Tres problemas en grafos *con costos*

a) Grafos direccionales:

- encontrar la *ruta más corta desde un vértice a todos los otros* —el algoritmo de Dijkstra

b) Grafos no direccionales:

- **encontrar el *árbol de cobertura de costo mínimo*, o *minimum spanning tree* (MST)**

c) Grafos direccionales:

- encontrar las *rutas más cortas entre todos los pares de vértices*

Mejorando la conectividad digital

- El gobernador de la Región del Maule ha decidido mejorar significativamente la conectividad digital de la región
- La idea es conectar mediante fibra óptica subterránea varios pares de puntos relevantes de la región —cada conexión tiene un costo
- Sólo que hay demasiados puntos relevantes como para hacerlo todo de una vez
- Lo prioritario es conectar las ciudades más pobladas, que tienen escuelas, universidades, hospitales, compañías de bomberos, supermercados, etc.
- Aprovechamos que si *A* está conectado con *B* y *B* con *C*, entonces automática-mente *A* está conectado con *C*

¿Cuál es la **forma más barata** de hacer esto?


VI REGION DEL LIB. GRAL. BERNARDO O'HIGGINS
VII REGION DEL MAULE
VII REGION DEL MAULE
VIII REGION DEL BIOBIO
Océano Pacífico
Mar Chileno
ARGENTINA
CHILE
Bucalemu
Boyeruca
Lo Valdivia
Lolol
El Guaico
Chépica
Convento Viejo
Auquinco
Quinta
Codegua
Morza
Huemul
Sierra Bellavista
La Rufina
Vn Tinguiririca 4300
Trompetilla
Termas del Flaco
Río Claro
Río Las Damas
Co Cayetano 2886
RESERVA NACIONAL LAGUNA TORCA
Llico
Lago Vichuquén
San Pedro de Alcántara
Ranguil
Est Las Palmas
Comalle
Teno
Rebeca
Río Teno
Rauco
La Montaña
El Manzano
Culenar
Romeral
Los Queñes
CURICO
Potrero Grande Chico
Río Teno
Cerro Torre de Santa Elena 3820
Las Leñas
Los Molles
60 Kms a Malargüe
Lipimávida
Pichibudi
Vichuquén
Pta Duao
Est Nilahue
Trincao
Hualañé
Iloca
Licantén
Palquibudi
Sagrada Familia
Lontué
Los Niches
Upeo
Monte Oscuro
El Planchón
Paso de Vergara 2502
Laguna Teno
Río Lontué
La Lora
Curepto
Villa Prat
Itahue
Molina
Potrero Grande
Quebrada Honda
Vn Peteroa 4151
Baños de Azufre
Río Claro
La Trinchera
Litú
Cumpeo
Culenar
Aduana
Paso de Potrerillos 2908
Río Colorado
Río Palos de San Pedro
Coipué
Gualleco
Botalcura
San Rafael
Astillero
Radal
RESERVA NACIONAL RADAL SIETE TAZAS
Putú
Junquillar
Quivolgo
Balseo
Batuco
Río Huenchullami
Pelarco
Río Lircay
Vn Descabezado Chico 3750
Vn Descabezado Grande 3830
Constitución
Pencahue
TALCA
Corinto
Curtiduría
Pichaman
Río Maule
Colín
Aurora
San Clemente
Corralones
Corel
Viches
RESERVA NACIONAL ALTOS DEL LIRCAY
Río Claro
Las Cañas
Cta Pellines
Nirivilo
Co Nirivilo 722
Maule
Duao
Río Maule
Hotel Casas El Colorado
Río Loanco
Río Purapel
San Javier
Mingre
Puente Pardo
Santa Ana
Río de la Invernada
Laguna La Invernada
Empedrado
Huerta del Maule
Villa Alegre
Colbún
Posada Paso Nevado
Armerillo
BARRERA
Lago Colbún
Lago Machicura
Paso Trolón 3044
Pahuil
Río Reloca
Las Campanas
Yerbas Buenas
Panimávida
Rari
Baños de La Monja
Quinamávida
Los Rabones
RESERVA NACIONAL LOS BELLOTOS
Las Garzas
Endesa
La Mina
115
Río Puelche
Melozal
Arbolillo
Palmilla
LINARES
Melado
Túnel Canal Melado
Robleria
El Salto
Paso Pehuenche o del Maule 2553
RESERVA NACIONAL FEDERICO ALBERT
La Aguada
Chanco
Tapihue
Río Ancoa
Campamento Ancoa
Co El Toro 3085
Vn San Pedro 3821
Pueblo Hundido
Villa Seca
Miraflores
Pelluhue
RESERVA NACIONAL LOS RUILES
Llepo
Mesamávida
Río Melado
Laguna del Maule
Laguna Negra
Curanipe
Los Nabos
Los Perales
Longaví
Melado
Aduana Pejerrey
Río d e la Puente
Pya Peuño
Los Cristales
CAUQUENES
Paso Hondo
Retiro
El Tránsito
Co El Buitre 2955
Laguna Fea
Pta Nugurue
Tregualemu
RESERVA NACIONAL LOS QUEULES
Hualve
Unicavén
Quella
Río Achibueno
Nvdo Longaví 3242
SIERRA DE CHOHICO
Río Barrancas
Pullay
Coronel
El Peñón
Parral
Ajial
Quinchimávida
Río Longaví
Laguna Vervarco Campos
Cancha Alegre
Tres Esquinas
Villa Rosa
San Pablo
51 Kms a Quirihue
60 Kms a Chillán
Buchupureo
El Parrón
Pocillas
Perquilauquén
Bullileo
Cobquecura
Emb Digua
Termas de Catillo
San Gregorio
Quirihue
Colmuyao
Torrecilla
San Carlos
Zemita
Lago Dial
Ninhue
Cachapoal
El Sauce
Trehuaco
Coipín
San Nicolás
BALSEO
Nahueltoro
San Fabián de Alicó
Vn Domuyo 4709
Río Perquilauquén
Los Sauces
Vegas de Itata

