Estrategias algorítmicas

Dividir para reinar

Backtracking

Algoritmos codiciosos

Programación dinámica

Estrategias algorítmicas

Dividir para reinar

Backtracking

Algoritmos codiciosos

Programación dinámica

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***

c) Grafos direccionales:

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

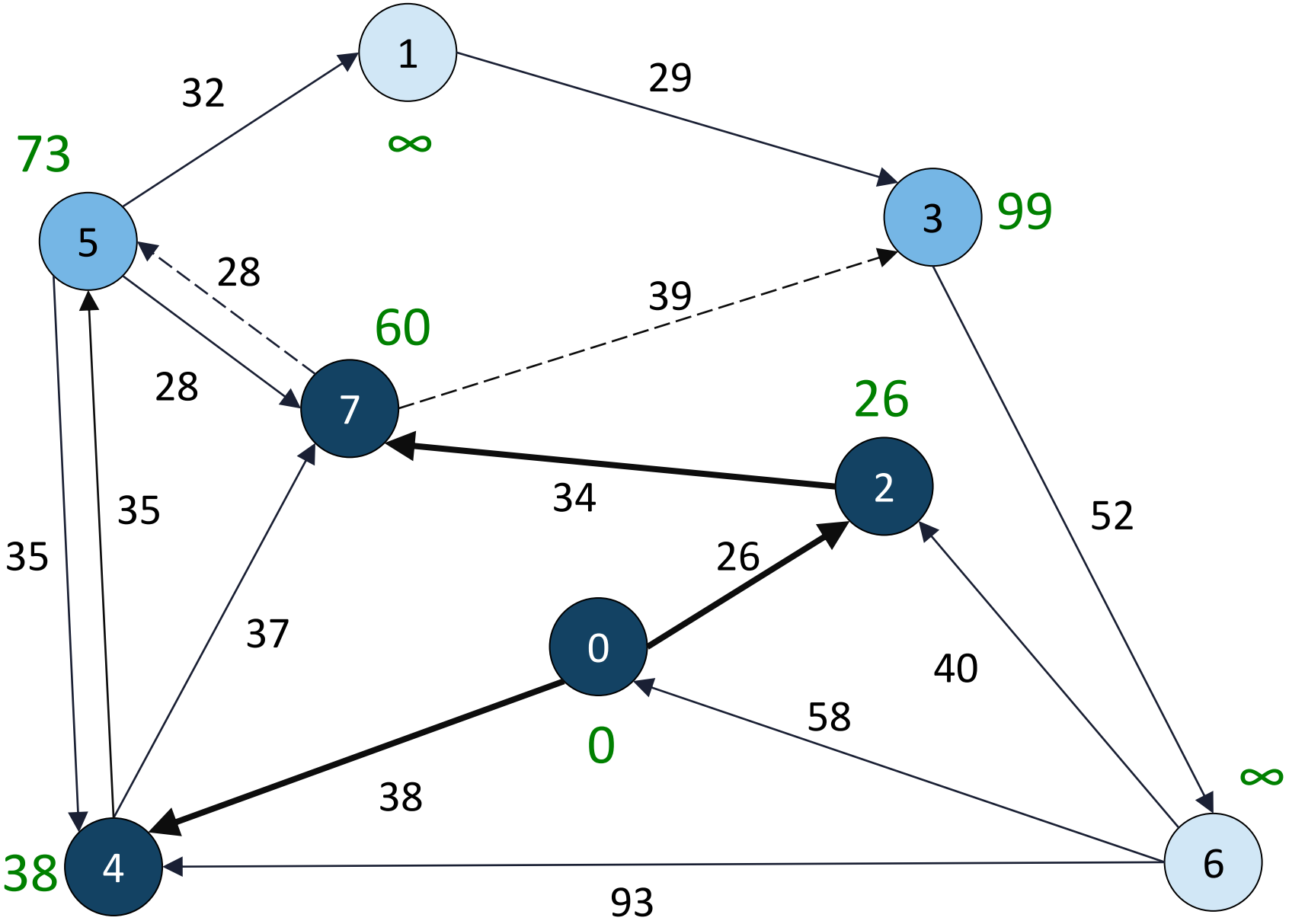
El algoritmo de Dijkstra es un **algoritmo codicioso**

