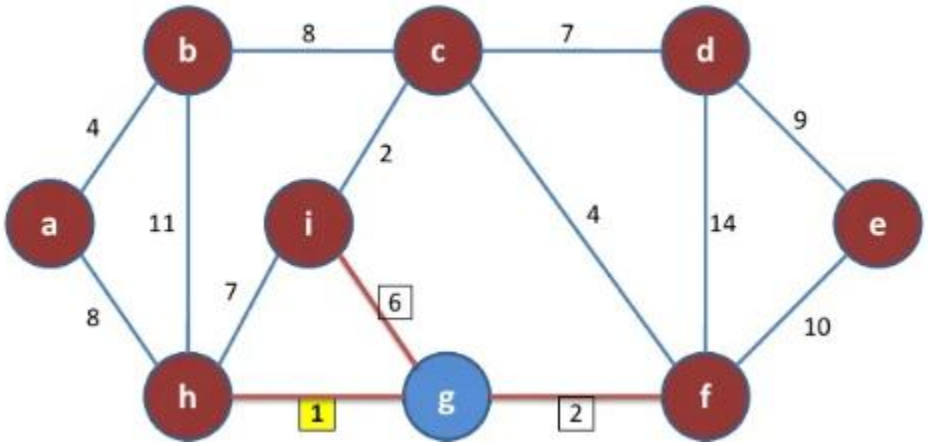
11-6-2025

Árbol Parcial mínimo de Prim

Práctica 4



Alumno: Adiel Hidalgo Ipiña

Grupo y registro: 6E2, 22310213

Carrera: Ingeniería en Mecatrónica

ENLACE DE GITHUB:

https://github.com/AdielHidalgo/Practica4_-rbolParcialm-nimodePrim_22310213.git

¿Qué es el Árbol Parcial mínimo de Prim?

El **algoritmo de Prim** es un método **codicioso (greedy)** que pertenece a la teoría de grafos. Su función es encontrar un **Árbol de Expansión Mínima (AEM)** dentro de un grafo que sea no dirigido, conexo y ponderado.

En términos más simples:

- **Grafo:** Una colección de puntos (llamados **vértices**) conectados por líneas (llamadas **aristas**).
- **Ponderado:** Cada conexión (arista) tiene un valor o costo asociado (su "peso").
- **Árbol de Expansión:** Es un subconjunto de las aristas que conecta todos los vértices sin formar ciclos.
- **Mínima:** El algoritmo de Prim garantiza que, de todos los posibles árboles de expansión, el que encuentra es aquel cuya suma de pesos es la menor posible.

El algoritmo funciona comenzando desde un vértice cualquiera y, paso a paso, agrega la arista más "barata" que conecte un vértice ya incluido en el árbol con uno que todavía está fuera, hasta que todos los vértices están conectados.

¿Para qué sirve?

El propósito fundamental del algoritmo de Prim es resolver problemas de **optimización de redes**. Sirve para encontrar la manera más económica o eficiente de conectar un conjunto de puntos, garantizando que todos estén enlazados. Su utilidad radica en minimizar costos, distancias, tiempo o cualquier otra métrica que pueda ser representada como el peso de una arista.

¿Cómo se implementa en el mundo?

El algoritmo de Prim tiene aplicaciones prácticas y cruciales en diversas industrias:

- **Redes de Telecomunicaciones:** Para diseñar el tendido de cables de fibra óptica o coaxiales que conecten ciudades, barrios o edificios utilizando la mínima cantidad de cable, lo que reduce drásticamente los costos de material e instalación.
- **Redes de Servicios Públicos:** En el diseño de redes eléctricas, tuberías de agua potable o gasoductos. El algoritmo ayuda a planificar la ruta que conecte todos los puntos de servicio (hogares, fábricas) con el mínimo costo de infraestructura.
- **Diseño de Circuitos Integrados (Chips):** Para conectar los diferentes componentes en una placa de circuito impreso (PCB) o dentro de un microchip (diseño VLSI)

utilizando la menor longitud de conductor posible. Esto ahorra espacio y reduce la latencia de la señal.

- **Logística y Transporte:** Aunque otros algoritmos son más comunes para rutas punto a punto, Prim puede usarse para diseñar redes de rutas base (por ejemplo, rutas de autobuses o de aerolíneas) que aseguren que todas las paradas o aeropuertos importantes estén conectados con una red de longitud total mínima.

¿Cómo lo implementarías en tu vida?

Podrías aplicar la lógica de Prim para optimizar tareas cotidianas:

- **Planificar una red Wi-Fi Mesh en casa:** Si tienes varios repetidores de señal (vértices) y quieres conectarlos entre sí y al router principal. El "peso" podría ser la distancia o la pérdida de señal entre ellos. Prim te ayudaría a diseñar una red estable con las conexiones más fuertes y directas posibles.
- **Organizar un recorrido para recoger amigos:** Si tienes que pasar por varias casas (vértices) para recoger a tus amigos para un viaje. El "peso" es el tiempo o la distancia de viaje entre cada casa. Prim te ayudaría a encontrar el conjunto de trayectos más corto que conecta a todos, minimizando la gasolina y el tiempo total del recorrido de recolección.

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

(Técnico de Análisis de Fallas en Flex para el proyecto de Cisco)

En tu rol, tu objetivo es encontrar el origen de una falla de la manera más rápida y eficiente posible. El algoritmo de Prim puede ayudarte a optimizar el **proceso de diagnóstico**.

Imagina una placa de circuito impreso (PCB) compleja de un equipo de Cisco que ha fallado.

- Los **vértices** del grafo serían los componentes clave o los puntos de prueba (test points) en la placa: el procesador, los módulos de memoria, los chips de red, los capacitores importantes, etc.
- El **peso** de las **aristas** entre dos puntos de prueba representaría el "costo" de realizar una prueba que los involucre. Este costo podría ser el **tiempo** que toma la prueba, la **complejidad** de la configuración del equipo de medición, o incluso una probabilidad inversa de encontrar la falla en esa sección.

Al aplicar el algoritmo de Prim, no estarías conectando los componentes físicamente, sino que estarías construyendo un **"árbol de diagnóstico de costo mínimo"**. El resultado sería una secuencia optimizada de pruebas que te garantizaría cubrir todos los puntos críticos con el menor esfuerzo o tiempo total. En lugar de seguir un guion de pruebas lineal, podrías seguir una ruta que, estadísticamente, te llevaría a la causa raíz de la falla de la forma más eficiente.