
  └─ Resolver recurrencias de algoritmos conocidos

SEMANA 3: Análisis de Complejidad (Big O)
├─ Teoría (3 días)
│ ├─ Notación O, Ω, Θ
│ ├─ Complejidades comunes: O(1), O(log n), O(n), O(n log n), O(n^2)
│ ├─ Análisis de caso mejor, peor, promedio
│ ├─ Complejidad espacial
│ └─ Amortized analysis
│
├─ Práctica (3 días)
│ ├─ Analizar complejidad de código propio
│ ├─ Optimizar algoritmos ineficientes
│ ├─ Comparar algoritmos de ordenamiento
│ └─ Medir tiempos reales vs teóricos
│
└─ Ejercicios (1 día)
  └─ Calcular Big O de 30 snippets de código

SEMANA 4: Proyecto - Analizador de Complejidad
├─ PROYECTO (6 días)
│ DESCRIPCIÓN: Herramienta que analiza complejidad de código Python
│
│ FEATURES:
│ ├─ Parser de código Python simple
│ ├─ Detectar loops, recursión
│ ├─ Estimar Big O automáticamente
│ ├─ Graficar crecimiento teórico
│ ├─ Benchmark: medir tiempo real
│ └─ Sugerencias de optimización
│

│ BONUS:
│ └─ Comparar diferentes implementaciones del mismo algoritmo
│
└─ Documentación (1 día)

ENTREGABLES MES 3:
☐ 30 problemas de combinatoria resueltos
☐ Todos los algoritmos recursivos clásicos implementados
☐ Proyecto analizador de complejidad
☐ Cheatsheet personal de Big O

MES 4 - ABRIL 2025: LÓGICA Y TEORÍA DE NÚMEROS

SEMANA 1: Lógica Proposicional
├─ Teoría (3 días)
│ ├─ Proposiciones y conectivos lógicos
│ ├─ Tablas de verdad
│ ├─ Equivalencias lógicas
│ ├─ Leyes de De Morgan
│ └─ Implicación y contrapositiva
│
├─ Práctica Código (3 días)
│ ├─ Evaluador de expresiones lógicas
│ ├─ Generador de tablas de verdad
│ ├─ Simplificador de expresiones
│ └─ SAT solver básico
│
└─ Ejercicios (1 día)
└─ 20 problemas de lógica proposicional

SEMANA 2: Teoría de Números Básica
├─ Teoría (3 días)
│ ├─ Divisibilidad y primos
│ ├─ MCD y MCM (Euclides)
│ ├─ Números primos: criba de Eratóstenes
│ ├─ Aritmética modular
│ └─ Congruencias
│
├─ Práctica Código (3 días)
│ ├─ Algoritmo de Euclides (recursivo e iterativo)
│ ├─ Criba de Eratóstenes optimizada
│ ├─ Test de primalidad
│ ├─ Factorización de enteros
│ └─ Operaciones en Zn
│
└─ Ejercicios (1 día)
└─ Project Euler problemas de números primos

SEMANA 3: Aplicaciones de Teoría de Números

- Teoría (2 días)
  - Teorema de Fermat
  - Función φ de Euler
  - Teorema chino del resto
  - Introducción a criptografía (RSA conceptual)
- Práctica Código (4 días)
  - Exponenciación modular rápida
  - Inverso modular
  - Implementar RSA simple
  - Generador de números pseudoaleatorios
- Ejercicios (1 día)
  - Problemas de criptografía básica

SEMANA 4: Proyecto Final Fase 1

- PROYECTO INTEGRADOR FASE 1 (7 días)

  TÍTULO: "Sistema de Análisis de Redes Sociales"

  DESCRIPCIÓN: Analizador de grafos sociales con métricas avanzadas

  COMPONENTES:
  - 1. Estructura de datos
    - Grafo de usuarios (nodos = personas, aristas = amistad)
    - Pesos en aristas (interacciones)
    - Atributos en nodos (perfil)
  - 2. Análisis de grafos
    - Comunidades (componentes conexas)
    - Personas influyentes (grado, betweenness)
    - Camino más corto entre usuarios
    - Clustering coefficient
  - 3. Recomendaciones
    - Amigos sugeridos (amigos de amigos)
    - Comunidades recomendadas
    - Detección de bots (análisis de patrones)
  - 4. Visualización
    - Grafo completo
    - Subgrafos de comunidades
    - Heatmap de interacciones
    - Dashboard con métricas
  - 5. Performance
    - Análisis de complejidad de cada operación
    - Optimización para grafos grandes (>10k nodos)
    - Benchmarking

│ TECNOLOGÍAS:
│ ├─ Python (código propio, sin NetworkX aún)
│ ├─ Matplotlib/Seaborn para viz
│ ├─ Dataset: generar o usar Kaggle
│ └─ GitHub con documentación completa
│
└─ Presentación (día extra)
  └─ Video demo + README + informe técnico

ENTREGABLES MES 4 / FASE 1:
☐ Proyecto integrador funcionando 100%
☐ Repositorio GitHub organizado con TODO el código de la fase
☐ 100+ problemas de LeetCode resueltos (fácil y medio)
☐ Notas consolidadas de Matemática Discreta
☐ Portfolio con 4 proyectos completados

HABILIDADES ADQUIRIDAS FASE 1:
✓ Dominio de estructuras de datos avanzadas
✓ Implementación de algoritmos de grafos
✓ Análisis de complejidad algorítmica
✓ Combinatoria y conteo aplicado
✓ Pensamiento recursivo
✓ Resolución de problemas complejos

FASE 2: ÁLGEBRA LINEAL APLICADA
Meses 5-8 (Mayo - Agosto 2025)
Dedicación: 60 min/día