El problema es descrito por un **grafo** (conexo) **no direccional con costos**

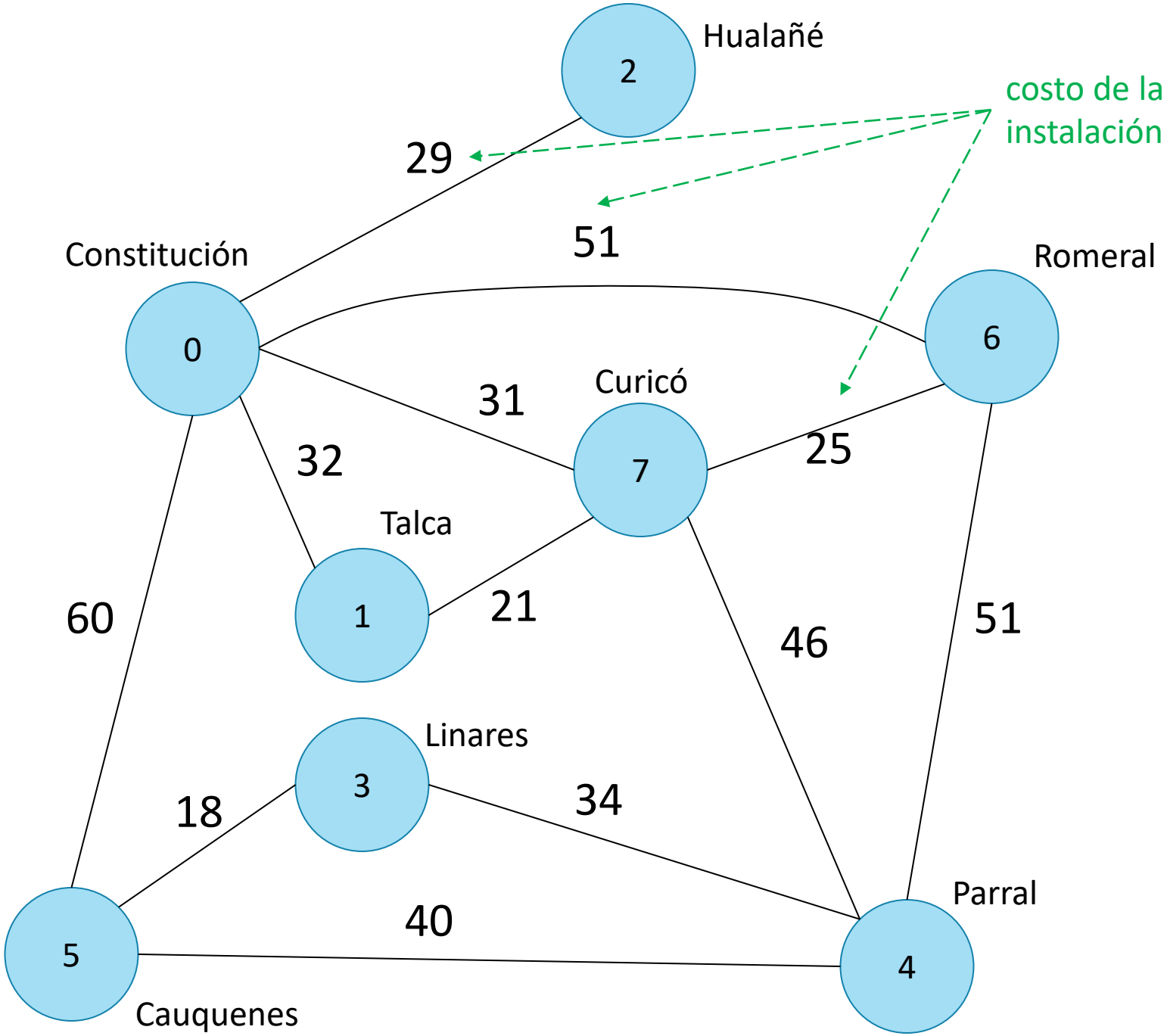
Las ciudades son los nodos, V

Las conexiones posibles entre pares de ciudades son las aristas, E

… y los costos de las conexiones son los costos de las aristas

El grafo es no direccional, ya que las conexiones de las que estamos hablando son no direccionales:

… si *u* está conectado con *v*, entonces *v* está conectado con *u*


Hualañé
2
costo de la instalación
29
51
Constitución
Romeral
0
6
31
Curicó
32
7
25
Talca
1
60
21
46
51
Linares
3
18
34
Parral
40
5
4
Cauquenes

La solución es un **subconjunto $T \subseteq E$** con las siguientes tres propiedades

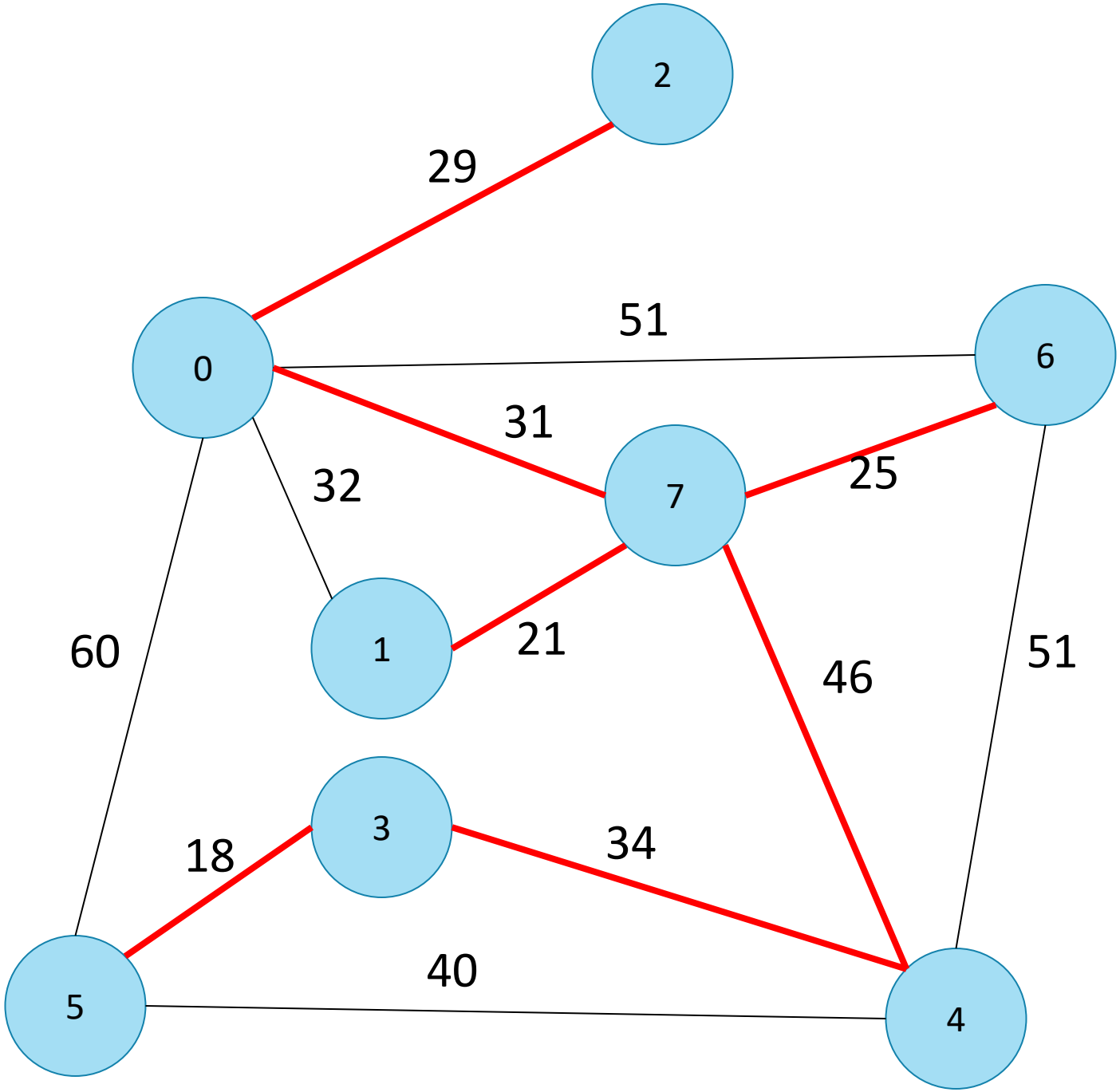
i) El grafo (V, T) es conexo → las aristas de T incluyen todos los vértices —forman una **cobertura**

