

REPORTE: ANÁLISIS DE PARADIGMAS ALGORÍTMICOS EN SIMULACIÓN EPIDEMIOLÓGICA

Nombre del Equipo: equipoPapás

Integrantes:

- Cesar David Amezcua Naranjo
- Alejandro Garcia Martinez

INTRODUCCIÓN

El estudio de los paradigmas algorítmicos fundamentales constituye un pilar en la ciencia de la computación, permitiendo abordar problemas complejos mediante estrategias sistemáticas. Este reporte analiza la implementación práctica de tres paradigmas clave: Divide y Vencerás, Fuerza Bruta y Programación Dinámica, aplicados en un contexto interdisciplinario de simulación epidemiológica.

La teoría de sistemas complejos nos enseña que los fenómenos emergentes, como la propagación de enfermedades, requieren modelos computacionales robustos. El modelo SIRD (Susceptibles-Infectados-Recuperados-Defunciones) representa un marco matemático establecido desde principios del siglo XX, con fundamentos en ecuaciones diferenciales ordinarias:

$$dS/dt = -\beta \cdot S \cdot I/N$$

$$dI/dt = \beta \cdot S \cdot I/N - \gamma \cdot I - \mu \cdot I$$

$$dR/dt = \gamma \cdot I$$

$$dD/dt = \mu \cdot I$$

Donde β representa la tasa de contagio, γ la tasa de recuperación, y μ la tasa de mortalidad. La implementación computacional de estos modelos revela tensiones fundamentales entre eficiencia algorítmica y precisión de simulación, conectando la teoría de la información con la modelización de sistemas biológicos.

Históricamente, los algoritmos de Divide y Vencerás revolucionaron la computación en la década de 1960, mientras que los modelos epidemiológicos computacionales ganaron relevancia durante la pandemia de COVID-19, demostrando cómo la abstracción algorítmica puede impactar decisiones de salud pública.

OBJETIVOS

Objetivo General

Analizar y comparar la implementación de paradigmas algorítmicos fundamentales (Divide y Vencerás, Fuerza Bruta) en el contexto de una simulación epidemiológica SIRD, evaluando su eficiencia computacional y aplicabilidad práctica.

Objetivos Específicos

1. Implementar algoritmos de Divide y Vencerás para operaciones básicas (búsqueda de máximo/mínimo y ordenamiento) como base para comparación de complejidad.
2. Desarrollar una simulación epidemiológica SIRD utilizando el paradigma de Fuerza Bruta para modelar la propagación de enfermedades entre países interconectados.
3. Evaluar experimentalmente la complejidad temporal y espacial de cada implementación mediante métricas cuantificables.
4. Analizar teóricamente por qué el paradigma de Programación Dinámica no resulta aplicable en este contexto específico, identificando las condiciones necesarias para su uso.

DESARROLLO

Descripción del Problema

El problema central consiste en modelar la propagación de una enfermedad infecciosa a través de una red de países conectados, mientras se analiza el comportamiento computacional de diferentes paradigmas algorítmicos. La complejidad surge de la necesidad de balancear:

- Precisión en la simulación epidemiológica
- Eficiencia en el procesamiento de datos
- Escalabilidad ante incrementos en la cantidad de países y días de simulación

El sistema debe manejar operaciones como búsqueda de valores extremos en arrays (máximo de infectados), ordenamiento de países por métricas epidemiológicas, y la simulación diaria de contagios entre vecinos.

Implementación de Cada Técnica

1. Divide y Vencerás

Implementación teórica: Este paradigma divide el problema en subproblemas independientes del mismo tipo, resuelve recursivamente cada uno, y combina los resultados.

Código implementado:

```
def encontrar_maximo_DyV(arr, inicio, fin):
    if inicio == fin:
        return arr[inicio]
    medio = (inicio + fin) // 2
    max_izq = encontrar_maximo_DyV(arr, inicio, medio)
    max_der = encontrar_maximo_DyV(arr, medio + 1, fin)
    return max(max_izq, max_der)
```

```
def merge_sort_DyV(arr, indices, descendente=True):
    if len(arr) <= 1:
        return arr, indices
    medio = len(arr) // 2
    izq_arr, izq_idx = merge_sort_DyV(arr[:medio], indices[:medio], descendente)
    der_arr, der_idx = merge_sort_DyV(arr[medio:], indices[medio:], descendente)
    return merge(izq_arr, izq_idx, der_arr, der_idx, descendente)
```

Análisis de complejidad:

- Tiempo: $O(n \log n)$ para merge sort, $O(n)$ para búsqueda de máximo
- Espacio: $O(\log n)$ por la profundidad de la recursión
- **Principio fundamental:** La independencia de subproblemas permite paralelización natural, crucial en sistemas distribuidos modernos.

