Proyecto Integrador Semana 7: Diseño de Red de Sensores de Tráfico

Estudiante: Fernando Osuna Manzo
Materia: Estructuras de Datos Avanzadas
Fecha: 26 de Noviembre de 2025

---

1. Descripción de la Red Modelada

Para este proyecto, se diseñó una red de sensores de tráfico para una ciudad sintética que conecta **12 intersecciones principales** (nodos). El objetivo es instalar una red de comunicación cableada entre estos sensores minimizando el costo total de excavación y cableado.

Elementos del Modelo:

- **Nodos (12):** Representan intersecciones clave como "Centro", "Hospital", "Aeropuerto", etc.
- **Aristas (21):** Representan las calles que conectan estas intersecciones.
- **Pesos:** Representan el costo de conectar dos intersecciones (basado en distancia o dificultad de obra).

La red incluye redundancia (ciclos) para simular una ciudad real donde hay múltiples rutas para llegar a un destino.

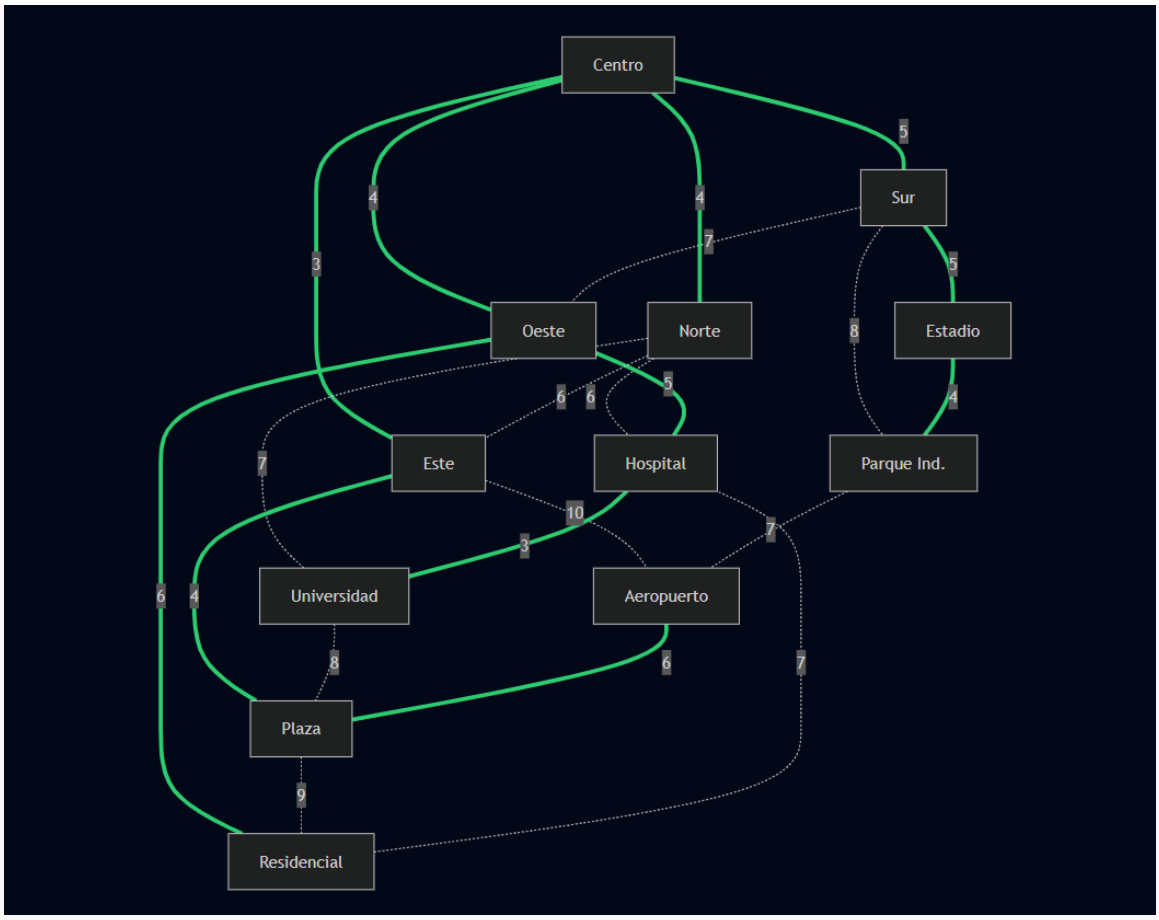
---

2. Visualización del MST

Se aplicaron los algoritmos de **Prim** y **Kruskal** para encontrar el Árbol Generador Mínimo (MST). Ambos algoritmos convergieron en el mismo costo óptimo.