ii) No hay otro subconjunto T que cumpla con i) y tenga menor **costo total** (la suma de los costos de las aristas de T) —es una cobertura de **costo mínimo**

iii) Las aristas de T no forman ciclos —forman un **árbol***

Es decir, **(V, T) es un árbol de cobertura de costo mínimo**, o ***MST***

* esta propiedad iii) de la solución se puede demostrar a partir de i) y ii)


2
29
0
51
6
31
32
7
25
1
21
60
46
51
3
18
34
5
40
4

MSTs

Originalmente, hace 100 años, redes de distribución de electricidad

Después, redes telefónicas

Hoy, redes de comunicación, eléctricas, hidráulicas, de computadores, de caminos, carreteras y autopistas, de tráfico aéreo

… incluso redes biológicas, químicas y físicas que se encuentran en la naturaleza

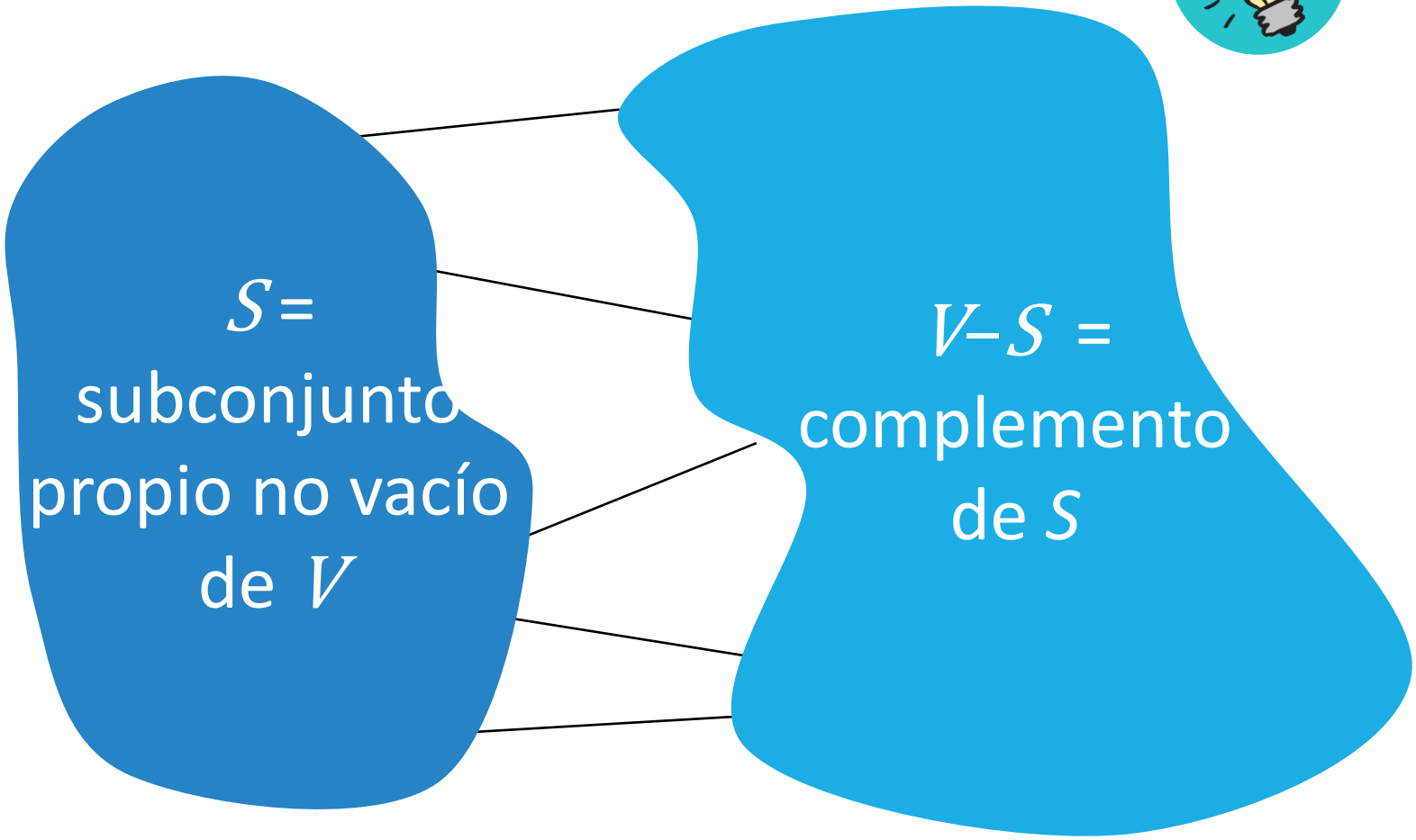
MSTs, **cortes** y aristas que **cruzan** el corte

Particionemos los vértices del grafo en dos (sub)conjuntos no vacíos, S y $V - S$ —la partición $(S, V - S)$ es un **corte** de V

Una arista **cruza** el corte si uno de sus extremos está en S y el otro en $V - S$

¿Qué podemos afirmar respecto a estas aristas y los **MST**?

El **corte** (S, V–S)
y las aristas que **cruzan** el corte



¿Cuál debería ser la siguiente arista a incluir?

Buscando un MST

Si para cada corte la arista de menor costo está en un **MST**

… ¿cómo podemos encontrar un **MST**?

… ¿podremos construirlo una arista a la vez?

Propiedades del MST

¿Hay alguna arista que **siempre** pertenezca a un **MST**?

¿Se cumple esto recursivamente? ¿En qué casos?

¿Podremos aprovecharlo en un algoritmo **codicioso**?

