

**Centro de Enseñanza Técnica
Industrial**

Ingeniería Mecatrónica



Samuel Josimar Orozco Torres

21110380 - 6E2

Índice

¿Qué es?	Pág. 3
¿Para qué sirve?	Pág. 5
¿Cómo se implementa en el mundo?	Pág. 10
¿Cómo lo implementarías en tu vida?	Pág. 12
¿Cómo lo implementarías en tu trabajo o trabajo de ensueño?	Pág. 13
Bibliografía	Pág. 14



¿Qué es?

En el mundo de las Matemáticas Avanzadas, el Algoritmo de Kruskal es una técnica esencial para resolver problemas relacionados con la teoría de grafos y la optimización de redes. Este potente método no sólo ayuda a comprender los conceptos subyacentes en las matemáticas de decisión, sino que también proporciona un enfoque sistemático para encontrar el árbol de expansión mínimo en un grafo no dirigido conectado.

El algoritmo de Kruskal encuentra un bosque de expansión mínima de un gráfico ponderado por borde no dirigido. Si el gráfico es conexo, encuentra un árbol de expansión mínimo. (Un árbol de expansión mínimo de un gráfico conectado es un subconjunto de los bordes que forman un árbol que incluye cada vértice, donde la suma de los pesos de todos los bordes del árbol se minimiza).

Para un gráfico desconectado, un bosque de expansión mínimo es compuesto por un árbol de expansión mínimo para cada componente conectado). Es un algoritmo codicioso en la teoría de grafos, ya que en cada paso agrega el siguiente borde de peso más bajo que no formará un ciclo al bosque de expansión mínimo.

El algoritmo puede resumirse en los siguientes pasos:

1. Coloca todos los vértices del grafo en árboles separados.
2. Ordena todas las aristas del grafo por orden ascendente de sus pesos.
3. Recorre las aristas ordenadas y realiza lo siguiente para cada arista:
 - Comprueba si los vértices finales de la arista están en árboles distintos.
 - Si los vértices están en árboles distintos, añade la arista al árbol de expansión mínima y fusiona los dos árboles.
 - Repite los pasos 3 hasta que todos los vértices estén incluidos en el árbol de expansión mínima o no haya más aristas que procesar.

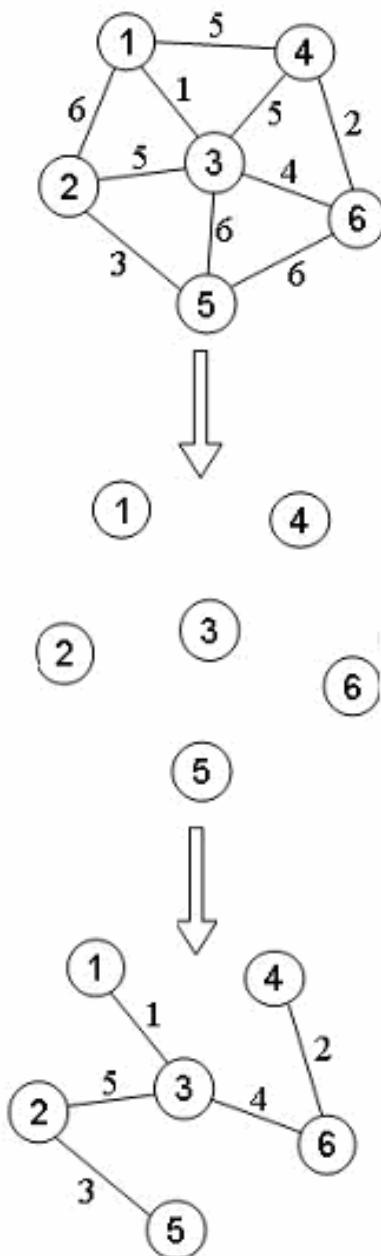


Figura 1. Ejemplo de Algoritmo de Kruskal.