MES 5 - MAYO 2025: VECTORES Y MATRICES

SEMANA 1: Fundamentos de Vectores
├─ Teoría (2 días)
│ ├─ Vectores en R^2 y R^3
│ ├─ Operaciones: suma, resta, escalar
│ ├─ Producto punto (dot product)
│ ├─ Producto cruz (cross product)
│ ├─ Norma y normalización
│ └─ Proyección vectorial
│
├─ Videos (1 día)
│ └─ 3Blue1Brown: Essence of Linear Algebra (capítulos 1-5)
│ ⭐⭐⭐⭐⭐ MIRAR COMPLETOS, SON ESPECTACULARES
│
├─ Práctica Código (3 días)
│ ├─ Clase Vector desde cero (sin NumPy)
│ │ ├─ __init__, __add__, __sub__, __mul__

```
│   │   ├── dot(), cross(), norm(), normalize()
│   │   └── angle_between(), projection()
│   │
│   ├── Visualización de vectores en 2D
│   └── Operaciones visualizadas paso a paso
│
└── Ejercicios (1 día)
    └── 20 problemas de vectores

SEMANA 2: Matrices Básicas
├── Teoría (2 días)
│   ├── Definición de matriz
│   ├── Operaciones: suma, resta, multiplicación escalar
│   ├── Multiplicación de matrices
│   ├── Transpuesta
│   ├── Identidad y matrices especiales
│   └── Determinante 2x2 y 3x3
│
├── Videos (1 día)
│   └── 3Blue1Brown: Linear transformations (capítulos 3-4)
│
├── Práctica Código (3 días)
│   ├── Clase Matrix desde cero
│   │   ├── Operaciones básicas
│   │   ├── Multiplicación de matrices (naive)
│   │   ├── Transpuesta
│   │   └── Determinante (método de cofactores)
│   │
│   ├── Optimización: multiplicación de Strassen (opcional)
│   └── Benchmarking: comparar con NumPy
│
└── Ejercicios (1 día)
    └── Operaciones matriciales manual y código

SEMANA 3: Transformaciones Lineales
├── Teoría (2 días)
│   ├── Transformaciones como matrices
│   ├── Rotación, escalado, reflexión
│   ├── Traslación (transformaciones afines)
│   ├── Composición de transformaciones
│   └── Interpretación geométrica
│
├── Videos (1 día)
│   └── 3Blue1Brown: Matrix multiplication as composition
│
├── Práctica Código (3 días)
│   ├── Matrices de transformación 2D
│   │   ├── rotation_matrix(angle)
│   │   ├── scale_matrix(sx, sy)
│   │   ├── shear_matrix(...)
```

│ │ └─ reflection_matrix(axis)
│ │
│ ├─ Aplicar transformaciones a puntos/figuras
│ └─ Animación de transformaciones
│
└─ Ejercicios (1 día)
  └─ Componer múltiples transformaciones

SEMANA 4: Proyecto - Motor Gráfico 2D
├─ PROYECTO (7 días)
│ TÍTULO: "Simple 2D Graphics Engine"
│
│ DESCRIPCIÓN: Motor de renderizado 2D usando solo álgebra lineal
│
│ FEATURES:
│ ├─ 1. Primitivas básicas
│ │ ├─ Puntos, líneas, triángulos, rectángulos
│ │ └─ Representación vectorial
│ │
│ ├─ 2. Transformaciones
│ │ ├─ Trasladar, rotar, escalar figuras
│ │ ├─ Transformaciones jerárquicas (padre-hijo)
│ │ └─ Animaciones suaves
│ │
│ ├─ 3. Cámara 2D
│ │ ├─ Viewport y transformación de vista
│ │ ├─ Zoom in/out
│ │ └─ Pan (mover cámara)
│ │
│ ├─ 4. Interactividad
│ │ ├─ Seleccionar y mover objetos
│ │ ├─ Aplicar transformaciones con teclado
│ │ └─ UI simple
│ │
│ └─ 5. Demo
│ ├─ Solar system simulation
│ ├─ Rotating shapes
│ └─ Interactive scene builder
│
│ RESTRICCIÓN: NO usar librerías de transformaciones,
│ solo Matplotlib para dibujar puntos/líneas
│
└─ Documentación completa

ENTREGABLES MES 5:
☐ Librería propia de vectores y matrices
☐ Motor gráfico 2D funcionando
☐ 10 animaciones/demos diferentes
☐ Comparación de rendimiento con NumPy

MES 6 - JUNIO 2025: SISTEMAS DE ECUACIONES Y ESPACIOS VECTORIALES

SEMANA 1: Sistemas de Ecuaciones Lineales
├─ Teoría (2 días)
│ ├─ Representación matricial Ax = b
│ ├─ Eliminación Gaussiana
│ ├─ Forma escalonada
│ ├─ Gauss-Jordan
│ ├─ Matriz inversa
│ └─ Casos: solución única, infinitas, sin solución
│
├─ Videos (1 día)
│ └─ 3Blue1Brown: Inverse matrices, rank, column space
│
├─ Práctica Código (3 días)
│ ├─ Implementar eliminación Gaussiana
│ ├─ Implementar Gauss-Jordan
│ ├─ Calcular matriz inversa
│ ├─ Resolver sistemas lineales
│ └─ Análisis de casos especiales
│
└─ Ejercicios (1 día)
   └─ 20 sistemas de ecuaciones de diferentes tipos

SEMANA 2: Espacios Vectoriales
├─ Teoría (3 días)
│ ├─ Definición de espacio vectorial
│ ├─ Subespacios
│ ├─ Independencia lineal
│ ├─ Base y dimensión
│ ├─ Column space, row space, null space
│ └─ Rank y nullity
│
├─ Videos (1 día)
│ └─ 3Blue1Brown: Abstract vector spaces
│
├─ Práctica Código (2 días)
│ ├─ Verificar independencia lineal
│ ├─ Encontrar base de un espacio
│ ├─ Calcular dimensión
│ └─ Column space y null space
│
└─ Ejercicios (1 día)
   └─ Problemas conceptuales de espacios vectoriales