El algoritmo de Kruskal

$\boldsymbol{kruskal}\big(\boldsymbol{G}(\boldsymbol{V},\boldsymbol{E})\big)$:

ordenar $\boldsymbol{E}$ por costo, de menor a mayor

$\boldsymbol{T} \leftarrow \emptyset$

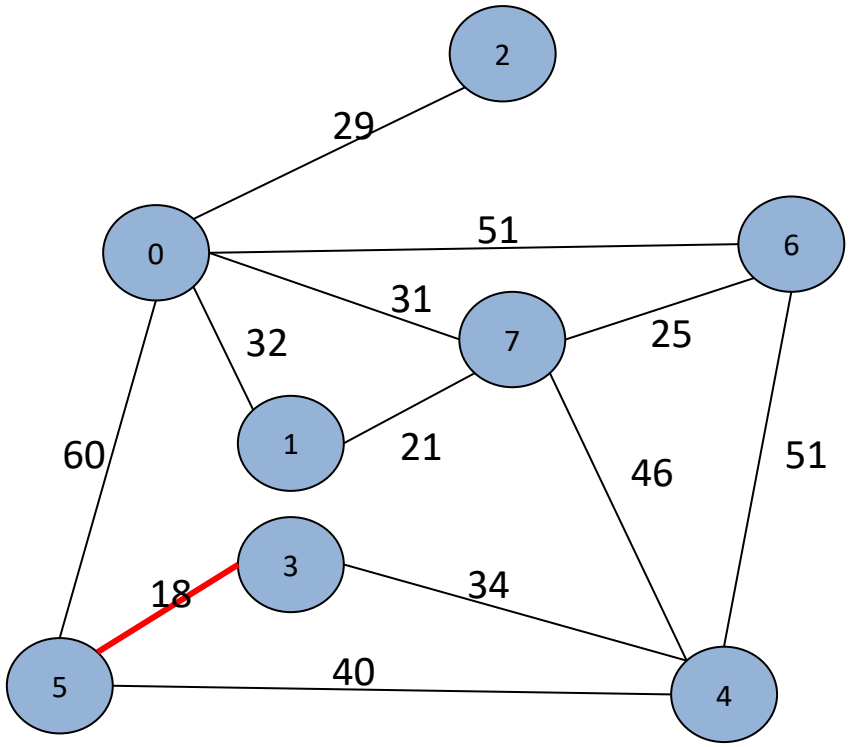
$\boldsymbol{foreach}\ \boldsymbol{e} \in \boldsymbol{E}$, en el orden obtenido arriba:

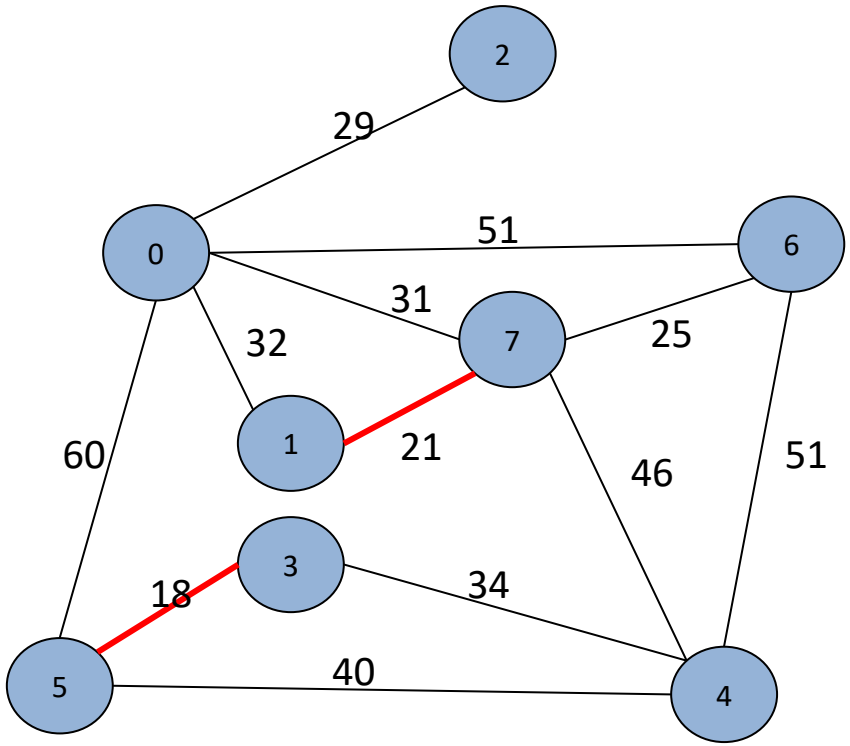
$\boldsymbol{if}$ agregar $\boldsymbol{e}$ a $\boldsymbol{T}$ no forma un ciclo:

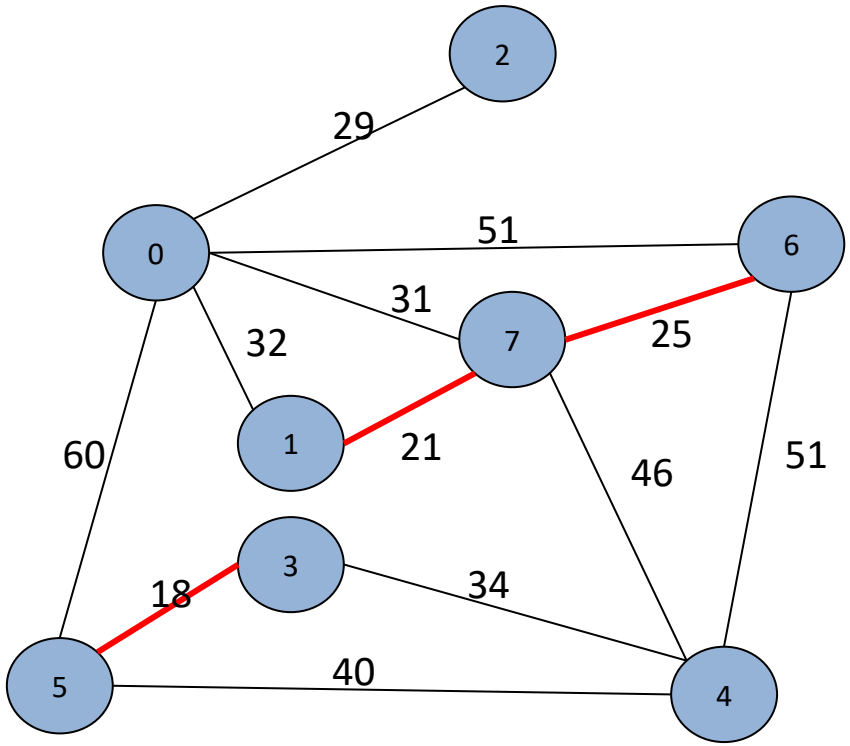
agregar $\boldsymbol{e}$ a $\boldsymbol{T}$