Características principales del Algoritmo de Kruskal

El Algoritmo de Kruskal tiene varias características importantes que lo hacen único y útil:

1. Enfoque codicioso: El algoritmo sigue un enfoque codicioso, lo que significa que en cada paso selecciona la arista de menor peso que no forma un ciclo en el árbol de expansión mínimo. El resultado es un algoritmo eficaz y rápido.
2. Estructura de datos Union-Find: El Algoritmo de Kruskal se beneficia del uso de una estructura de datos Union-Find, ya que mantiene un registro eficaz de los vértices que pertenecen a cada subárbol. Permite al algoritmo comprobar rápidamente si se puede añadir una arista sin crear un ciclo.
3. Complejidad temporal: La complejidad temporal del algoritmo es $O(|E| \log |E|)$, donde $|E|$ es el número de aristas del grafo. Esto hace que el Algoritmo de Kruskal sea una opción excelente para resolver problemas de grafos a gran escala.
4. Funciona en grafos desconectados: Si el grafo está desconectado, el Algoritmo de Kruskal producirá un bosque de mínima extensión, que es la colección de árboles de mínima extensión para cada componente conectado del grafo.

Comparación entre el Algoritmo de Kruskal y otras técnicas

El Algoritmo de Kruskal y el Algoritmo de Prim son dos técnicas muy utilizadas para hallar el árbol de expansión mínimo de un grafo ponderado, no dirigido y conectado. Aunque ambos algoritmos pretenden alcanzar el mismo resultado, difieren significativamente en su planteamiento e implementación.

- Selección de aristas: El Algoritmo de Kruskal se centra en seleccionar y añadir las aristas de menor peso que no formen un ciclo, mientras que el Algoritmo de Prim se centra en seleccionar las aristas expandiéndose continuamente desde un vértice inicial y eligiendo la arista de menor peso que conecte el árbol en crecimiento con un nuevo vértice.
- Punto de partida: El Algoritmo de Kruskal no requiere un vértice inicial, ya que considera todas las aristas del grafo independientemente de su relación con un vértice concreto. En cambio, el Algoritmo de Prim requiere un vértice inicial a partir del cual el algoritmo se expandirá para cubrir todo el grafo.
- Estructura de datos: El Algoritmo de Kruskal utiliza principalmente la estructura de datos Union-Find para gestionar conjuntos disjuntos de vértices, lo que ayuda a comprobar eficazmente la existencia de ciclos y fusionar árboles al añadir aristas al árbol de expansión mínimo. Por otra parte, el Algoritmo de Prim suele utilizar una cola de prioridad o una estructura de datos similar para llevar la cuenta de las aristas con el menor peso que quedan por añadir al árbol.

¿Para qué sirve?

El Algoritmo de Kruskal es uno de los varios algoritmos de árbol de expansión mínima disponibles para resolver problemas de matemáticas de decisión y teoría de grafos. Junto con el Algoritmo de Prim y el Algoritmo de Boruvka (también conocido como Algoritmo de Sollin), forman un conjunto de algoritmos muy utilizado y potente. Mientras que el Algoritmo de Prim se centra en el nivel de los vértices, y el Algoritmo de Boruvka trabaja en el nivel de los componentes, el Algoritmo de Kruskal opera en el nivel de las aristas. Esto permite al Algoritmo de Kruskal producir un árbol de extensión mínima sin tener en cuenta ningún punto de partida o vértice específico, lo que lo hace especialmente útil para los problemas de grafos sin nodo de partida natural. Además, la capacidad del Algoritmo de Kruskal para manejar grafos desconectados y producir un bosque de extensión mínima lo hace muy adecuado para resolver problemas en los que el grafo de entrada puede tener múltiples componentes conectados. Su versatilidad, combinada con su complejidad temporal relativamente baja, garantiza que el Algoritmo de Kruskal siga siendo una opción popular entre los algoritmos de árbol de expansión mínima disponibles para diversas aplicaciones.