SEMANA 3: Eigenvalues y Eigenvectors
├─ Teoría (3 días)
│ ├─ Definición de eigenvalue/eigenvector

```
│   ├── Ecuación característica
│   ├── Diagonalización
│   ├── Interpretación geométrica
│   └── Aplicaciones (sistemas dinámicos, Google PageRank)
│
├── Videos (1 día)
│   └── 3Blue1Brown: Eigenvectors and eigenvalues
│
├── Práctica Código (2 días)
│   ├── Calcular eigenvalues (método de potencias)
│   ├── Encontrar eigenvectors
│   ├── Diagonalizar matrices
│   └── Visualizar transformaciones con eigenvectors
│
└── Ejercicios (1 día)
    └── Problemas de eigenvalues/eigenvectors

SEMANA 4: Proyecto - Sistema de Ecuaciones Interactivo
├── PROYECTO (7 días)
│   TÍTULO: "Linear Algebra Visualizer & Solver"
│
│   DESCRIPCIÓN: Suite de herramientas para álgebra lineal
│
│   MÓDULOS:
│   ├── 1. Solver de Sistemas
│   │   ├── Input: sistema de ecuaciones
│   │   ├── Resolver paso a paso (Gauss-Jordan)
│   │   ├── Mostrar cada operación elemental
│   │   ├── Visualizar geométricamente (2D/3D)
│   │   └── Clasificar solución
│   │
│   ├── 2. Visualizador de Transformaciones
│   │   ├── Input: matriz de transformación
│   │   ├── Aplicar a grid de puntos
│   │   ├── Mostrar eigenvectors
│   │   ├── Animación de transformación
│   │   └── Descomposición en rotación/escala
│   │
│   ├── 3. Calculadora de Eigenvalues
│   │   ├── Calcular eigenvalues/eigenvectors
│   │   ├── Verificar con método de potencias
│   │   ├── Visualizar eigenvectors
│   │   └── Mostrar diagonalización
│   │
│   └── 4. Demos educativas
│       ├── PageRank de Google simplificado
│       ├── Sistema masa-resorte
│       └── Población de conejos/zorros (Lotka-Volterra)
│
│   TECH STACK:
```

│ ├─ Python + tu librería de álgebra lineal
│ ├─ Interface gráfica (Tkinter o web con Flask)
│ ├─ Matplotlib para visualizaciones
│ └─ Documentación tipo tutorial
│
└─ Testing y pulido

ENTREGABLES MES 6:
☐ Suite de herramientas de álgebra lineal
☐ 3 demos de aplicaciones reales
☐ Tutorial escrito de cada funcionalidad
☐ Comparación con software como Octave/MATLAB

MES 7 - JULIO 2025: DESCOMPOSICIÓN DE MATRICES Y OPTIMIZACIÓN

SEMANA 1: Descomposiciones de Matrices
├─ Teoría (3 días)
│ ├─ LU Decomposition
│ ├─ QR Decomposition (Gram-Schmidt)
│ ├─ Cholesky Decomposition
│ ├─ SVD (Singular Value Decomposition) - intro
│ └─ Aplicaciones de cada una
│
├─ Videos (1 día)
│ └─ MIT 18.06 - Factorizations (Strang)
│
├─ Práctica Código (2 días)
│ ├─ Implementar LU decomposition
│ ├─ Implementar QR (Gram-Schmidt)
│ ├─ Resolver sistemas usando factorizaciones
│ └─ Comparar eficiencia
│
└─ Ejercicios (1 día)
  └─ Casos de uso de cada descomposición

SEMANA 2: Least Squares y Regresión
├─ Teoría (2 días)
│ ├─ Problema de mínimos cuadrados
│ ├─ Normal equations: A^T A x = A^T b
│ ├─ Regresión lineal como least squares
│ ├─ Polynomial fitting
│ └─ Regularización (Ridge, Lasso - conceptual)
│
├─ Práctica Código (4 días)
│ ├─ Resolver least squares (QR decomposition)
│ ├─ Regresión lineal desde cero
│ ├─ Regresión polinomial
│ ├─ Visualización de ajuste

```
│   └── Métricas: MSE, R², residuals
│
└── Ejercicios (1 día)
    └── Datasets reales (Kaggle)

SEMANA 3: SVD y PCA
├── Teoría (3 días)
│   ├── SVD: A = UΣV^T
│   ├── Interpretación geométrica
│   ├── Aproximación de rango bajo
│   ├── PCA (Principal Component Analysis)
│   ├── Reducción de dimensionalidad
│   └── Aplicaciones: compresión, ML
│
├── Videos (1 día)
│   └── StatQuest: PCA explained
│
├── Práctica Código (2 días)
│   ├── SVD usando NumPy (comparar con implementación propia)
│   ├── Compresión de imágenes con SVD
│   ├── PCA desde cero
│   ├── Visualizar componentes principales
│   └── Reducir dimensionalidad de dataset
│
└── Ejercicios (1 día)
    └── Aplicar PCA a diferentes datasets

SEMANA 4: Proyecto - Image Processor
├── PROYECTO (7 días)
│   TÍTULO: "Linear Algebra Image Processing Suite"
│
│   DESCRIPCIÓN: Procesamiento de imágenes usando álgebra lineal
│
│   FEATURES:
│   ├── 1. Transformaciones básicas
│   │   ├── Rotación, escala, shearing
│   │   ├── Flip horizontal/vertical
│   │   └── Composición de transformaciones
│   │
│   ├── 2. Filtros lineales
│   │   ├── Blur (convolución con matriz)
│   │   ├── Sharpen
│   │   ├── Edge detection (Sobel)
│   │   └── Custom kernel creator
│   │
│   ├── 3. Compresión con SVD
│   │   ├── Descomponer imagen en RGB
│   │   ├── Aplicar SVD a cada canal
│   │   ├── Reconstruir con k valores singulares
│   │   ├── Slider para variar compresión
```

