**1. Redes Sociales**

Los grafos se utilizan para modelar redes sociales donde los nodos representan a las personas y las aristas representan las relaciones (amistades, seguidores, conexiones, etc.) entre ellas.

Ejemplo de Uso:

- Facebook: Modela las amistades entre usuarios.

- Twitter: Representa a los usuarios y sus seguidores.

**2. Sistemas de Recomendación**

Los grafos se utilizan en sistemas de recomendación para representar las relaciones entre usuarios y productos, servicios o contenidos.

Ejemplo de Uso:

- Netflix: Utiliza grafos para recomendar películas y series basándose en las preferencias y el historial de visualización de los usuarios.

- Amazon: Recomienda productos a los usuarios basándose en sus compras anteriores y en las compras de usuarios similares.

**3. Navegación y Mapas**

Los grafos modelan sistemas de transporte y mapas, donde los nodos representan lugares (ciudades, intersecciones) y las aristas representan rutas o carreteras.

Ejemplo de Uso:

- Google Maps: Utiliza grafos para encontrar la ruta más corta entre dos puntos, considerando el tráfico y otros factores.

- GPS: Los sistemas de navegación GPS utilizan grafos para guiar a los conductores a su destino.

**4. Bioinformática**

Los grafos se aplican en bioinformática para modelar relaciones biológicas y procesos biológicos complejos, como las interacciones entre proteínas.

Ejemplo de Uso:

- Redes de Interacción Proteica: Los grafos representan proteínas como nodos y las interacciones entre ellas como aristas.

- Secuenciación del Genoma: Utiliza grafos para ensamblar fragmentos de ADN y obtener la secuencia completa.

**5. Redes de Telecomunicaciones**

Los grafos modelan las infraestructuras de redes de telecomunicaciones, donde los nodos representan conmutadores, enrutadores y otros dispositivos de red, y las aristas representan los enlaces de comunicación entre ellos.

Ejemplo de Uso:

- Optimización de Redes: Las empresas de telecomunicaciones utilizan grafos para optimizar la distribución de datos y la configuración de redes.

- Detección de Fallos: Los grafos ayudan a identificar y aislar fallos en la red.

**Purpose and Scope:**

- **Dijkstra's Algorithm:** Designed to find the shortest path from a single source vertex to all other vertices in a graph. It works on graphs with non-negative weights.

- **Floyd-Warshall Algorithm:** Computes the shortest paths between all pairs of vertices in a graph. It can handle negative weights but no negative cycles.

**Complexity:**

- **Dijkstra's Algorithm:** More efficient for sparse graphs, with a time complexity of \(O(V^2)\) using a simple implementation or \(O(V \log V + E)\) with a priority queue (binary heap).

- **Floyd-Warshall Algorithm**: Better suited for dense graphs, with a time complexity of \(O(V^3)\). It uses dynamic programming to update shortest paths iteratively.

**Approach:**

- **Dijkstra's Algorithm:** Utilizes a greedy approach. It incrementally builds the shortest path tree, starting from the source vertex and expanding to the nearest vertices not yet processed.

- **Floyd-Warshall Algorithm:** Employs a dynamic programming approach. It iteratively updates the shortest paths by considering each vertex as an intermediate point in potential paths between other vertices.

**Data Structures:**

- **Dijkstra's Algorithm: Typically** uses a priority queue (min-heap) to efficiently select the next vertex with the smallest tentative distance.

- **Floyd-Warshall Algorithm:** Uses a 2D matrix to store shortest path distances between every pair of vertices, updating this matrix in each iteration.