Algoritmo de Kruskal - Puntos clave

- Algoritmo de Kruskal: Enfoque codicioso para encontrar el árbol de expansión mínimo de un grafo ponderado no dirigido, desarrollado por Joseph Kruskal en 1956.
- Pasos clave: Ordena las aristas por peso, añádelas al árbol de expansión mínima sin formar ciclos, fusiona los árboles correspondientes.
- Características: Enfoque codicioso, estructura de datos Union-Find, complejidad temporal $O(|E| \log |E|)$, funciona en grafos desconectados.
- Aplicaciones: Diseño de redes, análisis de conglomerados, redes de transporte.
- Comparación con el Algoritmo de Prim: Diferente selección de aristas, punto de partida, uso de estructura de datos, tipo de grafo y complejidad temporal.

Ventajas e inconvenientes de utilizar el Algoritmo de Kruskal en Matemáticas de la Decisión

El Algoritmo de Kruskal ofrece varias ventajas e inconvenientes cuando se aplica a otros problemas de matemáticas de decisión. A continuación, se describen los principales pros y contras de utilizar el Algoritmo de Kruskal:

Pros:

- El Algoritmo de Kruskal es relativamente sencillo de entender y aplicar, lo que lo convierte en una técnica accesible para estudiantes y profesionales de las matemáticas de decisión.
- El enfoque codicioso seguido en el Algoritmo de Kruskal suele proporcionar una gran eficacia en la resolución de problemas a gran escala, con una complejidad temporal menor en comparación con otros algoritmos de árbol de expansión mínima.
- El algoritmo es flexible, ya que puede trabajar tanto con grafos conectados como desconectados. Esto lo hace adecuado para una amplia gama de aplicaciones en las que otros algoritmos pueden no ser aplicables.
- El uso de la estructura de datos Union-Find en el Algoritmo de Kruskal permite una gestión eficaz de los conjuntos de vértices disjuntos, minimizando el riesgo de introducir ciclos al añadir aristas al árbol de expansión mínima.

Contras:

- En algunos casos, el Algoritmo de Prim puede tener un tiempo de ejecución global mejor, especialmente si se implementa con una cola de prioridad, lo que lo convierte en una opción más eficiente en determinados escenarios.
- El Algoritmo de Kruskal requiere ordenar todas las aristas del grafo, lo que puede resultar caro computacionalmente para grafos muy grandes.
- En grafos con pesos de arista densos o uniformes, el Algoritmo de Kruskal puede no ofrecer ventajas de rendimiento significativas sobre otros algoritmos de árbol de expansión mínima.
- La naturaleza codiciosa del algoritmo puede conducir a soluciones subóptimas en determinadas situaciones, como cuando la solución óptima requiere seleccionar primero las aristas de mayor peso.

Comprender los puntos fuertes y débiles del Algoritmo de Kruskal en relación con otras técnicas de árbol de envergadura mínima es vital a la hora de tomar decisiones informadas sobre qué algoritmo es el más adecuado para un problema determinado en matemáticas de la decisión y teoría de grafos.

