INTRODUCCIÓN

En esta semana se implementaron los algoritmos de rutas más utilizadas en sistemas GPS: Dijkstra y Floyd–Warshall. Estos permiten calcular rutas óptimas en un grafo con pesos, simular tráfico y obtener la mejor ruta entre puntos.

Durante la Semana 6 del proyecto, se implementaron ambos algoritmos utilizando el lenguaje de programación Python, integrándose dentro de una estructura llamada `WeightedGraph`. Esta implementación permite calcular rutas óptimas en un grafo, reconstruir los caminos obtenidos y analizar el comportamiento del sistema bajo diferentes condiciones, como la simulación de tráfico mediante modificaciones en los pesos de las aristas.

OBJETIVO

Implementar Dijkstra y Floyd–Warshall en Python dentro del proyecto, aplicarlos a un caso de rutas y analizar resultados.

DESARROLLO

El primer paso del desarrollo consistió en programar el algoritmo de Dijkstra con el objetivo de obtener la ruta más corta desde un nodo origen hacia el resto de los nodos del grafo. Este algoritmo utiliza una cola de prioridad para ir seleccionando el camino más corto disponible y actualizando las distancias mínimas de manera eficiente. Además, se implementó la reconstrucción del camino, permitiendo obtener no solo el costo total de la ruta, sino también los nodos que la conforman.

Posteriormente, se implementó el algoritmo de Floyd–Warshall, el cual permite calcular las distancias mínimas entre todos los pares de nodos del grafo mediante una matriz de distancias. Este algoritmo analiza todas las posibles combinaciones de rutas y actualiza los valores cuando encuentra un camino más corto. Gracias a este proceso, es posible obtener una matriz completa que muestra el costo mínimo entre cualquier punto del sistema.

pytest test_weighted_graph.py

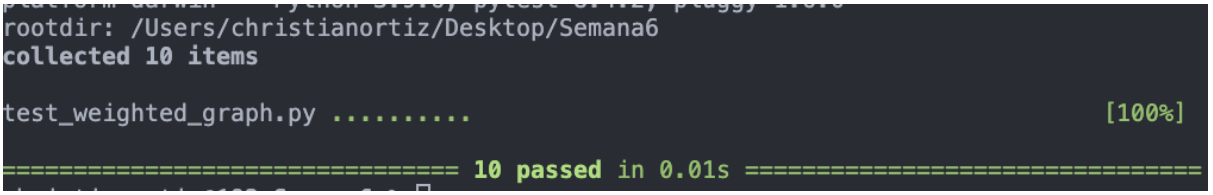

TABLA DE RUTAS

ORIGEN	DESTINO	RUTA	DISTANCIA
0	**1**	**0-1**	**7**
0	**2**	**0-2**	**9**
0	**5**	**0-2-5**	**11**
0	**3**	**0-2-3**	**20**
0	**4**	**0-2-3-4**	**26**

Comparación Dijkstra vs Floyd–Warshall

El algoritmo de Dijkstra resultó ser más rápido en este proyecto, ya que únicamente calcula las rutas más cortas desde un nodo origen, reduciendo la cantidad de operaciones necesarias. En cambio, Floyd–Warshall analiza todas las posibles combinaciones entre nodos, lo que aumenta considerablemente el tiempo de ejecución, aunque ofrece una matriz completa de distancias.

Dijkstra es más conveniente cuando se necesita obtener una ruta desde un punto específico, como en sistemas GPS o navegación en tiempo real. Por otro lado, Floyd–Warshall es más útil cuando se requiere conocer todas las rutas mínimas posibles dentro del sistema, como en análisis de accesibilidad o planificación global.

Reflexión: ¿Qué algoritmo escaló mejor en mi grafo?
el algoritmo que escaló mejor fue Dijkstra, ya que su tiempo de ejecución aumentó de forma mucho más eficiente conforme se agregaban nodos y aristas al grafo. Dijkstra únicamente calcula rutas mínimas desde un nodo origen, lo que reduce significativamente el trabajo necesario cuando no se requiere conocer todas las rutas posibles.

PROMPT 1 - IA

Escenario real	Problema	Algoritmo elegido	¿Por qué? (propiedades)	Desventaja del otro
🚗 GPS / Navegación en automóvil	Encontrar la mejor ruta desde tu ubicación hasta un destino específico	**Dijkstra**	Solo se necesita calcular rutas desde un origen → menor costo. Complejidad: O((V+E) log V). Excelente rendimiento en grafos grandes como mapas reales.	Floyd–Warshall calcularía rutas entre TODOS los puntos, desperdiciando recursos ($O(V^3)$)
🌐 Enrutamiento IP (routers)	Determinar la mejor salida hacia distintas redes desde un router	**Floyd–Warshall**	Necesita tabla completa de rutas entre todos los routers para decisiones rápidas. Calcula todas las distancias en 1 ejecución. Útil para redes estables.	Dijkstra tendría que ejecutarse varias veces (una por router), volviéndose más lento
📦 Logística de entregas (paquetería)	Optimizar rutas desde un centro de distribución hacia múltiples destinos	**Dijkstra**	Se tiene un origen fijo (almacén). Permite obtener rutas óptimas a muchos destinos sin cálculos innecesarios. Muy eficiente en grafos grandes y dispersos.	Floyd–Warshall tardaría mucho calculando rutas entre destinos que no se necesitan
🏫 Planificación escolar / campus	Determinar distancias mínimas entre todos los edificios	**Floyd–Warshall**	Necesitas todas las combinaciones edificio–edificio. Matriz completa permite consultas inmediatas. Complejidad fija $O(V^3)$ aceptable en grafos pequeños.	Dijkstra tendría que ejecutarse N veces para obtener toda la matriz → más lento
✈️ Aerolíneas / conexiones de vuelo	Encontrar ruta más barata entre ciudades específicas	**Dijkstra**	Búsqueda desde un aeropuerto origen hacia un destino. Evita calcular rutas innecesarias entre ciudades que no se usarán.	Floyd–Warshall escala mal con muchas ciudades, consume mucha memoria y CPU

CONCLUSIÓN

Durante la Semana 6 se logró integrar correctamente los algoritmos de Dijkstra y Floyd–Warshall dentro del proyecto, permitiendo calcular rutas óptimas en grafos con pesos. Ambos métodos fueron implementados y validados mediante pruebas unitarias, obteniendo un resultado satisfactorio con 10 pruebas aprobadas, lo que confirma el funcionamiento adecuado de las estructuras y algoritmos desarrollados.

Dijkstra demostró ser más eficiente para calcular rutas desde un único nodo origen, mostrando menor complejidad y mejor rendimiento en grafos medianos. Por otro lado, Floyd–Warshall permitió obtener todas las rutas óptimas entre pares de nodos, siendo más completo pero también más costoso computacionalmente.