Reporte Semanal

Ramirez Moreno
Mauricio Damian

1. INTRODUCCIÓN Y OBJETIVOS

Metas alcanzadas esta semana

Durante la Semana 6 trabajé en dos algoritmos clave para encontrar rutas óptimas en grafos con pesos:

- Implementé Dijkstra utilizando una cola de prioridad basada en SortedSet.
- Programé Floyd-Warshall para calcular distancias entre todos los pares.
- Añadí funciones para reconstruir caminos.
- Incorporé detección de ciclos negativos en Floyd-Warshall.
- Generé un cálculo de centralidad (distancia promedio por nodo).
- Validé todo con 10 pruebas unitarias (10/10 aprobadas).
- Integré estos algoritmos con el trabajo de semanas previas.

Descripción de la red urbana usada

- Analicé una red pequeña de 5 ciudades del área metropolitana de CDMX:
- Centro (A) – nodo principal.
- Ciudades B, C, D y E – conectadas entre sí.
- La red cuenta con 7 aristas bidireccionales, usando distancias reales aproximadas en kilómetros. Es un tamaño ideal para demostrar ambos algoritmos.

2. ARQUITECTURA DE LA IMPLEMENTACIÓN

Implementación de Dijkstra

Para Dijkstra utilicé una cola de prioridad con SortedSet<(double distancia, string nodo)>. El proceso fue:

- Inicializar todas las distancias a infinito, excepto el origen.
- Extraer siempre el nodo con la distancia más baja.
- Relajar las aristas de cada vecino.
- Actualizar distancias y volver a insertar en la cola si se encuentra una mejor ruta.

El comparador personalizado dentro del SortedSet evita conflictos cuando dos nodos tienen la misma distancia.

Complejidad:
≈ O((V+E) log V) → en esta red, ~28 operaciones.

Implementación de Floyd-Warshall

El enfoque para Floyd-Warshall fue:

- Construir una matriz de distancias con infinito, excepto la diagonal en cero.
- Cargar las distancias directas entre ciudades.
- Probar cada vértice como nodo intermedio.
- Revisar si aparece un ciclo negativo observando la diagonal.

- Complejidad:
  ≈ $O(V^3)$ → con 5 vértices: 125 operaciones.

3. RESULTADOS DE LAS PRUEBAS UNITARIAS (10/10)

Resumen de mis tests

Nombre del Test	Qué valida	Resultado
Dijkstra_Simple	Distancias básicas	PASS
Dijkstra_PesosCero	Aristas de peso 0	PASS
Dijkstra_Desconectado	Nodos inalcanzables	PASS
ReconstruirCamino	Camino óptimo	PASS
FloydWarshall_Simple	Distancias entre todos los pares	PASS
FloydWarshall_Negativos	Manejo de pesos negativos	PASS
CicloNegativo	Detecta ciclos negativos	PASS
Dijkstra_NodoInexistente	Error por nodo inválido	PASS
VerticeMasCentral	Vértice con menor distancia promedio	PASS
DistanciaMaxima	Diámetro ponderado	PASS

Me llamó la atención que mi código manejó correctamente todos los casos límite, incluso aquellos inesperados.

4. ANÁLISIS DE LA RED URBANA (5 CIUDADES)

Estructura de conexiones

Ciudad	Conexiones
Centro (A)	B (50.5), C (80.0), D (95.0)
Ciudad B	A (50.5), D (30.0)
Ciudad C	A (80.0), D (45.5), E (70.0)
Ciudad D	A (95.0), B (30.0), C (45.5), E (25.0)

Ciudad	Conexiones
Ciudad E	C (70.0), D (25.0)

Distancias mínimas desde Centro (Dijkstra)

A → A = 0.0 km

A → B = 50.5 km

A → C = 80.0 km

A → D = 80.5 km (mejor que el camino directo de 95 km)

A → E = 105.5 km

Hallazgo clave:
La ruta A → B → D ofrece un ahorro de 14.5 km (15.3%) frente a A → D directo.

Matriz generada por Floyd-Warshall

Desde / Hacia	Centro	B	C	D	E
Centro	0.0	50.5	80.0	80.5	105.5
B	50.5	0.0	130.0	30.0	55.0
C	80.0	130.0	0.0	45.5	70.0
D	80.5	30.0	45.5	0.0	25.0
E	105.5	55.0	70.0	25.0	0.0

Observaciones:

- B → C (130 km) es demasiado alto comparado con la ruta B → D → C (75.5 km).
- D → E (25 km) es la conexión más corta de toda la red.

5. ANÁLISIS DE CENTRALIDAD

Calcule la distancia promedio para cada ciudad:

Ciudad	Suma de distancias	Promedio	Observación
Centro	316.5	79.1	—

Ciudad	Suma de distancias	Promedio	Observación
B	265.5	66.4	Mejor que Centro
C	325.5	81.4	—
D	181.0	45.3	LA MÁS CENTRAL
E	255.5	63.9	—

Conclusión:
El vértice más central es Ciudad D, con la menor distancia promedio hacia el resto.

6. COMPARACIÓN: DIJKSTRA VS FLOYD-WARSHALL

Aspecto	Dijkstra	Floyd-Warshall
Caso típico	Un origen → todos	Todos ↔ todos
Complejidad	O((V+E)logV)	$O(V^3)$
Red de 5 ciudades	~28 operaciones	125 operaciones
Escalabilidad 1000 ciudades	Muy rápido	Extremadamente costoso
Pesos negativos	No	Sí (pero sin ciclos)
Detecta ciclos negativos	No	Sí
Reconstrucción de caminos	Fácil	Fácil

Recomendación práctica:

- Para navegación en tiempo real → Dijkstra
- Para análisis global o precálculo → Floyd-Warshall

7. PROBLEMAS DETECTADOS EN LA RED

1. Arista B ↔ C es ineficiente

Directa: 130 km

Alternativa B → D → C: 75.5 km
→ Es una conexión que no aporta valor.

2. Ruta Centro ↔ D directa poco útil

Directa: 95 km

Alternativa A → B → D: 80.5 km
→ Sería mejor mejorar esa vía o eliminar la directa.

Si yo fuera ingeniero urbano propondría:

1. Mejorar o eliminar la conexión B–C.
2. Analizar por qué Centro–D es tan costosa.
3. Verificar si D–E realmente debe ser tan corta.

8. CASOS DE USO ANALIZADOS

Caso 1: Ruta turística

Ruta recomendada:

Centro → B → D → E → C → Centro
Total: 255.5 km

Caso 2: Centro de distribución

Ciudad	Distancia promedio
Centro	79.1 km
D	45.3 km

→ D reduce 34 km promedio por entrega.

Caso 3: Resiliencia

- Si se cierra A → D:
- Alternativa: A → B → D
  → La red sigue completamente operativa.

9. CONCLUSIONES GENERALES

Lo que salió bien

- Dijkstra pasó todos los casos, incluyendo los extremos.
- El comparador del SortedSet evitó conflictos.
- Floyd-Warshall detectó ciclos negativos correctamente.
- Todo el sistema se integró con semanas anteriores.
- 10 pruebas unitarias confirmaron validez.

Lo que descubrí de la red

- Ciudad D es el mejor punto central.
- La conexión B–C es muy ineficiente.
- Dijkstra detectó rutas 15% más cortas que las “obvias”.