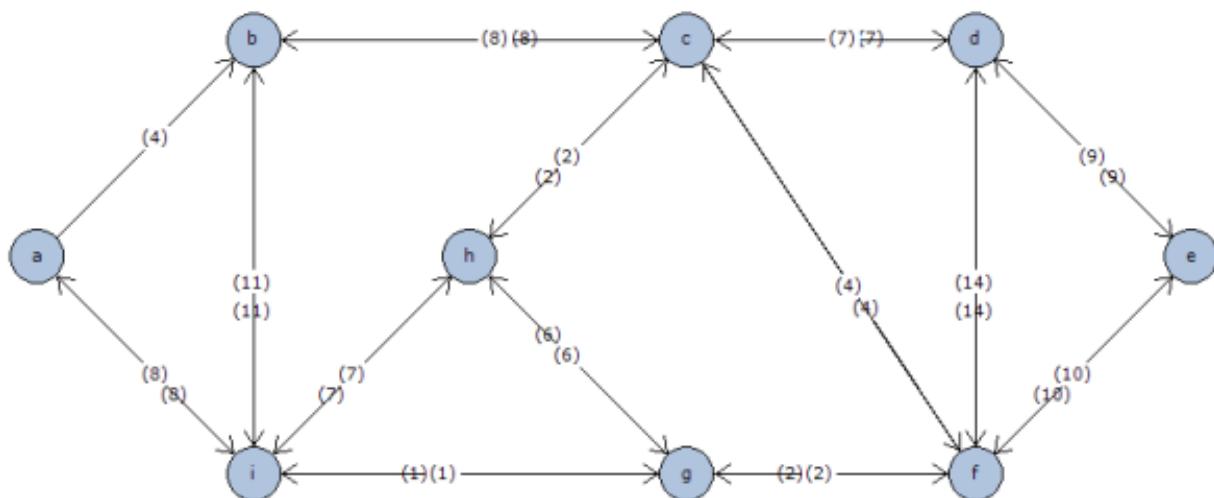
Ejemplo de árbol de coste total mínimo

Construya un grafo a partir de la siguiente matriz de datos:

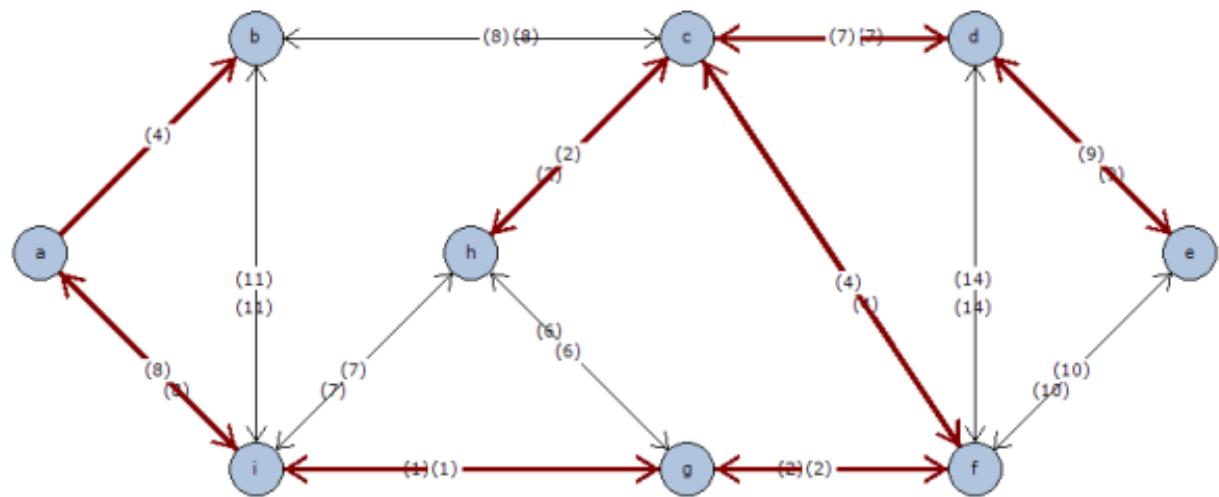
N1\N2	a	b	c	d	e	f	g	h	i
a		4							8
b			8						11
c		8		7		4		2	
d			7		9	14			
e				9		10			
f			4	14	10		2		
g						2		6	1
h			2				6		7
i	8	11				1	7		

Origen\Destino	a	b	c	d	e	f	g	h	i
a	4								8
b		8							11
c	8		7		4		2		
d		7		9	14				
e			9		10				
f		4	14	10		2			
g					2		6	1	
h		2				6		7	
i	8	11			1	7			

El grafo dibujado tendrá el siguiente aspecto:



No se requiere de la selección de ningún nodo para la ejecución del algoritmo, la representación del resultado será la siguiente.



Donde se muestra que el árbol de coste mínimo para este ejemplo tiene un coste total de 37 unidades.

