

# MST y algoritmo de Prim

Clase 21

IIC 2133 - Sección 1

Prof. Sebastián Buggedo

# Sumario

**Obertura**

MST

Algoritmo de Prim

Epílogo

# Conectividad digital

Consideremos el problema de mejorar la conectividad digital de la Región del Maule

- Objetivo: instalar fibra óptica subterránea entre pares de puntos relevantes
- Cada instalación de ese cableado tiene un costo
- Es prioritario conectar ciudades más pobladas

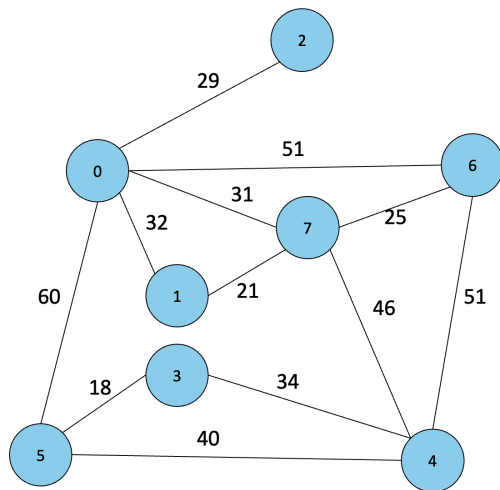
El desafío es **cubrir** con el menor costo

¿Qué tipo de grafo es más adecuado para representar el problema?

# Conectividad digital



# Conectividad digital



Usamos un grafo **no dirigido con costos**

# Conectividad digital

Usamos un grafo no dirigido con costos

- Cada nodo es un punto a conectar (ciudad, pueblo, etc)
- Cada arista tiene un costo de conexión
- Consideramos que cada nodo presente en el grafo, debe ser conectado

Para resolver el problema seleccionamos un subconjunto de aristas

- La suma de los costos debe ser **mínima**
- El **subgrafo** que solo considera esas aristas, debe ser **conexo**

¿Qué forma tiene el **subgrafo** que buscamos?

# Sumario

Obertura

**MST**

Algoritmo de Prim

Epílogo

# Árboles de cobertura mínimos

## Definición

Dado un grafo no dirigido  $G$ , un subgrafo  $T \subseteq G$  se dice un **árbol de cobertura mínimo** o **MST** de  $G$  si

1.  $T$  es un árbol
2.  $V(T) = V(G)$
3. No existe otro MST  $T'$  para  $G$  con menor costo total

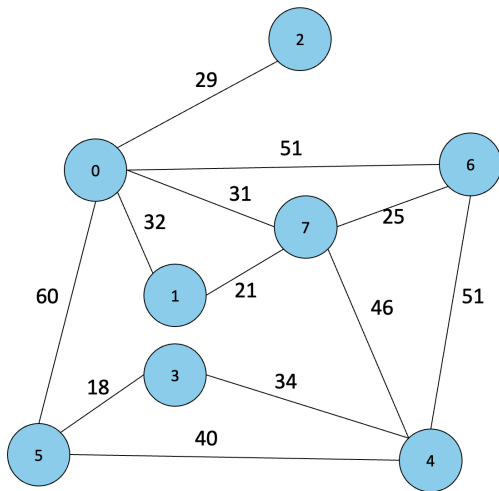
Es decir, si  $T$  es MST de  $G$ ,

1. no tiene ciclos
2. es una **cobertura** de los nodos de  $G$
3. tiene costo mínimo

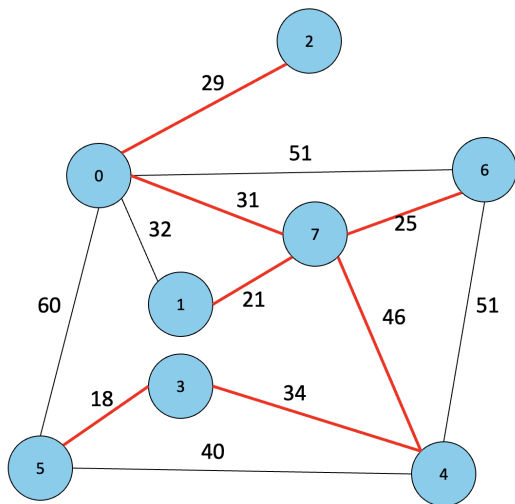
¿Puede haber más de un MST diferente para  $G$ ?



