Tarea #3

“Algoritmo de Dijkstra”

INTELIGENCIA ARTIFICIAL

18:00 – 19:00 Am

Profesor:

JOSE MARIO RIOS FELIX

Alumno:

Estrada Valenzuela Jesús Ernesto

7to semestre

ING. En Sistemas Computacionales

04/09/2025

Introducción

En el área de la informática y las ciencias de la computación, los algoritmos de búsqueda de caminos más cortos representan una de las herramientas más importantes para la resolución de problemas relacionados con grafos. Entre ellos, uno de los más conocidos y aplicados es el **algoritmo de Dijkstra**, desarrollado en 1956 por el científico neerlandés Edsger Wybe Dijkstra y publicado oficialmente en 1959 en el artículo *"A note on two problems in connexion with graphs"*. Este algoritmo fue ideado inicialmente para encontrar la ruta más eficiente en sistemas de telecomunicaciones, pero con el paso del tiempo se ha convertido en un pilar fundamental para aplicaciones modernas como sistemas de navegación GPS, redes de computadoras, logística, inteligencia artificial en videojuegos y muchas otras áreas.

Descripción del Algoritmo

El algoritmo de Dijkstra pertenece a la categoría de algoritmos voraces (greedy), lo que significa que en cada paso toma la decisión que parece más óptima localmente con la esperanza de que esta lleve a la solución global más adecuada. Su objetivo principal es determinar el **camino más corto desde un nodo origen hasta todos los demás nodos** de un grafo ponderado, siempre y cuando los pesos de las aristas sean no negativos.

El procedimiento consiste en ir explorando los nodos del grafo, acumulando distancias desde el nodo de inicio y actualizando dichas distancias cada vez que se encuentra un camino más corto. Una vez que un nodo ha sido procesado, se considera definitivo el valor de la distancia más corta hacia él.

En términos formales, el algoritmo opera sobre un grafo G=(V,E)G = (V, E)G=(V,E), donde VVV es el conjunto de vértices (nodos) y EEE es el conjunto de aristas (conexiones entre nodos). A cada arista se le asigna un peso positivo que representa una medida de distancia, costo o tiempo.

Funcionamiento Paso a Paso

El funcionamiento del algoritmo de Dijkstra puede resumirse en las siguientes etapas:

1. **Inicialización:** Se asigna una distancia de valor cero al nodo de origen y se establece la distancia infinita para todos los demás nodos.
2. **Selección:** Se escoge el nodo con la menor distancia provisional que aún no ha sido procesado.
3. **Relajación:** Para cada vecino de ese nodo, se calcula si la suma de la distancia actual más el peso de la arista hacia el vecino es menor que la distancia previamente registrada para ese nodo. En caso afirmativo, la distancia se actualiza.
4. **Marcado:** Una vez revisados todos los vecinos, se marca el nodo como visitado, indicando que ya no será modificado.
5. **Repetición:** El proceso se repite hasta que todos los nodos hayan sido visitados o, si se busca un nodo específico, hasta que este se procese.

Este procedimiento garantiza encontrar el camino más corto en grafos sin aristas con pesos negativos, ya que la estructura del algoritmo depende de la comparación de distancias acumuladas.

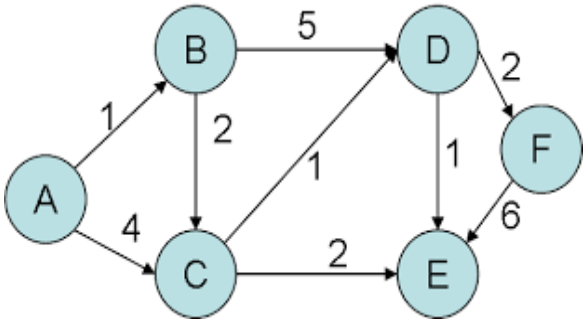
Complejidad Computacional

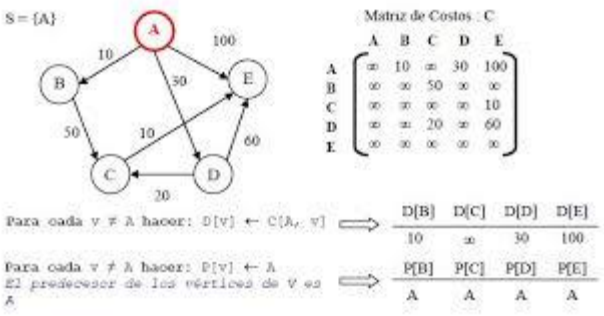
La eficiencia del algoritmo depende de la forma en que se represente el grafo y la estructura de datos utilizada para seleccionar el nodo con menor distancia:

- Si se utiliza una **matriz de adyacencia**, la complejidad es de **$O(V^2)$**, lo cual resulta adecuado para grafos pequeños o densos.
- Si se emplea una **lista de adyacencia junto con una cola de prioridad (PriorityQueue)**, la complejidad se reduce a **$O((V + E) \log V)$**, lo que lo hace más eficiente en grafos grandes y dispersos.

Ejemplo de Funcionamiento

Para ilustrar el funcionamiento del algoritmo, consideremos un grafo simple que representa un conjunto de ciudades conectadas por carreteras con diferentes distancias:





Este ejemplo muestra que el algoritmo no siempre elige el camino directo, sino el más corto acumulado.

