CETI.

Actividad: Algoritmo de Kruskal.

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 Kruskal?

El algoritmo de Kruskal, propuesto por el matemático estadounidense Joseph Kruskal en 1956, es una técnica para encontrar un árbol de expansión mínimo en un grafo ponderado. Este árbol recubridor mínimo conecta todos los vértices del grafo con el menor costo posible.

¿Para qué sirve?

El algoritmo de Kruskal se utiliza para resolver problemas de optimización en redes y sistemas de comunicación. Su objetivo principal es encontrar una red de conexiones que minimice el costo total, como en los siguientes casos:
Redes de Comunicación: En la planificación de redes de datos, como la construcción de redes de fibra óptica o la conexión de routers, Kruskal ayuda a minimizar los costos de instalación y mantenimiento.
Diseño de Circuitos Electrónicos: En la fabricación de chips y placas electrónicas, Kruskal se aplica para conectar componentes con la menor longitud de cable posible.
Planificación de Rutas: En logística y transporte, el algoritmo de Kruskal optimiza las rutas de entrega y minimiza los gastos de combustible y tiempo.

¿Cómo se implementa en el mundo?

El algoritmo de Kruskal se implementa en sistemas de navegación GPS, en la planificación de redes de comunicación, en la logística de distribución y en la optimización de recursos en general.

¿Cómo lo implementarías en tu vida?

En mi vida cotidiana, podría aplicar el algoritmo de Kruskal para planificar mis desplazamientos diarios. Por ejemplo, al elegir la ruta más corta para ir al trabajo o al calcular el tiempo estimado de llegada a una cita.

¿Cómo lo implementarías en el diseño de videojuegos?

En el diseño de videojuegos, Kruskal es útil para crear mapas con caminos interesantes y desafiantes. Puede utilizarse para generar laberintos o para diseñar niveles en los que los jugadores deben explorar áreas conectadas de manera eficiente.

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 }
        };

        Kruskal(graph);
    }

    static void Kruskal(int[,] graph)
    {
        int n = graph.GetLength(0);
        List<Edge> edges = new List<Edge>();

        // Crear una lista de aristas con sus pesos
        for (int i = 0; i < n; i++)
        {
            for (int j = i + 1; j < n; j++)
            {
                if (graph[i, j] != 0)
                {
                    edges.Add(new Edge(i, j, graph[i, j]));
                }
            }
        }

        // Ordenar las aristas por peso
        edges.Sort((a, b) => a.Weight.CompareTo(b.Weight));

        int[] parent = new int[n];
        for (int i = 0; i < n; i++)
        {
            parent[i] = i;
```

```
        }

        List<Edge> minimumSpanningTree = new List<Edge>();

        foreach (var edge in edges)
        {
            int root1 = Find(parent, edge.Source);
            int root2 = Find(parent, edge.Destination);

            if (root1 != root2)
            {
                minimumSpanningTree.Add(edge);
                Union(parent, root1, root2);
            }
        }

        Console.WriteLine("Árbol de Mínimo Coste (Kruskal):");
        foreach (var edge in minimumSpanningTree)
        {
            Console.WriteLine($"Arista ({edge.Source},
{edge.Destination}) con peso {edge.Weight}");
        }
    }

    static int Find(int[] parent, int vertex)
    {
        if (parent[vertex] != vertex)
        {
            parent[vertex] = Find(parent, parent[vertex]);
        }
        return parent[vertex];
    }

    static void Union(int[] parent, int root1, int root2)
    {
        parent[root2] = root1;
    }
}

class Edge
{
    public int Source { get; }
    public int Destination { get; }
    public int Weight { get; }
```

```
    public Edge(int source, int destination, int weight)
    {
        Source = source;
        Destination = destination;
        Weight = weight;
    }
}
```

Explicación

- Se crea una lista de aristas a partir de la matriz de adyacencia.
- Las aristas se ordenan por peso.
- Se utiliza la estructura de datos de conjunto disjunto para evitar ciclos en el árbol.
- Se construye el árbol de expansión mínimo (MST) mediante Kruskal.

Este algoritmo es útil en redes de comunicación, planificación de rutas y diseño de circuitos electrónicos.

Fuentes Bibliográficas

Kruskal, J. B. (1956). On the shortest spanning subtree of a graph and the traveling salesman problem. Proceedings of the American Mathematical Society, 7(1), 48-50.
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 Joseph Kruskal y en el libro “Introduction to Algorithms” de Cormen, Leiserson, Rivest y Stein.
1: Kruskal, J. B. (1956). On the shortest spanning subtree of a graph and the traveling salesman problem. Proceedings of the American Mathematical Society, 7(1), 48-50. 2: Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to Algorithms (3rd ed.). MIT Press.