CENTRO DE ENSEÑASA TÉCNICA INDUSTRIAL

INTELIGENCIA ARTIFICIAL

PRÁCTICA 5 SIMULADOR ÁRBOL DE MÁXIMO Y MÍNIMO COSTE KRUSKAL.

PRECIADO MARTINEZ BRUNO AUGUSTO

N: 21110311

6 E

Parte Teórica

¿Qué ES? Un simulador de "Árbol de Máximo y Mínimo Coste" en el contexto del algoritmo de Kruskal se refiere a un programa o aplicación que permite a los usuarios ingresar un grafo ponderado (un conjunto de vértices y aristas, donde cada arista tiene un peso) y luego calcula y muestra tanto el Árbol de Expansión Mínima (Minimum Spanning Tree, MST) como el Árbol de Máximo Coste utilizando el algoritmo de Kruskal.

El Árbol de Expansión Mínima (MST) es un subconjunto del grafo original que conecta todos los vértices sin formar ciclos y tiene el peso total mínimo. El Árbol de Máximo Coste es lo opuesto, es un subconjunto que conecta todos los vértices pero con el peso total máximo. Ambos árboles son útiles en diferentes contextos y problemas de optimización.

¿Para qué sirve? Educación y Aprendizaje: Estos simuladores son utilizados en entornos educativos para enseñar y aprender sobre algoritmos de árboles de expansión, en particular, el algoritmo de Kruskal

Depuración y Entendimiento de Código: Los desarrolladores de software también pueden usar estos simuladores para depurar y entender cómo funcionan los algoritmos de Kruskal en su propio código. Pueden ingresar datos específicos y ver cómo se comporta su implementación del algoritmo.

Investigación y Análisis: Los investigadores y analistas pueden usar simuladores para explorar problemas de grafos y redes y evaluar diferentes enfoques para resolverlos. Esto puede ser útil en campos como logística, transporte, telecomunicaciones y más.

¿Cómo se implementa en el mundo? Redes de Comunicación: En las redes de telecomunicaciones y en la construcción de redes de comunicación, el algoritmo de Kruskal se utiliza para conectar torres de señal, estaciones de repetición y otros puntos de acceso de manera que se minimice la longitud total de cable o la infraestructura necesaria.

Logística y Transporte: En la planificación de rutas de transporte, como la entrega de mercancías o la optimización de rutas de camiones, se utiliza Kruskal para encontrar rutas eficientes y económicas que conecten diferentes ubicaciones.

Distribución de Energía: En la distribución de energía eléctrica, el algoritmo de Kruskal se emplea para conectar subestaciones y líneas de transmisión de manera que se minimicen las pérdidas de energía y se optimice la distribución.

Construcción de Carreteras y Rutas: En ingeniería civil y planificación de transporte, se emplea el MST de Prim para determinar la disposición de carreteras, rutas de ferrocarril y otros medios de transporte para conectar ciudades o ubicaciones de manera eficiente.

¿Cómo lo implementarías en tu vida? Al momento de diseñar un circuito que sea de dificultad algo serio para tener un ordenamiento más óptico y una mejor optimización de la energía en toso el proyecto

¿Cómo lo implementarías en tu trabajo o tu trabajo de ensueño cuanto tenga que hacer un manejo muy delicado de información entre barias partes para minimizar el trafico de información por ejemplo recibiendo datos en tiempo real de varias cámaras que estén entrenando para poder hacer que manejen por si solas

Código

```python
class Edge:
    def __init__(self, src, dest, weight):
        self.src = src
        self.dest = dest
        self.weight = weight

def find_parent(parent, vertex):
    if parent[vertex] == -1:
        return vertex
    return find_parent(parent, parent[vertex])

def kruskal_min_max_spanning_tree(edges, num_vertices, min_cost=True):
    edges.sort(key=lambda edge: edge.weight, reverse=min_cost)

    parent = [-1] * num_vertices
    min_max_tree = []

    for edge in edges:
        src_parent = find_parent(parent, edge.src)
        dest_parent = find_parent(parent, edge.dest)

        if src_parent != dest_parent:
            min_max_tree.append(edge)
            parent[src_parent] = dest_parent

    return min_max_tree

def display_tree(tree, min_tree=True):
    if min_tree:
        print("Árbol de Expansión Mínima:")
    else:
        print("Árbol de Máximo Coste:")
    total_weight = 0
    for edge in tree:
        print(f"{edge.src} - {edge.dest} : {edge.weight}")
        total_weight += edge.weight
    print(f"Peso Total del Árbol: {total_weight}")

def main():
    num_vertices = int(input("Ingrese el número de vértices: "))
    num_edges = int(input("Ingrese el número de aristas: "))

    edges = []
    for _ in range(num_edges):
```

```
    parent = [-1] * num_vertices
    min_max_tree = []

    for edge in edges:
        src_parent = find_parent(parent, edge.src)
        dest_parent = find_parent(parent, edge.dest)

        if src_parent != dest_parent:
            min_max_tree.append(edge)
            parent[src_parent] = dest_parent

    return min_max_tree

def display_tree(tree, min_tree=True):
    if min_tree:
        print("Árbol de Expansión Mínima:")
    else:
        print("Árbol de Máximo Coste:")
    total_weight = 0
    for edge in tree:
        print(f"{edge.src} - {edge.dest} : {edge.weight}")
        total_weight += edge.weight
    print(f"Peso Total del Árbol: {total_weight}")

def main():
    num_vertices = int(input("Ingrese el número de vértices: "))
    num_edges = int(input("Ingrese el número de aristas: "))

    edges = []
    for _ in range(num_edges):
        src, dest, weight = map(int, input("Ingrese (src dest peso): ").split())
        edges.append(Edge(src, dest, weight))

    min_tree = kruskal_min_max_spanning_tree(edges, num_vertices, min_cost=True)
    max_tree = kruskal_min_max_spanning_tree(edges, num_vertices, min_cost=False

    display_tree(min_tree, min_tree=True)
    display_tree(max_tree, min_tree=False)

if __name__ == "__main__":
    main()
```

Resultado

```
Ingrese el número de vértices: 5
Ingrese el número de aristas: 7
Ingrese (src dest peso):  0 1 2
Ingrese (src dest peso):  0 2 1
Ingrese (src dest peso):  1 2 3
Ingrese (src dest peso):  1 3 5
Ingrese (src dest peso):  1 4 4
Ingrese (src dest peso):  2 4 7
Ingrese (src dest peso):  3 4 9
Árbol de Expansión Mínima:
3 - 4 : 9
2 - 4 : 7
1 - 3 : 5
0 - 1 : 2
Peso Total del Árbol: 23
Árbol de Máximo Coste:
0 - 2 : 1
0 - 1 : 2
1 - 4 : 4
1 - 3 : 5
Peso Total del Árbol: 12
```