**Algoritmo de la ruta más corta de Dijkstra**

### **Propósito y Usos**

Con el algoritmo de Dijkstra, puedes encontrar la ruta más corta o el **camino más corto** entre los nodos de un grafo. Específicamente, puedes encontrar el camino más corto desde un nodo (llamado el **nodo de origen**) a todos los otros nodos del grafo, generando un árbol del camino más corto.

Este algoritmo es usado por los dispositivos GPS para encontrar el camino más corto entre la ubicación actual y el destino del usuario. Tiene amplias aplicaciones en la industria, especialmente en aquellas áreas que requieren modelar redes.

### **Aspectos básicos del algoritmo de Dijkstra**

* El algoritmo de Dijkstra básicamente inicia en el nodo que escojas (el nodo de origen) y analiza el grafo para encontrar el camino más corto entre ese nodo y todos los otros nodos en el grafo.
* El algoritmo mantiene un registro de la distancia conocida más corta desde el nodo de origen hasta cada nodo y actualiza el valor si encuentra un camino más corto.
* Una vez que el algoritmo ha encontrado el camino más corto entre el nodo de origen y otro nodo, ese nodo se marca como "visitado" y se agrega al camino.
* El proceso continúa hasta que todos los nodos en el grafo han sido añadidos al camino. De esta forma, tenemos un camino que conecta al nodo de origen con todos los otros nodos siguiendo el camino más corto posible para llegar a cada uno de ellos.

### **Requisitos**

El algoritmo de Dijkstra solo puede aplicarse a grafos con arcos cuyos valores o pesos son **positivos**. Esto se debe a que durante es proceso, los valores de los arcos deben ser sumados para encontrar el camino más corto.

Si existe un valor negativo en el grafo, el algoritmo no funcionará correctamente. Una vez que el nodo se marca como "visitado", el camino actual hacia ese nodo se marca como el camino más corto para alcanzar ese nodo pero los valores negativos pueden cambiar esto si el valor total puede ser reducido luego de este paso.

## **En resumen**

* Los grafos son usados para modelar conexiones entre objetos, personas o entidades. Tienen dos elementos principales: nodos y arcos. Los nodos representan objetos y los arcos representan las conexiones entre esos objetos.
* El algoritmo de Dijkstra encuentra el camino más corto entre un nodo dado (el nodo de origen) y todos los otros nodos del grafo.
* Este algoritmo usa los valores de los arcos para encontrar el camino que minimiza el valor total entre el nodo de origen y los demás nodos del grafo. Este valor depende de lo que representa el valor de los arcos en el grafo. Puede ser, por ejemplo, tiempo, costo o distancia.

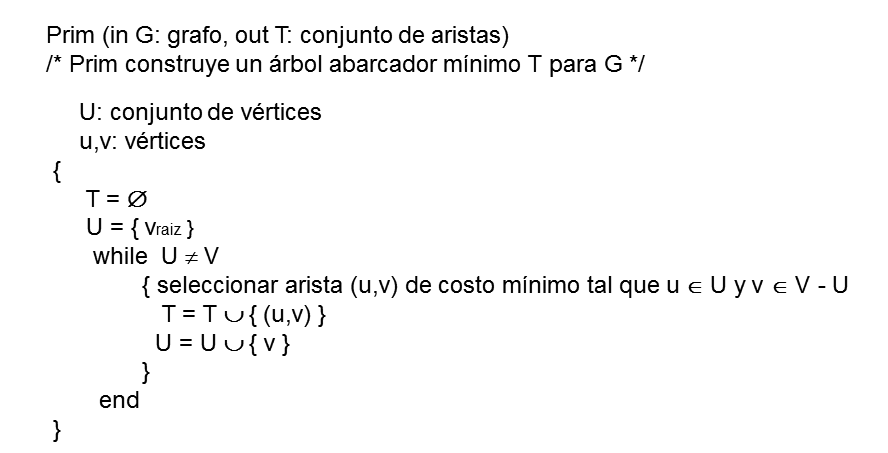
# Objetivo y Descripción

Sea **G** = ( **V** , **A** ), donde **V** es el conjunto de vértices del grafo **G** y **A** es el conjunto de aristas del mismo; **n** es la cantidad de vértices del conjunto **V** y **a** es la cantidad de aristas del conjunto **A**.

