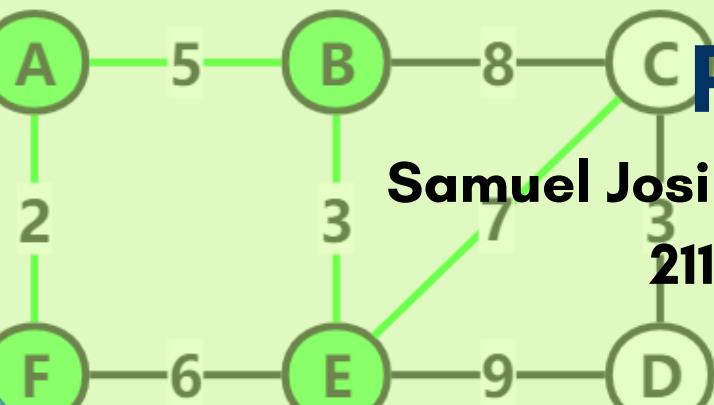
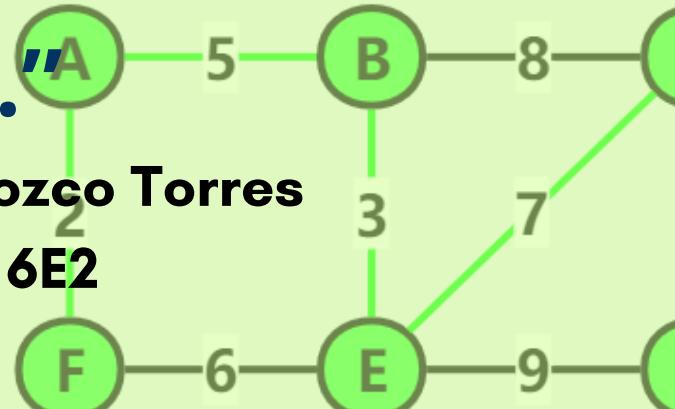


## Práctica 4

# “Árbol Parcial mínimo de Prim.”



Samuel Josimar Orozco Torres  
21110380 - 6E2



# Índice

<b>¿Qué es?</b>	<b>Pág. 3</b>
<b>¿Para qué sirve?</b>	<b>Pág. 5</b>
<b>¿Cómo se implementa en el mundo?</b>	<b>Pág. 6</b>
<b>¿Cómo lo implementarías en tu vida?</b>	<b>Pág. 7</b>
<b>¿Cómo lo implementarías en tu trabajo o trabajo de ensueño?</b>	<b>Pág. 8</b>
<b>Bibliografía</b>	<b>Pág. 9</b>



## ¿Qué es?

Un árbol ayuda a conectar los vértices de una gráfica. En el caso no dirigido cualquier árbol tiene una cantidad fija de aristas. Pero si asignamos pesos a las aristas, podemos comparar árboles entre sí.

El algoritmo prim fue desarrollado en 1930 por el matemático checo Vojtěch Jarník. Es un algoritmo ambicioso que encuentra un árbol de expansión mínimo para un grafo ponderado no dirigido. Esto significa que encuentra un subconjunto de los bordes que forma un árbol que incluye cada vértice, donde se minimiza el peso total de todos los bordes del árbol. El algoritmo funciona creando este árbol un vértice a la vez, desde un vértice inicial arbitrario, paso a paso agregando la conexión más barata posible desde el árbol a otro vértice.

Estos algoritmos encuentran el bosque de expansión mínimo en un grafo posiblemente desconectado; en cambio, la forma más básica del algoritmo de Prim sólo encuentra árboles de expansión mínimos en los grafos conectados. Sin embargo, al ejecutar el algoritmo de Prim por separado para cada componente conectado del grafo, se denomina bosque de expansión mínimo.

### Las principales características son:

- Su implementación es solo para grafo no dirigido.
- El proceso se realiza sólo en aristas con costos positivos.
- Cuando el grafo es conectado
  - Las aristas resultantes componen un árbol
- Cuando el grafo no está conectado,
  - Encuentra un árbol de expansión mínimo para cada componente conectado.
  - Las aristas resultantes conforman un bosque.

