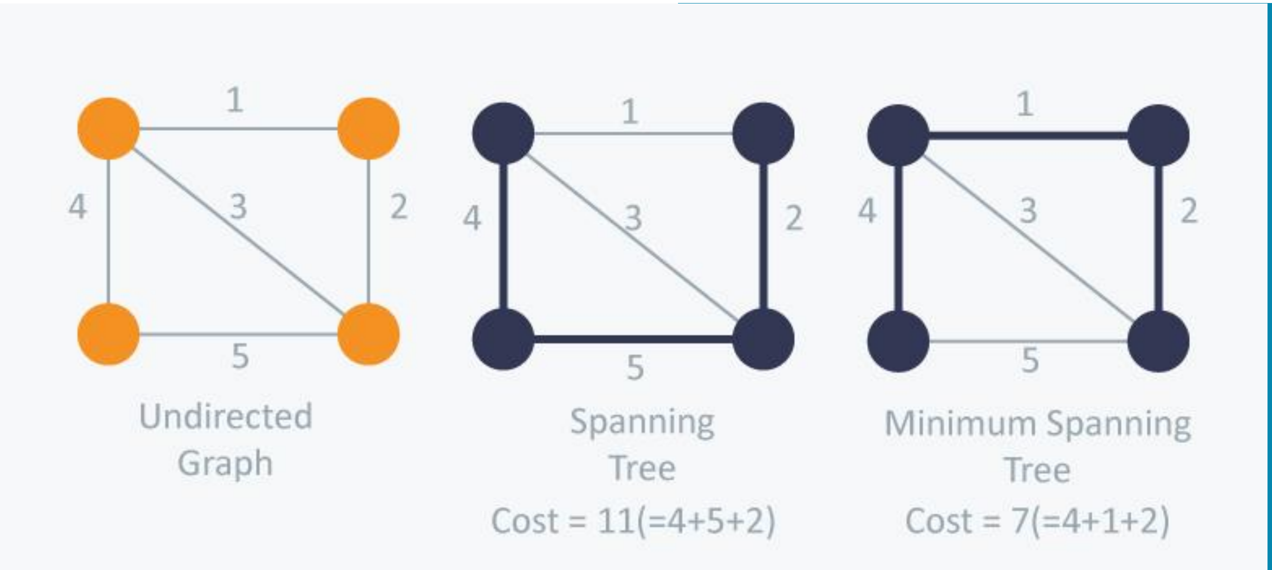


Complejidad Algorítmica

Unidad 2: Algoritmos voraces, programación dinámica y problemas P-NP

Módulo 10: Árbol de Expansión Mínima (MST)

Ing. Patricia Reyes Silva
pcsiprey@upc.edu.pe

Complejidad Algorítmica

Semana 10 / Sesión 1

MÓDULO 10: Arboles de Expansión Mínima (MST)

Contenido

1. Conceptos básicos y MST
2. Algoritmos MST
   2.1. Algoritmo Kruskal
   2.2. Algoritmo PRIM
3. Aplicaciones del MST

Preguntas

1. Conceptos básicos y MST

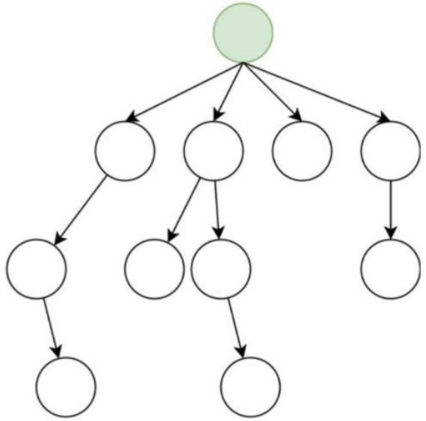
¿Qué es un árbol dentro de la teoría de grafos?

- Antes de responder, formalicemos algunos conceptos que ya conocemos.

Árbol: es <u>una estructura de datos que simula una estructura de árbol jerárquica</u> compuesta por un <u>conjunto finito de uno o más nodos</u> tales que:

- Hay <u>un nodo</u> especialmente designado llamado **raíz**.
- Los nodos restantes se dividen en n>=0 **<u>conjuntos disjuntos</u>** $T_1, T_2, T_3, \ldots, T_n$

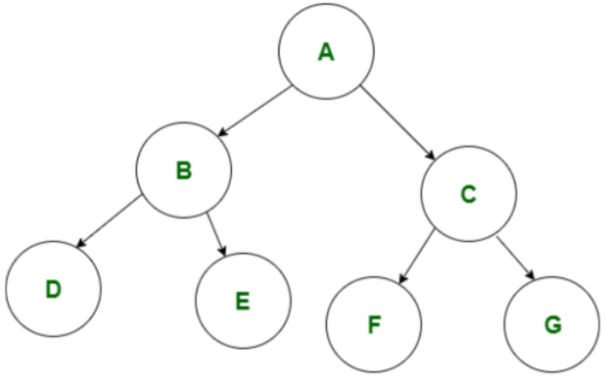
  donde $T_1, T_2, T_3, \ldots, T_n$ se denomina <u>subárboles o hijos de la raíz</u>.



Árbol General

Puede tener cero o muchos subárboles secundarios desordenados

vs

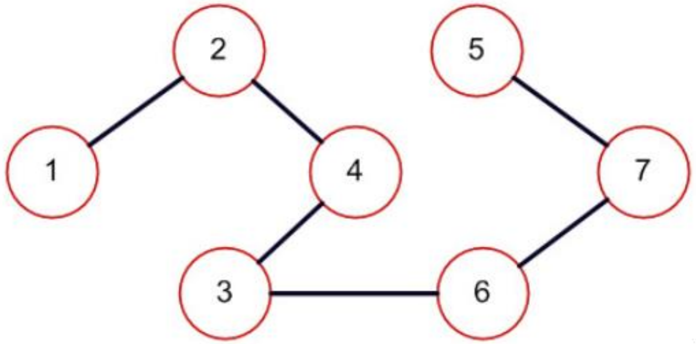


Árbol binario

Cada nodo puede tener como máximo dos nodos (subárbol izquierdo y derecho), ordenados.