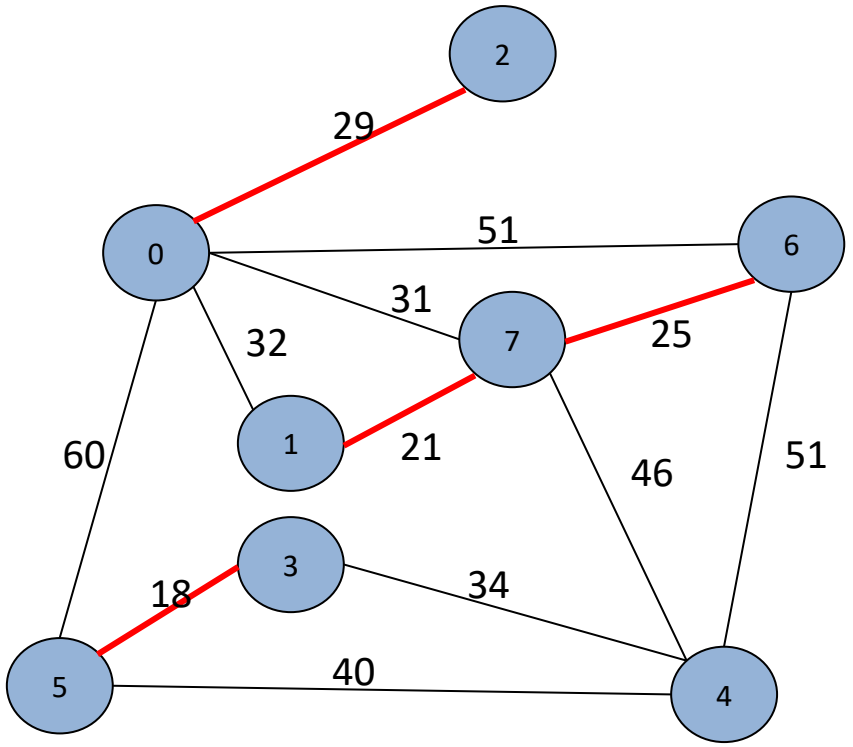
$\boldsymbol{return}\ \boldsymbol{T}$

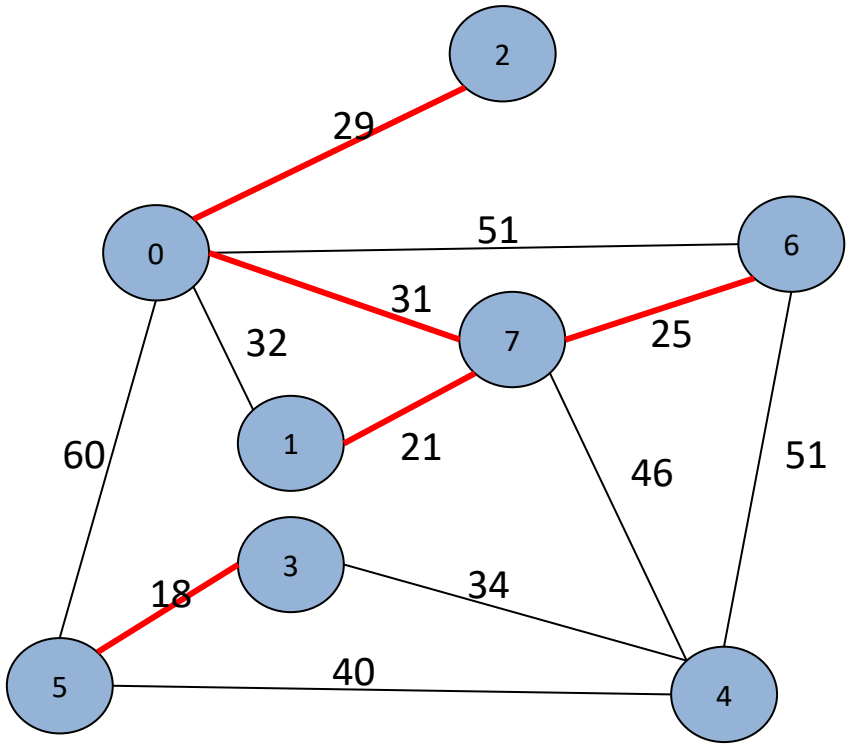
`kruskal` en acción: luego de ordenar las aristas no decrecientemente por costo, la arista de menor costo es (3,5): $\omega(3,5) = 18$



