ceti

CENTRO DE ENSEÑANZA TÉCNICA INDUSTRIAL

INTELIGENCIA ARTIFICIAL

Saul Armando Cuenca Martínez
21110324

"ÁRBOL PARCIAL MÍNIMO DE PRIM"

Parte Teórica

¿Qué es el Árbol Parcial Mínimo de Prim ?

Es un algoritmo utilizado en teoría de grafos para encontrar un árbol de expansión mínima en un grafo no dirigido y ponderado. En un grafo ponderado, el peso de un subgrafo es la suma de los pesos de cada una de las aristas en el subgrafo. Así, un árbol generador mínimo (MST por sus siglas en inglés) es un subgrafo ponderado no dirigido que es un árbol generador con peso mínimo.

¿Para qué sirve el Árbol Parcial Mínimo de Prim?

En otras palabras, busca un subconjunto de aristas que conectan todos los nodos del grafo y tengan el menor peso total posible. Este subconjunto forma un árbol que abarca todos los nodos y, al mismo tiempo, minimiza la suma de los pesos de las aristas.

El algoritmo de Prim se llama "parcial mínimo" porque construye un árbol de expansión mínimo paso a paso, añadiendo una arista a la vez. Comienza con un nodo inicial (que puede ser cualquier nodo del grafo) y, en cada paso, agrega la arista más corta que conecta un nodo en el árbol parcial con un nodo fuera del árbol parcial. El proceso continúa hasta que todos los nodos estén incluidos en el árbol.

¿Cómo se implementa el Árbol Parcial Mínimo de Prim en el mundo?

Redes de Comunicación: En redes de comunicación, como las redes de telecomunicaciones y la infraestructura de Internet, el algoritmo de Prim se utiliza para diseñar una red eficiente que conecta nodos (como centrales

telefónicas o servidores) con el costo mínimo. Esto ayuda a minimizar los costos de comunicación y asegura una conectividad confiable.

Diseño de Circuitos: En ingeniería eléctrica y electrónica, se utiliza el Árbol Parcial Mínimo de Prim para diseñar circuitos impresos y conexiones entre componentes electrónicos, minimizando la longitud de las pistas y, por lo tanto, reduciendo el tiempo de viaje de las señales y los costos de producción.

Logística y Transporte: En la planificación de rutas y logística, el algoritmo de Prim se aplica para encontrar rutas óptimas para camiones de reparto, servicios de entrega, rutas de transporte público y cualquier otro sistema de transporte, minimizando distancias o costos.

Diseño de Mapas: En la creación de mapas y cartografía, se utiliza el Árbol Parcial Mínimo de Prim para la generación de mapas eficientes y conectividad de carreteras o rutas de navegación, lo que facilita la planificación de viajes y la navegación.

Redes de Distribución de Energía: En el campo de la distribución de energía eléctrica, se emplea el algoritmo de Prim para diseñar una red de distribución eficiente, minimizando las pérdidas de energía y garantizando una distribución confiable.

Optimización de Costos: En general, el Árbol Parcial Mínimo de Prim se utiliza en situaciones donde es necesario conectar nodos o ubicaciones con un costo asociado a las conexiones. En tales casos, el algoritmo ayuda a minimizar los costos totales mientras se asegura la conectividad de todos los puntos.

¿Cómo lo implementarías el Árbol Parcial Mínimo de Prim en tu vida?

Toma de Decisiones Financieras: Cuando tomas decisiones financieras, como inversiones, puedes aplicar la idea de buscar oportunidades que ofrezcan el máximo rendimiento con el menor riesgo o costo.

Gestión del Tiempo: La gestión del tiempo puede beneficiarse de la aplicación de principios de optimización. Priorizar tareas y actividades importantes puede ayudarte a ser más eficiente y productivo.

Planificación de Carrera: En tu carrera, puedes aplicar la lógica de encontrar caminos de desarrollo que te lleven a tus metas profesionales minimizando obstáculos y costos innecesarios.

Organización del Hogar: Al organizar y diseñar tu espacio en casa, puedes aplicar conceptos de eficiencia para minimizar desplazamientos y ahorrar tiempo y energía.

Búsqueda de Empleo: Cuando buscas empleo, puedes aplicar el concepto de encontrar oportunidades que se adapten a tus objetivos profesionales y minimizar costos, como tiempo y gastos de desplazamiento.