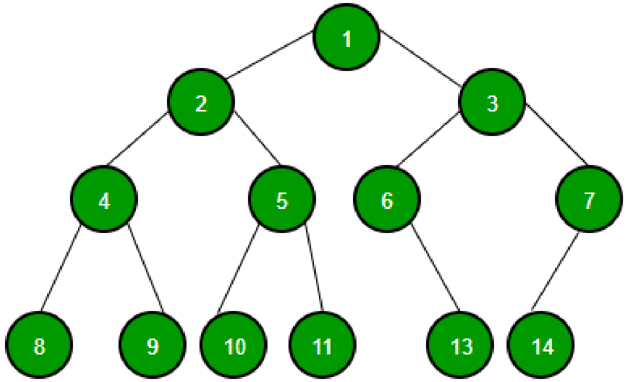
1. Conceptos básicos y MST

¿Todo grafo es un árbol?



Grafo

vs



Árbol binario

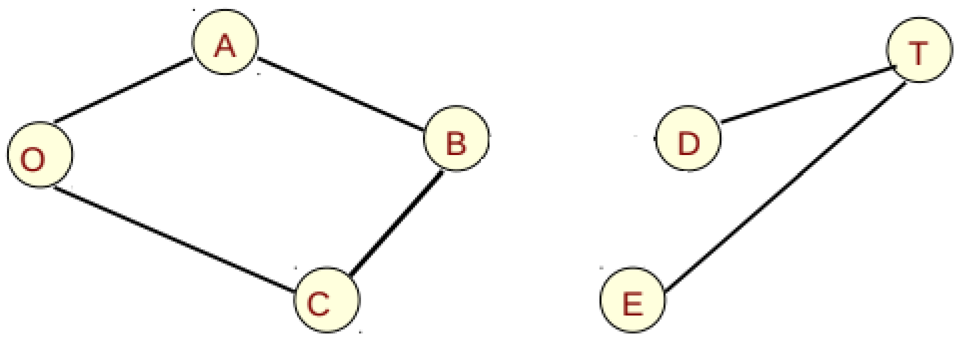
1. Conceptos básicos y MST

¿Hay diferencias entre un grafo y un árbol?

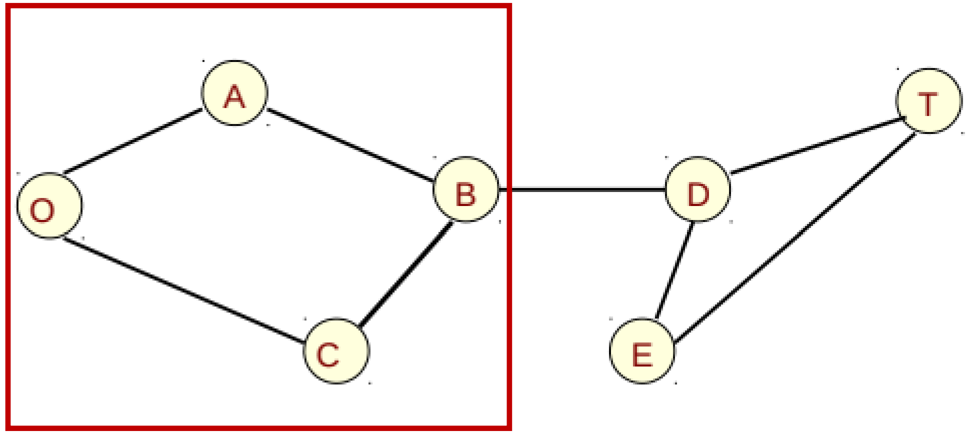
GRAFO	ARBOL
1. El Grafo es una estructura de datos no lineal.	1. Igual.
2. Es una colección de vértices/nodos y aristas.	2. Igual.
3. Cada nodo puede tener cualquier número de aristas.	3. Pueden contener cualquier número de nodos secundarios (subárboles). Pero en el caso de los árboles binarios, cada nodo puede tener como máximo dos nodos secundarios.
4. No hay un nodo único llamado raíz en el grafo.	**4. Hay un nodo único llamado raíz** en los árboles.
5. Se puede formar un ciclo.	5. No existe ningún ciclo.
6. Aplicación: Encontrar la ruta más corta.	6. Aplicación: árboles de juego, árboles de decisión.

1. Conceptos básicos y MST

Sea el siguiente grafo: ¿Es un árbol?

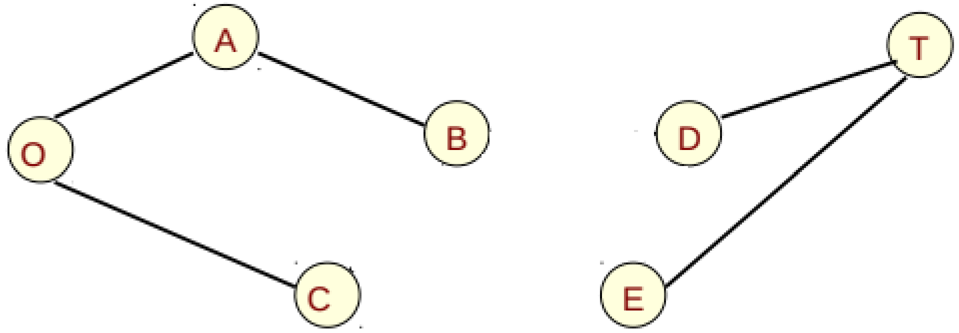


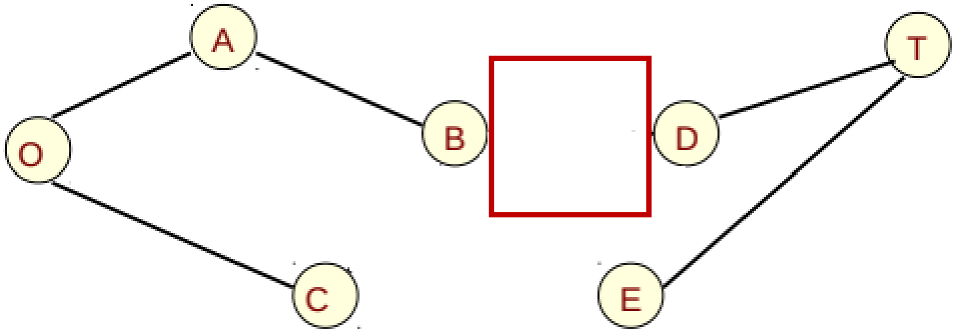
NO, es un árbol porque una red o grafo con ciclo, ¡no es un árbol!



