

Inteligencia Artificial

Práctica 3

**"Simulador de Algoritmo de
Dijkstra."**

Samuel Josimar Orozco Torres

21110380 - 6E2

Índice

¿Qué es?

Pág. 3

¿Para qué sirve?

Pág. 5

¿Cómo se implementa en el mundo?

Pág. 8

¿Cómo lo implementarías en tu vida?

Pág. 10

¿Cómo lo implementarías en tu trabajo o trabajo de ensueño?

Pág. 11

Bibliografía

Pág. 12



¿Qué es?

El algoritmo de Dijkstra es un método para encontrar las rutas más cortas desde un vértice de un grafo hasta todos los demás.

- El algoritmo de Dijkstra se utiliza para encontrar la ruta más corta desde el vértice inicial del grafo hasta todos los demás.
- El algoritmo de Dijkstra considera todos los vértices de manera equivalente y siempre elige el vértice con la distancia más pequeña hasta él.
- El algoritmo de Dijkstra puede ser lento para grafos grandes, ya que explora todos los vértices.

Los grafos son estructuras de datos usadas para representar "conexiones" entre pares de elementos.

- Estos elementos se llaman nodos. Representan objetos reales, personas o entidades.
- Las conexiones entre los nodos se llaman aristas o arcos.

Los nodos se representan como círculos de colores y los arcos se representan como líneas que conectan los círculos.

Dos nodos están conectados si existe un arco entre ellos.

Los grafos se utilizan para representar objetos, sus interconexiones y las relaciones entre ellos.

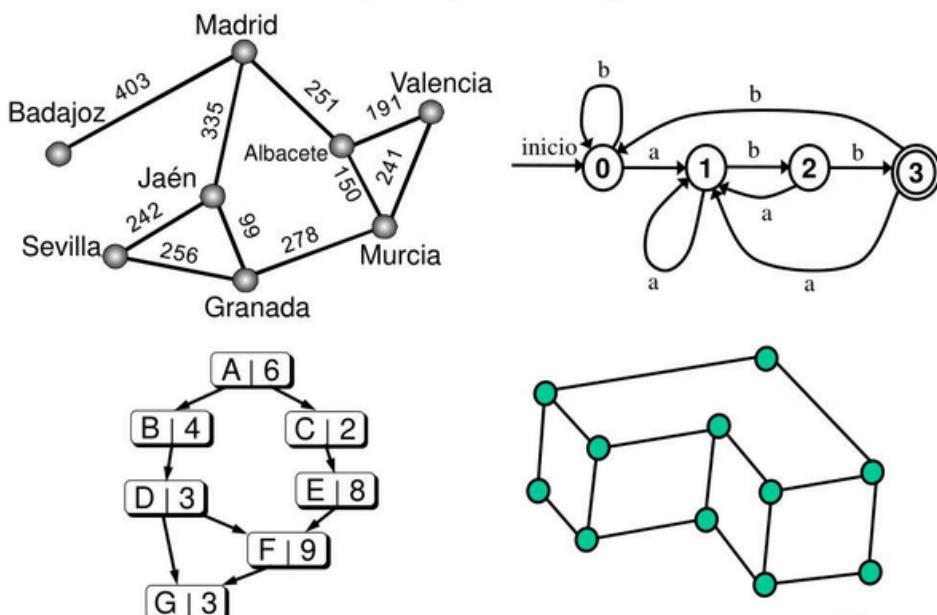


Figura 1. Ejemplo de grafos.

Los grafos pueden ser:

- No dirigido: si para cada par de nodos conectados, puedes ir de un nodo al otro en ambas direcciones.
- Dirigido: si para cada par de nodos conectados, solo puedes ir de un nodo a otro en una dirección específica. Usamos flechas en lugar de líneas sencillas para representar arcos dirigidos.

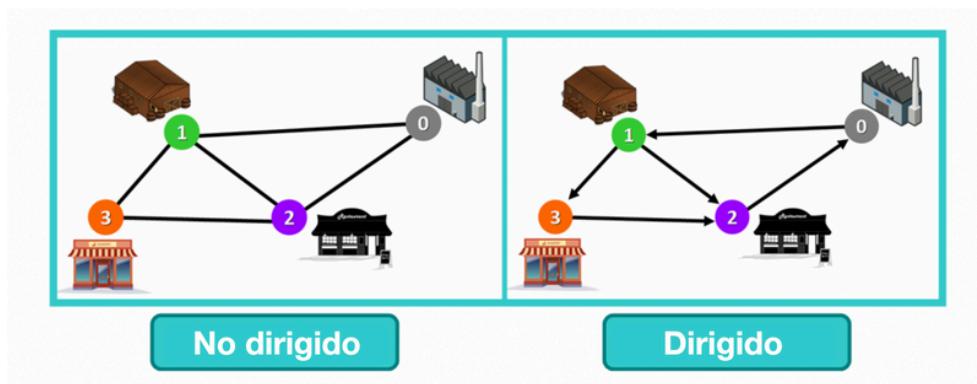


Figura 2. Ejemplo grafo no dirigido y dirigido.