│ │ └─ Comparar tamaño vs calidad
│ │
│ ├─ 4. Reducción de dimensionalidad
│ │ ├─ PCA en imágenes
│ │ ├─ Reducir features
│ │ └─ Reconstruir imagen
│ │
│ └─ 5. Regresión en imágenes
│ ├─ Ajustar tendencia de color
│ ├─ Interpolación
│ └─ Inpainting básico
│
│ INTERFACE:
│ ├─ GUI con sliders y controles
│ ├─ Before/After comparison
│ ├─ Exportar resultados
│ └─ Batch processing
│
└─ Demo y documentación

ENTREGABLES MES 7:
☐ Image processing suite completo
☐ 5+ filtros diferentes implementados
☐ Compresión SVD funcionando
☐ Análisis de compresión vs calidad

MES 8 - AGOSTO 2025: ÁLGEBRA LINEAL EN MACHINE LEARNING

SEMANA 1: Gradientes y Optimización
├─ Teoría (3 días)
│ ├─ Gradiente de funciones multivariables
│ ├─ Gradient descent
│ ├─ Jacobian y Hessian
│ ├─ Learning rate
│ └─ Convergencia
│
├─ Práctica Código (3 días)
│ ├─ Gradient descent desde cero
│ ├─ Optimizar función cuadrática
│ ├─ Visualizar proceso de descenso
│ ├─ Stochastic gradient descent
│ └─ Mini-batch gradient descent
│
└─ Ejercicios (1 día)
└─ Optimizar diferentes funciones

SEMANA 2: Neural Networks Básico (matemáticas)
├─ Teoría (3 días)

│ ├─ Perceptrón
│ ├─ Activación functions
│ ├─ Feedforward: multiplicaciones matriciales
│ ├─ Backpropagation (chain rule)
│ ├─ Weight updates
│ └─ Loss functions
│
├─ Práctica Código (3 días)
│ ├─ Implementar perceptrón
│ ├─ Red neuronal simple (1 hidden layer)
│ ├─ Backpropagation desde cero
│ ├─ Entrenar en XOR problem
│ └─ Visualizar decision boundaries
│
└─ Ejercicios (1 día)
   └─ Entrenar en diferentes datasets

SEMANA 3: Aplicaciones de Álgebra Lineal en ML
├─ Teoría (2 días)
│ ├─ Distancia y similitud (Euclidean, cosine)
│ ├─ K-Means clustering (álgebra lineal)
│ ├─ K-Nearest Neighbors
│ ├─ Support Vector Machines (conceptual)
│ └─ Kernel trick
│
├─ Práctica Código (4 días)
│ ├─ Implementar K-Means desde cero
│ ├─ KNN classifier
│ ├─ Comparar métricas de distancia
│ └─ Visualizar clusters y fronteras
│
└─ Ejercicios (1 día)
   └─ Clustering de datasets reales

SEMANA 4: Proyecto Final Fase 2
├─ PROYECTO INTEGRADOR FASE 2 (7 días)
│ TÍTULO: "Recommender System from Scratch"
│
│ DESCRIPCIÓN: Sistema de recomendaciones tipo Netflix/Amazon
│   usando SOLO álgebra lineal
│
│ COMPONENTES:
│ ├─ 1. Data representation
│ │ ├─ User-Item matrix
│ │ ├─ Sparse matrix handling
│ │ └─ Feature vectors
│ │
│ ├─ 2. Collaborative Filtering
│ │ ├─ User-based: similitud de coseno
│ │ ├─ Item-based: similitud de ítems

│ │ ├─ Matrix factorization (SVD)
│ │ └─ Predicción de ratings
│ │
│ ├─ 3. Content-based Filtering
│ │ ├─ Feature extraction
│ │ ├─ Vectorización de contenido
│ │ ├─ Similitud de contenido
│ │ └─ Hybrid approach
│ │
│ ├─ 4. Dimensionality Reduction
│ │ ├─ PCA en features
│ │ ├─ Latent factors (SVD)
│ │ └─ Visualización en 2D/3D
│ │
│ ├─ 5. Evaluation
│ │ ├─ Train/test split
│ │ ├─ RMSE, MAE
│ │ ├─ Precision@K, Recall@K
│ │ └─ A/B testing simulado
│ │
│ └─ 6. Demo Application
│ ├─ Web interface (Flask)
│ ├─ User profile
│ ├─ Recomendaciones personalizadas
│ ├─ Explicación de por qué se recomienda
│ └─ Filtros y categorías
│
│ DATASETS:
│ ├─ MovieLens (películas)
│ ├─ O dataset propio de música/libros
│ └─ Mínimo 10k usuarios, 1k ítems
│
│ RESTRICCIÓN: Implementar algoritmos desde cero,
│ puede usar NumPy pero no sklearn
│
└─ Presentación y deployment

ENTREGABLES MES 8 / FASE 2:

☐ Sistema de recomendaciones funcionando
☐ Web app deployada
☐ Informe técnico comparando métodos
☐ Portfolio actualizado con proyectos de álgebra lineal
☐ Todas las implementaciones en GitHub

HABILIDADES ADQUIRIDAS FASE 2:

✓ Dominio completo de álgebra lineal aplicada
✓ Implementación de algoritmos de ML desde cero
✓ Procesamiento y análisis de imágenes
✓ Optimización numérica
✓ Reducción de dimensionalidad

✓ Sistemas de recomendación

FASE 3: PROBABILIDAD Y ESTADÍSTICA
Meses 9-12 (Septiembre - Diciembre 2025)
Dedicación: 60 min/día