1. Conceptos básicos y MST

Sea el siguiente grafo: ¿Es un árbol?

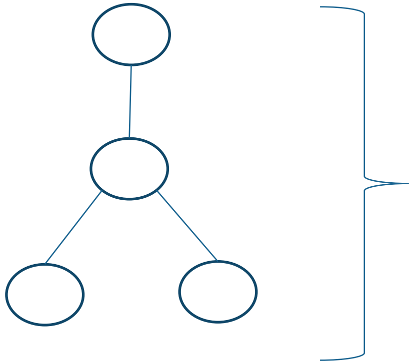




- **NO**, es un árbol porque una red NO CONEXA, ¡no es un árbol! Es un bosque.
- Un bosque es una colección disjunta de árboles.

CARACTERISTICAS DE UN ARBOL

- **Es un grafo acíclico conexo**, en otras palabras, una árbol es un grafo conexo sin ciclos.
- Los bordes de un árbol se conocen como ramas.
- Los elementos de los árboles se llaman sus nodos.
- Los nodos sin nodos secundarios se denominan nodos hoja.
- Un árbol con 'n' vértices tiene aristas 'n-1'.



ARBOL:

Grafo acíclico conexo con nodos-1 aristas

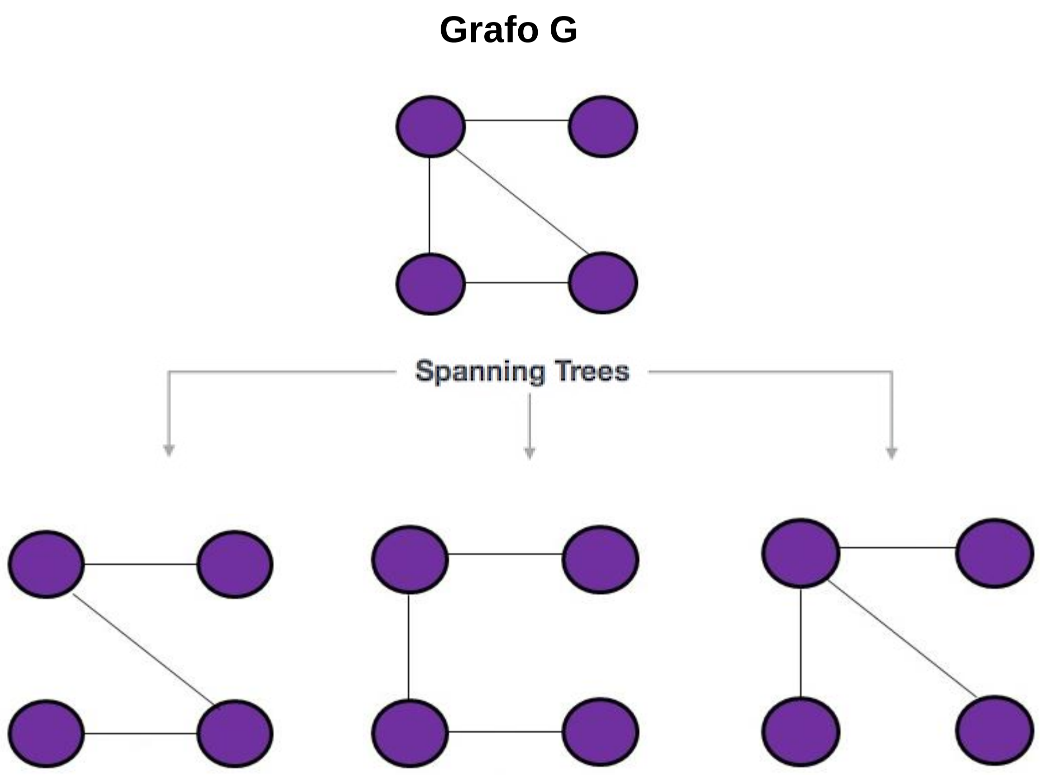
1. Conceptos básicos y MST

ARBOL DE EXPANSION (Spanning Tree)

Dado un grafo no dirigido y conexo G = (V,A), un árbol de expansión del grafo G :

1. Es un árbol que se extiende de G, es decir, incluye todos los vértices de G.

2. Es un subgrafo de G, donde cada borde en el árbol pertenece a G.



1. Conceptos básicos y MST

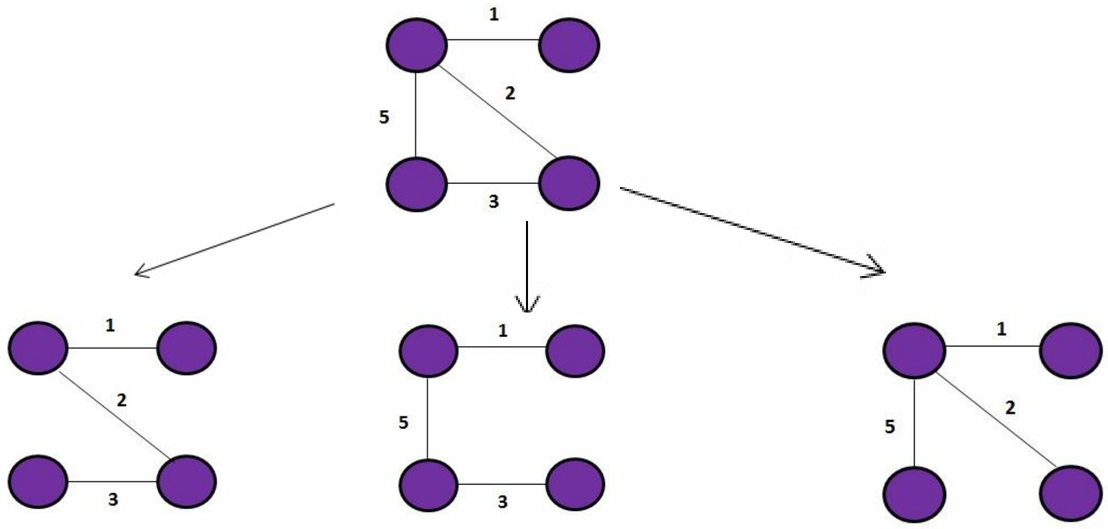
- **Minimum spanning tree (MST)** o **Minimum weight spanning tree**, significa árbol de expansión mínimo o árbol de expansión de peso mínimo**,** y es un modelo de optimización de redes.
- Un árbol de expansión mínima es un **subconjunto de los bordes ponderados de un grafo no dirigido** que conecta todos los vértices entre sí, **sin ningún ciclo** y con el objetivo que el **peso total de los bordes sea el mínimo posible**.