- a ----(4)---> b
- d ----(7)---> c
- e ----(9)---> d
- f ----(4)---> c
- g ----(2)---> f
- h ----(2)---> c
- i ----(8)---> a
- i ----(1)---> g

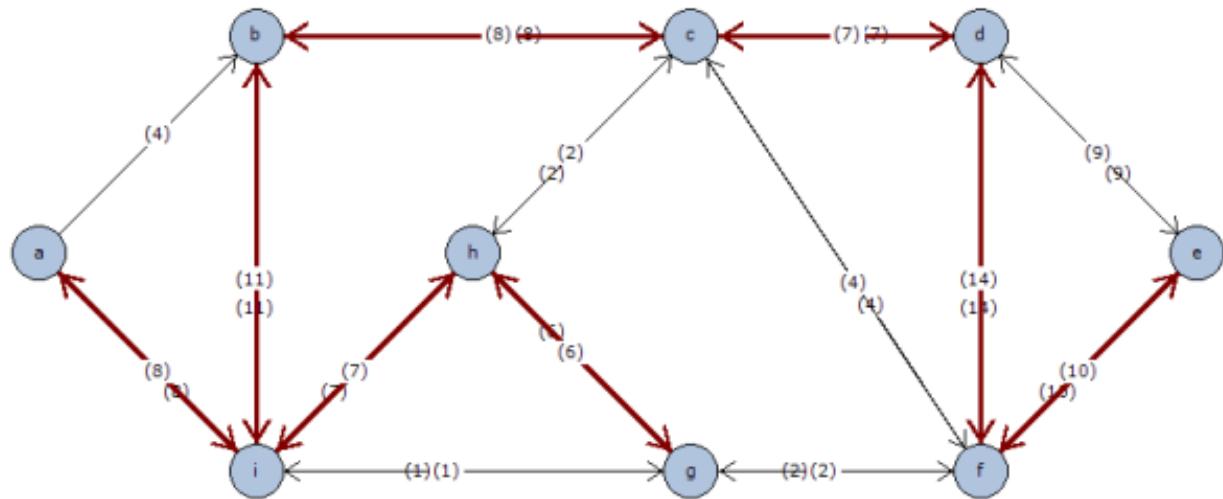
Coste total = 37

Matriz de Arcos del árbol con coste mínimo:

N1\N2	a	b	c	d	e	f	g	h	i
a	0	1	0	0	0	0	0	0	0
b	0	0	0	0	0	0	0	0	0
c	0	0	0	0	0	0	0	0	0
d	0	0	1	0	0	0	0	0	0
e	0	0	0	1	0	0	0	0	0
f	0	0	1	0	0	0	0	0	0
g	0	0	0	0	0	1	0	0	0
h	0	0	1	0	0	0	0	0	0
i	1	0	0	0	0	0	1	0	0

Árbol de coste máximo

Del mismo modo se puede calcular el árbol de coste máximo, que en este ejemplo tendrá un coste total de 71 unidades, tal y como muestran la siguiente figura y la tabla de resultados.



c ----(8)---> b

d ----(7)---> c

f ----(14)---> d

f ----(10)---> e

h ----(6)---> g

i ----(8)---> a

i ----(11)---> b

i ----(7)---> h

Coste total = 71

Matriz de Arcos del árbol con coste máximo:

N1\N2	a	b	c	d	e	f	g	h	i
a	0	0	0	0	0	0	0	0	0
b	0	0	0	0	0	0	0	0	0
c	0	1	0	0	0	0	0	0	0
d	0	0	1	0	0	0	0	0	0
e	0	0	0	0	0	0	0	0	0
f	0	0	0	1	1	0	0	0	0
g	0	0	0	0	0	0	0	0	0
h	0	0	0	0	0	0	1	0	0
i	1	1	0	0	0	0	0	1	0

¿Cómo se implementa en el mundo?

