**De una secuencia a una red viva: Havel–Hakimi y la tecnología blockchain**

**Introducción: el enigma de una secuencia**

Partimos de algo mínimo: una lista de grados. La pregunta es directa: ¿existe un grafo simple cuyos vértices tengan exactamente esos grados? En esta aplicación, esa lista se escribe en el campo de Havel–Hakimi —por ejemplo, “3,3,2,2”— y, al pulsar **Iniciar**, el enigma se vuelve construcción. Lo que era una fila de enteros se materializa en un overlay sobre el que ya se puede medir, recorrer y difundir información.

**La abstracción: de números a red**

La clave es la abstracción. Igual que Euler “olvidó” la geografía de Königsberg para quedarse con la estructura de conexiones, aquí nos centramos en quién se conecta con quién. El algoritmo de Havel–Hakimi (HH) ordena la secuencia, atiende al vértice más demandante, lo enlaza con los siguientes y reduce; vuelve a ordenar y repite. Si el proceso termina en ceros sin contradicciones, la secuencia era realizable y el grafo queda construido. La implementación ejecuta esa reducción paso a paso y la hace visible en la interfaz, de modo que la idea se vuelve tangible.

**Fundamentos clave**

Dos reglas enmarcan el juego. La primera es la paridad: la suma de los grados debe ser par o no habrá grafo posible. La segunda es el tope por vértice: ningún grado puede exceder n−1, porque nadie puede conectarse con más nodos de los que hay. La inicialización de tu app valida ambas de entrada. El éxito se reconoce cuando todos los grados remanentes llegan a cero; si en cambio surge un grado negativo o un vértice intenta conectarse con más vecinos de los disponibles, HH detiene la construcción con un diagnóstico claro.

**El teorema compañero: Erdős–Gallai en visión rápida**

Donde Euler resolvía “¿hay paseo euleriano?”, Erdős–Gallai responde “¿esta secuencia es realizable?”. La demo usa HH porque, además de decidir, **construye** el grafo. Erdős–Gallai, en cambio, ofrece un criterio elegante para reconocer secuencias gráficas de un vistazo. Son dos puertas a la misma casa: una certifica la posibilidad, la otra te entrega la red lista para experimentar.

**Del papel al código: cómo HH levanta el overlay**

Un overlay es una red lógica que se construye por encima de la red física. Internet pone los cables y routers; el overlay decide quién habla con quién. En redes P2P, ese overlay es el grafo de vecinos que cada nodo mantiene para intercambiar mensajes sin un servidor central; los pares se conectan entre sí, comparten datos y se coordinan de forma distribuida.

En mi demo genero el overlay con el algoritmo Havel–Hakimi a partir de una secuencia de grados: primero verifico que la secuencia sea gráfica y luego construyo un grafo simple que la realize.

Sobre esa malla ejecuto un protocolo gossip: elijo un origen y configuro el fanout, que es cuántos vecinos intenta contactar cada nodo informado en cada ronda para reenviar el mensaje o bloque. La visualización muestra los nodos con su grado inicial y residual, el origen destacado y las aristas que se activan en cada paso, mientras el panel reporta la ronda, el porcentaje de nodos alcanzados y las aristas usadas; también se pueden cambiar los layouts para observar con claridad la topología resultante.

Este ejercicio tiene aplicaciones directas. En blockchain permite explorar cómo distintas mallas P2P afectan la latencia de propagación de transacciones y bloques, el costo de ancho de banda y la robustez frente a caídas o comportamientos maliciosos.

En redes distribuidas ayuda a diseñar topologías que sostengan buen desempeño con recursos limitados, ajustando fanout y conectividad según la carga. En ciberseguridad facilita ensayar escenarios de ataque y contingencia sobre la estructura misma de la red. En analítica y docencia funciona como laboratorio visual para comprender, medir y explicar difusión, conectividad y resiliencia sin perderse en abstracciones.

La idea central es sencilla: no alcanza con tener una red; importa qué red construimos.

**De una secuencia a una red viva: Havel–Hakimi, gossip y blockchain**

**Del enigma a la construcción**

Todo comienza con una lista de grados. La pregunta es directa: **¿existe un grafo simple con exactamente esos grados?** En el módulo de **Havel–Hakimi (HH)** se ingresa, por ejemplo, 3,3,2,2, se pulsa **Iniciar** y el enigma se vuelve construcción visible. La secuencia se ordena, HH atiende al vértice más demandante, lo enlaza con los siguientes, descuenta una unidad en sus grados residuales y repite. Si el proceso concluye en ceros, la secuencia era realizable: la lista de números se transforma en **malla**.