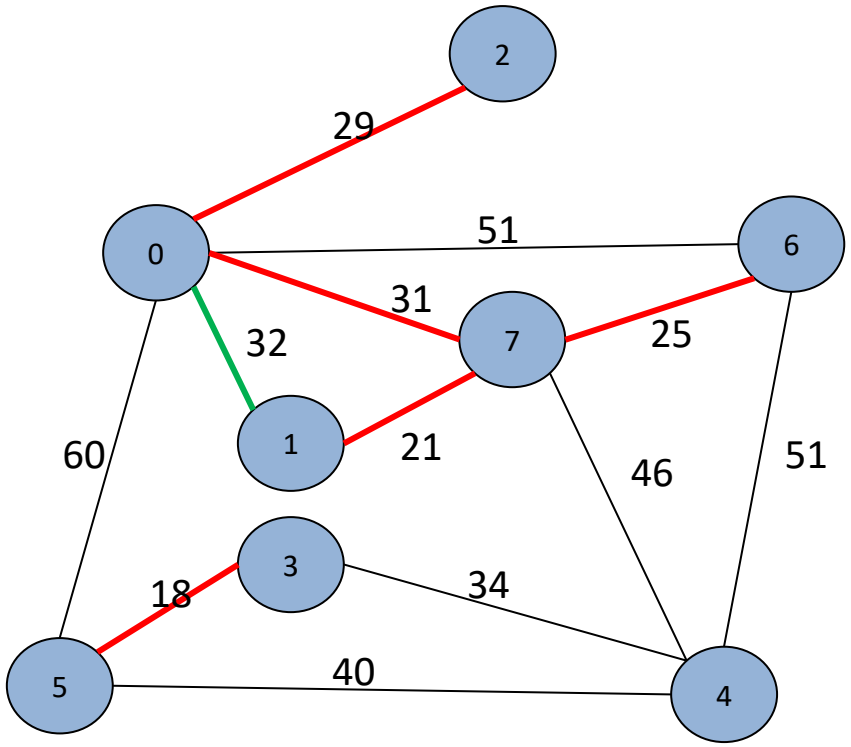

2
29
0
51
6
31
7
25
32
1
21
60
46
51
3
18
34
5
40
4


2
29
0
51
6
31
7
25
32
1
21
60
46
51
3
18
34
5
40
4


2
29
0
51
6
31
7
25
32
1
21
60
46
51
3
18
34
5
40
4

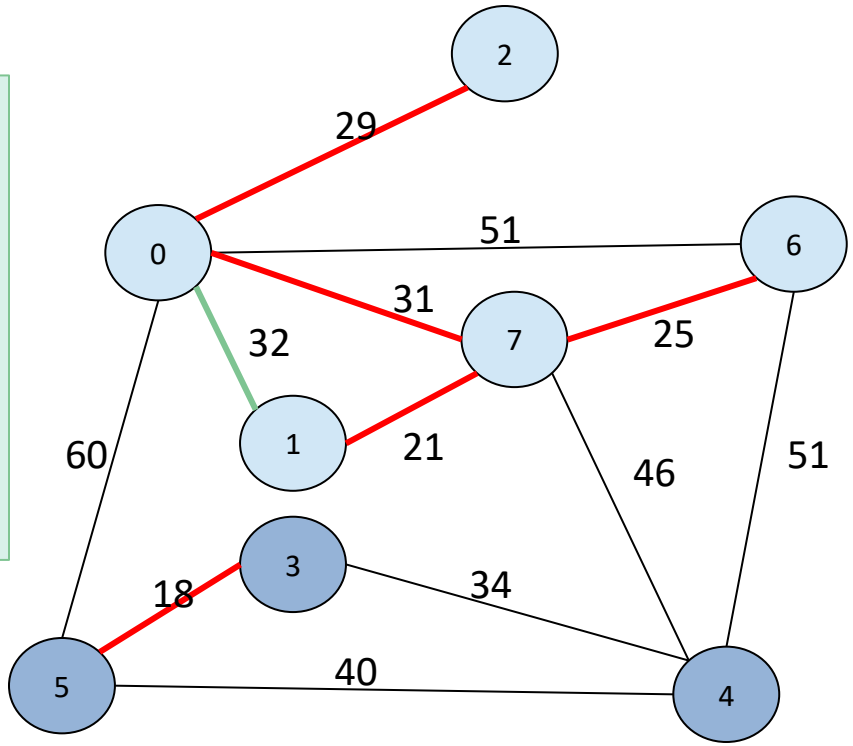

2
29
0
51
6
31
7
25
32
1
21
60
46
51
3
18
34
5
40
4

La arista (0, 1) es la próxima arista que debemos considerar, con costo $\omega(0, 1) = 32$, pero *la descartamos* ya que forma un ciclo con las aristas (1, 7) y (7, 0); y por lo tanto no contribuye a conectar entre sí nodos que aún no estén conectados

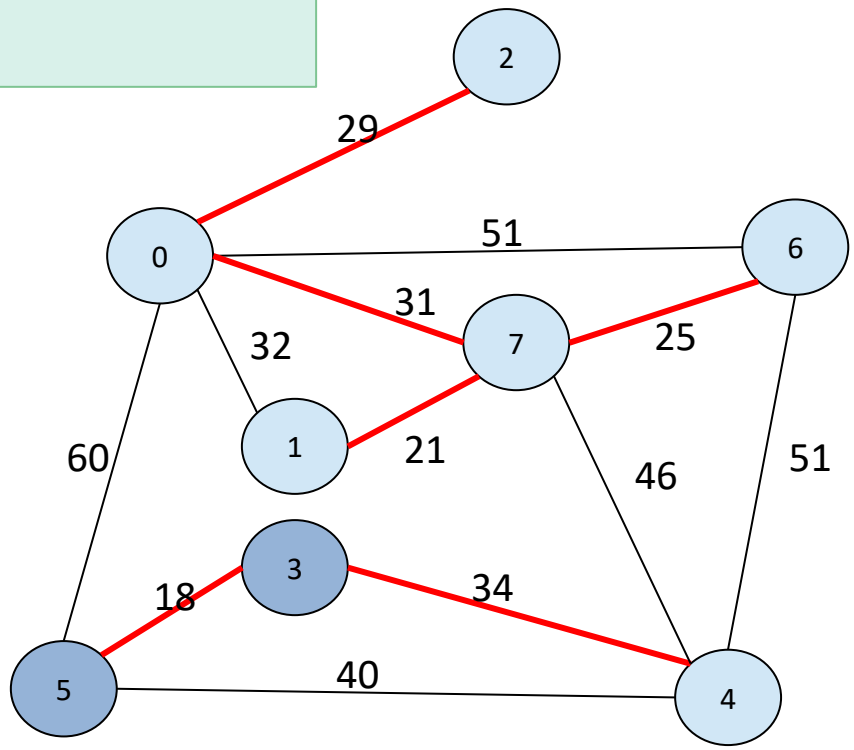


A propósito, ¿cuál es en este caso el *corte* del que hablamos en la diap. 9? En cada paso, al considerar la próxima arista de menor costo, (*u*, *v*), el corte queda formado por el subconjunto de nodos conectados a *u* mediante aristas que ya están en el MST (las aristas rojas), y por su complemento; o, alternativamente, por el subconjunto de nodos conectados a *v* mediante aristas que ya están en el MST, y por su complemento.

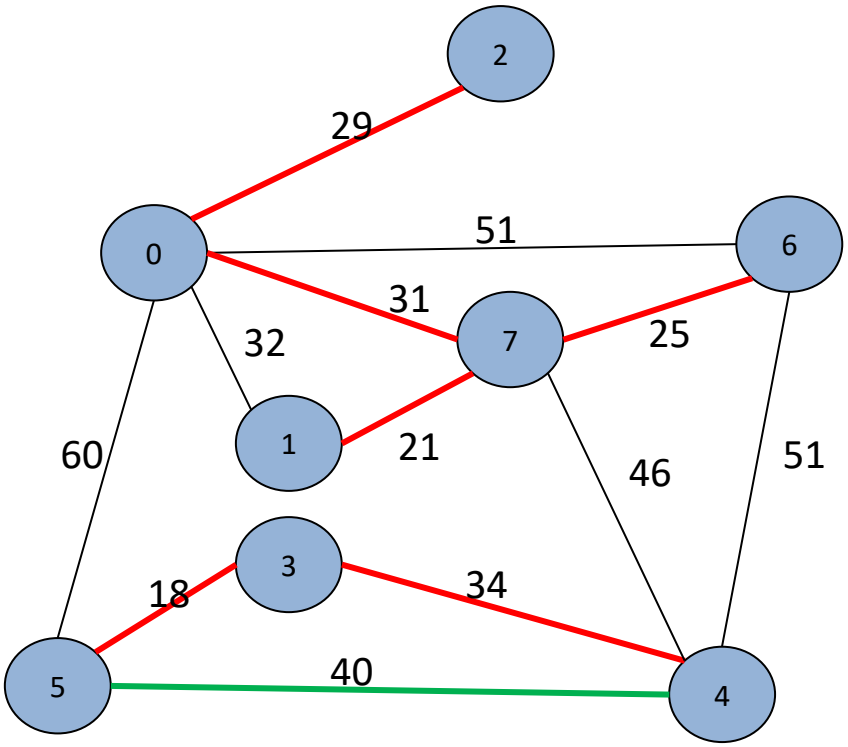
Así, al considerar la arista (0, 1), el corte es ({**0**, 2, 7, **1**, 6}, {3, 4, 5}), y vemos que ambos nodos 0 y 1 quedan en el mismo subconjunto, por lo que la arista (0, 1) **no cruza** el corte