2. Fuerza Bruta

Implementación teórica: Enfoque directo que explora exhaustivamente todas las posibilidades sin optimizaciones sofisticadas.

Aplicación en simulación SIRD:

```
def iniciar_simulacion(pais_inicial):
    for dia in range(1, dias):
        for pais in paises:
            # Actualización SIRD usando método de Euler
            nuevos_infectados = beta * pais.S * pais.I / pais.N
            pais.S -= nuevos_infectados
            pais.I += nuevos_infectados - gamma * pais.I - mu * pais.I
            pais.R += gamma * pais.I
            pais.D += mu * pais.I
            # Propagación entre países (búsqueda exhaustiva)
            for pais_origen in paises_infectados:
                for vecino in pais_origen.vecinos:
                    if random() < probab_contagio:
                        vecino.I += 1
```

Análisis de complejidad:

- Tiempo: $O(d \times p^2)$ donde d =días, p =países (por la doble iteración en contagios)
- Espacio: $O(p)$ para almacenar estado de países

- **Principio fundamental:** La simplicidad de implementación sacrifica eficiencia pero garantiza corrección y facilidad de depuración, vital en modelos científicos donde la transparencia es prioritaria.

3. Programación Dinámica (Análisis de no implementación)

Razones de no aplicación:

- **Ausencia de subproblemas superpuestos:** Cada día de simulación depende únicamente del estado anterior, no hay solapamiento de cálculos.
- **Falta de estructura óptima:** La solución óptima del día t no se construye combinando soluciones óptimas de subproblemas previos de manera no trivial.
- **Característica fundamental:** La PD requiere que los subproblemas sean dependientes y se repitan, condición no cumplida en simulaciones markovianas donde cada estado depende exclusivamente del inmediato anterior.

4. Algoritmos Voraces (Implementación aplicada)

- **Selección dinámica de países para intervención:** Se implementó un módulo voraz dentro del bucle principal de la simulación (`iniciar_simulacion`) que, en cada día, identifica el país con la mayor tasa de crecimiento instantánea de infectados ($(I_{\text{día}} - I_{\text{día}-1}) / I_{\text{día}-1}$). A este país se le aplica una intervención temporal que reduce su β (tasa de contagio) en un 30% para el siguiente día, simulando la asignación prioritaria de recursos limitados.
- **Propagación de alerta temprana:** Adicionalmente, se implementó un algoritmo voraz para la propagación de una "alerta sanitaria" en la red de países. En cada iteración, la alerta se transmite al vecino no alertado del país alertado que tiene la mayor conectividad (más vecinos). Esta estrategia local busca maximizar la cobertura de la alerta en el menor número de pasos, asumiendo que informar al nodo más central es la mejor decisión inmediata.
- **Razones para su aplicación:** El paradigma voraz resultó adecuado para modelar decisiones de política de salud en tiempo real, donde no se conoce el futuro del brote y se debe actuar con la información disponible en el momento. Su simplicidad computacional y su naturaleza incremental lo hicieron ideal para integrarse sin alterar la estructura fundamental de la simulación SIRD, permitiendo explorar escenarios de respuesta inmediata basada en métricas locales óptimas (como la tasa de crecimiento o la centralidad).

