Universidad Michoacana de San Nicolás de Hidalgo
Facultad de Ingeniería Eléctrica
Análisis de Algoritmos

Especificaciones del trabajo: Implementación del algoritmo de Floyd–Warshall

Indicaciones generales

- **Trabajo:** Programar, documentar y verificar el algoritmo de **Floyd–Warshall** para caminos más cortos (all-pairs shortest paths).
- **Lenguaje:** Lenguaje C
- **Entrega:** Código fuente + reporte técnico + evidencias de ejecución.

1) Objetivo

Desarrollar un programa que:

1. Lea o construya un **grafo dirigido o no dirigido** con pesos (matriz de adyacencia).
2. Calcule la **matriz de distancias mínimas y matriz de recorridos** entre todos los pares de nodos con Floyd–Warshall.
3. Genere **corridas de ejemplo** (inputs y outputs) y un **reporte técnico** con verificación de complejidad y conclusiones.

2) Alcance y requisitos funcionales

2.1 Entrada

El programa debe aceptar al menos una de estas opciones (puedes implementar ambas):

- **A) Archivo de texto** con:
  - n (número de nodos)
  - n x n pesos en matriz (usa un valor especial para “no hay arista”, por ejemplo INF).
- **B) Entrada por consola** (mismo formato).

Convenciones obligatorias:

- Los nodos se identifican como 1..n o A...Z (elige uno y sé consistente).
- Debe permitirse:
  - Aristas con peso positivo
  - Peso cero

2.2 Salida

Tu programa debe imprimir y/o guardar:

1. **Matriz final de distancias** D (con INF donde no exista camino).
2. **Matriz final de recorridos**:
3. Para al menos **3 pares (origen, destino)**:
    - Distancia mínima
    - Ruta reconstruida (ej. 1 → 3 → 4)

3) Requisitos técnicos de implementación

3.1 Estructuras mínimas

- D: matriz n x n (distancias)
- R: matriz n x n (rutas)
- INF: constante grande (ej. 10^15) o float('inf').

3.2 Inicialización esperada

- D[i][i] = 0
- Si hay arista i→j con peso w, entonces D[i][j] = w
- Si no hay arista, D[i][j] = INF

4) Casos de prueba (corridas)

Entrega evidencias (capturas o logs) de al menos:

Grafos:

- Debes mostrar:
    - Matriz inicial
    - Matriz final
    - 3 rutas reconstruidas

5) Reporte técnico (estructura)

5.1 Resumen técnico

Incluye:

- Qué problema resuelve Floyd–Warshall
- Supuestos (matriz de adyacencia, INF, etc.)
- Qué entrega tu programa

5.2 Desarrollo / Metodología

- Representación del grafo
- Inicialización
- Cómo reconstruyes rutas
- Consideraciones de INF y desbordamientos (si aplica)

5.3 Verificación de complejidad

Debes justificar:

- **Tiempo:** O(n3)O(n^3)O(n3) por los 3 ciclos anidados.

5.4 Corridas de ejemplo (evidencias)

Incluye tablas/capturas de 3 casos.

5.5 Conclusiones

Mínimo 5–10 líneas sobre:

- Ventajas
- Limitaciones: O(n3)O(n^3)O(n3) no escala a grafos enormes
- Qué aprendiste (rutas, matrices, verificación)

6) Entregables

1. **Código fuente** (carpeta /src)
2. **Ejecutable o instrucciones de ejecución**:
    - README.md con cómo correr y ejemplos de entrada
3. **Reporte técnico** en PDF
4. **Archivos de prueba** (carpeta /tests) con tus matrices de entrada

7) Criterios de evaluación (rúbrica breve)

- **Correctitud del algoritmo (40%)**: D correcta, INF correcto, casos desconectados.
- **Reconstrucción de rutas (20%)**: rutas válidas, coherentes con D.
- **Calidad del código (15%)**: modularidad, nombres, comentarios, manejo de errores.
- **Reporte técnico (20%)**: resumen, metodología, complejidad, corridas.
- **Presentación y reproducibilidad (5%)**: README, inputs claros, salidas legibles.