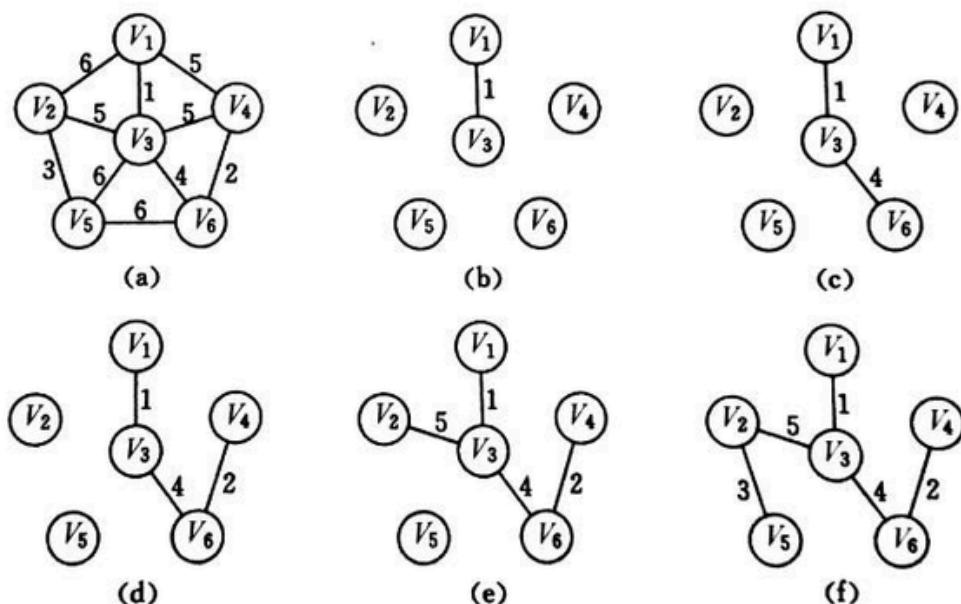


Figura 1. Ejemplo de algoritmo prim.

## Árboles de peso mínimo

Para cualquier gráfica G conexa en n vértices podemos encontrar como subgráfica a un árbol que use todos los vértices. Esta es una manera «económica» de conectar a todos los vértices de la gráfica pues usa únicamente  $n-1$  aristas. De hecho, es la menor cantidad de aristas que se pueden usar y entonces en términos de números de aristas cualquier árbol serviría.

Los vértices pueden representar casas y las aristas potenciales lugares donde se pueden colocar cables eléctricos. El peso en cada arista puede representar lo que costaría colocar un cable ahí. Nos gustaría que la red eléctrica quede toda conexa, pero gastando lo menos posible en cables.

Aplicaciones como la anterior ayudan a motivar la siguiente definición.

Definición. El peso de un árbol T que es subgráfica de una gráfica ponderada G es la suma de los pesos de las aristas de T.

A partir de aquí podemos plantear el siguiente problema.

Problema. Dada una gráfica conexa con aristas ponderadas G, encontrar un árbol que use todos los vértices y que sea de peso mínimo.

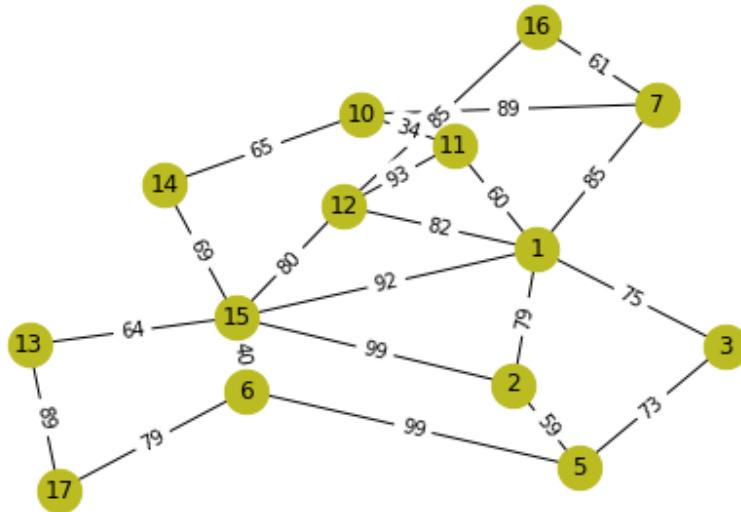


Figura 2. Ejemplo de gráfica conexa con aristas ponderadas.