El algoritmo de Kruskal se utiliza en el mundo real para resolver problemas de optimización relacionados con redes de conexión mínima. Se aplica en diversas áreas donde es necesario construir sistemas eficientes de conexión, como infraestructura, comunicación y logística.

Redes de Transporte

- Construcción de carreteras o vías férreas:

Kruskal se utiliza para conectar ciudades minimizando los costos de construcción, representando cada ciudad como un nodo y las posibles rutas entre ellas como aristas con pesos (costos).

- Sistemas de distribución:

Empresas de logística pueden usar el algoritmo para optimizar rutas de entrega o crear redes de transporte mínimas entre almacenes y puntos de venta.



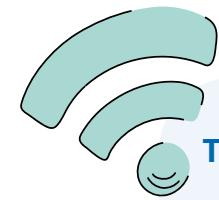
Redes Eléctricas y de Servicios

- Diseño de redes eléctricas:

Para conectar estaciones eléctricas o transformadores con el menor coste de cableado, Kruskal ayuda a evitar ciclos y asegura que todas las estaciones estén conectadas de manera eficiente.

- Abastecimiento de agua o gas:

Se usa para construir sistemas de tuberías optimizados, minimizando el costo total de instalación.



Telecomunicaciones y Redes de Datos

- Optimización de redes de fibra óptica o cables:

Empresas de telecomunicaciones implementan Kruskal para construir infraestructuras de conexión entre ciudades con el menor costo posible.

- Diseño de redes de computadoras:

En grandes organizaciones, Kruskal ayuda a establecer conexiones mínimas entre servidores y equipos, optimizando recursos.



Análisis de Datos y Computación

- Agrupamiento de datos (clustering):

En aprendizaje automático, Kruskal se utiliza en algunos algoritmos de agrupamiento jerárquico para construir relaciones mínimas entre puntos de datos.

- Redes sociales:

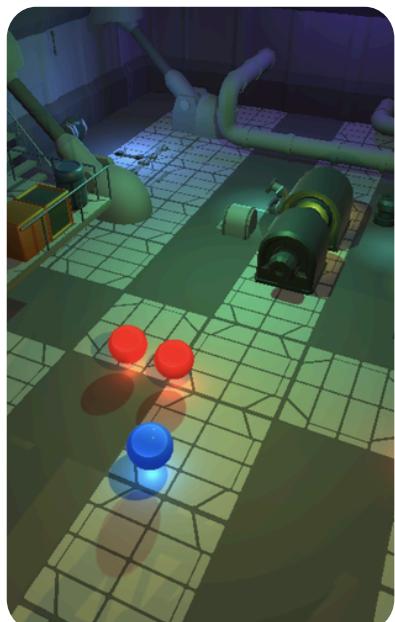
Se puede usar para analizar y conectar comunidades dentro de una red social con el menor número de relaciones necesarias.



Juegos y Simulaciones

- Diseño de mapas en videojuegos:

Kruskal se utiliza para generar caminos mínimos entre puntos de interés en un mapa, como ciudades o estaciones en un juego de estrategia.



¿Cómo lo implementarías en tu vida?

Como estudiante de ingeniería mecatrónica, puedes implementar el algoritmo de Kruskal en varias situaciones para optimizar tiempo y recursos.

Diseño de sistemas eléctricos y cableado



Conexión de sensores y actuadores:

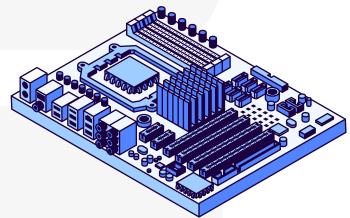
Al desarrollar un proyecto donde se necesite conectar múltiples sensores y actuadores a un microcontrolador, el algoritmo de Kruskal puede ayudar a diseñar el sistema de cableado más eficiente, reduciendo el costo y el uso de materiales.