¿Qué es un MST?

(Árbol de Expansión Mínimo)

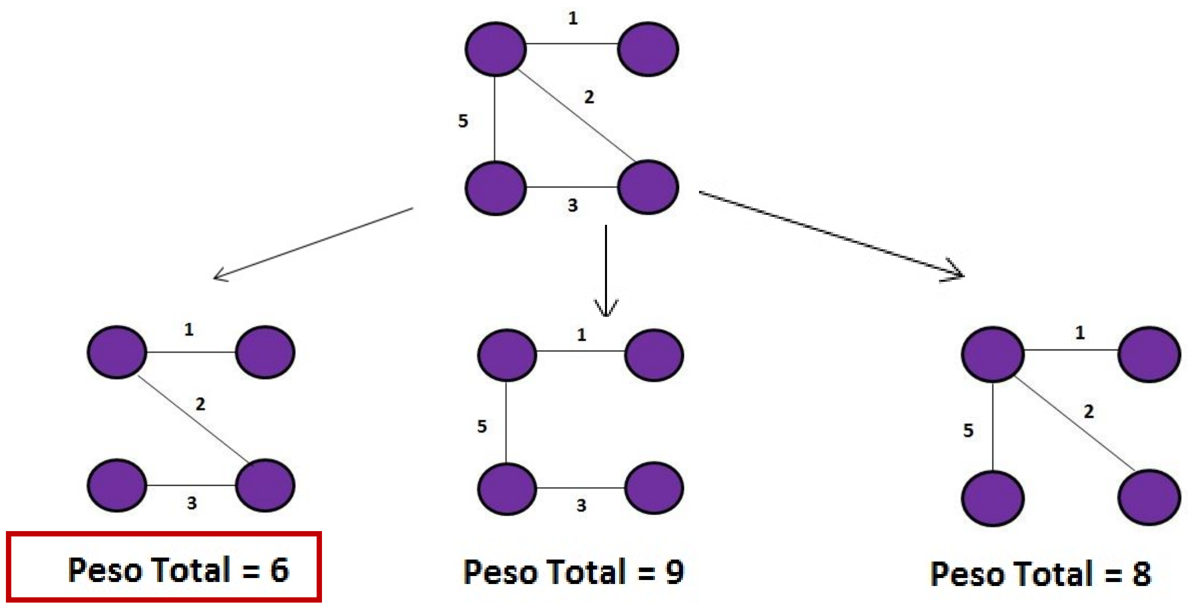
Sea G (V,A)

Grafo G ponderado con 4 vértices y 4 aristas



1. Conceptos básicos y MST

Sea G (V,A) ➡ Grafo G ponderado con 4 vértices y 4 aristas



<u>Árbol de expansión mínimo (o de peso mínimo)</u>

- El <u>costo del árbol de expansión es la suma de los pesos de todos los bordes del árbol</u>.
- Puede haber muchos árboles de expansión (a partir de un grafo).
- El **árbol de expansión mínimo** es el **árbol de expansión donde el costo es mínimo** entre todos los árboles de expansión.
- También puede haber muchos árboles de expansión mínimos.

2. Algoritmos MST

¿Qué algoritmos solucionan el problema del MST?

Existen dos Algoritmos famosos para resolver este problema de calculo del árbol de expansión mínimo:

- **Algoritmo Kruskal**
- **Algoritmo Prim**

Es común en ambos algoritmos:

- Utilizar la propiedad anterior.
- Ser de tipo voraz (codicioso o greedy), es decir, que seleccionan uno de los candidatos con el criterio que es mejor en cada momento (menor costo).
- Giran en torno a verificar si al agregar un borde o un vértice se crea un ciclo o no.

2. Algoritmos MST

2.1. Algoritmo de KRUSKAL

- El **algoritmo de Kruskal** <u>construye el árbol de expansión agregando aristas</u> una por una en un árbol de expansión en crecimiento.
- El **algoritmo de Kruskal** sigue un enfoque codicioso (voraz o greedy), ya que en cada iteración encuentra un borde que tiene el menor peso y lo agrega al árbol de expansión en crecimiento.

<u>Pasos del algoritmo</u>

- Ordenar los bordes del gráfico con respecto a sus pesos.
- Agregar bordes al MST desde el borde con el peso más pequeño hasta el borde con el peso más grande.
- <u>Solo agregar bordes que no formen un ciclo</u>, bordes que conecten solo componentes desconectados.
- La forma más común de averiguar si dos componentes están desconectados, es aplicando el algoritmo **Union Find**. Este divide los vértices en grupos y nos permite verificar si dos vértices pertenecen al mismo grupo o no y, por lo tanto, decidir si agregar un borde crea un ciclo.

