Inteligencia Artificial

Práctica 3: Algoritmo de Dijkstra

Profesor: Mauricio Alejandro Cabrera Arellano

Alumno: Omar Josue Munguia Camacho

Registro: 21110391

Grupo: 6E2

- ¿Qué es el Algoritmo de Dijkstra?

El Algoritmo de Dijkstra es un método de búsqueda en grafos que se utiliza para encontrar las rutas más cortas desde un nodo inicial a todos los demás nodos en un grafo ponderado. Este algoritmo fue desarrollado por el científico informático holandés Edsger W. Dijkstra en 1956 y publicado en 1959. Es una técnica fundamental en la teoría de grafos y se utiliza en diversas aplicaciones de la informática y la ingeniería.

Su funcionamiento es de la siguiente manera:

El Algoritmo de Dijkstra funciona encontrando la ruta más corta desde un nodo inicial a todos los demás nodos en un grafo ponderado.

- Pasos del Algoritmo de Dijkstra

1) Inicialización:

   - Distancias: Se establece la distancia del nodo inicial a sí mismo como 0 y la distancia a todos los demás nodos como infinita.
   - Cola de Prioridad: Se utiliza una cola de prioridad (min-heap) para seleccionar el nodo con la distancia mínima no visitado.
   - Nodos Visitados: Se mantiene un conjunto de nodos visitados para evitar procesar el mismo nodo más de una vez.
   - Nodos Previos: Se almacena el nodo previo para cada nodo, permitiendo reconstruir la ruta más corta al final.

2) Bucle Principal:

   - Mientras la cola de prioridad no esté vacía:

     - Extraer el nodo con la menor distancia acumulada de la cola de prioridad.

     - Si el nodo ya ha sido visitado, se salta.

     - Se marca el nodo como visitado.

     - Se examinan todos los vecinos del nodo actual:

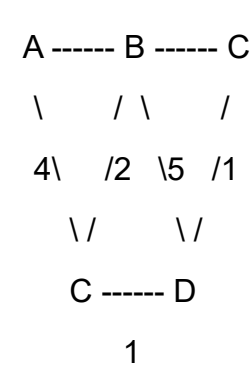
       - ✓ Se calcula la distancia a través del nodo actual a cada vecino.
       - ✓ Si esta nueva distancia es menor que la distancia conocida, se actualiza la distancia y se agrega el vecino a la cola de prioridad.
       - ✓ Se actualiza el nodo previo del vecino para poder reconstruir la ruta más corta.

3) Reconstrucción de la Ruta:

   - Una vez que se han procesado todos los nodos, se puede utilizar el registro de nodos previos para reconstruir las rutas más cortas desde el nodo inicial a cualquier otro nodo.

- Ejemplo de Funcionamiento

Supongamos un grafo simple:



- Paso a Paso

1) Inicialización:

- Distancias: `{'A': 0, 'B': ∞, 'C': ∞, 'D': ∞}`
- Cola de Prioridad: `[(0, 'A')]`
- Nodos Visitados: `{}`
- Nodos Previos: `{'A': None, 'B': None, 'C': None, 'D': None}`

2) Primer Iteración:

- Extraer: `(0, 'A')`
- Visitar: `'A'`
- Actualizar distancias y nodos previos para vecinos de 'A':

- `'B'`: Nueva distancia = 0 + 1 = 1 (`distancias['B']` se actualiza a 1, `previous_nodes['B']` se actualiza a 'A')

- `'C'`: Nueva distancia = 0 + 4 = 4 (`distancias['C']` se actualiza a 4, `previous_nodes['C']` se actualiza a 'A')

- Cola de Prioridad: `[(1, 'B'), (4, 'C')]`

3) Segunda Iteración:

- Extraer: `(1, 'B')`

- Visitar: `'B'`

- Actualizar distancias y nodos previos para vecinos de 'B':

- `'C'`: Nueva distancia = 1 + 2 = 3 (mejora la distancia de 4 a 3, `distancias['C']` se actualiza a 3, `previous_nodes['C']` se actualiza a 'B')

- `'D'`: Nueva distancia = 1 + 5 = 6 (`distancias['D']` se actualiza a 6, `previous_nodes['D']` se actualiza a 'B')

- Cola de Prioridad: `[(3, 'C'), (4, 'C'), (6, 'D')]`

4) Tercera Iteración:

- Extraer: `(3, 'C')`

- Visitar: `'C'`

- Actualizar distancias y nodos previos para vecinos de 'C':

- `'D'`: Nueva distancia = 3 + 1 = 4 (mejora la distancia de 6 a 4, `distancias['D']` se actualiza a 4, `previous_nodes['D']` se actualiza a 'C')

- Cola de Prioridad: `[(4, 'C'), (6, 'D'), (4, 'D')]`

5) Cuarta Iteración:

- Extraer: `(4, 'C')`

- `'C'` ya ha sido visitado, se salta.