# Árboles de cobertura mínimos



# Árboles de cobertura mínimos



# Árboles de cobertura mínimos

Los MST están presentes en la solución de múltiples problemas de conectividad

- Redes de distribución eléctrica
- Redes telefónicas
- Comunicaciones, computacionales, tráfico aéreo. . .
- Incluso redes biológicas, químicas y físicas

# Árboles de cobertura mínimos

Para atacar el problema algorítmicamente, dado  $G$  no dirigido

- Llamamos **corte** a una **partición**  $(V_1, V_2)$  de  $V(G)$

$$V_1, V_2 \neq \emptyset, \quad V_1 \cup V_2 = V(G), \quad V_1 \cap V_2 = \emptyset$$

- Diremos que una arista **cruza el corte** si uno de sus extremos está en  $V_1$  y el otro en  $V_2$

¿Qué podemos afirmar respecto a los MST y las aristas que cruzan un corte dado?

# Árboles de cobertura mínimos

Si  $T$  es un MST de  $G$ , y  $P = (V_1, V_2)$  es un corte de  $G$

- Sabemos que al menos una arista que cruza  $P$  está incluida en  $T$
- De lo contrario, no habría camino entre nodos de  $V_1$  y  $V_2$
- Por lo tanto:  $T$  no sería de cobertura

Si  $P = (V_1, V_2)$  es un corte de  $G$

- Sabemos que cada arista de corte tiene un costo asociado
- La arista más barata **siempre** se incluye en **algún** MST

¿Cómo podemos usar estos hechos para construir un MST desde cero?

# Sumario

Obertura

MST

**Algoritmo de Prim**

Epílogo

# Algoritmo de Prim

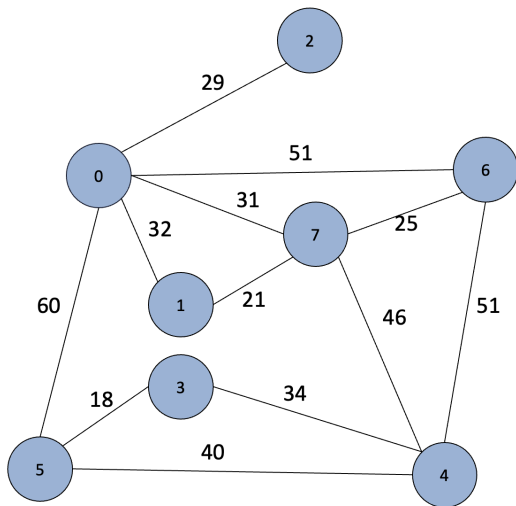
La idea detrás del **algoritmo de Prim** es utilizar las aristas que cruzan cortes para guiar la construcción

Para un grafo  $G = (V, E)$  y un nodo inicial  $v$

1. Sean  $R = \{v\}$  y  $\bar{R} = V - R$
2. Sea  $e$  la arista de menor costo que cruza de  $R$  a  $\bar{R}$
3. Sea  $u$  el nodo de  $e$  que pertenece a  $\bar{R}$
4. Agregar  $e$  al MST. Eliminar  $u$  de  $\bar{R}$  y agregarlo a  $R$
5. Si quedan elementos en  $\bar{R}$ , volver al paso 2.

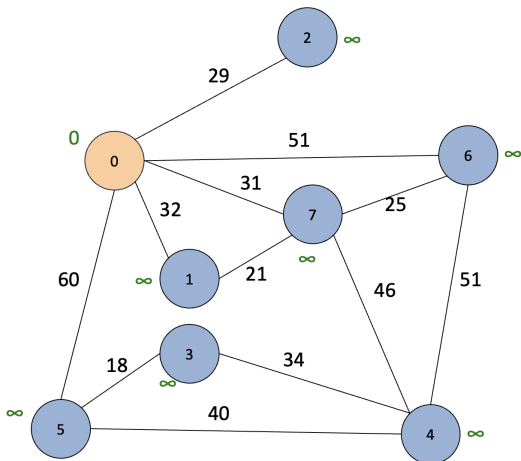
¿Cómo hacer eficiente el paso 2?