Veamos un ejemplo…

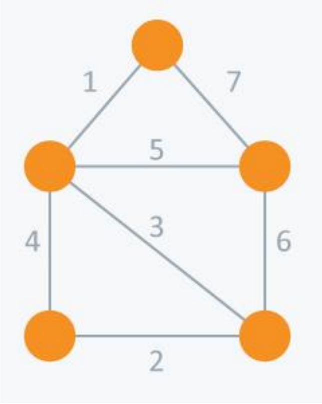
2. Algoritmos MST

2.1. Algoritmo de KRUSKAL

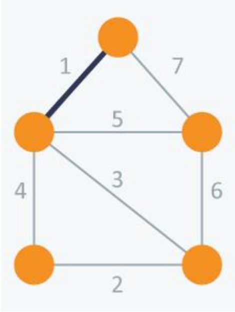
En el algoritmo de Kruskal, <u>en cada iteración seleccionaremos la arista con el peso más bajo sin que forme un ciclo.</u>

Ejemplo # 1:

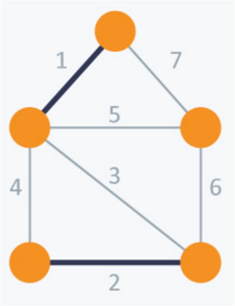
G(V,A)



1. Comenzaremos primero con el borde de peso más bajo, es decir, los bordes con peso 1.

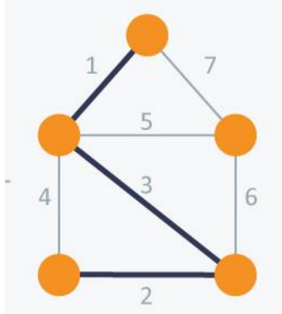


2. Después de eso, seleccionaremos el segundo borde de peso más bajo, es decir, el borde con peso 2.

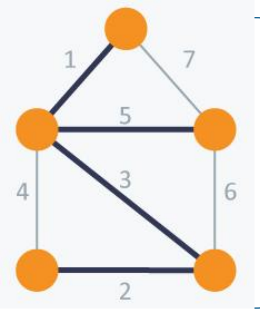


Observamos que los bordes 1 y 2 son totalmente disjuntos.

3. El próximo borde será el tercer borde ponderado más bajo, el borde con peso 3, que conecta las dos partes separadas del grafo



4. No podemos elegir el borde con peso 4, porque con eso se creará un ciclo y no podemos tener ningún ciclo. Por tanto elegimos el borde con peso 5.



Ignoramos también los bordes con pesos 6 y 7 porque también estarían formando ciclos.

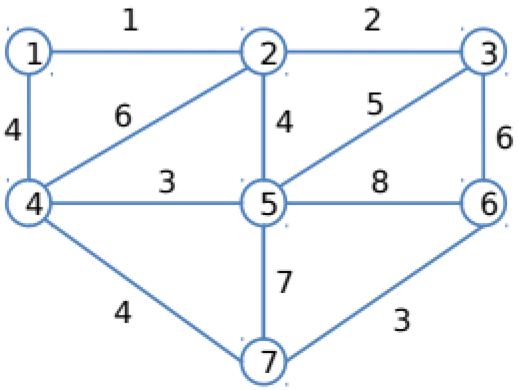
Finalmente, terminamos con un **árbol de expansión mínimo** con un **costo total del MST = 11** (suma de los costos de los bordes 1 + 2 + 3 + 5)

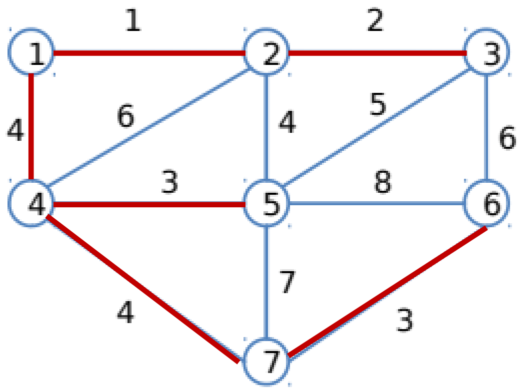
2. Algoritmos MST

2.1. Algoritmo de KRUSKAL

Ejemplo #2: Encontrar el árbol de expansión mínima por el algoritmo de Kruskal.

G(V,A)

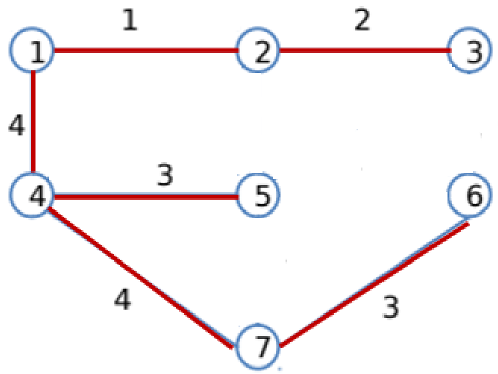




Costos de las aristas

1-2 = 1
2-3 = 2
4-5 = 3
6-7 = 3
1-4 = 4
4-7 = 4

Árbol de expansión mínimo



Costo total del MST = 17

2. Algoritmos MST

2.1. Algoritmo de KRUSKAL

Implementación

Necesitamos:

- Ordenar las aristas de G, de menor a mayor: O(a log a).
- Saber si una arista dada (v, w) provocará un ciclo.
  - ➢ ¿Cómo comprobar rápidamente si (v, w) forma un ciclo?
    - ✓ Una arista (v, w) forma un ciclo si v y w están en el mismo **componente conexo**.
    - ✓ La relación “están en el mismo componente conexo” es una relación de equivalencia.

Pseudocodigo

Sea el grafo G= (V, A)

1. Empezar con un grafo sin aristas: G’= (V, Ø)
2. Seleccionar la arista de menor coste de A.
   - ➢ Si la arista seleccionada forma un ciclo en G’, eliminarla.
   - ➢ Si no, añadirla a G’.
3. Repetir el paso 2. hasta tener n-1 aristas

```
KRUSKAL(G):
A = ∅
For each vertex v ∈ G.V:
    MAKE-SET(v)
For each edge (u, v) ∈ G.E ordered by increasing order by weight(u, v):
    if FIND-SET(u) ≠ FIND-SET(v):
    A = A ∪ {(u, v)}
    UNION(u, v)
return A
```

2. Algoritmos MST

2.2. Algoritmo PRIM

- El algoritmo de Prim también usa el enfoque Greedy (voraz o codicioso) para encontrar el árbol de expansión mínimo.
- En el Algoritmo de Prim hacemos crecer el árbol de expansión desde una posición inicial.
- A diferencia de un borde en Kruskal, agregamos un vértice al árbol de expansión creciente en Prim.