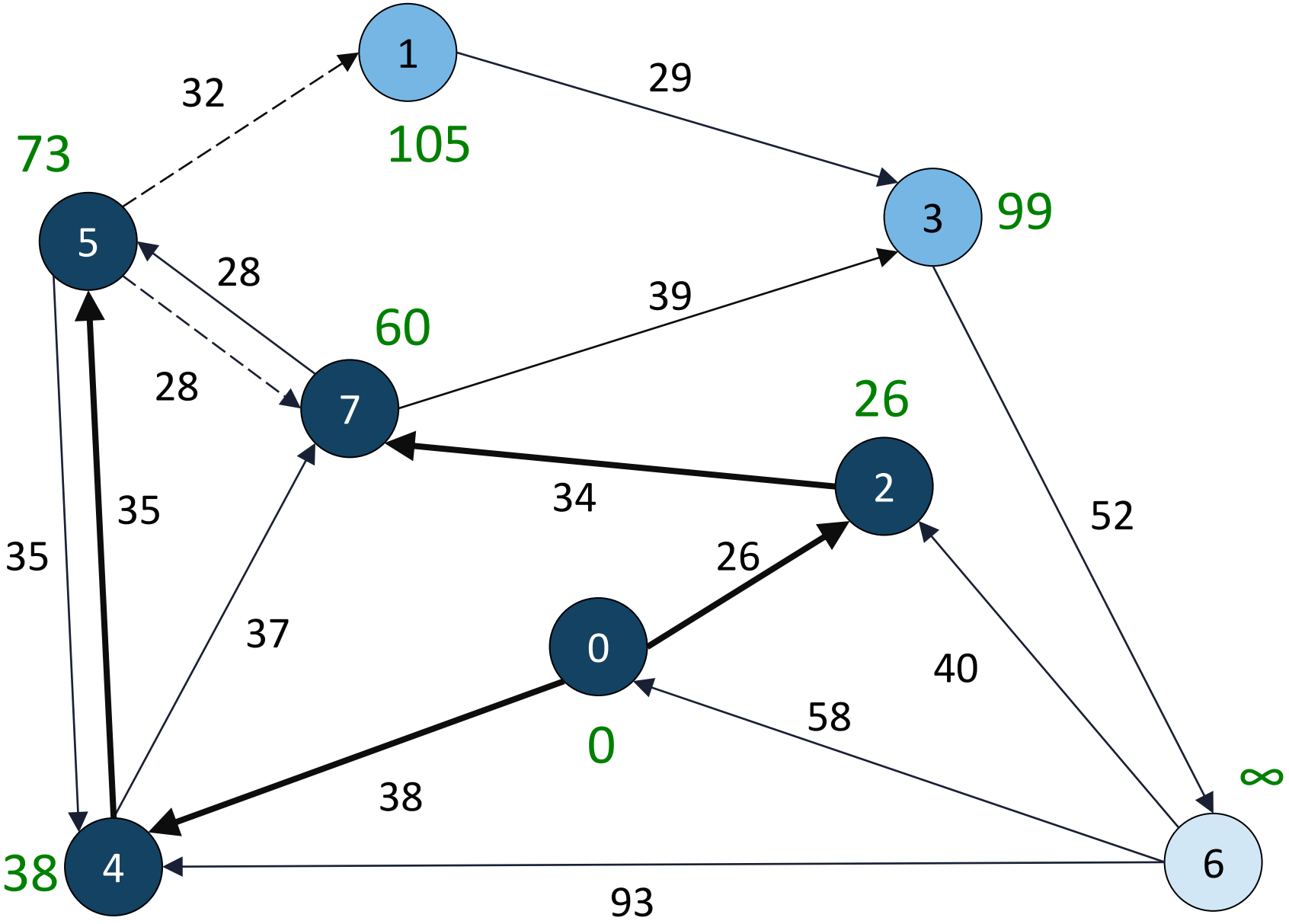
En cada paso, el algoritmo hace una elección que corresponde a un **óptimo local**:

- toma la mejor decisión que puede, con la información que tiene hasta ese momento

… con la esperanza de que al hacer la última elección haya logrado el (un) **óptimo global:**

- algunos algoritmos codiciosos efectivamente encuentran el óptimo global (p.ej., Dijkstra)
- … pero no todos


1
∞
3
99
5
73
7
60
2
26
0
0
4
38
6
∞
32
29
28
39
28
35
35
34
52
26
37
40
58
38
93

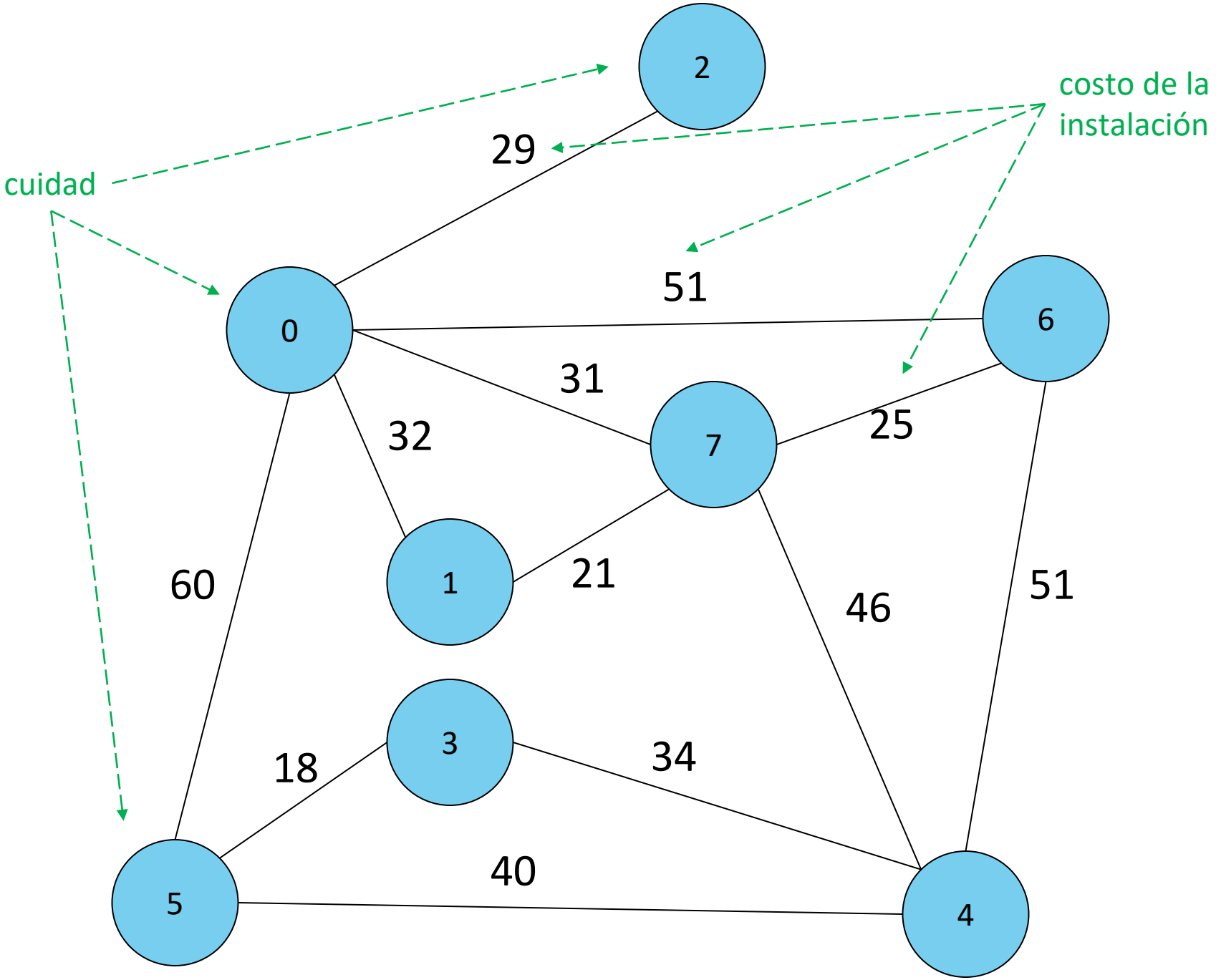

1
105
29
32
73
5
3
99
28
39
60
28
7
26
2
34
35
52
35
26
37
0
40
58
0
38
∞
38
4
6
93

Conectividad digital

- El intendente de la Región del Maule ha decidido mejorar significativamente la conectividad digital de la región
- La idea es instalar fibra óptica subterránea entre todos los pares de puntos relevantes de la región —cada instalación tiene un costo
- Sólo que hay demasiada fibra óptica que instalar como para hacerlo todo de una vez
- Lo prioritario es conectar las ciudades más pobladas, que tienen escuelas, universidades, hospitales, compañías de bomberos, super-mercados, etc.