## ¿Para qué sirve?

El algoritmo de Prim es un método voraz (greedy) que asegura encontrar siempre el árbol generador mínimo de un grafo ponderado y conexo.

1. Greedy approach (voraz): En cada paso, el algoritmo selecciona la arista con menor peso que conecta un nodo dentro del árbol actual con uno que no está en él.

2. Requisitos del grafo:

- Conexo: Todos los nodos deben estar alcanzables desde cualquier otro nodo.
- Ponderado: Cada arista debe tener un peso o coste asociado.

3. Eficiencia:

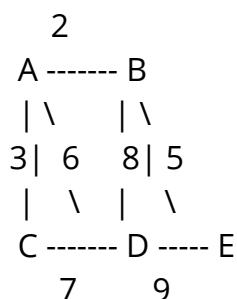
- Implementación básica:  $O(V^2)$ , útil para grafos densos (muchas aristas).
- Con cola de prioridad (Heap de Fibonacci o Min-Heap):  $O(E \log V)$ , más eficiente para grafos dispersos (menos aristas).

## Funcionamiento paso a paso

- Elegir un nodo inicial arbitrario y marcarlo como parte del árbol generado.
- Encontrar la arista de menor peso que conecta un nodo en el árbol con uno fuera de él.
- Añadir esa arista y el nuevo nodo al árbol.
- Repetir hasta que todos los nodos estén incluidos.

## Ejemplo simple:

Considera el siguiente grafo ponderado:

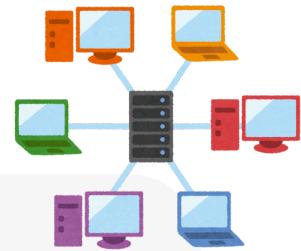


1. Empezamos en el nodo A.
2. Seleccionamos la arista más pequeña conectada a A, que es A-B(2).
3. Ahora consideramos todas las aristas conectadas a A y B. La más pequeña es A-C(3).
4. Repetimos este proceso hasta incluir todos los nodos. El árbol generador mínimo es:

Árbol Generador Mínimo: {A-B, A-C, C-D, B-E}

Peso total:  $2+3+7+5=17$

## ¿Cómo se implementa en el mundo?



### Redes de computadoras:

Diseñar infraestructuras de conexión con el menor coste posible, como cableado entre servidores.



### Energía:

Diseñar redes eléctricas optimizando los costes del tendido de cables y minimizando pérdidas.



### Construcción y transporte:

Planificación de carreteras o tuberías que conecten varias localidades con el menor coste.



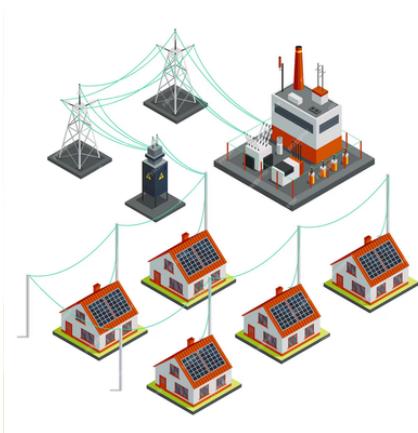
### Clustering:

Identificar grupos en grafos basados en las conexiones mínimas (por ejemplo, en análisis de redes sociales).



### Gráficos y visualización:

Determinar caminos mínimos en representaciones gráficas o en motores de videojuegos.



## ¿Cómo lo implementarías en tu vida?

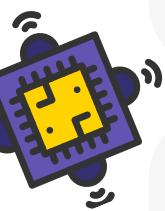
Como estudiante de ingeniería mecatrónica, puedes implementar el algoritmo de Árbol Parcial mínimo de Prim en varias situaciones para optimizar tiempo y recursos.



### Proyectos de domótica y automatización

En casas inteligentes o sistemas de automatización:

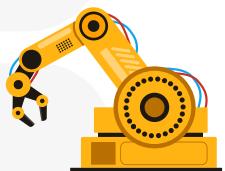
Optimizando la instalación de dispositivos: Por ejemplo, si se están instalando interruptores y sensores en diferentes puntos de una casa, el algoritmo de Prim puede ayudar a diseñar una red de conexión eficiente.



### Optimización de redes de sensores

En aplicaciones de Internet de las Cosas (IoT), el algoritmo de Prim se puede usar para:

Conectar sensores de forma eficiente: Por ejemplo, en un invernadero inteligente, conectar sensores de humedad y temperatura a un controlador minimizando el cableado.