Pasos del algoritmo

- Mantener dos conjuntos disjuntos de vértices. Uno que contiene vértices que están en el árbol de expansión en crecimiento y el segundo que no está en el árbol de expansión en crecimiento.
- Seleccionar el vértice menos costoso que esté conectado al primer árbol de expansión en crecimiento y que no esté en el segundo árbol de expansión en crecimiento.
- Insertar los vértices, que están conectados al primer árbol de expansión en crecimiento, en la cola de prioridad.
- Consultar por los ciclos. Marcar los nodos que ya han sido seleccionados e insertar solo aquellos nodos en la cola de prioridad que no estén marcados.

Veamos a través de un ejemplo…

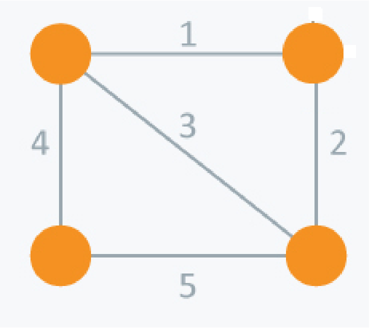
2. Algoritmos MST

2.2. Algoritmo PRIM

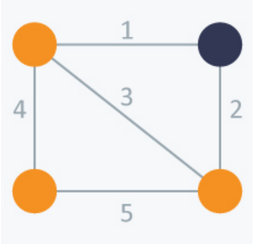
En el algoritmo de PRIM, al igual que en Kruskal, en cada iteración seleccionaremos la arista con el peso más bajo sin que forme un ciclo.

Ejemplo # 1:

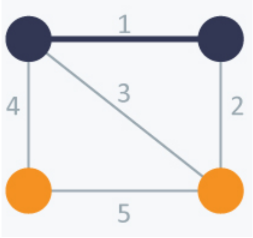
G(V,A)



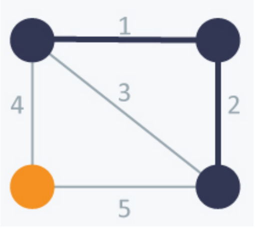
1. Comenzaremos con un nodo arbitrario (no importa cuál) y lo marcaremos.



2. En cada iteración marcaremos un nuevo vértice que sea contiguo al que ya hemos marcado (primero la arista con borde 1 y luego 2).

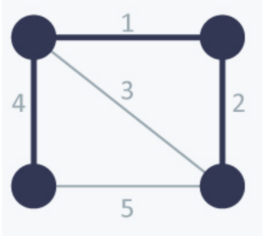


Costo mst= 1



Costo mst= 1+2

3. No elegimos el borde con peso 3 porque estaría formando un ciclo, en cambio, elegimos 4 por ser el de menor valor.



Costo mst= 1+2 + 4

Finalmente, terminamos con un **árbol de expansión mínimo**

Costo total del MST = 7

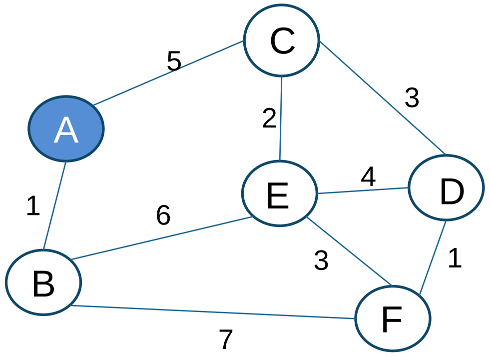
(suma de los costos de los bordes 1 + 2 + 4)

2. Algoritmos MST

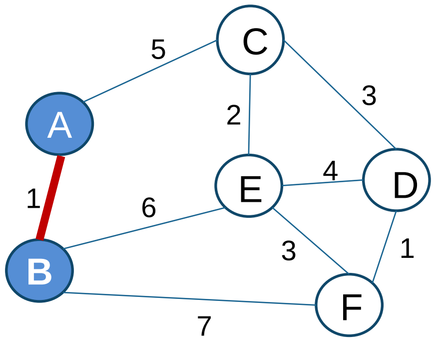
2.2. Algoritmo PRIM

Ejemplo # 2: Encontrar el árbol de expansión mínima por el algoritmo de Prim.

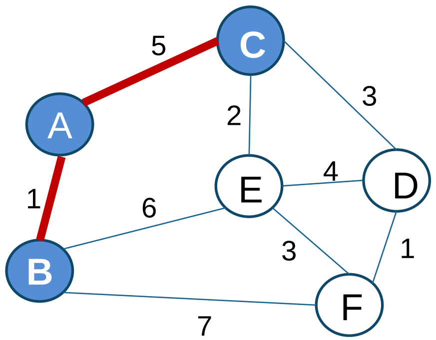
G(V,A)



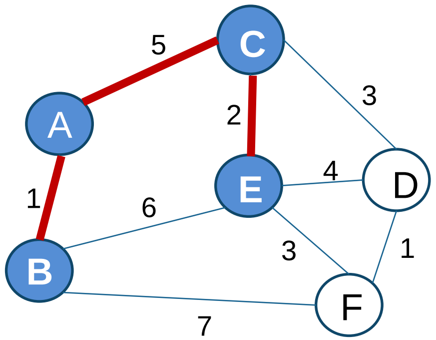
Aleatoriamente, elegimos el nodo A para iniciar.