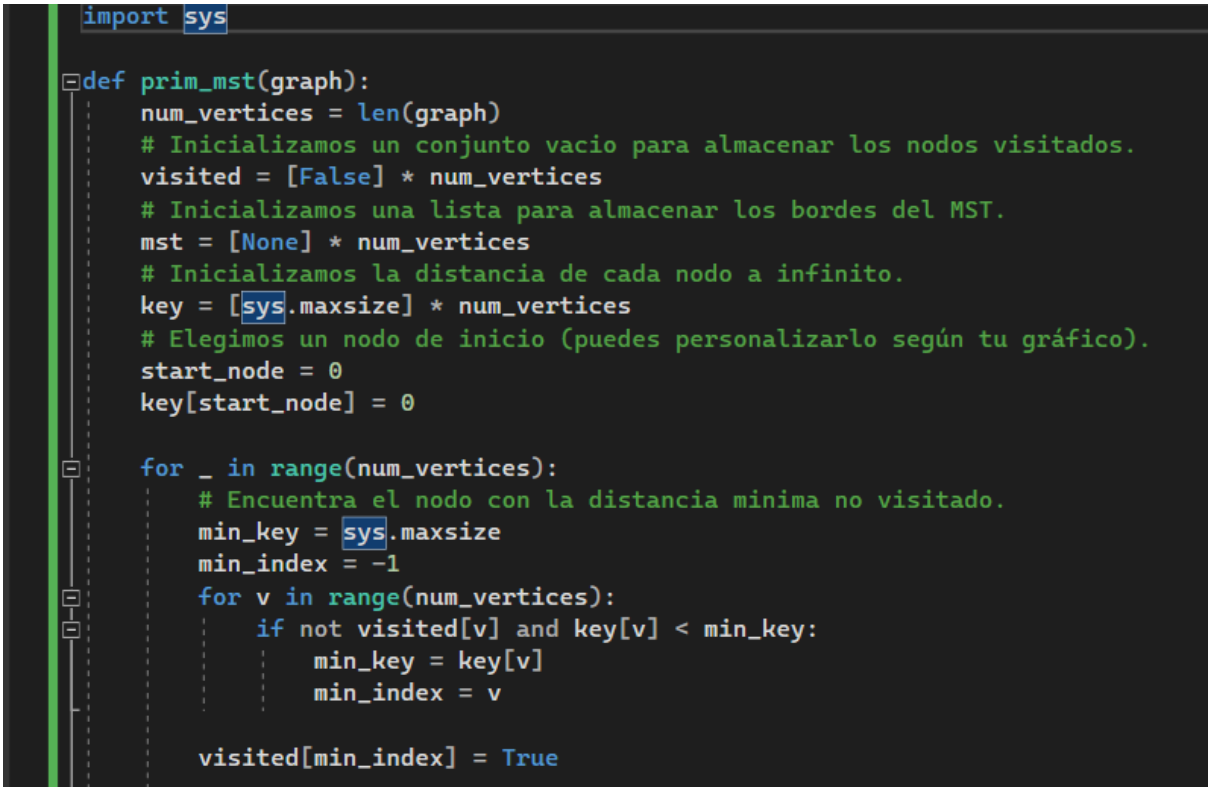
Gestión de Proyectos Personales: En proyectos personales, como la planificación de un evento, puedes aplicar la lógica de minimizar costos y tiempos al seleccionar proveedores, ubicaciones y actividades.

Organización de Actividades Diarias: En tu rutina diaria, puedes aplicar la lógica de priorizar tareas basadas en su importancia y urgencia, lo que te permite optimizar el tiempo y los recursos disponibles.

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

De igual forma que la práctica anterior, esto nos ayudará personalmente y laboralmente, me gusto que nuestro profesor nos mostrará que siempre hay alguna forma mejor de hacer las cosas como lo es utilizando esto que sacando información siempre llegaremos al error o a nuestros objetivos. Principalmente siempre teniendo en cuenta nuestros objetivos e implementarlo a nuestra vida.