¿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
Llepo
Mesamávida
Pelluhue
RESERVA NACIONAL LOS RUILES
Curanipe
Los Nabos
Los Perales
Longaví
Melado
Aduana Pejerrey
Río Achibueno
Nvdo Longaví 3242
Co El Buitre 2955
Laguna del Maule
Laguna Negra
Laguna Fea
Río Melado
Río d e la Puente
Pya Peuño
Pta Nugurue
CAUQUENES
Los Cristales
Retiro
Paso Hondo
Hualve
Unicavén
Quella
El Tránsito
Río Longaví
Tregualemu
RESERVA NACIONAL LOS QUEULES
Pullay
Coronel
El Peñon
Parral
Ajial
Villa Rosa
San Pablo
Quinchimávida
Cancha Alegre
51 Kms a Quirihue
60 Kms a Chillán
Tres Esquinas
Buchupureo
Cobquecura
El Parrón
Pocillas
Perquilauquén
Emb Digua
Termas de Catillo
San Gregorio
Río Perquilauquén
Laguna Varvaco Campos
SIERRA DE CHOHICO
Río Barrancas
Quirihue
Colmuyao
Torrecilla
San Carlos
Zemita
Ninhue
Cachapoal
El Sauce
BALSEO
Nahueltoro
Lago Dial
Mela
Trehuaco
Coipin
San Nicolás
San Fabián de Alicó
Los Sauces
Vn Domuyo 4709
Vegas de Itata
35°
36°
71°
72°