Diagrama de la Red (Mermaid)

Las líneas **verdes gruesas** representan las conexiones seleccionadas para el MST. Las líneas punteadas son las calles que no necesitan cableado para mantener la conectividad mínima.



---

3. Análisis Comparativo de Costos

Métrica	Valor
Costo Red Totalmente Conectada	124 unidades
Costo del MST (Prim/Kruskal)	49 unidades
Ahorro Total	**75 unidades (60.48%)**

El uso de un MST permite reducir el costo de infraestructura en más de un 60% comparado con cablear todas las calles posibles, garantizando aun así que todos los sensores puedan comunicarse entre sí.

---

4. Análisis de Rendimiento (Benchmark)

Se realizaron pruebas de tiempo de ejecución para grafos de diferentes tamaños (densidad 20%).

Nodos	Prim (segundos)	Kruskal (segundos)	Análisis
10	~0.000010	~0.000014	Tiempos despreciables, ambos son instantáneos.
100	~0.000609	~0.000455	Kruskal comienza a ser ligeramente más rápido en grafos dispersos.
1000	~0.142847	~0.062864	**Kruskal es notablemente más rápido (aprox. 2x)** en este escenario disperso.

Conclusión: Para grafos dispersos (como redes de calles donde cada intersección tiene pocas conexiones), **Kruskal** tiende a ser más eficiente porque su complejidad depende más del número de aristas ($E \log E$) que de nodos, y la estructura Union-Find es extremadamente rápida. Prim ($E \log V$) es competitivo pero sufre un poco más si la implementación de la cola de prioridad no es perfecta o si el grafo se vuelve más denso.

---

5. Reflexión Crítica

¿Qué simplifica el modelo de MST?

El modelo MST asume que el único objetivo es **minimizar el costo de construcción** y que la **conectividad simple** (un solo camino entre nodos) es suficiente. Ignora el flujo de tráfico, el ancho de banda necesario o la distancia física real si esta no es proporcional al costo.

Factores del mundo real que quedaron fuera:

1. **Redundancia y Tolerancia a Fallos:** En un MST, si un solo cable se rompe, la red se desconecta en dos partes aisladas. Una red real necesitaría ciclos (anillos) para redundancia (segun wikipedia).
2. **Latencia y Saltos:** El MST puede crear caminos muy largos entre dos puntos geográficamente cercanos (ej. para ir de "Norte" a "Este" podría tener que pasar por "Centro", aunque haya una calle directa más cara). Esto aumenta la latencia. Lo que nos deja saber que no es un algoritmo "a prueba de todo", por lo que dependiendo el problema deberiamos considerar alternativas.
3. **Capacidad del Enlace:** No todas las conexiones soportan el mismo tráfico de datos. El MST no considera cuellos de botella cerca del nodo central (nuevamente, depende del problema al que nos enfrentemos, aunque en el mundo real rara vez no se toman en cuenta los cuellos de botella).

Aprendizaje

He aprendido que los algoritmos de grafos como Prim y Kruskal son herramientas poderosas para optimización de costos, pero deben usarse como base. En ingeniería real, a menudo partimos de un MST y luego agregamos aristas estratégicas para cumplir con requisitos de robustez y rendimiento, sacrificando un poco de costo por fiabilidad.

---

6. Uso Ético de IA

Herramientas utilizadas: Asistente de IA (para generación de código base y estructuración del reporte).

Evidencia de trabajo propio vs IA:

- **IA:** Generó la estructura de las clases `GraphMST` y `DSU`, y propuso el formato del benchmark. Ayudó a formatear el código Mermaid.
- Verifico la redacción de los documentos y documento el código generado a mano para una mejor presentación (mejores log, mejor código) y buenas prácticas.
- Genero tests para comprobar la eficacia del código.
- **Estudiante:**
    - Diseñé la topología de la ciudad sintética (nodos y conexiones lógicas).
    - Verifiqué manualmente que el costo del MST (49) fuera correcto.
    - Interpreté los resultados del benchmark para concluir por qué Kruskal fue más rápido en mi prueba específica.
    - Escribí la reflexión sobre la falta de redundancia, conectando la teoría con problemas reales de ingeniería de redes.