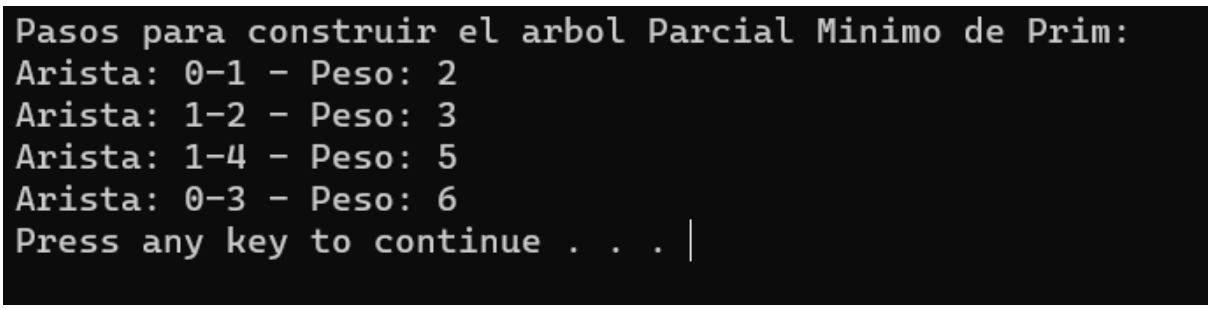
Simulador en un programa



```
import sys

def prim_mst(graph):
    num_vertices = len(graph)
    # Inicializamos un conjunto vacio para almacenar los nodos visitados.
    visited = [False] * num_vertices
    # Inicializamos una lista para almacenar los bordes del MST.
    mst = [None] * num_vertices
    # Inicializamos la distancia de cada nodo a infinito.
    key = [sys.maxsize] * num_vertices
    # Elegimos un nodo de inicio (puedes personalizarlo según tu gráfico).
    start_node = 0
    key[start_node] = 0

    for _ in range(num_vertices):
        # Encuentra el nodo con la distancia minima no visitado.
        min_key = sys.maxsize
        min_index = -1
        for v in range(num_vertices):
            if not visited[v] and key[v] < min_key:
                min_key = key[v]
                min_index = v

        visited[min_index] = True
```

```
        # Agregamos el borde a la lista del MST.
        if mst[min_index] is not None:
            print(f"Arista: {mst[min_index]} - Peso: {key[min_index]}")

        for v in range(num_vertices):
            if (
                graph[min_index][v] != 0
                and not visited[v]
                and graph[min_index][v] < key[v]
            ):
                key[v] = graph[min_index][v]
                mst[v] = f"{min_index}-{v}"

# Ejemplo de grafo ponderado en forma de matriz de adyacencia.
graph = [
    [0, 2, 0, 6, 0],
    [2, 0, 3, 8, 5],
    [0, 3, 0, 0, 7],
    [6, 8, 0, 0, 9],
    [0, 5, 7, 9, 0],
]

print("Pasos para construir el arbol Parcial Minimo de Prim:")
prim_mst(graph)
```


```
        # Agregamos el borde a la lista del MST.
        if mst[min_index] is not None:
            print(f"Arista: {mst[min_index]} - Peso: {key[min_index]}")

        for v in range(num_vertices):
            if (
                graph[min_index][v] != 0
                and not visited[v]
                and graph[min_index][v] < key[v]
            ):
                key[v] = graph[min_index][v]
                mst[v] = f"{min_index}-{v}"

# Ejemplo de grafo ponderado en forma de matriz de adyacencia.
graph = [
    [0, 2, 0, 6, 0],
    [2, 0, 3, 8, 5],
    [0, 3, 0, 0, 7],
    [6, 8, 0, 0, 9],
    [0, 5, 7, 9, 0],
]

print("Pasos para construir el arbol Parcial Minimo de Prim:")
prim_mst(graph)
```



https://github.com/SaulCuenca/IA-P2/tree/main/PR4

References

El algoritmo de Prim. (2020, November 14). Fundamental Nerve. Retrieved October 26, 2023, from https://alejandrosanchezyali.blogspot.com/2020/11/al-algoritmo-de-prim.html

Martínez, L. I. (n.d.). *Árboles de peso mínimo: algoritmos de Prim y Kruskal — Matemáticas Discretas para Ciencia de Datos*. Matemáticas Discretas para Ciencia de Datos. Retrieved October 26, 2023, from https://madi.nekomath.com/P5/ArbolPesoMin.html