## Árboles de cobertura mínimos

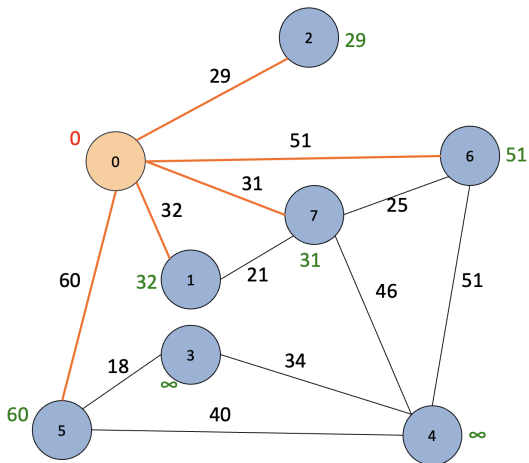




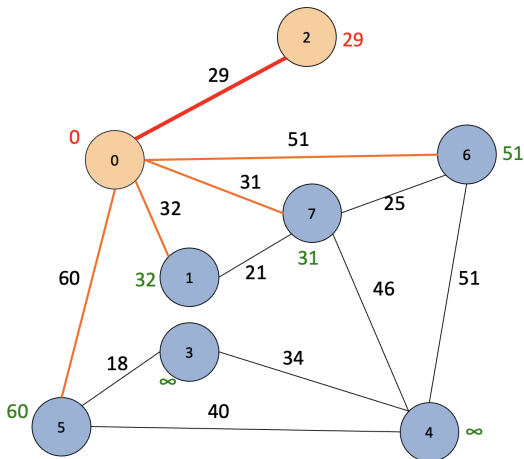
# Árboles de cobertura mínimos



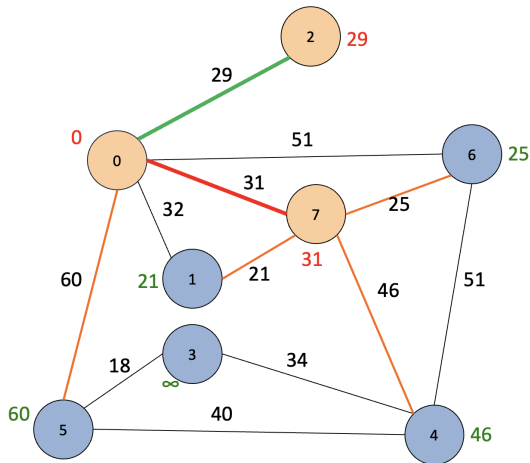
# Árboles de cobertura mínimos



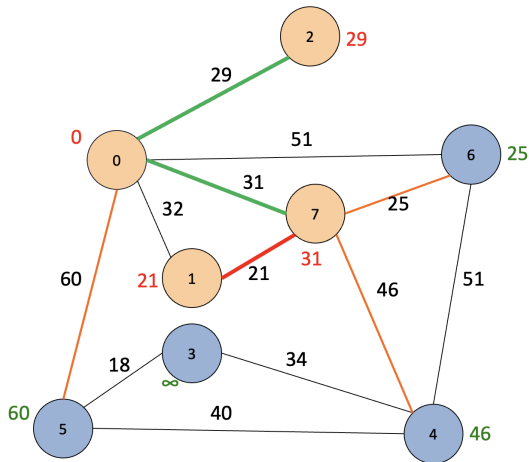
# Árboles de cobertura mínimos



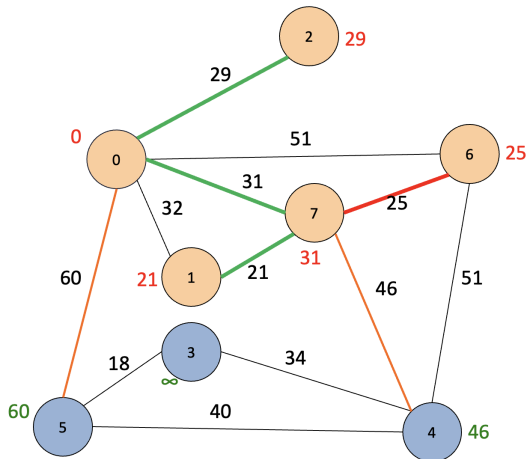
# Árboles de cobertura mínimos



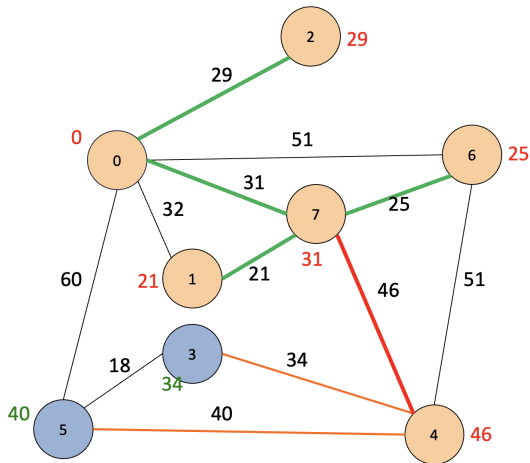
# Árboles de cobertura mínimos



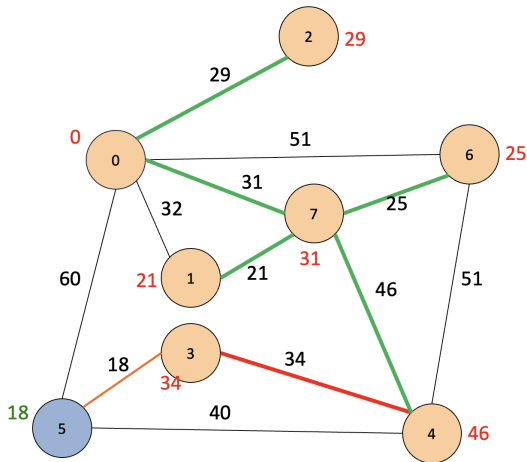
# Árboles de cobertura mínimos



# Árboles de cobertura mínimos

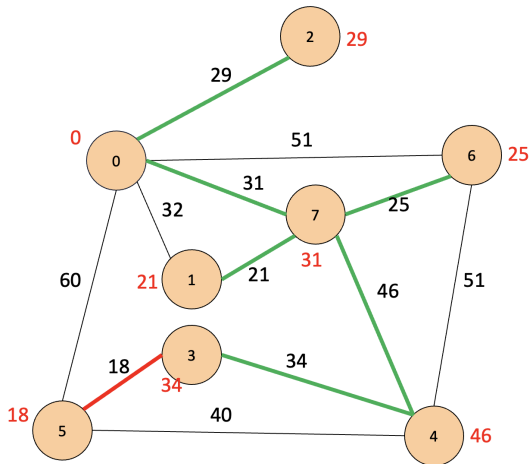


# Árboles de cobertura mínimos

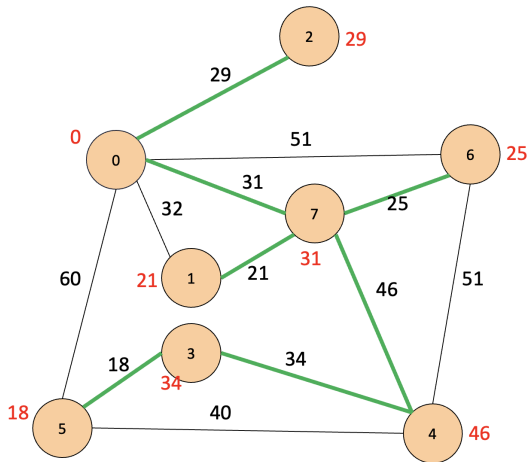




# Árboles de cobertura mínimos



# Árboles de cobertura mínimos



# Algoritmo de Prim

Prim( $s$ ):

```
1   $Q \leftarrow$  cola de prioridades vacía
2   $T \leftarrow$  lista vacía
3  for  $u \in V - \{s\}$  :
4       $d[u] \leftarrow \infty$ ;  $\pi[u] \leftarrow \emptyset$ ; Insert( $Q, u$ )
5   $d[s] \leftarrow 0$ ;  $\pi[s] \leftarrow \emptyset$ ; Insert( $Q, s$ )
6  while  $Q$  no está vacía :
7       $u \leftarrow$  Extract( $Q$ )