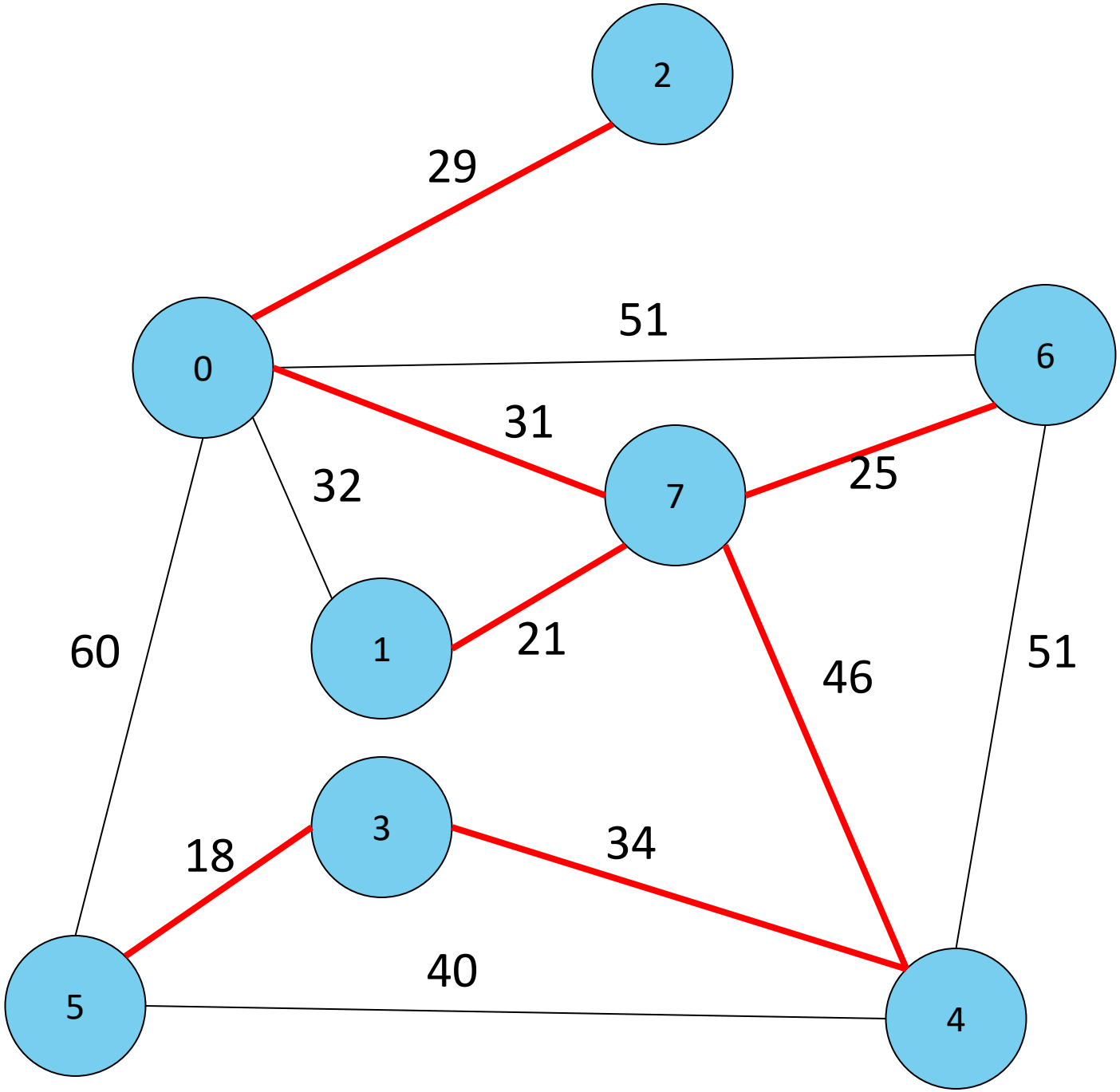

2
costo de la instalación
cuidad
29
51
0
6
31
32
7
25
60
1
21
46
51
3
18
34
40
5
4

MST: *Minimum spanning tree*

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

La solución es un **subconjunto *T* de las aristas** tal que:

- las aristas de *T* no forman ciclos —forman un **árbol**
- las aristas de *T* incluyen todos los vértices —forman una **cobertura**
- no hay otro árbol de cobertura con menor costo total (la suma de los costos de las aristas de *T*) —es de **costo mínimo**


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

MSTs

Originalmente, hace 100 años, redes de distribución de potencia

Después, redes telefónicas

Hoy, redes de comunicación, eléctricas, hidráulicas, de computadores, de carreteras, de tráfico aéreo