Aplicaciones del Algoritmo

El algoritmo de Dijkstra tiene múltiples aplicaciones en la vida real, entre las que se destacan:

1. **Sistemas de navegación GPS:** Determinar la ruta más corta o más rápida entre dos puntos de un mapa digital.
2. **Redes de telecomunicaciones e Internet:** Optimizar la transmisión de paquetes en protocolos de enrutamiento como OSPF.
3. **Logística y transporte:** Minimizar los costos de distribución o tiempos de entrega en redes de transporte.
4. **Inteligencia artificial en videojuegos:** Permite que los personajes controlados por computadora encuentren rutas en mapas complejos.
5. **Robótica:** Se utiliza en la planificación de trayectorias para robots móviles y vehículos autónomos.

Ventajas y Desventajas

Ventajas:

- Es sencillo de comprender e implementar.
- Es muy eficiente para grafos sin aristas de peso negativo.
- Se adapta bien a múltiples representaciones de grafos.
- Constituye la base para algoritmos más avanzados como A*.

Desventajas:

- No funciona correctamente en grafos que poseen aristas con pesos negativos.
- En grafos muy grandes, puede consumir muchos recursos computacionales si no se utilizan estructuras optimizadas.

Variantes y Algoritmos Relacionados

El algoritmo de Dijkstra ha inspirado variantes y otros algoritmos que solucionan problemas relacionados:

- **Algoritmo de Bellman-Ford:** Permite manejar grafos con aristas de peso negativo, aunque es menos eficiente.
- **Algoritmo de Floyd-Warshall:** Encuentra los caminos más cortos entre todos los pares de nodos de un grafo.
- **Algoritmo A* (A estrella):** Utiliza heurísticas para acelerar la búsqueda hacia un nodo destino específico.
- **Algoritmo de Johnson:** Optimiza la búsqueda de caminos en grafos grandes con posibles pesos negativos pero sin ciclos negativos.

CODIGO DE EJEMPLO EN JAVA

```
import java.util.*;


public class Dijkstra {
  static class Edge {
    int to, weight;
    Edge(int t, int w) { to = t; weight = w; }
  }


  public static int[] dijkstra(List<List<Edge>> graph, int source) {
    int n = graph.size();
    int[] dist = new int[n];
    Arrays.fill(dist, Integer.MAX_VALUE);
    dist[source] = 0;
```

```
    PriorityQueue<int[]> pq = new PriorityQueue<>(Comparator.comparingInt(a -> a[0]));
    pq.offer(new int[]{0, source}); // [distancia, nodo]

    boolean[] visited = new boolean[n];

    while (!pq.isEmpty()) {
      int[] current = pq.poll();
      int d = current[0], u = current[1];

      if (visited[u]) continue;
      visited[u] = true;

      for (Edge edge : graph.get(u)) {
        int v = edge.to, w = edge.weight;
        if (dist[u] != Integer.MAX_VALUE && d + w < dist[v]) {
          dist[v] = d + w;
          pq.offer(new int[]{dist[v], v});
        }
      }
    }
    return dist;
  }

  public static void main(String[] args) {
    int n = 6; // número de nodos
```

```
    List<List<Edge>> graph = new ArrayList<>();
    for (int i = 0; i < n; i++) graph.add(new ArrayList<>());

    // Agregar aristas (ejemplo)
    graph.get(0).add(new Edge(1, 7));
    graph.get(0).add(new Edge(2, 9));
    graph.get(0).add(new Edge(5, 14));
    graph.get(1).add(new Edge(2, 10));
    graph.get(1).add(new Edge(3, 15));
    graph.get(2).add(new Edge(3, 11));
    graph.get(2).add(new Edge(5, 2));
    graph.get(3).add(new Edge(4, 6));
    graph.get(4).add(new Edge(5, 9));

    int source = 0;
    int[] dist = dijkstra(graph, source);

    System.out.println("Distancias desde el nodo " + source + ":");
    for (int i = 0; i < dist.length; i++) {
        System.out.println("Hasta nodo " + i + " -> " + dist[i]);
    }
  }
}
```

Conclusión

El algoritmo de Dijkstra constituye una de las aportaciones más influyentes a la teoría de grafos y la informática en general. Su capacidad para calcular caminos más cortos en redes con pesos positivos lo ha convertido en una herramienta indispensable en campos como la navegación satelital, la logística empresarial, el diseño de redes de comunicación y la inteligencia artificial. Aunque presenta limitaciones frente a grafos con aristas de peso negativo, su simplicidad, eficiencia y adaptabilidad lo mantienen como uno de los algoritmos más utilizados en la práctica. Su estudio no solo es relevante en el ámbito académico, sino que también tiene un alto impacto en la vida cotidiana gracias a sus aplicaciones en tecnologías que usamos diariamente.

Referencias

- Dijkstra, E. W. (1959). *A note on two problems in connexion with graphs*. Numerische Mathematik, 1, 269–271.
- Wikipedia. (2025). *Algoritmo de Dijkstra*. Recuperado de: https://es.wikipedia.org/wiki/Algoritmo_de_Dijkstra
- FreeCodeCamp. (2023). *Algoritmo de la ruta más corta de Dijkstra – introducción gráfica*. Recuperado de: https://www.freecodecamp.org/espanol/news/algoritmo-de-la-ruta-mas-corta-de-dijkstra-introduccion-grafica/
- Baeldung. (2024). *Dijkstra's Algorithm in Java*. Recuperado de: https://www.baeldung.com/java-dijkstra
- GeeksForGeeks. (2024). *Dijkstra's shortest path algorithm in Java using PriorityQueue*. Recuperado de: https://www.geeksforgeeks.org/java/dijkstras-shortest-path-algorithm-in-java-using-priorityqueue/