MES 9 - SEPTIEMBRE 2025: PROBABILIDAD FUNDAMENTAL

SEMANA 1: Fundamentos de Probabilidad
├─ Teoría (3 días)
│ ├─ Espacio muestral y eventos
│ ├─ Axiomas de probabilidad
│ ├─ Probabilidad condicional
│ ├─ Independencia de eventos
│ ├─ Regla de la multiplicación
│ └─ Regla de la suma (eventos mutuamente excluyentes)
│
├─ Práctica Código (3 días)
│ ├─ Simulador de experimentos aleatorios
│ ├─ Dados, monedas, cartas
│ ├─ Verificar probabilidades teóricas vs simuladas
│ ├─ Ley de grandes números
│ └─ Visualización de convergencia
│
└─ Ejercicios (1 día)
└─ 30 problemas de probabilidad básica

SEMANA 2: Teorema de Bayes
├─ Teoría (2 días)
│ ├─ Partición del espacio muestral
│ ├─ Teorema de probabilidad total
│ ├─ Teorema de Bayes
│ ├─ Prior, likelihood, posterior
│ └─ Aplicaciones: diagnóstico, spam filtering
│
├─ Práctica Código (4 días)
│ ├─ Implementar Naive Bayes classifier
│ ├─ Spam filter desde cero
│ ├─ Text classification
│ ├─ Bayesian updating visualizado
│ └─ Comparar con sklearn
│
└─ Ejercicios (1 día)
└─ Problemas de Bayes aplicados

SEMANA 3: Variables Aleatorias Discretas

├── Teoría (3 días)
│ ├── Definición de variable aleatoria
│ ├── PMF (Probability Mass Function)
│ ├── CDF (Cumulative Distribution Function)
│ ├── Esperanza y varianza
│ ├── Distribución Bernoulli
│ ├── Distribución Binomial
│ └── Distribución de Poisson
│
├── Práctica Código (3 días)
│ ├── Clase DiscreteRV
│ ├── Calcular esperanza, varianza
│ ├── Simular distribuciones
│ ├── Graficar PMF y CDF
│ └── Aplicaciones reales
│
└── Ejercicios (1 día)
└── Problemas con diferentes distribuciones

SEMANA 4: Variables Aleatorias Continuas
├── Teoría (3 días)
│ ├── PDF (Probability Density Function)
│ ├── CDF para continuas
│ ├── Distribución Uniforme
│ ├── Distribución Exponencial
│ ├── Distribución Normal (Gaussiana)
│ ├── Teorema del Límite Central
│ └── Z-scores y tabla normal
│
├── Práctica Código (3 días)
│ ├── Clase ContinuousRV
│ ├── Generar muestras de distribuciones
│ ├── Visualizar PDFs
│ ├── Demostrar Teorema Límite Central
│ └── Aplicaciones: tiempos de espera, errores
│
└── Ejercicios (1 día)
└── Problemas de distribuciones continuas

ENTREGABLES MES 9:
☐ Librería de probabilidad propia
☐ Naive Bayes classifier funcionando
☐ 50 problemas de probabilidad resueltos
☐ Visualizaciones de todas las distribuciones

MES 10 - OCTUBRE 2025: ESTADÍSTICA DESCRIPTIVA E INFERENCIAL

SEMANA 1: Estadística Descriptiva

├── Teoría (2 días)
│   ├── Medidas de tendencia central (media, mediana, moda)
│   ├── Medidas de dispersión (rango, varianza, desv. estándar)
│   ├── Cuartiles y percentiles
│   ├── Outliers (IQR method)
│   ├── Skewness y kurtosis
│   └── Visualizaciones: histogramas, boxplots, violinplots
│
├── Práctica Código (4 días)
│   ├── Implementar todas las medidas desde cero
│   ├── Análisis exploratorio de datos (EDA)
│   ├── Detector de outliers
│   ├── Reportes automáticos de datasets
│   └── Dashboard de estadísticas
│
└── Ejercicios (1 día)
    └── Analizar 5 datasets de Kaggle

SEMANA 2: Correlación y Regresión
├── Teoría (2 días)
│   ├── Covarianza
│   ├── Correlación de Pearson
│   ├── Correlación de Spearman
│   ├── Correlación ≠ Causalidad
│   ├── Regresión lineal simple (repaso)
│   ├── Regresión múltiple
│   └── R^2 y coeficiente de determinación
│
├── Práctica Código (4 días)
│   ├── Calcular correlaciones
│   ├── Heatmap de correlación
│   ├── Regresión múltiple desde cero
│   ├── Feature selection
│   └── Análisis de residuales
│
└── Ejercicios (1 día)
    └── Predecir variables en datasets reales

SEMANA 3: Inferencia Estadística
├── Teoría (3 días)
│   ├── Población vs muestra
│   ├── Distribución muestral
│   ├── Error estándar
│   ├── Intervalos de confianza
│   ├── Hipótesis nula y alternativa
│   ├── Tipos de errores (Tipo I, Tipo II)
│   ├── P-value
│   ├── Pruebas t (t-test)
│   └── Chi-cuadrado
│

├─ Práctica Código (3 días)
│ ├─ Simulación de muestreo
│ ├─ Calcular intervalos de confianza
│ ├─ Implementar t-test
│ ├─ Chi-cuadrado test
│ └─ Interpretar p-values
│
└─ Ejercicios (1 día)
   └─ Tests de hipótesis en datasets