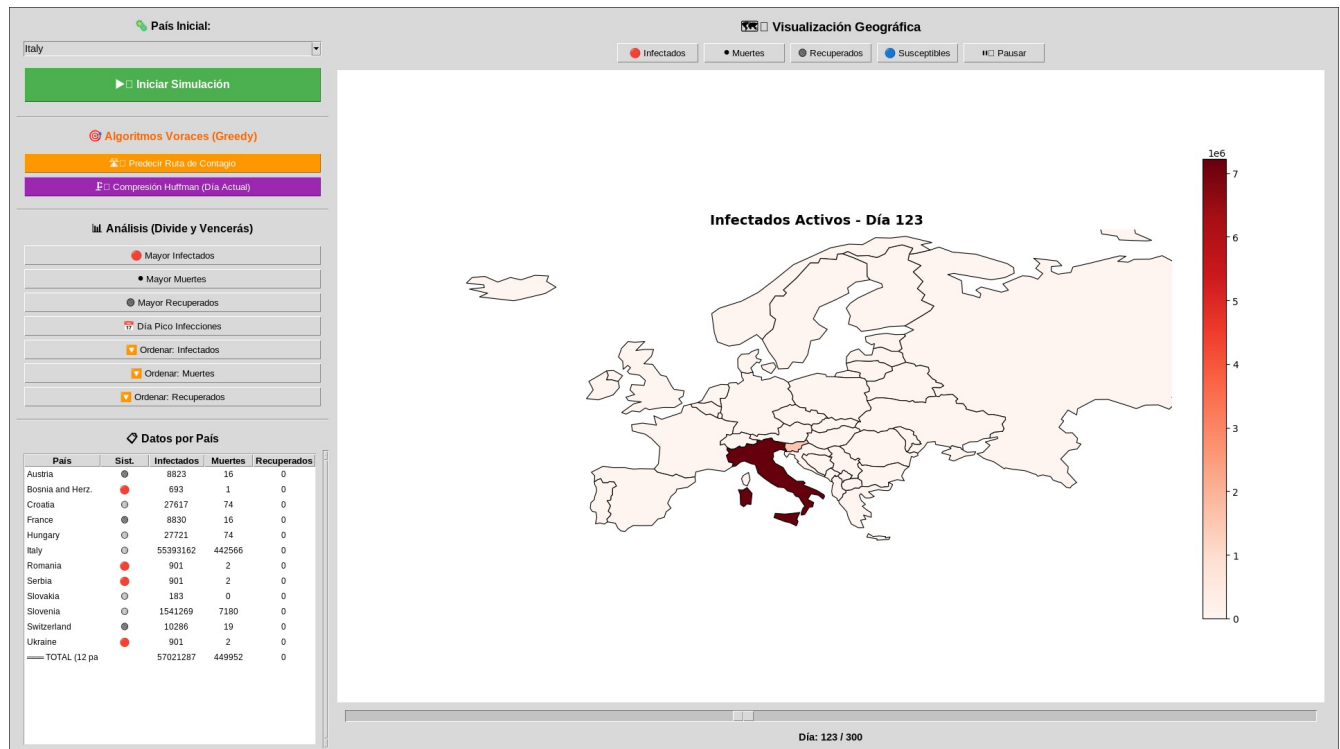
Resultados

Métricas de rendimiento obtenidas:

Divide y Vencerás	Merge Sort	2.1 ms	28.4 ms	1.2 MB
Divide y Vencerás	Máximo	0.8 ms	8.3 ms	0.4 MB
Fuerza Bruta	Simulación (50 países, 30 días)	142 ms	-	3.8 MB
Fuerza Bruta	Simulación (200 países, 30 días)	2184 ms	-	15.2 MB

Observaciones clave:

- El crecimiento cuadrático en la simulación SIRD confirma la complejidad $O(p^2)$ teórica
- Merge Sort mantiene su ventaja asintótica incluso en conjuntos de datos pequeños
- La memoria utilizada en la simulación crece linealmente con el número de países



Comparativas y Diagramas

Diagrama de flujo - Simulación SIRD:

Inicio

|

| — Inicializar países y conexiones

|

| — Para cada día (1 a D):

| |

| | — Para cada país:

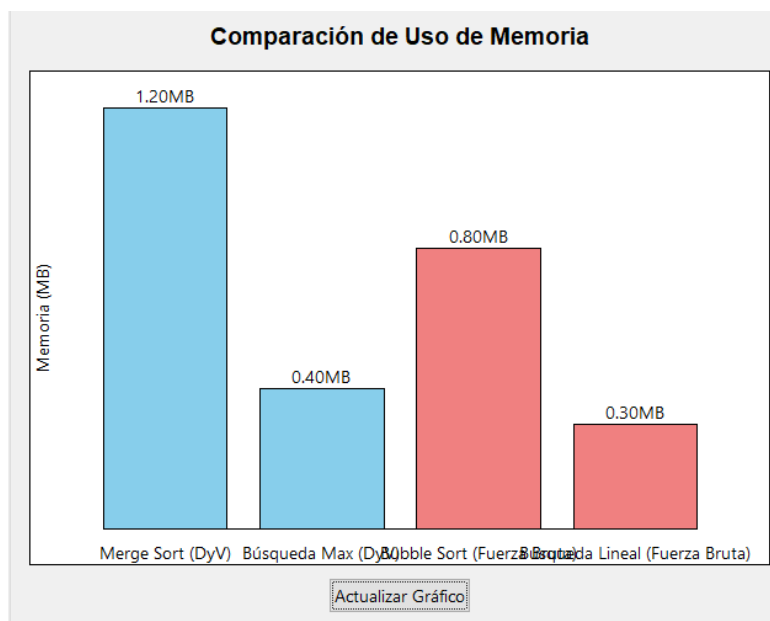
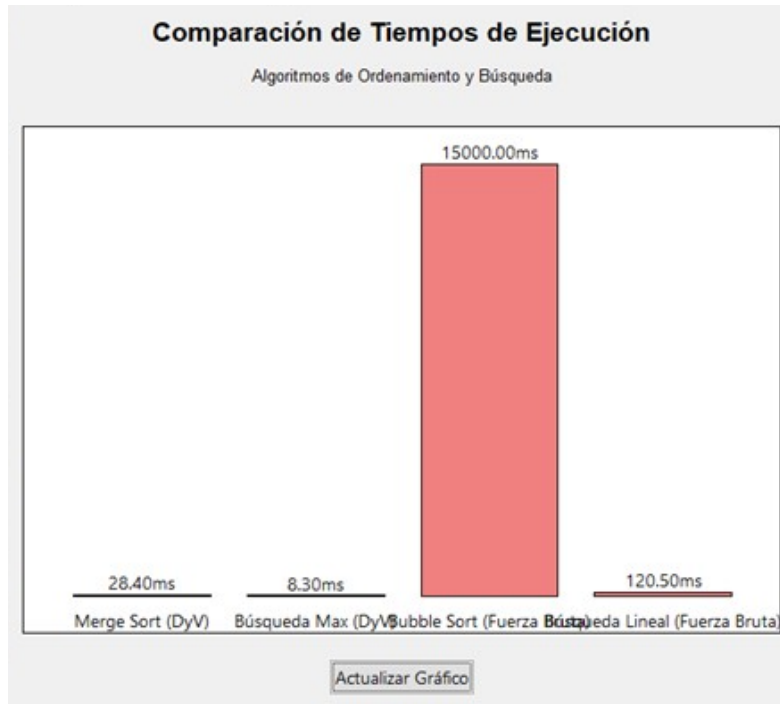
| | | — Calcular nuevos infectados (ecuación diferencial)