Optimización de redes en robots móviles

Comunicación entre módulos:

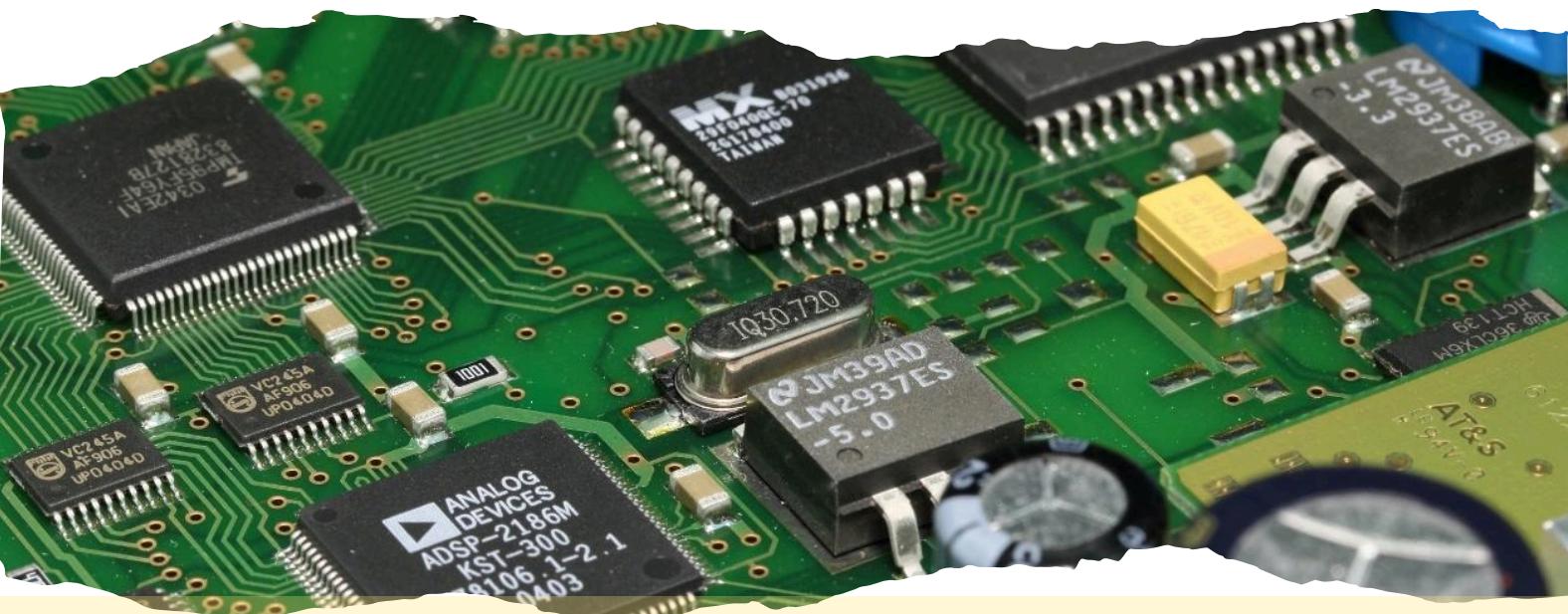
En un robot autónomo con múltiples módulos (como cámaras, motores y sensores), Kruskal puede determinar cómo interconectar estos módulos usando el menor número de cables o rutas de comunicación inalámbrica.