El objetivo del algoritmo de ***Prim*** es construir, a partir de un grafo G no dirigido, pesado y conexo, un árbol abarcador de costo mínimo (MST del inglés Minimum Spanning Tree), es decir, encuentra un conjunto **T** de **n-1** aristas pertenecientes al conjunto **A** que conectan todos los vértices del grafo; donde el costo total de todas las aristas del árbol es el menor posible.

El algoritmo de ***Prim*** comienza inicializando el conjunto **T** (MST resultante) en vacío y el conjunto **U** con un vértice de**V** (vraiz) que será la raíz del árbol resultante. El árbol representado por el conjunto **T** crece arista por arista. En cada paso del algoritmo se encuentra la arista de menor costo (**u**,**v**) que conecta **U** con **V** - **U**, es decir, **u** está en **U**(vértices de árbol) y **v** está en **V** - **U**(vértices del grafo que aún no se incorporaron al árbol) y luego se agrega **v** a **U** y la arista (**u**,**v**) a **T**. Este paso se repite hasta que **U** sea igual a **V**.

Veamos a continuación un resumen del algoritmo:



Veamos un ejemplo de la construcción del MST:

# Algoritmo de Kruskal

El **algoritmo de Kruskal** es un [algoritmo](https://es.wikipedia.org/wiki/Algoritmo) de la [teoría de grafos](https://es.wikipedia.org/wiki/Teor%C3%ADa_de_grafos) para encontrar un [árbol recubridor mínimo](https://es.wikipedia.org/wiki/%C3%81rbol_recubridor_m%C3%ADnimo) 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 mínimo. Si el grafo no es conexo, entonces busca un bosque expandido mínimo (un *árbol expandido mínimo* para cada [componente conexa](https://es.wikipedia.org/w/index.php?title=Componente_conexa_(teor%C3%ADa_de_los_grafos)&action=edit&redlink=1)). Este algoritmo toma su nombre de [Joseph Kruskal](https://es.wikipedia.org/wiki/Joseph_Kruskal), quien lo publicó por primera vez en 1956.[1](https://es.wikipedia.org/wiki/Algoritmo_de_Kruskal#cite_note-Kruskal-1)​[2](https://es.wikipedia.org/wiki/Algoritmo_de_Kruskal#cite_note-Algorithms-2)​. Otros algoritmos que sirven para hallar el **árbol de expansión mínima** o **árbol recubridor mínimo** son el [algoritmo de Prim](https://es.wikipedia.org/wiki/Algoritmo_de_Prim), el algoritmo del borrador inverso y el [algoritmo de Boruvka](https://es.wikipedia.org/wiki/Algoritmo_de_Boruvka).

## **Descripción[**[**editar**](https://es.wikipedia.org/w/index.php?title=Algoritmo_de_Kruskal&action=edit&section=1)**]**

El algoritmo de Kruskal es un ejemplo de [algoritmo voraz](https://es.wikipedia.org/wiki/Algoritmo_voraz) que funciona de la siguiente manera:

* se crea un bosque *B* (un conjunto de árboles), donde cada vértice del grafo es un [árbol](https://es.wikipedia.org/wiki/%C3%81rbol_(teor%C3%ADa_de_grafos)) separado
* se crea un conjunto *C* que contenga a todas las aristas del grafo
* mientras *C* es *no vacío*
  + eliminar una arista de peso mínimo de *C*
  + si esa arista conecta dos árboles diferentes se añade al bosque, combinando los dos árboles en un solo árbol
  + en caso contrario, se desecha la arista

Al acabar el algoritmo, el bosque tiene un solo componente, el cual forma un árbol de expansión mínimo del grafo.

En un **árbol de expansión mínimo** se cumple:

* la cantidad de aristas del árbol es la cantidad de nodos menos uno (1).