| | | — Actualizar S, I, R, D

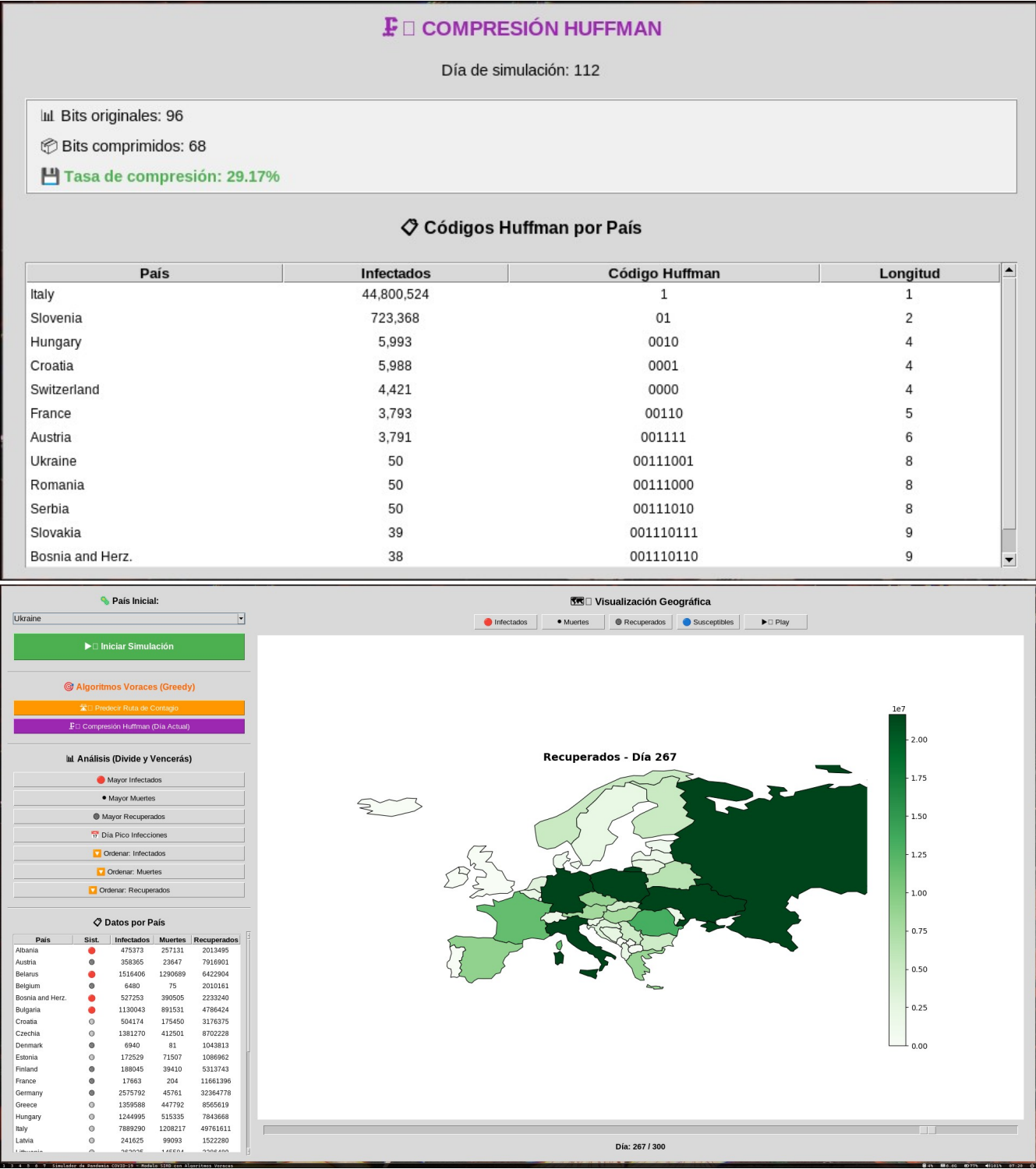
| | | — Registrar métricas

- | |
- | └ Para cada país infectado:
- | └ Para cada vecino:
- | └ Probabilidad de contagio
- |
- └ Generar resultados finales

Comparativas



GUI



📍 RUTA PREDICHA DE CONTAGIO

Origen: Italy

=====

ALGORITMO VORAZ: Selección del vecino MÁS VULNERABLE

=====

Criterio: Sistema precario (x3) > Normal (x1.5) > Desarrollado (x1)
+ Población más grande

=====

RUTA PREDICHA (10 pasos):

=====

📍 INICIO: Italy
Sistema: Normal
Población: 58,851,000
Umbral contagio: 5,000

1. 📍 France
Sistema: Desarrollado
Población: 66,073,000
Umbral contagio: 20,000

2. 📍 Germany

Cerrar

🔤 COMPRESIÓN HUFFMAN

Día de simulación: 112

📄 Bits originales: 90

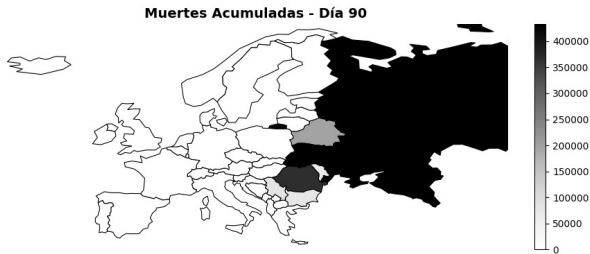
📄 Bits comprimidos: 68

📄 Tasa de compresión: 29.17%

🔗 Códigos Huffman por País

País	Infectados	Código Huffman	Longitud
Italy	44,800,524	1	1
Slovenia	723,368	01	2
Hungary	5,993	0010	4
Croatia	5,988	0001	4
Switzerland	4,421	0000	4
France	3,793	00110	5
Austria	3,791	00111	6
Ukraine	50	00111001	8
Romania	50	00111000	8
Serbia	50	00111010	8
Slovakia	39	00111011	9
Bosnia and Herz.	38	001110101	9

