Análisis Comparativo: Dijkstra vs Floyd-Warshall

Este documento presenta 5 escenarios del mundo real para comparar la idoneidad de los algoritmos de Dijkstra y Floyd-Warshall.

Escenario	Algoritmo Elegido	Justificación (Por qué)	Propiedades Clave	Desventaja del Otro
1. Navegación GPS (Google Maps)	**Dijkstra** (o A*)	Solo necesitamos la ruta óptima desde un punto de origen (mi ubicación) a un destino.	**Complejidad:** $O(E + V\log V)$. Eficiente para grafos dispersos (mapas de carreteras) y consulta de fuente única.	**Floyd-Warshall:** $O(V^3)$ es computacionalmente inviable para millones de intersecciones y calcula rutas irrelevantes.
2. Protocolo de Enrutamiento OSPF	**Dijkstra**	Cada router calcula independientemente la tabla de rutas desde *sí mismo* hacia el resto de la red.	**Fuente Única:** Cada nodo actúa como fuente única. Se ejecuta localmente en cada dispositivo.	**Floyd-Warshall:** Requiere conocimiento global y cálculo de pares que el router local no necesita (ej. ruta entre dos routers remotos).
3. Arbitraje de Divisas	**Floyd-Warshall**	Necesitamos detectar ciclos negativos (oportunidades de ganancia infinita) en un grafo de tipos de cambio.	**Ciclos Negativos:** FW maneja pesos negativos y detecta ciclos revisando la diagonal ($dist[i][i] < 0$).	**Dijkstra:** No soporta pesos negativos; entraría en bucles infinitos o daría resultados incorrectos.

4. Análisis de Latencia en Malla Densa	**Floyd-Warshall**	En una red de servidores "full-mesh" (todos conectados con todos), queremos la matriz completa de latencias.	**Grafos Densos:** En grafos donde $E \approx V^2$, la complejidad de FW es competitiva y su implementación matricial es simple y cache-friendly.	**Dijkstra:** Ejecutarlo V veces añade overhead de gestión de colas de prioridad que no compensa en grafos muy densos y pequeños.
5. Planificación Logística (TSP pequeño)	**Floyd-Warshall**	Para optimizar rutas de reparto entre 50 almacenes, necesitamos la distancia entre *cada par* para alimentar un algoritmo TSP.	**Pre-cómputo:** Genera la matriz de adyacencia completa necesaria para heurísticas de optimización combinatoria.	**Dijkstra:** Implementar V ejecuciones requiere más código y gestión de estructuras de datos para obtener el mismo resultado matricial.

Reflexión Post-IA

¿Los escenarios coinciden con la intuición?

Sí, la distinción principal radica en **Fuente Única vs. Todos los Pares** y **Grafos Dispersos vs. Densos/Pequeños**. Dijkstra domina en escalas grandes y rutas específicas, mientras que Floyd-Warshall brilla en análisis estructural completo de grafos pequeños o cuando existen pesos negativos.

Casos donde ambos son válidos

En grafos **dispersos y pequeños** (ej. < 500 nodos, pocas aristas), ejecutar Dijkstra V veces ($V \cdot E\log\ V$) puede ser teóricamente más rápido que FW (V^3), pero FW suele preferirse por la simplicidad de implementación (3 bucles for) y menor constante oculta si no se requiere optimización extrema.

Criterio Adicional Importante

Facilidad de Implementación y Depuración: Floyd-Warshall es extremadamente compacto (4-5 líneas de lógica core), lo que reduce la probabilidad de bugs en comparación con la gestión de colas de prioridad y estados de visita de Dijkstra. Esto lo hace ideal para concursos de programación o scripts rápidos donde V es pequeño.