Un grafo ponderado es un grafo cuyos arcos tienen un "peso", "valor", o "costo" asociado. El valor de cada arco puede representar la distancia, tiempo, u otro valor que modele la conexión entre el par de nodos que conecta.

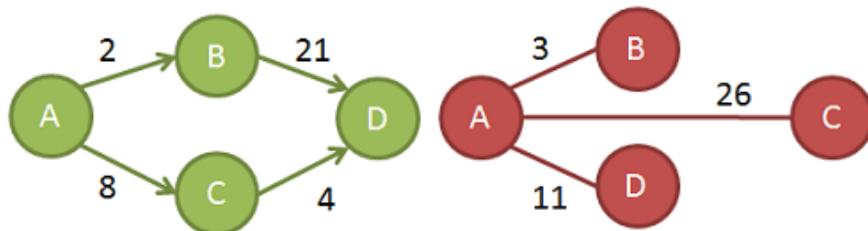


Figura 3. Ejemplo grafo ponderado.

¿Para qué sirve?

Con el algoritmo de Dijkstra, puedes encontrar la ruta más corta o el camino más corto entre los nodos de un grafo. Específicamente, puedes encontrar el camino más corto desde un nodo (llamado el nodo de origen) a todos los otros nodos del grafo, generando un árbol del camino más corto.

Este algoritmo es usado por los dispositivos GPS para encontrar el camino más corto entre la ubicación actual y el destino del usuario. Tiene amplias aplicaciones en la industria, especialmente en aquellas áreas que requieren modelar redes.

Aspectos básicos del algoritmo de Dijkstra

- El algoritmo de Dijkstra básicamente inicia en el nodo que escogas (el nodo de origen) y analiza el grafo para encontrar el camino más corto entre ese nodo y todos los otros nodos en el grafo.
- El algoritmo mantiene un registro de la distancia conocida más corta desde el nodo de origen hasta cada nodo y actualiza el valor si encuentra un camino más corto.
- Una vez que el algoritmo ha encontrado el camino más corto entre el nodo de origen y otro nodo, ese nodo se marca como "visitado" y se agrega al camino.
- El proceso continúa hasta que todos los nodos en el grafo han sido añadidos al camino. De esta forma, tenemos un camino que conecta al nodo de origen con todos los otros nodos siguiendo el camino más corto posible para llegar a cada uno de ellos.

Requisitos

El algoritmo de Dijkstra solo puede aplicarse a grafos con arcos cuyos valores o pesos son positivos. Esto se debe a que, durante el proceso, los valores de los arcos deben ser sumados para encontrar el camino más corto.

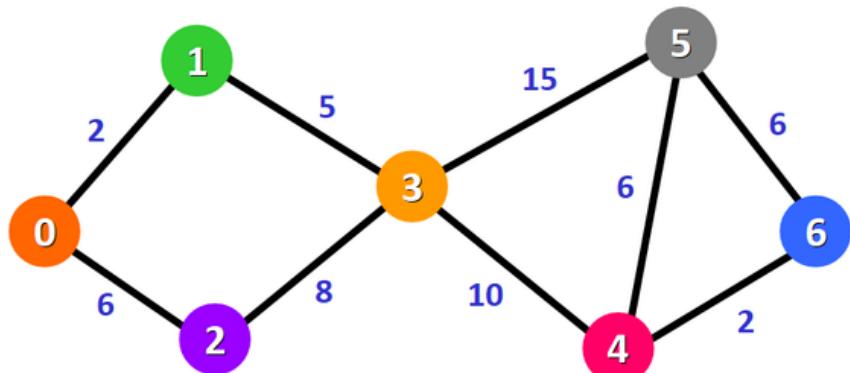
Si existe un valor negativo en el grafo, el algoritmo no funcionará correctamente. Una vez que el nodo se marca como "visitado", el camino actual hacia ese nodo se marca como el camino más corto para alcanzar ese nodo, pero los valores negativos pueden cambiar esto si el valor total puede ser reducido luego de este paso.