SEMANA 4: Proyecto - Statistical Analysis Tool
├─ PROYECTO (7 días)
│ TÍTULO: "AutoStats: Automated Statistical Analysis"
│
│ DESCRIPCIÓN: Herramienta que analiza datasets automáticamente
│
│ FEATURES:
│ ├─ 1. Data Profiling
│ │ ├─ Cargar CSV/Excel
│ │ ├─ Detectar tipos de datos
│ │ ├─ Missing values analysis
│ │ ├─ Estadísticas descriptivas completas
│ │ └─ Distribution fitting
│ │
│ ├─ 2. Visualización automática
│ │ ├─ Histogramas
│ │ ├─ Boxplots
│ │ ├─ Scatter matrices
│ │ ├─ Correlation heatmap
│ │ └─ Pair plots
│ │
│ ├─ 3. Outlier Detection
│ │ ├─ IQR method
│ │ ├─ Z-score method
│ │ ├─ Isolation Forest (opcional)
│ │ └─ Visualización de outliers
│ │
│ ├─ 4. Hypothesis Testing
│ │ ├─ Suggest appropriate tests
│ │ ├─ Execute tests
│ │ ├─ Interpret results
│ │ └─ Recommendations
│ │
│ ├─ 5. Regression Analysis
│ │ ├─ Auto feature selection
│ │ ├─ Multiple regression
│ │ ├─ Diagnostics plots
│ │ └─ Predictions
│ │
│ └─ 6. Report Generation

│   ├── PDF report con todos los análisis
│   ├── Gráficos embebidos
│   ├── Interpretaciones en lenguaje natural
│   └── Recomendaciones de análisis adicional
│
│ TECH STACK:
│ ├── Python + Pandas + Matplotlib/Seaborn
│ ├── Web UI (Streamlit o Flask)
│ ├── Export a PDF (reportlab)
│ └── GitHub con ejemplos
│
└── Testing con datasets variados

ENTREGABLES MES 10:
☐ AutoStats tool funcionando
☐ 10 reports de diferentes datasets
☐ Tutorial de uso
☐ Comparación con herramientas comerciales

MES 11 - NOVIEMBRE 2025: PROBABILIDAD APLICADA Y PROCESOS

SEMANA 1: Distribuciones Multivariables
├── Teoría (3 días)
│ ├── Joint distributions
│ ├── Marginal distributions
│ ├── Conditional distributions
│ ├── Independence
│ ├── Covariance y correlation
│ └── Multivariate normal distribution
│
├── Práctica Código (3 días)
│ ├── Simular distribuciones conjuntas
│ ├── Visualizar 2D distributions
│ ├── Calcular marginales
│ ├── Conditional sampling
│ └── Multivariate Gaussian
│
└── Ejercicios (1 día)
  └── Problemas de probabilidad conjunta

SEMANA 2: Cadenas de Markov
├── Teoría (3 días)
│ ├── Definición de cadena de Markov
│ ├── Matriz de transición
│ ├── Estados y transiciones
│ ├── Estado estacionario
│ ├── Cadenas absorbentes
│ └── Aplicaciones: PageRank, modelado de textos

│
├─ Práctica Código (3 días)
│ ├─ Implementar Markov Chain
│ ├─ Simular estados
│ ├─ Encontrar distribución estacionaria
│ ├─ Text generation con Markov
│ └─ PageRank implementado
│
└─ Ejercicios (1 día)
   └─ Modelar sistemas con Markov chains

SEMANA 3: Monte Carlo Methods
├─ Teoría (2 días)
│ ├─ Simulación de Monte Carlo
│ ├─ Estimación de π
│ ├─ Integración Monte Carlo
│ ├─ Importance sampling
│ └─ MCMC (Markov Chain Monte Carlo) - intro
│
├─ Práctica Código (4 días)
│ ├─ Calcular π con Monte Carlo
│ ├─ Integración numérica
│ ├─ Optimización estocástica
│ ├─ Simulación de sistemas complejos
│ └─ Análisis de convergencia
│
└─ Ejercicios (1 día)
   └─ Problemas que requieren simulación

SEMANA 4: Bootstrapping y Resampling
├─ Teoría (2 días)
│ ├─ Bootstrap method
│ ├─ Intervalos de confianza bootstrap
│ ├─ Cross-validation
│ ├─ Permutation tests
│ └─ Aplicaciones en ML
│
├─ Práctica Código (4 días)
│ ├─ Implementar bootstrap
│ ├─ Bootstrap confidence intervals
│ ├─ K-fold cross-validation
│ ├─ Leave-one-out CV
│ └─ Permutation tests
│
└─ Ejercicios (1 día)
   └─ Validar modelos con resampling

ENTREGABLES MES 11:
☐ Librería de métodos Monte Carlo
☐ Text generator con Markov chains

☐ PageRank implementado

☐ Bootstrap toolkit

MES 12 - DICIEMBRE 2025: PROBABILIDAD EN MACHINE LEARNING

SEMANA 1: Bayesian Machine Learning

├─ Teoría (3 días)
│ ├─ Bayesian vs Frequentist
│ ├─ Prior, likelihood, posterior
│ ├─ Conjugate priors
│ ├─ Bayesian inference
│ ├─ Bayesian linear regression
│ └─ Bayesian networks - intro
│
├─ Práctica Código (3 días)
│ ├─ Bayesian updating
│ ├─ Bayesian linear regression
│ ├─ Comparison con regresión frecuentista
│ ├─ Uncertainty quantification
│ └─ Simple Bayesian network
│
└─ Ejercicios (1 día)
   └─ Problemas de inferencia Bayesiana

SEMANA 2: Expectation-Maximization (EM)

├─ Teoría (3 días)
│ ├─ Maximum Likelihood Estimation
│ ├─ Missing data problem
│ ├─ EM Algorithm
│ ├─ Gaussian Mixture Models (GMM)
│ └─ K-Means como caso especial
│
├─ Práctica Código (3 días)
│ ├─ Implementar EM algorithm
│ ├─ GMM desde cero
│ ├─ Clustering con GMM
│ ├─ Comparar con K-Means
│ └─ Visualizar convergencia
│
└─ Ejercicios (1 día)
   └─ Aplicar GMM a diferentes datasets