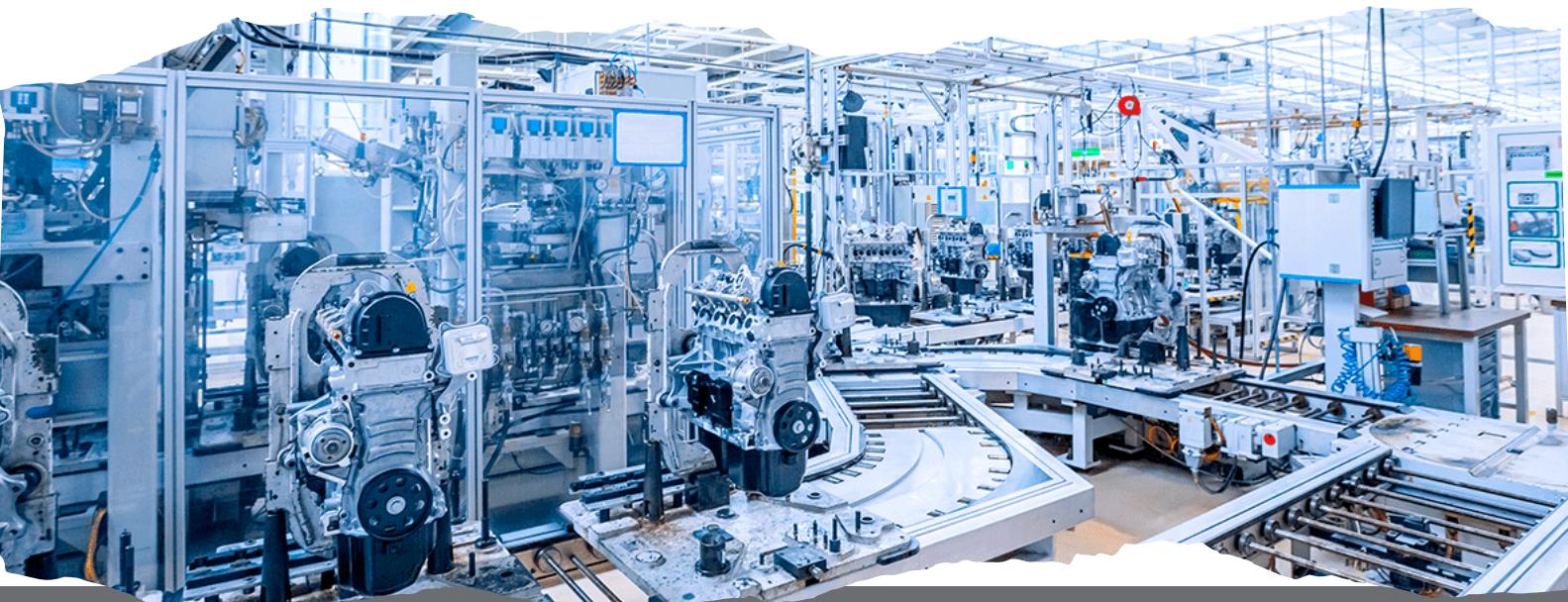


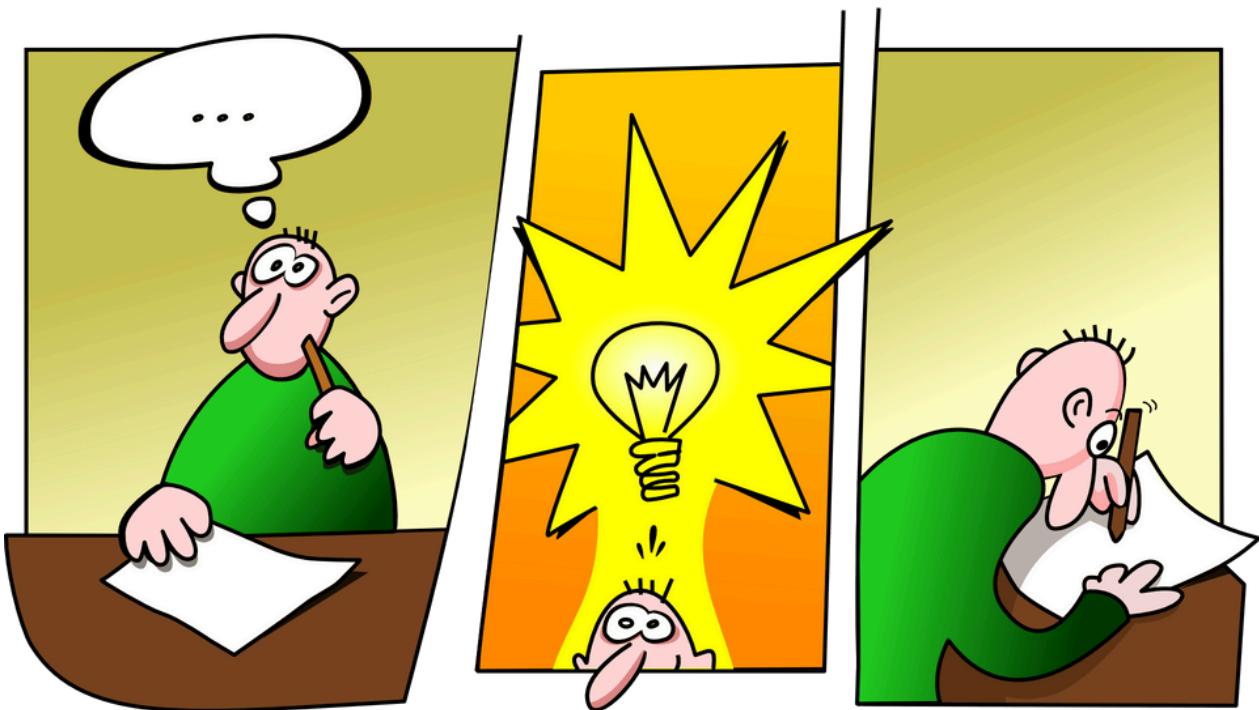
### Infraestructura robótica

Al diseñar robots o sistemas robóticos:

Planificando rutas en brazos robóticos: Un brazo robótico con múltiples articulaciones puede representarse como un grafo donde cada articulación es un nodo. Usar Prim podría ayudar a optimizar el recorrido para reducir energía o tiempo.

Optimización de redes de comunicación en equipos robóticos: Trabajando con varios robots colaborativos, usar el algoritmo para minimizar las conexiones de comunicación entre ellos (como en redes inalámbricas).





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

Mi trabajo de ensueño, que se centra en el diseño de robots, brazos robóticos y vehículos a control remoto, implementaría el algoritmo de Árbol Parcial mínimo de Prim de varias maneras para optimizar las operaciones y mejorar la eficiencia de los sistemas.

El primer escenario sería para la optimización de trayectorias internas al diseñar un brazo robótico con varias articulaciones, modelado como un grafo donde los nodos son las articulaciones y las aristas representarían las posibles conexiones con sus costos asociados, al igual la conexión de sensores y actuadores, el algoritmo de Prim puede ayudar a diseñar una red de cableado mínima que conecte todos los sensores con el controlador principal.

Por ejemplo, el diseño de un robot para ensamblaje automatizado:

1. Escenario: Estás diseñando un brazo robótico para ensamblar piezas en una línea de producción.
2. Problema: El brazo necesita sensores y actuadores distribuidos en diferentes partes para detectar y manipular las piezas.
3. Solución: Representa los puntos donde se colocan sensores y actuadores como nodos en un grafo. Las aristas representan el cableado posible entre ellos, con pesos que indican la longitud del cable o el costo de conexión.
4. Aplicación: Usa Prim para calcular la red de conexión mínima entre todos los puntos, asegurándote de reducir al máximo los materiales y el tiempo de instalación.

# Bibliografía

- Prim - Familia de funciones — pgRouting Manual (3.4). (s. f.).  
<https://docs.pgrouting.org/3.4/es/prim-family.html>
- Árboles de peso mínimo: algoritmos de Prim y Kruskal — Matemáticas Discretas para Ciencia de Datos. (s. f.). <https://madi.nekomath.com/P5/ArbolPesoMin.html>
- De la Paz, A. (2020, 20 marzo). Algoritmos de Kruskal y Prim. Copyright (C) 2020.  
<https://www.wextensible.com/temas/voraces/kruskal-prim.html>