- Cola de Prioridad: `[(4, 'D'), (6, 'D')]`

6) Quinta Iteración:

- Extraer: `(4, 'D')`

- Visitar: `'D'`

- `'D'` no tiene vecinos que mejoren la distancia.

- Cola de Prioridad: `[(6, 'D')]`

7) Sexta Iteración:

- Extraer: `(6, 'D')`

- `'D'` ya ha sido visitado, se salta.

- Cola de Prioridad: `[]`

- Resultados Finales

- Distancias: {'A': 0, 'B': 1, 'C': 3, 'D': 4}

- Nodos Previos: {'A': None, 'B': 'A', 'C': 'B', 'D': 'C'}

El algoritmo termina cuando se han visitado todos los nodos posibles, proporcionando las distancias más cortas desde el nodo inicial a todos los demás nodos y los nodos previos necesarios para reconstruir las rutas más cortas.

- ¿Para qué sirve el Algoritmo de Dijkstra?

El Algoritmo de Dijkstra se utiliza para resolver problemas de rutas más cortas en grafos ponderados. Es útil en situaciones donde se necesita determinar la ruta más eficiente o de menor costo entre dos puntos, considerando que las conexiones (aristas) entre los puntos (nodos) tienen diferentes pesos (costos o distancias).

- ¿Cómo se implementa en el mundo?

El Algoritmo de Dijkstra se implementa en diversas áreas y aplicaciones del mundo real, tales como:

- Sistemas de Navegación GPS: Los dispositivos de navegación utilizan este algoritmo para calcular las rutas más rápidas y eficientes entre ubicaciones geográficas.

- Redes de Computadoras: Enrutadores y protocolos de red, como el Protocolo de Estado de Enlace (OSPF), emplean Dijkstra para determinar las rutas más cortas y óptimas para la transmisión de datos.
- Planificación de Transporte y Logística: Se utiliza para optimizar rutas de transporte y entrega, minimizando el tiempo y el costo de desplazamiento.
- Juegos y Simulaciones: En los videojuegos, se usa para que los personajes no jugadores (NPC's) encuentren caminos más cortos y realistas en su entorno.

➢ ¿Cómo se puede implementar en la vida diaria?

El Algoritmo de Dijkstra puede ser implementado en la vida diaria de las siguientes maneras:

- Planificación de Viajes: Ayuda a encontrar la ruta más rápida o económica al viajar de un lugar a otro, ya sea en coche, a pie o en transporte público.
- Gestión de Tiempo: Optimiza la agenda diaria al determinar el orden más eficiente para realizar múltiples tareas en diferentes ubicaciones.

- Aplicaciones Móviles: Las aplicaciones de mapas y navegación que se utilizan cotidianamente para desplazarse por la ciudad implementan este algoritmo para ofrecer rutas eficientes.

➢ ¿Cómo se implementaría en el trabajo?

En el entorno laboral, el Algoritmo de Dijkstra puede ser aplicado de varias formas, dependiendo del campo y la industria:

- Logística y Cadena de Suministro: Optimiza las rutas de distribución y entrega de productos, reduciendo costos y mejorando la eficiencia.
- Ingeniería de Redes: Diseña y optimiza redes de comunicación, asegurando que los datos viajen por las rutas más eficientes.
- Gestión de Proyectos: Identifica la secuencia más corta y eficiente de tareas en proyectos complejos, especialmente en proyectos de construcción o desarrollo de software.
- Análisis de Datos y Toma de Decisiones: Ayuda a analizar redes de datos y a tomar decisiones informadas sobre la mejor manera de manejar flujos de información y recursos.
- Investigación Operativa: Utiliza el algoritmo para resolver problemas de optimización en la investigación de operaciones, mejorando procesos y sistemas en diversas industrias.

En resumen, el Algoritmo de Dijkstra es una herramienta poderosa y versátil que encuentra aplicaciones en una amplia gama de campos y situaciones, desde la navegación y la logística hasta la planificación de proyectos y la optimización de redes. Su capacidad para encontrar rutas más cortas y eficientes lo hace invaluable en muchas áreas de la vida diaria y profesional.