SEMANA 3: Hidden Markov Models

├─ Teoría (3 días)
│ ├─ Definición de HMM
│ ├─ Forward algorithm
│ ├─ Viterbi algorithm
│ ├─ Baum-Welch algorithm

```
│   └── Aplicaciones: speech recognition, POS tagging
│
├── Práctica Código (3 días)
│   ├── Implementar HMM
│   ├── Forward-backward algorithm
│   ├── Viterbi para secuencias
│   ├── Entrenar HMM (Baum-Welch)
│   └── Aplicación: POS tagger simple
│
└── Ejercicios (1 día)
    └── Secuencias con HMM

SEMANA 4: Proyecto Final Fase 3
├── PROYECTO INTEGRADOR FASE 3 (7 días)
│   TÍTULO: "Probabilistic Prediction System"
│
│   DESCRIPCIÓN: Sistema de predicción con cuantificación de incertidumbre
│
│   OPCIÓN A: Predictor de series temporales
│   ├── Cargar serie temporal (stock, weather, etc.)
│   ├── Análisis estadístico de la serie
│   ├── Modelo probabilístico (ARIMA, HMM, o Bayesian)
│   ├── Predicciones con intervalos de confianza
│   ├── Bootstrap para uncertainty quantification
│   ├── Visualización de predicciones
│   └── Comparación de modelos
│
│   OPCIÓN B: Fraud Detection System
│   ├── Dataset de transacciones
│   ├── Feature engineering
│   ├── Bayesian classifier
│   ├── GMM para detección de anomalías
│   ├── Probabilidades de fraude
│   ├── Threshold optimization
│   └── Evaluation metrics
│
│   OPCIÓN C: A/B Testing Platform
│   ├── Simulador de experimentos
│   ├── Hypothesis testing automatizado
│   ├── Bayesian A/B testing
│   ├── Sequential testing
│   ├── Sample size calculator
│   ├── Power analysis
│   └── Dashboard de resultados
│
│   REQUERIMIENTOS COMUNES:
│   ├── Implementaciones desde cero
│   ├── Visualizaciones claras
│   ├── Intervalos de confianza
│   ├── Uncertainty quantification
```

│ ├─ Testing exhaustivo
│ └─ Deployment (web app)
│
└─ Presentación final

ENTREGABLES MES 12 / FASE 3:
☐ Proyecto probabilístico desplegado
☐ Todas las implementaciones en GitHub
☐ Report técnico de métodos probabilísticos
☐ Portfolio con 3 fases completadas

HABILIDADES ADQUIRIDAS FASE 3:
✓ Fundamentos sólidos de probabilidad
✓ Estadística inferencial aplicada
✓ Métodos de Monte Carlo
✓ Machine Learning probabilístico
✓ Manejo de incertidumbre
✓ A/B testing y experimentación

==========

🎯 AÑO 2: ESPECIALIZACIÓN Y MAESTRÍA

==========

FASE 4: CÁLCULO Y OPTIMIZACIÓN
Meses 13-16 (Enero - Abril 2026)
Dedicación: 60 min/día

MES 13 - ENERO 2026: CÁLCULO DIFERENCIAL

SEMANA 1-2: Límites y Continuidad
├─ Teoría
│ ├─ Concepto de límite
│ ├─ Propiedades de límites
│ ├─ Límites infinitos
│ ├─ Continuidad
│ └─ Teorema del valor intermedio
│
├─ Videos
│ └─ 3Blue1Brown: Essence of Calculus (caps 1-3)
│
├─ Práctica Código
│ ├─ Calcular límites numéricamente
│ ├─ Visualizar límites
│ └─ Verificar continuidad
│
└─ Ejercicios

└─ 40 problemas de límites

SEMANA 3-4: Derivadas
├─ Teoría
│ ├─ Definición de derivada
│ ├─ Reglas de derivación
│ ├─ Chain rule, product rule, quotient rule
│ ├─ Derivadas de funciones comunes
│ ├─ Derivadas implícitas
│ └─ Aplicaciones: tasas de cambio, velocidad
│
├─ Videos
│ └─ 3Blue1Brown: Derivatives (caps 2-5)
│
├─ Práctica Código
│ ├─ Derivación numérica
│ ├─ Automatic differentiation básico
│ ├─ Visualizar tangentes
│ └─ Problemas de optimización simple
│
└─ Ejercicios
└─ 50 problemas de derivadas

PROYECTO MES 13:
└─ Visualizador interactivo de derivadas
├─ Graficar función y derivada
├─ Mostrar tangentes
├─ Animaciones de límite de derivada
└─ Optimización visual

MES 14 - FEBRERO 2026: CÁLCULO INTEGRAL Y CÁLCULO MULTIVARIABLE

SEMANA 1-2: Integrales
├─ Teoría
│ ├─ Integral definida (área bajo curva)
│ ├─ Teorema fundamental del cálculo
│ ├─ Antiderivadas
│ ├─ Técnicas de integración
│ └─ Aplicaciones
│
├─ Videos
│ └─ 3Blue1Brown: Integration
│
├─ Práctica Código
│ ├─ Métodos numéricos: Riemann, trapezoidal, Simpson
│ ├─ Monte Carlo integration
│ └─ Visualizar aproximaciones
│

└─ Ejercicios

SEMANA 3-4: Cálculo Multivariable
├─ Teoría
│ ├─ Derivadas parciales
│ ├─ Gradiente
│ ├─ Jacobiano
│ ├─ Hessiano
│ ├─ Optimización multivariable
│ └─ Lagrange multipliers
│
├─ Práctica Código
│ ├─ Gradient descent (repaso con cálculo)
│ ├─ Newton's method
│ ├─ Optimización con restricciones
│ └─ Visualizar gradientes
│
└─ Proyecto: Optimizer toolkit

MES 15 - MARZO 2026: OPTIMIZACIÓN AVANZADA

CONTENIDO:
├─ Linear Programming (Simplex)
├─ Convex optimization
├─ Gradient-based methods
│ ├─ SGD variants (Momentum, Adam, RMSprop)
│ └─ Adaptive learning rates
├─ Derivative-free optimization
├─ Genetic algorithms
└─ Simulated annealing