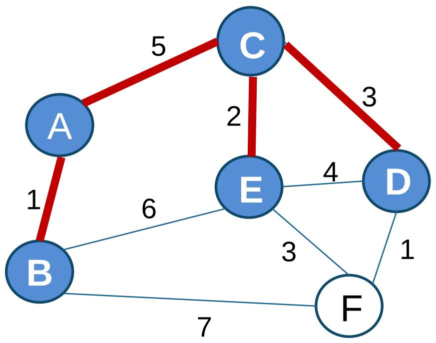
(1) A-B = 1



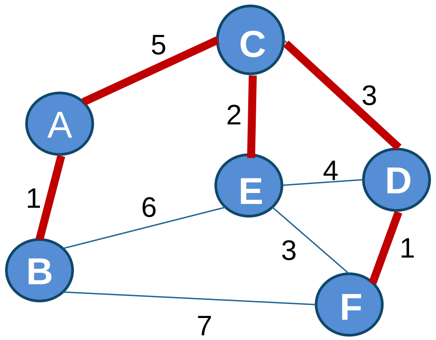
(2) A-C = 5 (porque de B-E y B-F el costo > 5)



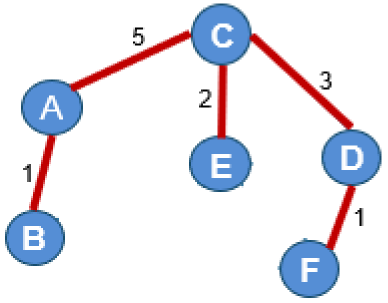
(3) C-E = 2



(4) C-D = 3



(5) D-F = 1



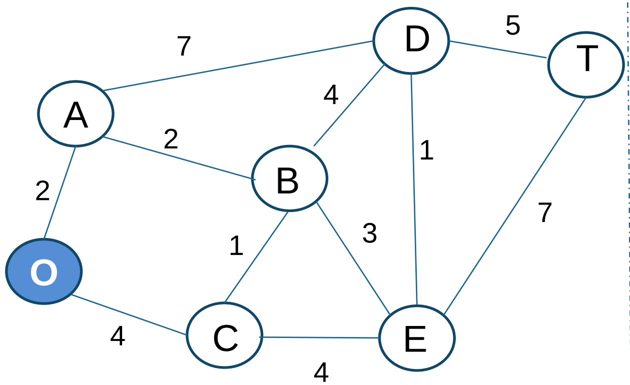
Costo total del MST = 12

2. Algoritmos MST

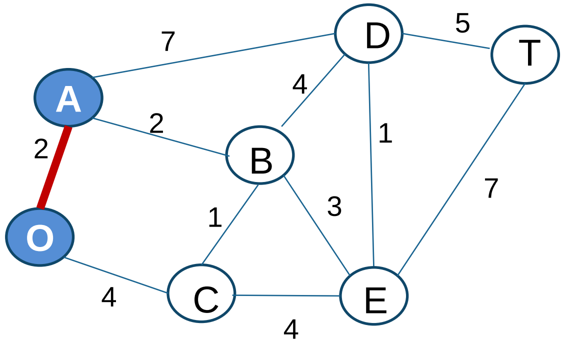
2.2. Algoritmo PRIM

Ejemplo # 3: Encontrar el árbol de expansión mínima por el algoritmo de Prim.

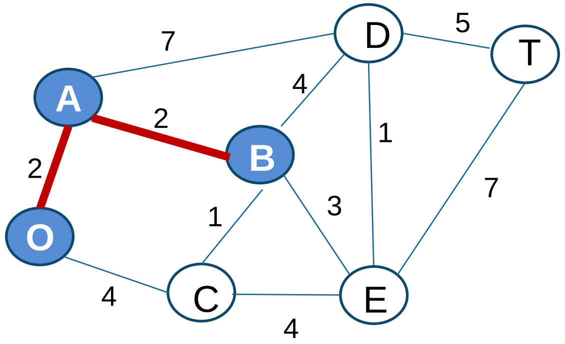
G(V,A)



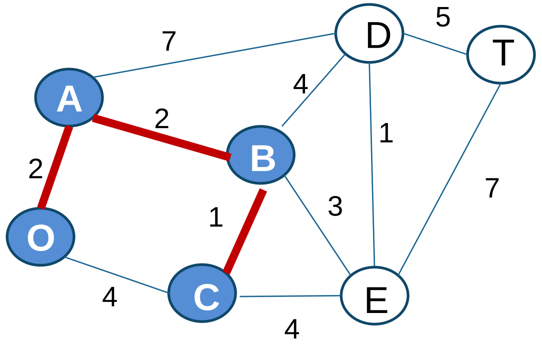
Aleatoriamente, elegimos el nodo “O” para iniciar.