En la visualización, cada nodo aparece con su etiqueta (v1, v2, …) y el par **residual / inicial** (como 0 / 6). Las aristas van emergiendo a medida que HH avanza y el grafo se estabiliza; se puede seguir paso a paso o en modo automático.

**Uso esencial de la demo**

El panel de **Overlay (Havel–Hakimi)** recibe la secuencia, propone ejemplos rápidos y también genera familias completas: redes **k-regulares** de tamaño configurable o secuencias aleatorias ya garantizadas como gráficas. Al iniciar, la aplicación valida dos reglas que evitan falsos arranques: **paridad de la suma** y **ningún grado mayor que n−1n-1n−1**. Si algo no cuadra, el sistema lo informa e indica el paso exacto donde se volvió imposible.

A partir del grafo construido, el módulo de **Gossip** simula la difusión P2P. Se selecciona el **fanout** (cuántos vecinos contacta cada nodo por ronda) y el **origen**; la animación resalta el origen con un halo, marca en amarillo las aristas activas de cada ronda y actualiza el estado de los nodos conforme se informan. En el encabezado se muestran **ronda**, **cobertura** (nodos alcanzados) y **aristas usadas** en tiempo real. Una **matriz de adyacencia** permite inspeccionar la conectividad exacta.

**Lo que ocurre internamente**

HH no es solo un verificador: **construye** el grafo. En cada ciclo toma el mayor grado residual, enlaza hacia los siguientes y reduce. En la simulación **se nota** cómo descienden los grados residuales, cómo quedan fijadas las aristas y cómo se estabiliza la topología. Si en algún punto un vértice requiere más vecinos de los disponibles o algún residual cae por debajo de cero, la ejecución se detiene con un diagnóstico claro.

Una vez creada la malla, el **gossip** opera como un rumor digital: la frontera de informados intenta comunicar a nuevos vecinos (hasta el fanout por nodo) y la onda se propaga por rondas. No hay servidor central; la **geometría del overlay** determina velocidad de propagación, consumo de ancho de banda y robustez ante fallos.

**Layouts que ayudan a pensar**

La misma red puede visualizarse como **anillo**, **radial por grado**, **cuadrícula** o **sunflower**. No es un detalle meramente estético: cambia la **legibilidad** de la estructura. El radial agrupa por grado (útil para detectar hubs), el anillo facilita seguir saltos de ronda y el sunflower distribuye bien redes medianas y grandes. Un botón de **reordenar** rota/permuta sin alterar la estructura, útil para observar la difusión desde otros ángulos.

**Métricas que importan**

La **cobertura por rondas** indica cuán rápido un bloque se vuelve conocimiento común. **Aristas usadas** aproximan el costo de ancho de banda por ronda. La relación **fanout–rondas** muestra el umbral de **rendimientos decrecientes**: pasar de 2 a 3 puede recortar una ronda entera; de 5 a 6 el efecto suele ser marginal, aunque el tráfico duplicado aumenta. Con la **matriz de adyacencia** se confirman cuellos de botella: hubs que concentran tránsito o comunidades apenas conectadas.

**Blockchain y P2P: por qué esta simulación es relevante**

En redes blockchain, la forma del overlay condiciona la **latencia de propagación** de transacciones y bloques, el **consumo de ancho de banda** y la **resiliencia** frente a ataques (eclipse, particiones) o fallos. Con HH se diseñan familias de mallas (regulares, casi regulares, con mínimos/máximos de grado), y con gossip se comprueba su efecto: ¿cuántas rondas hacen falta para 90% de cobertura? ¿cuál es el costo por bloque? ¿qué sucede si se desconecta un 10% de nodos al azar o cae un hub? Los controles de fanout, origen y layout permiten estudiar respuestas **en contexto**.

**Ejemplo con la imagen compartida**

En la captura se observa una red con **grado objetivo 6** (los nodos muestran 0 / 6 tras finalizar HH). Un origen —marcado con aro ámbar— inicia la difusión; las aristas amarillas señalan los contactos de la ronda actual. La densidad sugiere **diámetro** pequeño: con fanouts moderados, la cobertura crece deprisa en las primeras rondas y se estabiliza pronto. Al repetir la simulación con origen periférico o con fanout = 2, se prolonga la “cola” para alcanzar a los últimos nodos: un comportamiento esperado que conviene medir antes de ajustar una malla P2P real.

**Cierre**

No alcanza con “tener una red”: **importa qué red se construye**. Con **Havel–Hakimi** una especificación local de grados se convierte en overlay concreto; con **gossip** se observa, ronda a ronda, cómo circula la información sobre esa geometría. Ese ciclo —diseño, simulación, ajuste— es la base de redes P2P mejor pensadas: rápidas, frugales y resistentes.