PROYECTO:
└─ Optimization library comparando métodos

MES 16 - ABRIL 2026: ECUACIONES DIFERENCIALES

CONTENIDO:
├─ ODEs básicas
├─ Métodos numéricos: Euler, Runge-Kutta
├─ Sistemas de ODEs
├─ Aplicaciones: física, biología, economía
└─ Análisis de estabilidad

PROYECTO FASE 4:
└─ Physics Engine 2D
  ├─ Simulaciones con ODEs

├─ Colisiones
├─ Springs, pendulums
└─ Optimización de trayectorias

FASE 5: ALGORITMOS AVANZADOS Y COMPLEJIDAD
Meses 17-20 (Mayo - Agosto 2026)

MES 17: Dynamic Programming avanzado
MES 18: Algoritmos de strings y patrones
MES 19: Algoritmos geométricos
MES 20: Complejidad computacional y NP-completeness

PROYECTO FASE 5:
└─ Algorithm Visualizer completo
  ├─ Todos los algoritmos importantes
  ├─ Análisis de complejidad
  └─ Educational tool

FASE 6: TÓPICOS AVANZADOS Y PROYECTO FINAL
Meses 21-24 (Septiembre - Diciembre 2026)

MES 21: Information Theory y Criptografía
MES 22: Teoría de juegos
MES 23: Computación cuántica (intro)
MES 24: PROYECTO FINAL INTEGRADOR

PROYECTO FINAL (4 semanas):
Elegir UNO de:

A) Compiler from scratch
  └─ Lexer, parser, optimizer, code gen
  └─ Usa: grafos, árboles, optimización

B) Deep Learning Framework
  └─ Mini TensorFlow/PyTorch
  └─ Usa: álgebra lineal, cálculo, optimización

C) Database Management System
  └─ Query optimizer, indexing, transactions
  └─ Usa: árboles, grafos, probabilidad

D) Distributed System Simulator
  └─ Consensus, fault tolerance
  └─ Usa: probabilidad, grafos, teoría de juegos

==========

📊 SISTEMA DE SEGUIMIENTO Y EVALUACIÓN

EVALUACIÓN MENSUAL:
├─ ¿Completé todos los temas teóricos?
├─ ¿Implementé los algoritmos en código?
├─ ¿Terminé el proyecto del mes?
├─ ¿Resolví los ejercicios propuestos?
└─ ¿Actualicé GitHub y portfolio?

EVALUACIÓN TRIMESTRAL (cada fase):
├─ Proyecto integrador completado
├─ Todos los conceptos consolidados
├─ Portfolio actualizado
└─ Preparación para siguiente fase

MÉTRICAS DE PROGRESO:
├─ Problemas de LeetCode resueltos
├─ Proyectos en GitHub
├─ Conceptos dominados (checklist)
└─ Tiempo invertido (tracking)

AJUSTES AL ROADMAP:
- Revisar cada 3 meses
- Ajustar según dificultad real
- Agregar/quitar temas según interés
- Mantener flexibilidad

💡 TIPS PARA SEGUIR EL ROADMAP

1. CONSISTENCIA sobre intensidad
   └─ Mejor 45 min/día que 5 horas el sábado

2. NO adelantarse si no dominaste el tema actual
   └─ Mejor base sólida que cobertura superficial

3. SIEMPRE implementar en código
   └─ Si no lo programaste, no lo aprendiste

4. PROYECTOS son más importantes que ejercicios
   └─ Portfolio > cantidad de problemas resueltos

5. DOCUMENTAR todo en GitHub
   └─ README claros, código comentado

6. COMPARTIR progreso
   └─ Blog, Twitter, LinkedIn

7. No compararte con otros
   └─ Tu único competidor eres tú de ayer

8. Cuando te trabes, BUSCA OTRA EXPLICACIÓN
   └─ 3Blue1Brown casi siempre ayuda

9. Aplicar a PROBLEMAS REALES
   └─ Busca datasets, competencias, proyectos open source

10. DISFRUTAR el proceso
   └─ Las matemáticas son hermosas cuando las entendés

==========

🎓 AL FINALIZAR EL ROADMAP (24 MESES)

==========

HABRÁS LOGRADO:

CONOCIMIENTOS:
✓ Matemática Discreta nivel experto
✓ Álgebra Lineal aplicada a ML/gráficos
✓ Probabilidad y Estadística para Data Science
✓ Cálculo y Optimización
✓ Algoritmos avanzados

HABILIDADES:
✓ Implementar algoritmos complejos desde cero
✓ Analizar y optimizar código
✓ Diseñar sistemas eficientes
✓ Aplicar matemáticas a problemas reales
✓ Machine Learning con fundamentos sólidos

PORTFOLIO:
✓ 10+ proyectos completos
✓ 500+ problemas de LeetCode
✓ GitHub con código de calidad profesional
✓ Blog/documentación técnica

PREPARACIÓN:
✓ Lista para ingeniería en software
✓ Fundamentos para cualquier especialización
✓ Nivel de desarrollador sobresaliente

PRÓXIMOS PASOS:
├─ Decidir: ¿Ingeniería o Licenciatura?
├─ O ir directo al mercado laboral
├─ Especializarse: IA, sistemas, gráficos, etc.
└─ Contribuir a open source

==========

📅 INICIO: ENERO 2025
🎯 META: DESARROLLADOR SOBRESALIENTE
💪 ACTITUD: PERSISTENCIA Y PASIÓN
==========

¡VAMOS ERNESTO, VOS PODÉS!
Ya aprobaste matemática discreta contra todo pronóstico.
Esto es un maratón, no un sprint.
Paso a paso, tema a tema, proyecto a proyecto.
En 2 años vas a ser imparable.

==========