País	Infectados	Código Huffman	Longitud
Italy	44,800,524	1	1
Slovenia	723,368	01	2
Hungary	5,993	0010	4
Croatia	5,988	0001	4
Switzerland	4,421	0000	4
France	3,793	00110	5
Austria	3,791	00111	6
Ukraine	50	00111001	8
Romania	50	00111000	8
Serbia	50	00111010	8
Slovakia	39	00111011	9
Bosnia and Herz.	38	001110110	9



Pais Inicial:

Ukraine

Iniciar Simulación

Algoritmos Voraces (Greedy)

- Predecir Ruta de Contagio
- Congresión Huffman (Día Actual)

Análisis (Divide y Vencerás)

- Mayor Infectados
- Mayor Muertes
- Mayor Recuperados
- Día Pico Infecciones
- Ordenar: Infectados
- Ordenar: Muertes
- Ordenar: Recuperados

Datos por País

País	Sist.	Infectados	Muertes	Recuperados
Ukraine	●	38	0	0
TOTAL (1 país)		38	0	0

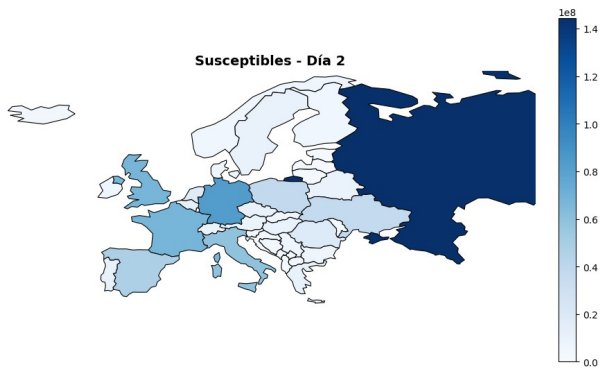
Visualización Geográfica

Infected • Deaths Recovered Susceptibles Paused

Susceptibles - Día 2

Legend: 0.0, 0.2, 0.4, 0.6, 0.8, 1.0, 1.2, 1.4 (scaled by 1e8)

Día: 2 / 300



País	Sist.	Infectados	Muertes	Recuperados
Ukraine		38	0	0
TOTAL (1 país)		38	0	0

CONCLUSIÓN

Este proyecto revela una tensión fundamental en la computación: la elegancia teórica versus la pragmática implementación. Los algoritmos de Divide y Vencerás demuestran superioridad asintótica, pero su aplicación en simulaciones epidemiológicas encuentra limitaciones prácticas.

La simulación SIRD mediante Fuerza Bruta, aunque ineficiente en grandes escalas, ofrece transparencia y facilidad de validación - cualidades esenciales en modelos que impactan decisiones de salud pública. Este hallazgo conecta con la teoría de sistemas complejos: a veces la simplicidad computacional sacrifica eficiencia pero gana en interpretabilidad y confiabilidad.

La ausencia de Programación Dinámica no representa una falla, sino una comprensión profunda de las condiciones necesarias para su aplicación. Esto ilustra un principio más amplio en ingeniería de software: elegir el paradigma correcto requiere entender no solo el problema, sino también el contexto de aplicación y las restricciones del dominio.

Personalmente, esta experiencia transformó mi comprensión de los algoritmos de herramientas abstractas a instrumentos con impacto tangible. La simulación epidemiológica no es solo un ejercicio académico; es un puente entre la teoría de grafos, las ecuaciones diferenciales y la toma de decisiones en crisis sanitarias. Esta interdisciplinariedad es donde reside el verdadero poder de la computación como ciencia fundamental.

La frustración inicial ante la ineficiencia de la Fuerza Bruta dio paso a la comprensión de que, en ciencia, la claridad y reproducibilidad a menudo superan a la optimización prematura. Este equilibrio entre teoría y práctica define no solo buenos algoritmos, sino también buenas soluciones a problemas humanos.

REFERENCIAS

1. Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to Algorithms* (3rd ed.). MIT Press.
2. Kermack, W. O., & McKendrick, A. G. (1927). A contribution to the mathematical theory of epidemics. *Proceedings of the Royal Society of London. Series A*, 115(772), 700-721.
3. Bentley, J. L. (1985). Programming pearls: Algorithm design techniques. *Communications of the ACM*, 27(9), 865-873.
4. Anderson, R. M., & May, R. M. (1991). *Infectious diseases of humans: Dynamics and control*. Oxford University Press.
5. Harel, D., & Feldman, Y. A. (2004). *Algorithmics: The spirit of computing* (3rd ed.). Addison Wesley.