

BioSim – Simulador Algorítmico de Propagación y Contención de Epidemias

Objetivo general

Diseñar un sistema que modele cómo se propaga una enfermedad (virus, bacteria, o mutación genética) a través de una red de individuos y territorios, y que permita **predecir, optimizar y contener** la propagación usando **múltiples paradigmas algorítmicos**, cada uno resolviendo un subproblema específico.

El reto no es solo “simular”, sino **diseñar algoritmos eficientes y analizarlos formalmente**, comparando sus complejidades y rendimientos experimentales.

Escenario base

1. Se tiene una **población de N individuos** conectados por una **red de contactos**
 - Cada territorio tiene un conjunto M de individuos, en donde se relacionan por proximidad o contacto, con cierta probabilidad de contagio
 2. El virus tiene **características de mutación** (propagación, letalidad, tiempo de recuperación).
 3. El sistema debe resolver **ocho subproblemas**, cada uno mapeado a un paradigma algorítmico distinto, todos parte del mismo modelo integral.
-

Mapa de subproblemas y paradigmas

Fase	Subproblema a resolver	Complejidad de algoritmo involucrado
1. Inicialización y análisis de datos	Leer poblaciones por nombre. Para cada población, listar sus individuos ordenados por: (1) <i>riesgo DESC</i> , (2) <i>tiempo_de_infección</i>	Cada conjunto de datos deberá ser ordenado usando: 4 algoritmos $O(n \log n)$ diferentes

	ASC (los None /no infectados al final), (3) <i>nombre ASC</i>	
2. Detección de brotes	Determinar los primeros focos y zonas conectadas.	Algoritmo menor a $O(n^3)$
3. Propagación temporal	Calcular la expansión paso a paso del contagio.	Simular contagio acumulativo y recuperación (almacenar estados por día), algoritmo menor o igual a $O(n \log n)$
4. Minimización del riesgo total	Seleccionar subconjunto mínimo de nodos a aislar (sumando los riesgos individuales de cada nodo sin considerar como se conectan entre sí)	Algoritmo de complejidad menor a $O(n^2)$
5. Identificación de rutas críticas de contagio	Encontrar el “camino de mayor probabilidad de infección” entre dos nodos.	Complejidad a alcanzar $O((n + m) \log n)$; donde n:número de nodos; m:número de aristas
6. Cálculo de rutas óptimas de contención	Hallar la red mínima de vacunación o aislamiento.	Algoritmo de complejidad $O((n + m) \log n)$
7. Clustering de cepas similares	Agrupar variantes según similitud genética o contagio.	Construcción $O(nL)$, Búsqueda o consulta por cepa: $O(L)$ n: número de variantes; L: longitud del nombre o cadena
8. Almacenamiento eficiente y consulta rápida	Acceso a historiales de contagios, información individual, cepas o variantes.	Todos con complejidad $O(1)$. Se puede ocupar el mismo algoritmo aquí para todo

Sugerencia de información en las entidades a ocupar:

- **individuos:** id, nombre, territorio_id, grado_inicial, riesgo_inicial
- **territorios:** territorio_id, nombre, M
- **conexiones_territorio:** u_territorio, v_territorio, peso_proximidad (≥ 30 filas)
- **contactos:** u_individuo, v_individuo, prob_contagio
- **cepas:** cepa_id, nombre, beta, letalidad, gamma_recuperación
- **semillas:** individuo_id, t0, cepa_id

⚠ **DATOS PRECARGADOS OBLIGATORIOS:**

50 variantes/cepas, 20 territorios/poblaciones, ≥ 30 conexiones entre territorios (aristas inter-territoriales) y 10 pacientes contagiados (semillas iniciales de contagio)

⚠ **Prohibido usar algoritmos mayores a $O(n^3)$ o factoriales en cualquier funcionalidad.**

Entregables

✦ 1. Informe Escrito

- ◆ Descripción de los algoritmos utilizados.
- ◆ Justificación de cada algoritmo y comparación con otras opciones.
- ◆ Cálculo de la **eficiencia (mejor, peor y caso promedio) de manera detallada y demostración de las cotas de complejidad**
- ◆ Explicación detallada del código con una prueba de escritorio **en el mejor, peor y caso promedio**

✦ 2. Código Fuente

- ◆ Implementación del sistema en un lenguaje de programación C.
- ◆ Comentarios en el código explicando la lógica utilizada.

✦ 3. Presentación del código guiado (máximo 15 minutos)

- ◆ Durante la presentación, se deberá mostrar el **funcionamiento del sistema completo** siguiendo la secuencia de **escenarios** que corresponden a los subproblemas del proyecto *BioSim*
- Después de cada demostración breve, se explicará la parte del código que implementa ese escenario, indicando el tipo de algoritmo utilizado y su complejidad.

◆ Escenarios a mostrar:

1. **Inicialización y análisis de datos:** lectura de poblaciones, ordenamiento de individuos por distintos criterios (riesgo, tiempo de infección, nombre) usando varios algoritmos $O(n \log n)$.
2. **Detección de brotes:** identificación de los primeros focos y zonas conectadas.
3. **Propagación temporal:** simulación paso a paso del contagio y recuperación acumulada.
4. **Minimización del riesgo total:** selección del subconjunto mínimo de nodos a aislar.

5. **Identificación de rutas críticas:** cálculo del camino de mayor probabilidad de infección.
 6. **Rutas óptimas de contención:** construcción de la red mínima de vacunación o aislamiento.
 7. **Clustering de cepas similares:** agrupamiento de variantes según similitud genética o patrón de contagio.
 8. **Almacenamiento y consulta eficiente:** acceso rápido a historiales, nodos clave y variantes mediante hashing o estructuras balanceadas.
-

Aspectos Obligatorios

- ✓ Cada funcionalidad debe usar un algoritmo eficiente y adecuado.
- ✓ No se debe repetir el mismo algoritmo para distintas funcionalidades, incluso si tienen la misma complejidad.
- ✓ El código debe ser limpio, modular y documentado.