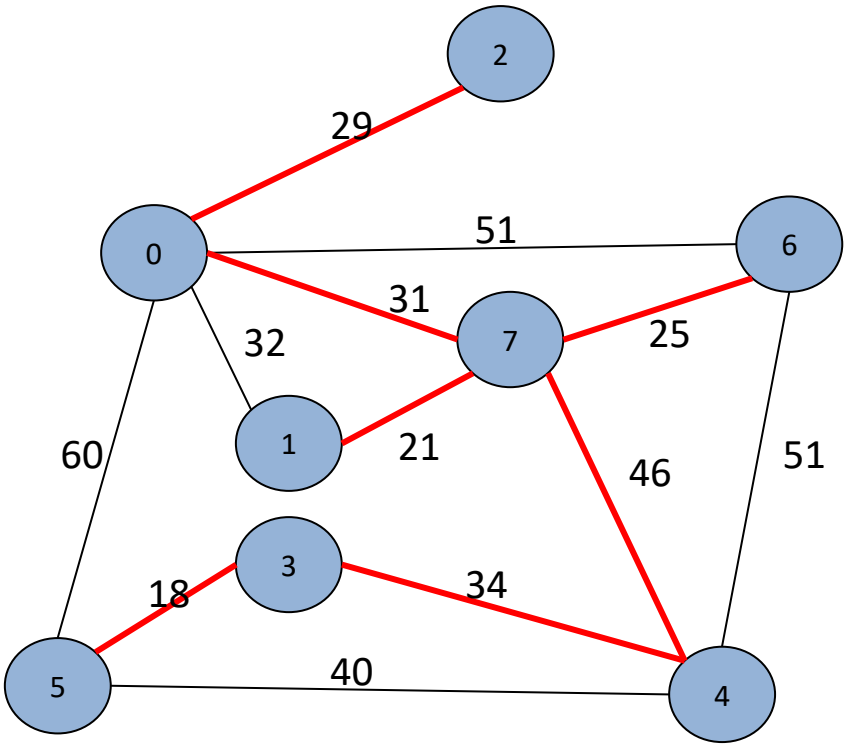
Ahora la arista considerada es la arista (3, 4), de costo $\omega(3, 4) = 34$; y el corte puede ser ({**3**, 5}, {0, 1, 2, **4**, 6, 7}), o bien ({**4**}, {0, 1, 2, **3**, 5, 6, 7}).
En ambos casos, la arista (3, 4) **cruza** el corte, es decir, no forma un ciclo con las aristas (rojas) que ya son parte del MST, y por lo tanto la agregamos a la solución



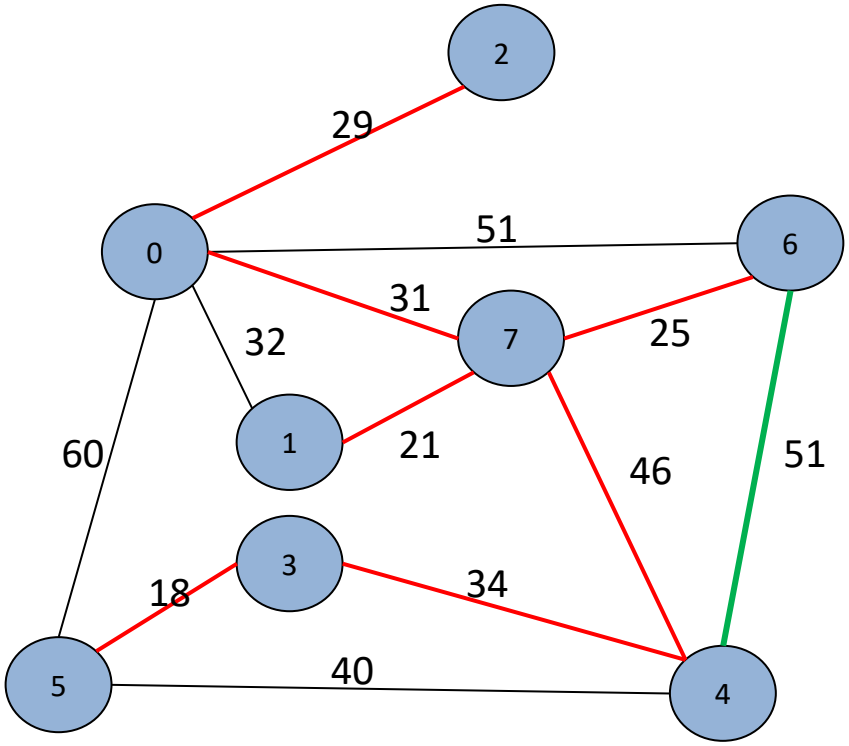
Si la arista considerada forma un ciclo con aristas que ya están en el MST, significa que *no cruza el corte* y la descartamos

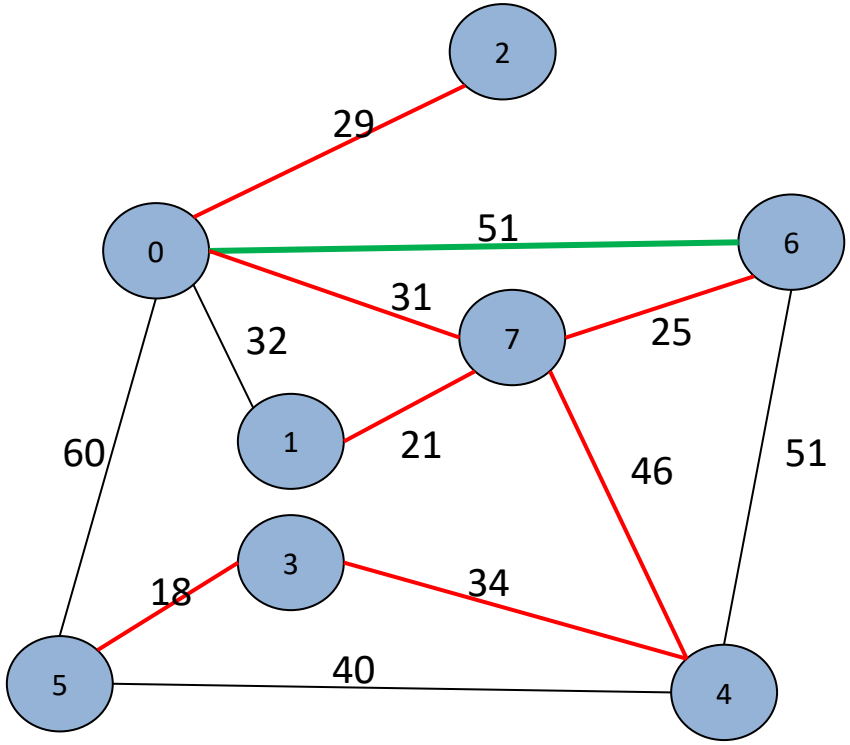


Si el grafo tiene $|V|$ vértices, entonces el MST tiene $|V|-1$ aristas



Una vez que el MST tiene $|V|-1$ aristas, cualquiera otra arista forma un ciclo … y en la práctica no es necesario revisar las aristas restantes





Corrección de **`kruskal`**

Para demostrar que **`kruskal`** es correcto

… basta demostrar que dado cualquier corte, la arista de menor costo que cruza el corte está en el MST:

- haciendo el supuesto de que todos los costos son distintos, se puede demostrar (más o menos fácilmente) por contradicción

… y luego demostrar que `kruskal` efectivamente implementa esta estrategia —*hints*:

- ya vimos cómo define `kruskal` el corte (V_1, V_2)
- ¿cómo elije `kruskal` la arista de menor costo que cruza el corte anterior?