8       $T \leftarrow T \cup \{(\pi[u], u)\}$ 
9      for  $v \in \alpha[u]$  :
10         if  $v \in Q$  :
11             if  $d[v] > cost(u, v)$  :
12                  $d[v] \leftarrow cost(u, v)$ 
13                  $\pi[v] \leftarrow u$ 
14  return  $T$ 
```

Importante: la cola de prioridades usa  $d[v]$  como prioridad de  $v$

## Demostración

**Finitud.** Mismo argumento que en Dijkstra.

**Correctitud.** Fijaremos nuestra atención en la línea 8 del algoritmo, justo luego de agregar a  $T$  una nueva arista. Denotaremos por  $G_n$  al subgrafo de  $G$  tal que considera solo los primeros  $n$  nodos extraídos de  $Q$  con todas sus aristas de  $G$ .

Probaremos por inducción sobre el número de iteraciones la propiedad

$P(n) :=$  En la iteración  $n$ -ésima, la línea 8 guarda en  $T$  un MST para el subgrafo  $G_n$

## Demostración

**C.B.** Para  $n = 1$ , extraemos de  $Q$  el nodo inicial y agregamos una arista *dummy*. Como el grafo que debemos cubrir es  $G_1 = (\{s\}, \emptyset)$  y  $T$  no cubre nada más que el nodo inicial, es efectivamente un MST para  $G_1$ .