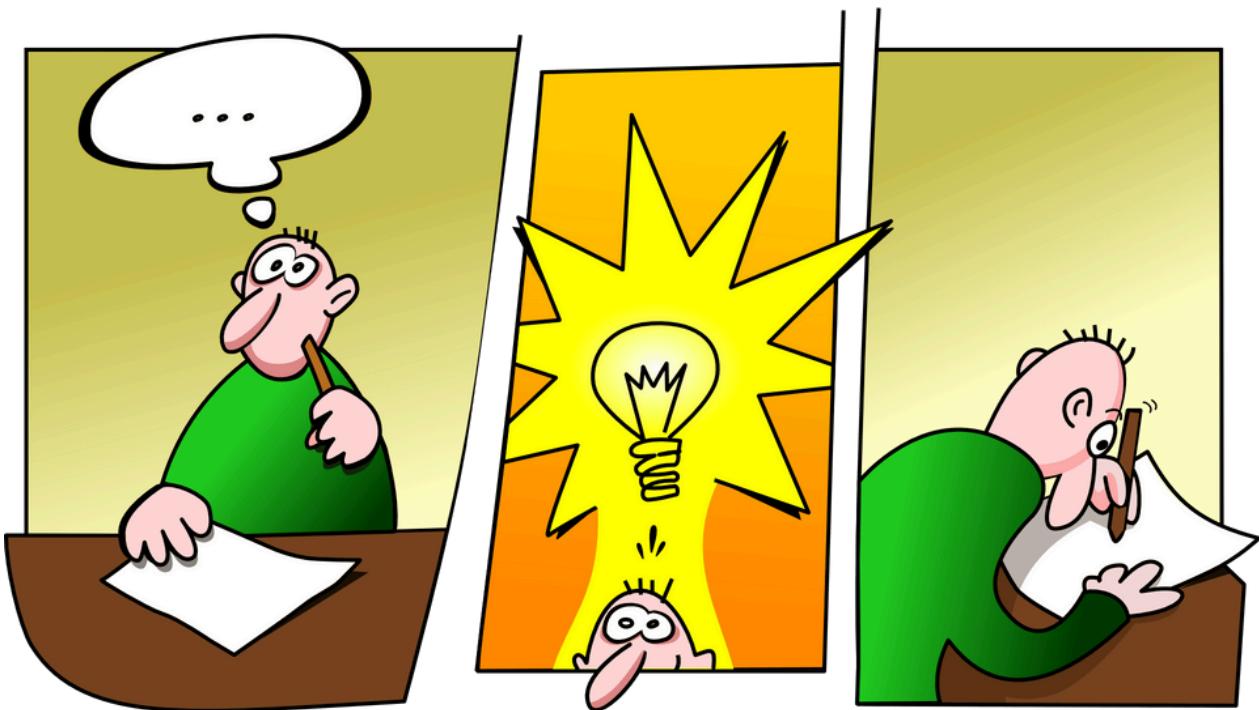


Diseño de placas de circuito impreso (PCB)

Optimización de trazados:

Usar Kruskal para optimizar las conexiones entre componentes en una placa de circuito. Esto ayuda a minimizar las pistas y reducir interferencias eléctricas.





¿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 Kruskal de varias maneras para optimizar las operaciones y mejorar la eficiencia de los sistemas.

Se puede implementar para optimizar recursos y conexiones en proyectos como:

- Brazos robóticos: Minimizar el cableado interno entre sensores y actuadores para reducir peso y costos.
- Robots móviles: Planificar rutas óptimas entre estaciones o diseñar redes de comunicación eficiente entre robots colaborativos.
- Circuitos y PCB: Optimizar trazados en placas de circuito, reduciendo material y espacio.
- Vehículos autónomos: Conectar sensores y actuadores al sistema central con el menor costo de cableado.

Bibliografía

- Algoritmo de Kruskal _ AcademiaLab. (s. f.). https://academia-lab.com/encyclopedia/algoritmo-de-kruskal/#google_vignette
- Árboles de peso mínimo: algoritmos de Prim y Kruskal — Matemáticas Discretas para Ciencia de Datos. (s. f.-b). <https://madi.nekomath.com/P5/ArbolPesoMin.html>
- Algoritmo de Kruskal: grafos, conectividad | StudySmarter. (s. f.). StudySmarter ES. <https://www.studysmarter.es/resumenes/matematicas/matematicas-de-la-decision/algoritmo-de-kruskal/>
- algoritmo_kruskal [Grafos - software para la construcción, edición y análisis de grafos.]. (s. f.). https://arodrigu.webs.upv.es/grafos/doku.php?id=algoritmo_kruskal