CETI.

Actividad: Algoritmo dijkstra.

Nombre: David Alejandro Cisneros Delgadillo.
Registro: 21110381.
Grupo: 6°E.
Parcial 3.
Fecha: *10/06/2024.*
Nombre de la asignatura: Inteligencia Artificial.

¿Qué es el Algoritmo de Dijkstra?

El algoritmo de Dijkstra, concebido por el científico de la computación holandés Edsger W. Dijkstra en 1956 y publicado tres años después, es una técnica para encontrar las rutas más cortas entre nodos en un grafo ponderado. Este grafo puede representar, por ejemplo, redes de carreteras, conexiones de vuelos o cualquier otro sistema de relaciones con costos asociados.

¿Para qué sirve?

El algoritmo de Dijkstra encuentra la ruta más corta desde un nodo de origen hacia todos los demás nodos en el grafo. Además, puede utilizarse para hallar la ruta más corta hacia un nodo de destino específico. Su aplicación es amplia y diversa:
Redes de Transporte: En la planificación de rutas para vehículos, trenes o aviones, Dijkstra ayuda a determinar la ruta más eficiente en términos de tiempo o distancia.
Redes de Comunicación: En protocolos de enrutamiento de redes de datos, como OSPF (Open Shortest Path First) e IS-IS (Intermediate System to Intermediate System), Dijkstra se utiliza para encontrar las rutas más cortas entre routers.
Logística y Distribución: En la gestión de flotas de transporte, Dijkstra optimiza las entregas y minimiza los costos.
Diseño de Circuitos Electrónicos: En el diseño de chips y placas electrónicas, Dijkstra ayuda a encontrar las conexiones más cortas entre componentes.

¿Cómo se implementa en el mundo?

El algoritmo de Dijkstra se implementa en sistemas de navegación GPS, en la optimización de rutas de entrega de paquetes, en la planificación de vuelos y en la administración de redes de comunicación. Su uso es fundamental en la infraestructura tecnológica moderna.
¿Cómo lo implementarías en tu vida?
En mi vida cotidiana, podría aplicar el algoritmo de Dijkstra para planificar mis desplazamientos diarios. Por ejemplo, al elegir la ruta más rápida para ir al trabajo o al calcular el tiempo estimado de llegada a una cita médica.

¿Cómo lo implementarías en el diseño de videojuegos?

En el diseño de videojuegos, Dijkstra es útil para crear rutas óptimas para personajes no jugadores (NPCs), como en juegos de estrategia o aventuras. También se puede emplear para generar laberintos o mapas con caminos interesantes y desafiantes.

Algoritmo:

```
using System;
using System.Collections.Generic;

class Program
{
    static void Main()
    {
        // Crear un grafo de ejemplo (representado como una matriz de
adyacencia)
        int[,] graph = {
            { 0, 4, 2, 0, 0 },
            { 4, 0, 1, 5, 0 },
            { 2, 1, 0, 8, 10 },
            { 0, 5, 8, 0, 2 },
            { 0, 0, 10, 2, 0 }
        };

        int startVertex = 0; // Nodo de inicio (puedes cambiarlo según tu
necesidad)

        Dijkstra(graph, startVertex);
    }

    static void Dijkstra(int[,] graph, int start)
    {
        int n = graph.GetLength(0);
        int[] distance = new int[n];
        bool[] visited = new bool[n];

        for (int i = 0; i < n; i++)
        {
            distance[i] = int.MaxValue;
            visited[i] = false;
        }

        distance[start] = 0;

        for (int count = 0; count < n - 1; count++)
        {
            int u = GetMinDistanceVertex(distance, visited);
            visited[u] = true;

            Console.WriteLine($"Paso {count + 1}: Visitando nodo {u}");
```

```
            for (int v = 0; v < n; v++)
            {
                if (!visited[v] && graph[u, v] != 0 && distance[u] !=
int.MaxValue &&
                    distance[u] + graph[u, v] < distance[v])
                {
                    distance[v] = distance[u] + graph[u, v];
                }
            }
        }

        Console.WriteLine("\nDistancias mínimas desde el nodo de
inicio:");
        for (int i = 0; i < n; i++)
        {
            Console.WriteLine($"Nodo {i}: {distance[i]}");
        }
    }

    static int GetMinDistanceVertex(int[] distance, bool[] visited)
    {
        int minDist = int.MaxValue;
        int minVertex = -1;

        for (int v = 0; v < distance.Length; v++)
        {
            if (!visited[v] && distance[v] < minDist)
            {
                minDist = distance[v];
                minVertex = v;
            }
        }

        return minVertex;
    }
}
```

Este código simula el algoritmo de Dijkstra para encontrar las distancias mínimas desde un nodo de inicio en un grafo. Cada paso se muestra en la consola. Puedes modificar la matriz graph y el nodo de inicio según tus necesidades.

Fuentes Bibliográficas

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. Numerische Mathematik, 1(1), 269-271.
Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to Algorithms (3rd ed.). MIT Press.

Este ensayo se basa en la obra de Edsger W. Dijkstra y en el libro “Introduction to Algorithms” de Cormen, Leiserson, Rivest y Stein.
1: Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. Numerische Mathematik, 1(1), 269-271. 2: Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to Algorithms (3rd ed.). MIT Press.