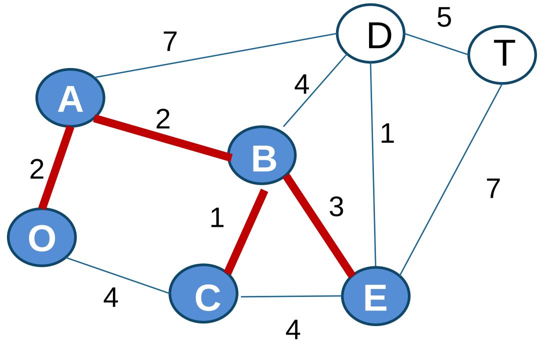
(1) O-A = 2



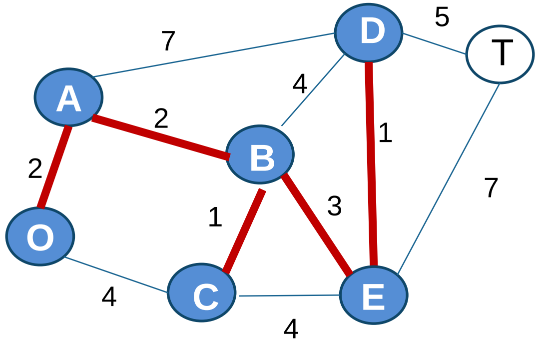
(2) A-B = 2



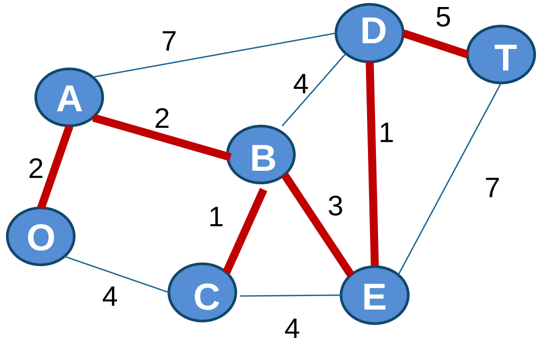
(3) B-C = 1



(4) B-E = 3



(5) E-D = 1



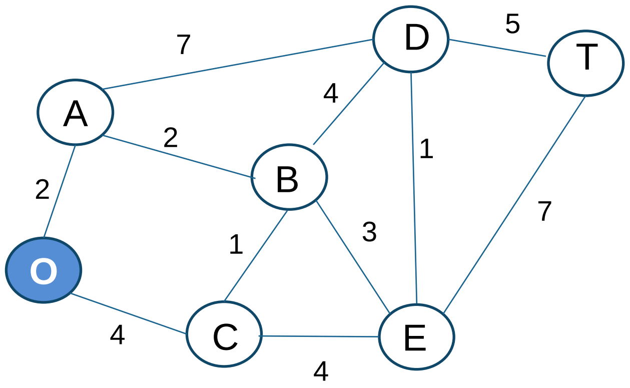
(6) D-T = 5

2. Algoritmos MST

2.2. Algoritmo PRIM

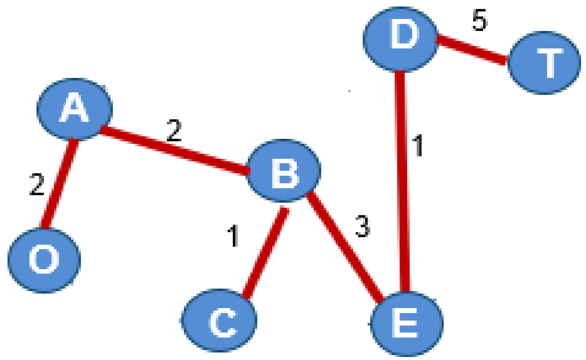
Ejemplo # 3: Encontrar el <u>árbol de expansión mínima</u> por el algoritmo de <u>Prim.</u>

G(V,A)



Finalmente, terminamos con un **árbol de expansión mínimo**

O-A = 2
A-C = 5
B-C = 1
B-E = 3
E-D = 1
D-T = 5



Costo total del MST = 17
(2+5+1+3+1+5)

2. Algoritmos MST

2.2. Algoritmo PRIM

Implementación

1. Empezar en un vértice cualquiera v. El árbol consta inicialmente sólo del nodo v.
2. En el resto de vértices, buscar el que esté más próximo a v (es decir, con la arista (v, w) de coste mínimo). Añadir w y la arista (v, w) al árbol.
3. Buscar el vértice más próximo a cualquiera de estos dos. Añadir ese vértice y la arista al árbol de expansión.
4. Repetir sucesivamente hasta añadir los n vértices.

- El árbol T aumenta un vértice cada vez.
- El array d[v] contiene el menor costo de la arista que conecta v con el árbol.
- Tiene una complejidad O(n2).

Pseudocodigo

```
T = ∅;
U = { 1 };
while (U ≠ V)
    let (u, v) be the lowest cost edge such that u ∈ U and v ∈ V - U;
    T = T ∪ {(u, v)}
    U = U ∪ {v}
```

- La solución se construye poco a poco, empezando con una solución "vacía".
- Implícitamente, el algoritmo maneja los conjuntos:
  - **V:** Vértices del grafo.
  - **U:** Vértices añadidos a la solución.
  - **V-U:** Vértices que quedan por añadir.

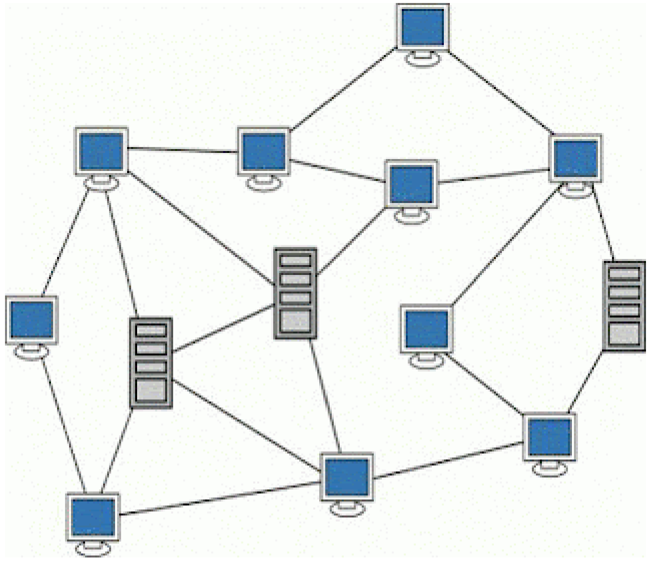
3. Aplicaciones de MST

- Varios algoritmos de búsqueda de rutas, entre ellos, el **algoritmo de Dijkstra** y el **algoritmo de búsqueda A***, construyen internamente un árbol de expansión como paso intermedio para resolver el problema.

- Las personas a menudo utilizamos algoritmos que construyen gradualmente un árbol de expansión (o muchos árboles similares) como pasos intermedios en el proceso de encontrar el árbol de expansión mínimo,

Principales aplicaciones

- Minimizar:
  - ➢ El costo de las redes eléctricas (para la transmisión de energía eléctrica de alto voltaje).
  - ➢ Las conexiones de cableado de equipos eléctricos.
  - ➢ Las conexiones de tuberías para conectar diferentes localidades.
  - ➢ El costo total de los trayectos en las redes de transporte.
  - ➢ Las conexiones de cableado en las redes de telecomunicaciones.
  - ➢ Las rutas terrestres, para conectar un grupo de computadoras en una red cableada que se encuentran a distancias variables.
  - ➢ Las rutas aéreas. Siendo los vértices del grafo las ciudades y los bordes las rutas entre las ciudades, cuanto más se tenga que viajar, más costará.



PREGUNTAS

Dudas y opiniones