Procedimiento

El algoritmo de Dijkstra comenzará inicialmente con distancias infinitas e intentará mejorarlas paso a paso.

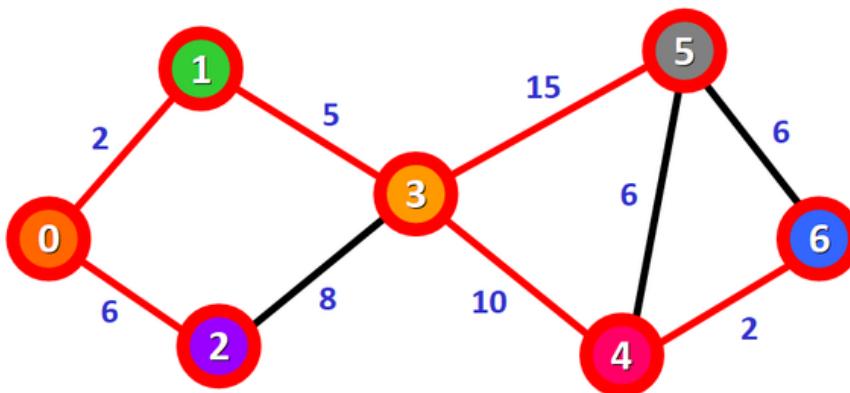
- Marca todos los nodos sin mirar. Crear un conjunto de todos los nodos no vistos llamados el conjunto no previsto.
- Asignar a cada nodo a Distancia provisional valor: establecerlo a cero para nuestro nodo inicial y para infinito para todos los demás nodos. Durante el funcionamiento del algoritmo, la distancia tentativa de un nodo v es la longitud del camino más corto descubierto hasta ahora entre el nodo v y el Nodo. Puesto que inicialmente ningún camino es conocido por cualquier otro vértice que la fuente misma (que es un camino de la longitud cero), todas las otras distancias tentativas se fijan inicialmente para el infinito.
- Para el nodo actual, considere a todos sus vecinos no vistos y calcula sus distancias tentativas a través del nodo actual. Compare la distancia tentativa recién calculada a la que se asigna actualmente al vecino y la asigne la más pequeña. Por ejemplo, si el nodo actual A está marcado con una distancia de 6, y el borde que lo conecta con un vecino B tiene longitud 2, entonces la distancia a B a través de A será $6 + 2 = 8$. Si B fue marcado previamente con una distancia mayor a 8 entonces cambiar a 8. De lo contrario, se mantendrá el valor actual.
- Cuando terminemos de considerar a todos los vecinos no vistos del nodo actual, marque el nodo actual como visitado y retírelo del conjunto no visto. Un nodo visitado nunca será revisado de nuevo (esto es válido y óptimo en relación con el comportamiento en el paso 6.: que los próximos nodos a visitar siempre estarán en el orden de 'la distancia más reducida de nodo inicial primero' así que cualquier visita después tendría una mayor distancia).
- Si el nodo de destino se ha marcado visitado (cuando se planea una ruta entre dos nodos específicos) o si la distancia más pequeña entre los nodos en el conjunto no previsto es infinidad (cuando se planea una trasversal completa; se produce cuando no hay conexión entre el nodo inicial y los nodos no visibles), entonces parar. El algoritmo ha terminado.
- De lo contrario, seleccione el nodo no visto que está marcado con la distancia más pequeña tentativa, fijarlo como el nuevo actual nodo, y volver al paso 3.

Al planificar una ruta, en realidad no es necesario esperar hasta que el nodo de destino sea "visitado" como arriba: el algoritmo puede detenerse una vez que el nodo de destino tenga la distancia tentativa más pequeña entre todos los "no visitados" nodos (y por lo tanto podría ser seleccionado como el próximo 'actual').



Distancia: Nodos por visitar: {0, 1, 2, 3, 4, 5, 6}

0: 0
1: ∞
2: ∞
3: ∞
4: ∞
5: ∞
6: ∞



Distancia: Nodos por visitar: {0, 1, 2, 3, 4, 5, 6}

0: 0
1: 2
2: 6
3: 7
4: 10
5: ∞
6: 19

Figura 4. Ejemplo de Algoritmo de Dijkstra.

¿Cómo se implementa en el mundo?