Un “detalle” no menor

$\boldsymbol{kruskal}\big(\boldsymbol{G}(\boldsymbol{V},\boldsymbol{E})\big)$:

ordenar $\boldsymbol{E}$ por costo, de menor a mayor

$\boldsymbol{T} \leftarrow \emptyset$

$\boldsymbol{foreach}\ \boldsymbol{e} \in \boldsymbol{E}$, en el orden obtenido arriba:

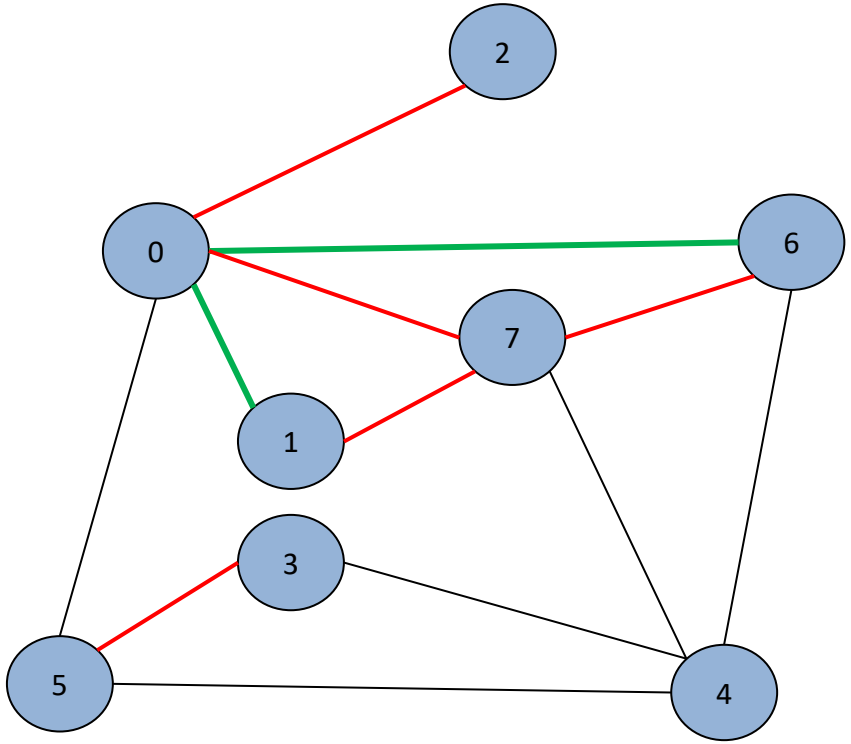
$\boldsymbol{if}$ agregar $\boldsymbol{e}$ a $\boldsymbol{T}$ no forma un ciclo:

agregar $\boldsymbol{e}$ a $\boldsymbol{T}$

$\boldsymbol{return}\ \boldsymbol{T}$

¿cómo revisamos esto eficientemente?

Agregar (*u*,*v*) forma un ciclo **ssi** *u* y *v* están en el mismo subárbol (de aristas rojas)



Un nodo puede pertenecer a un solo sub-árbol del grafo

Los **conjuntos** de nodos de cada sub-árbol son **disjuntos**

¿Cómo podemos modelar esto para aprovecharlo?

`kruskal` con el vocabulario de conjuntos disjuntos

$\boldsymbol{kruskal}\big(\boldsymbol{G}(\boldsymbol{V},\boldsymbol{E})\big)$:

ordenar $\boldsymbol{E}$ por costo, de menor a mayor

considerar cada nodo como formando un conjunto por sí mismo

$\boldsymbol{T} \leftarrow \emptyset$

$\boldsymbol{foreach}$ $(\boldsymbol{u},\boldsymbol{v}) \in \boldsymbol{E}$, en el orden obtenido arriba:

$\boldsymbol{if}$ si $\boldsymbol{u}$ y $\boldsymbol{v}$ no están en el mismo conjunto:

agregar $(\boldsymbol{u},\boldsymbol{v})$ a $\boldsymbol{T}$

unir los conjuntos de $\boldsymbol{u}$ y $\boldsymbol{v}$

$\boldsymbol{return}$ $\boldsymbol{T}$

`kruskal` con las operaciones de conjuntos disjuntos

$kruskal\big(G(V,E)\big)$:

ordenar E por costo, de menor a mayor

$foreach\ v \in V$: $make\ set(v)$

$T \leftarrow \emptyset$

$foreach\ (u,v) \in E$, en el orden obtenido arriba:

$if\ find\ (u) \neq find\ (v)$:

$T \leftarrow T \cup \{(u,v)\}$

$union(u,v)$

$return\ T$

¿Cuál es la complejidad de `kruskal`, si representamos los conjuntos como árboles?

Primero, hay que ordenar las $|E|$ aristas → $O(E \log E)$

Luego, hay que construir $|V|$ conjuntos (de un elemento cada uno) → $O(V)$

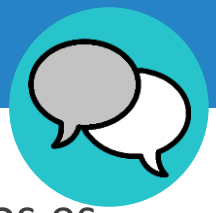
Durante la ejecución del segundo *loop*, se realizan $|V|-1$ uniones

… y $2|E|$ operaciones *find*

Cada *union* toma O(1) → O(V) para el total de $|V|-1$ operaciones *union*

¿Cuánto toman en total las $2|E|$ operaciones *find*?

¿Cuánto toman en total las 2|*E*| operaciones *find*?



El costo promedio de una operación *find* en un conjunto de *n* elementos es O(log**n*) —usando unión por rango y compresión de ruta

El *n* más pequeño para el cual log**n* es 5 es $n = 2^{16} = 65536$

… y va a quedarse en 5 para todos los $n \leq 2^{65536}$

→ en cualquier uso práctico, log**n* es constante (aunque en teoría tiende a ∞)

Así, las 2|*E*| operaciones *find* toman O(*E*log**E*)

… y la complejidad de **`kruskal`** es

O(*E*log*E*) + O(V) + O(*E*log**E*)

= O(*E*log*E*)

= **O(*E*log*V*)**, ya que |E| = O(V^2)