Centro de Enseñanza Técnica Industrial

Inteligencia Artificial

6°E1

Victor Abel Moreno Magaña 21110345

¿Qué es el árbol de Máximo y Mínimo coste Kruskal?

El árbol de Máximo y Mínimo coste Kruskal es un algoritmo para encontrar un árbol recubridor máximo o mínimo en un grafo conexo y ponderado. Es decir, busca un subconjunto de aristas que, formando un árbol, incluyen todos los vértices y donde el valor de la suma de todas las aristas del árbol es el máximo o el mínimo, respectivamente.

¿Para qué sirve el árbol de Máximo y Mínimo coste Kruskal?

El árbol de Máximo y Mínimo coste Kruskal se utiliza en una amplia gama de aplicaciones, que incluyen:

- ***Redes***: El árbol de Máximo y Mínimo coste Kruskal se utiliza para encontrar el camino más largo o más corto entre dos puntos en una red, como una red de carreteras o una red de telecomunicaciones.
- ***Distribución***: El árbol de Máximo y Mínimo coste Kruskal se utiliza para encontrar la ruta más eficiente para distribuir bienes o servicios.
- ***Construcción***: El árbol de Máximo y Mínimo coste Kruskal se utiliza para encontrar la ruta más económica o más costosa para construir una red de infraestructura, como una red de carreteras o una red de tuberías.

¿Cómo se implementa el árbol de Máximo y Mínimo coste Kruskal en el mundo?

El árbol de Máximo y Mínimo coste Kruskal se implementa en una amplia gama de productos y servicios, que incluyen:

- ***Navegadores GPS***: Los navegadores GPS utilizan el árbol de Máximo y Mínimo coste Kruskal para encontrar el camino más largo o más corto entre dos puntos.
- ***Sistemas de entrega***: Los sistemas de entrega utilizan el árbol de Máximo y Mínimo coste Kruskal para encontrar la ruta más eficiente para entregar productos.
- ***Software de diseño de redes***: El software de diseño de redes utiliza el árbol de Máximo y Mínimo coste Kruskal para encontrar la ruta más económica o más costosa para construir una red de infraestructura.

¿Cómo implementaría el árbol de Máximo y Mínimo coste Kruskal en mi vida?

Podría implementar el árbol de Máximo y Mínimo coste Kruskal en mi vida para encontrar el camino más largo o más corto entre dos puntos en mi ciudad. Por ejemplo, podría usar el algoritmo para encontrar el camino más rápido para ir del trabajo a casa o para encontrar el camino más largo para visitar a un amigo.

¿Cómo lo implementaría en mi trabajo o en mi trabajo de ensueño?

En mi trabajo actual, podría utilizar el árbol de Máximo y Mínimo coste Kruskal para optimizar las rutas de entrega de los productos. Por ejemplo, podría usar el algoritmo para encontrar la ruta más eficiente para entregar un paquete a un cliente.

En mi trabajo de ensueño, podría utilizar el árbol de Máximo y Mínimo coste Kruskal para ayudar a las personas a encontrar el camino más largo o más corto para acceder a servicios esenciales, como atención médica o educación. Por ejemplo, podría usar el algoritmo para encontrar el camino más rápido para llegar a un hospital o para encontrar el camino más largo para llegar a una escuela.

Ejemplo de implementación en la vida real

Imagine que vive en una ciudad con un sistema de transporte público. El sistema de transporte público está formado por una red de autobuses, trenes y tranvías.

Para encontrar el camino más largo o más corto entre dos puntos en la ciudad, podría utilizar el árbol de Máximo y Mínimo coste Kruskal para encontrar un árbol recubridor máximo o mínimo de la red de transporte público. Este árbol recubridor máximo o mínimo representaría la ruta más eficiente para viajar entre cualquier dos puntos de la ciudad.

El árbol de Máximo y Mínimo coste Kruskal podría implementarse utilizando un algoritmo voraz. El algoritmo comenzaría con un conjunto vacío que consta de un solo vértice, que representa el punto de partida. Luego, el algoritmo agregaría una arista a la vez al árbol, siempre que la arista agregada no creará un ciclo en el árbol. La arista agregada sería la arista que tiene el mayor o menor costo, respectivamente.

El algoritmo continuaría agregando aristas al árbol hasta que todos los vértices de la red de transporte público estuvieran incluidos en el árbol. El árbol resultante sería el árbol recubridor máximo o mínimo de la red de transporte público.

Este árbol recubridor máximo o mínimo podría utilizarse para encontrar el camino más largo o más corto entre cualquier dos puntos de la ciudad. Por ejemplo, si quisiera encontrar el camino más largo para ir del trabajo a casa, podría utilizar el árbol recubridor máximo para encontrar la ruta que pasa por la mayor cantidad de autobuses, trenes y tranvías

Código

```python
import sys

def prim_mst(graph):
    num_vertices = len(graph)
    # Inicializamos un conjunto vacío 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 mínima 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
```

```python
        # 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)
```

```
C:\Users\victo\python.exe
Pasos para construir el Arbol de Minimo Coste de Kruskal:
Arista: (0-2) - Peso: 0
Arista: (0-4) - Peso: 0
Arista: (2-3) - Peso: 0
Arista: (0-1) - Peso: 2
```

https://github.com/VictorM8464/6-E1_21110345_PR4.git?authuser=1

Bibliografía

Adrianblade. (s. f.). GitHub - adrianblade/java-algorithm. GitHub. https://github.com/adrianblade/java-algorithm

colaboradores de Wikipedia. (2020). Algoritmo de Kruskal. Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Algoritmo_de_Kruskal#:~:text=El%20algoritmo%20de%20Kruskal%20es,del%20%C3%A1rbol%20es%20el%20m%C3%ADnimo