El algoritmo de Dijkstra se utiliza ampliamente en la resolución de problemas reales donde es necesario encontrar la ruta más corta o menos costosa entre ubicaciones.



Sistemas de navegación y GPS:

En aplicaciones de navegación como Google Maps y Waze, el algoritmo de Dijkstra es la base para calcular la ruta óptima entre dos puntos en función de la distancia, el tiempo estimado o el tráfico actual. Las carreteras y caminos son representados como un grafo, donde los cruces son nodos y las carreteras entre ellos son las aristas con un peso determinado (como la distancia o el tiempo de viaje).



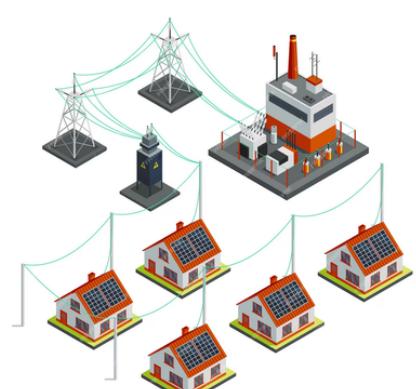
Redes de telecomunicaciones:

En redes de Internet y telecomunicaciones, se utiliza para la gestión y optimización de las rutas de datos. Los routers aplican variantes de Dijkstra para encontrar la mejor ruta a través de la red, minimizando el tiempo de transmisión y el consumo de recursos de red. Cada nodo es un router y las conexiones entre ellos son las aristas con peso basado en métricas como latencia o ancho de banda.



Sistemas de energía:

En la distribución de electricidad, el algoritmo puede aplicarse para optimizar el flujo de energía desde las plantas generadoras hasta las zonas de consumo, reduciendo pérdidas en las líneas de transmisión y distribuyendo la carga de forma eficiente.



Videojuegos y simulaciones:

En videojuegos que requieren el cálculo de rutas para personajes o enemigos (pathfinding), como en juegos de estrategia o de mundo abierto, se usa para que los personajes encuentren el camino más rápido hacia un objetivo, evitando obstáculos.



Logística y planificación de rutas:

En compañías de logística como Amazon o DHL, se utiliza Dijkstra para planificar rutas de entrega óptimas, minimizando el costo o el tiempo de entrega. También se puede aplicar en la planificación de rutas para servicios de transporte como autobuses o trenes.



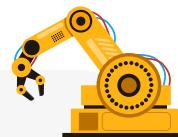
Inteligencia artificial y robótica:

En robótica, Dijkstra ayuda a que los robots autónomos encuentren rutas óptimas para navegar por espacios complejos, ya sea en el interior de edificios, en líneas de ensamblaje o en almacenes.



¿Cómo lo implementarías en tu vida?

Como estudiante de ingeniería mecatrónica, puedes implementar el algoritmo de Dijkstra en varias situaciones para optimizar tiempo y recursos.

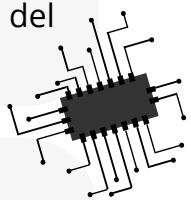


Proyectos de robótica o automatización

Aplicar Dijkstra para que un robot encuentre la ruta más corta en un espacio con obstáculos, por ejemplo, en un prototipo de robot autónomo.

Optimización en diseño de circuitos

En circuitos electrónicos (como en proyectos con Arduino), utilizar Dijkstra para optimizar la disposición de componentes y minimizar la longitud de los cables, lo cual reduce interferencias y pérdidas de señal, mejorando la eficiencia del circuito.