**H.I.** Suponemos que en la iteración  $n$ ,  $T$  tiene guardado un MST para cubrir el subgrafo  $G_n$  con costo mínimo.

**T.I.** Sea  $u$  el  $(n + 1)$ -ésimo nodo extraído de la cola  $Q$ . Sea además  $T$  la variable guardado en la iteración anterior.

- Como  $u$  fue extraído en la iteración  $n + 1$ , significa que es el nodo con la arista más barata para conectar directamente con **algún nodo** de  $G_n$ .
- A saber, dicho nodo es  $\pi[u]$ .
- Al agregar la arista  $(\pi[u], u)$  a  $T$ , obtenemos un nuevo conjunto  $T'$  de  $G_{n+1}$ . Basta argumentar sus propiedades.

## Demostración

### $T'$ es **cubrimiento**

- Como  $T$  es MST para  $G_n$  por **H.I.**, entonces conecta todos los nodos de  $G_n$  (incluyendo a  $\pi[u]$ ).
- Al agregar  $(\pi[u], u)$ , se conecta también a  $u$ . Por lo tanto,  $T'$  cubre todo  $G_{n+1}$

### $T'$ es **árbol**

- Por **H.I.**,  $T$  es árbol.
- Al agregar  $(\pi[u], u)$ , se conecta un nodo ya incluído en  $G_n$  con uno **no incluído**, de forma que no se producen ciclos y  $T'$  no es árbol

### $T'$ es de **costo mínimo**

- Por **H.I.**,  $T$  es de costo mínimo para  $G_n$ .
- La elección de  $u$  asegura que se agrega la arista más barata para conectar  $u$  a  $G_n$ . Se concluye que  $T'$  es de costo mínimo.

## Demostración

De lo anterior se concluye que  $P(n)$  es cierta. En particular, como  $G_{|V|} = G$

$P(|V|)$  verdadera  $\Leftrightarrow$  Prim entrega MST para  $G$



No olvidar: no necesariamente hay un único MST para  $G$

# Complejidad

Considerando que usamos una cola de prioridad implementada como heap

- Extracción del más prioritario  $\mathcal{O}(\log(V))$
- Actualización de prioridad cuando cambia el costo  $\mathcal{O}(\log(V))$

La complejidad en el peor caso viene dada por las iteraciones del **while**

- El loop ocurre  $|V|$  veces  $\mathcal{O}(V)$
- En cada loop se extrae un nodo  $\mathcal{O}(\log(V))$
- Cada arista se revisa exactamente una vez  $\mathcal{O}(E)$
- Por cada revisión se hace una actualización de costo  $\mathcal{O}(\log(V))$
- Total  $\mathcal{O}((V + E) \log(V))$

Podemos simplificar este último resultado



# Complejidad

Recordemos que para grafos conexos existe una relación entre  $|V|$  y  $|E|$

- Si  $G$  es un árbol,  $E = V - 1$
- Si  $G$  es conexo cualquiera,  $E \geq V - 1$

Concluimos que  $V \in \mathcal{O}(E)$  para grafos conexos

- Luego,  $(V + E) \in \mathcal{O}(E)$

El algoritmo de Prim toma tiempo  $\mathcal{O}(E \log(V))$

# Algoritmo de Prim: una versión más concreta

Prim( $s$ ):

```
1    $Q \leftarrow$  cola de prioridades con  $s$ 
2    $T \leftarrow$  lista vacía
3    $x.key \leftarrow 0$ ;  $x.parent \leftarrow \emptyset$ 
4   while  $Q$  no está vacía :
5        $u \leftarrow \text{Extract}(Q)$ ;  $u.color \leftarrow$  negro
6       if  $u.parent \neq \emptyset$  :
7            $T \leftarrow T \cup \{(u.parent, u)\}$ 
8       for  $v \in \alpha[u] \wedge v.color \neq \text{negro}$  :
9           if  $v \in Q$  :
10               $\text{Insert}(Q, v)$ 
11           if  $v.key > \text{cost}(u, v)$  :
12               $v.key \leftarrow \text{cost}(u, v)$ 
13               $v.parent \leftarrow u$ 
14   return  $T$ 
```

Suponemos que inicialmente  $v.key \leftarrow \infty$  para todo  $v$

# Sumario

Obertura

MST

Algoritmo de Prim

**Epílogo**

# Epílogo

Vea

**www.menti.com**

Introduce el código

**5551 1581**



O usa el código QR