Optimización de trayectorias para brazos robóticos

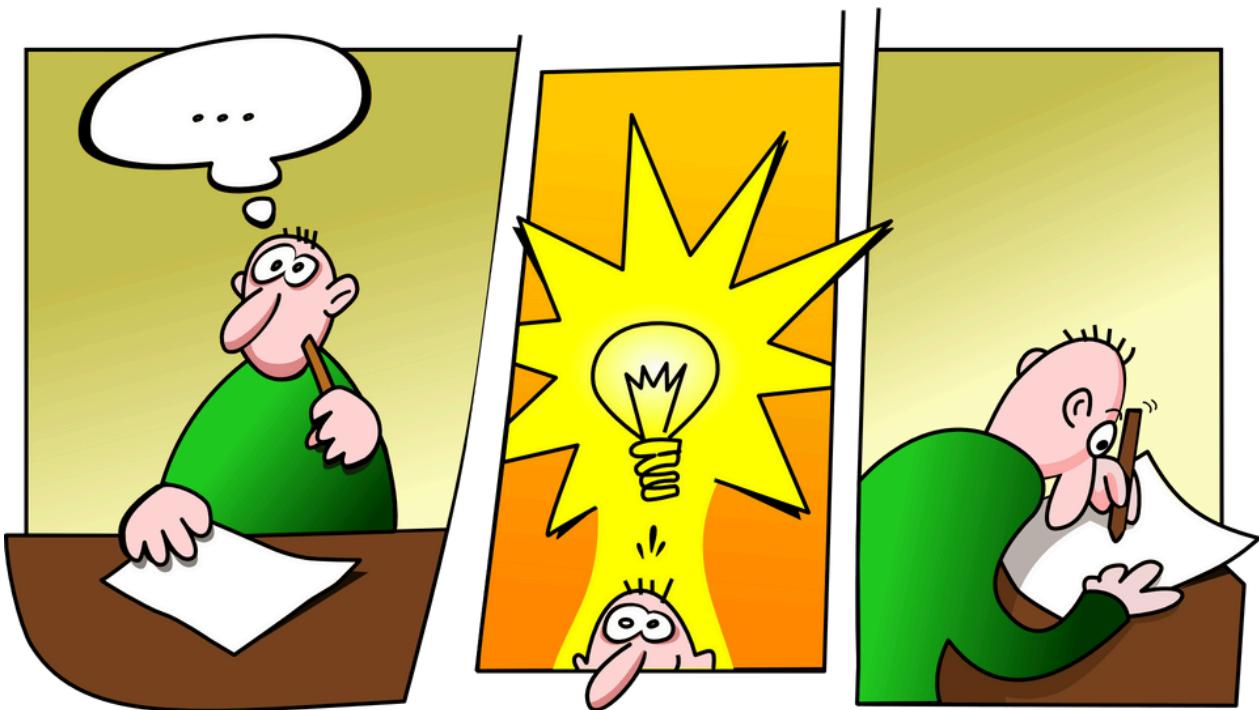
En proyectos de robótica con brazos manipuladores, el algoritmo permite calcular la trayectoria óptima de movimiento en el espacio de trabajo. Cada posición del brazo en su espacio de trabajo se considera un nodo, y el peso puede representar la energía o el tiempo requerido para moverse de una posición a otra, ayudando a reducir el consumo de energía o tiempo de operación.



Análisis y simulación de redes de sensores

En proyectos de IoT o en redes de sensores, Dijkstra es útil para optimizar el flujo de información entre nodos, mejorando el rendimiento de la red en términos de consumo de energía y rapidez de transmisión de datos.





¿Cómo lo implementarías en tu trabajo o tu trabajo de ensueño?

Mi trabajo de ensueño, que se centra en el diseño de robots, brazos robóticos y vehículos a control remoto, implementaría el algoritmo de Dijkstra de varias maneras para optimizar las operaciones y mejorar la eficiencia de los sistemas.

Uno de los mayores retos en robótica es calcular la trayectoria más eficiente para que un robot o un brazo robótico se mueva de un punto a otro sin chocar con obstáculos. Utilizando el algoritmo de Dijkstra, modelaría el espacio de trabajo del robot como un grafo, donde cada posición sería un nodo y las posibles rutas entre ellas serían las aristas con un peso basado en el tiempo o la energía requerida para realizar el movimiento. De esta forma, el algoritmo me permitiría encontrar la trayectoria más corta o eficiente, minimizando el tiempo de operación y el consumo de energía.

Bibliografía

- Byte, L. (2024, 24 octubre). El Algoritmo de Dijkstra: Cómo Funciona y Dónde se Usa» CódigoNautas. CódigoNautas. <https://codigonautas.com/algoritmo-dijkstra-que-es-ejemplo/>
- Navone, E. C. (2023, 2 agosto). Algoritmo de la ruta más corta de Dijkstra - Introducción gráfica y detallada. freeCodeCamp.org. <https://www.freecodecamp.org/espanol/news/algoritmo-de-la-ruta-mas-corta-de-dijkstra-introduccion-grafica/>
- Algoritmo De Dijkstra _ AcademiaLab. (s. f.). <https://academialab.com/encyclopedia/algoritmo-de-dijkstra/>
- VGA. Un visualizador genérico de algoritmos. El Algoritmo de Dijkstra o de Caminos mínimos. (s. f.). <https://atlas.uned.es/algoritmos/voraces/dijkstra.html>