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

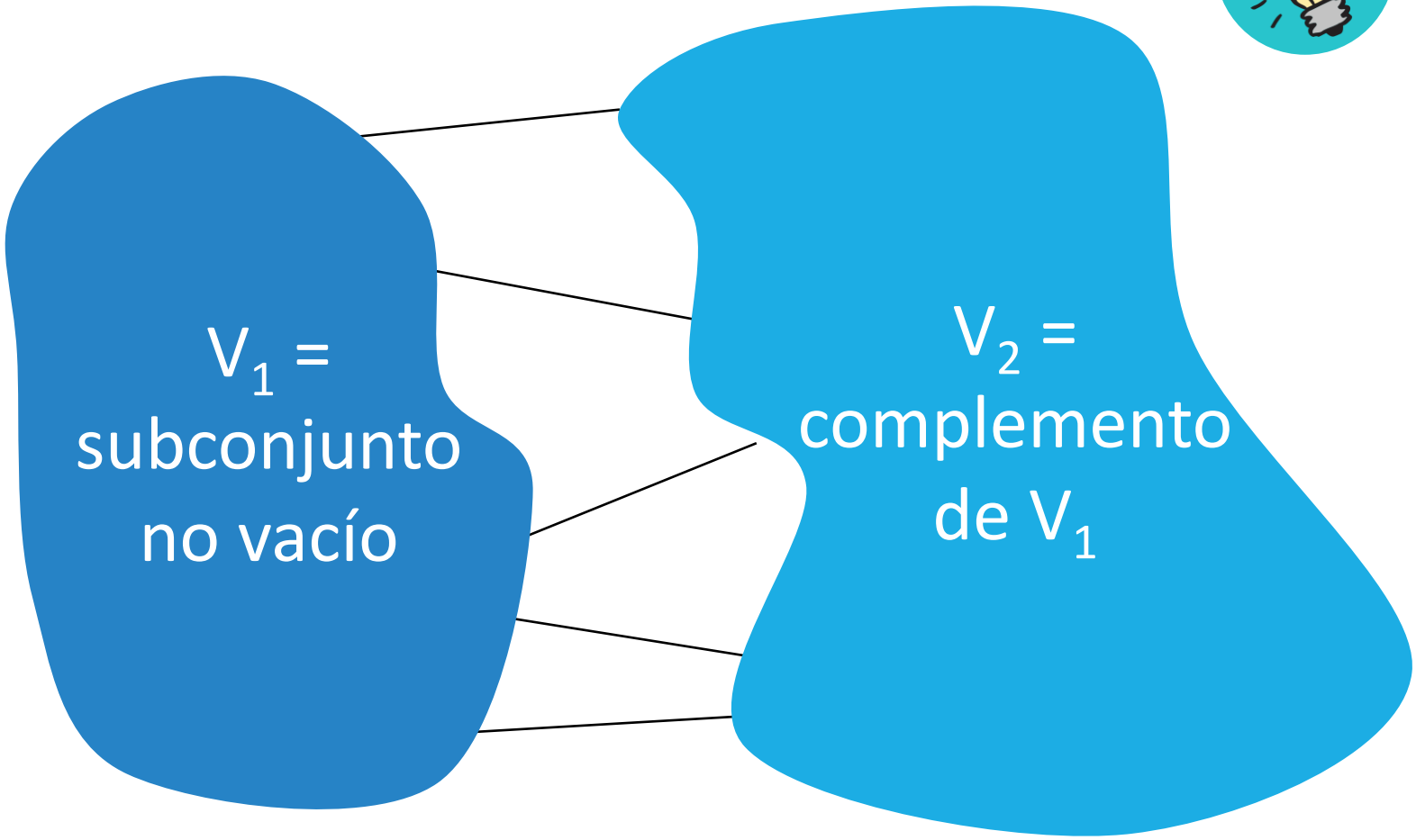
MSTs, **cortes** y aristas que **cruzan** el corte

Cortemos (particionemos) los vértices del grafo en dos (sub) conjuntos no vacíos, V_1 y V_2

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 estas aristas y los **MST**?

El corte (V_1, V_2) y las aristas que cruzan el corte



¿Cuál debería ser la siguiente arista, y siguiente nodo, a incluir?

Buscando un MST

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

… ¿cómo podemos encontrar un **MST**?

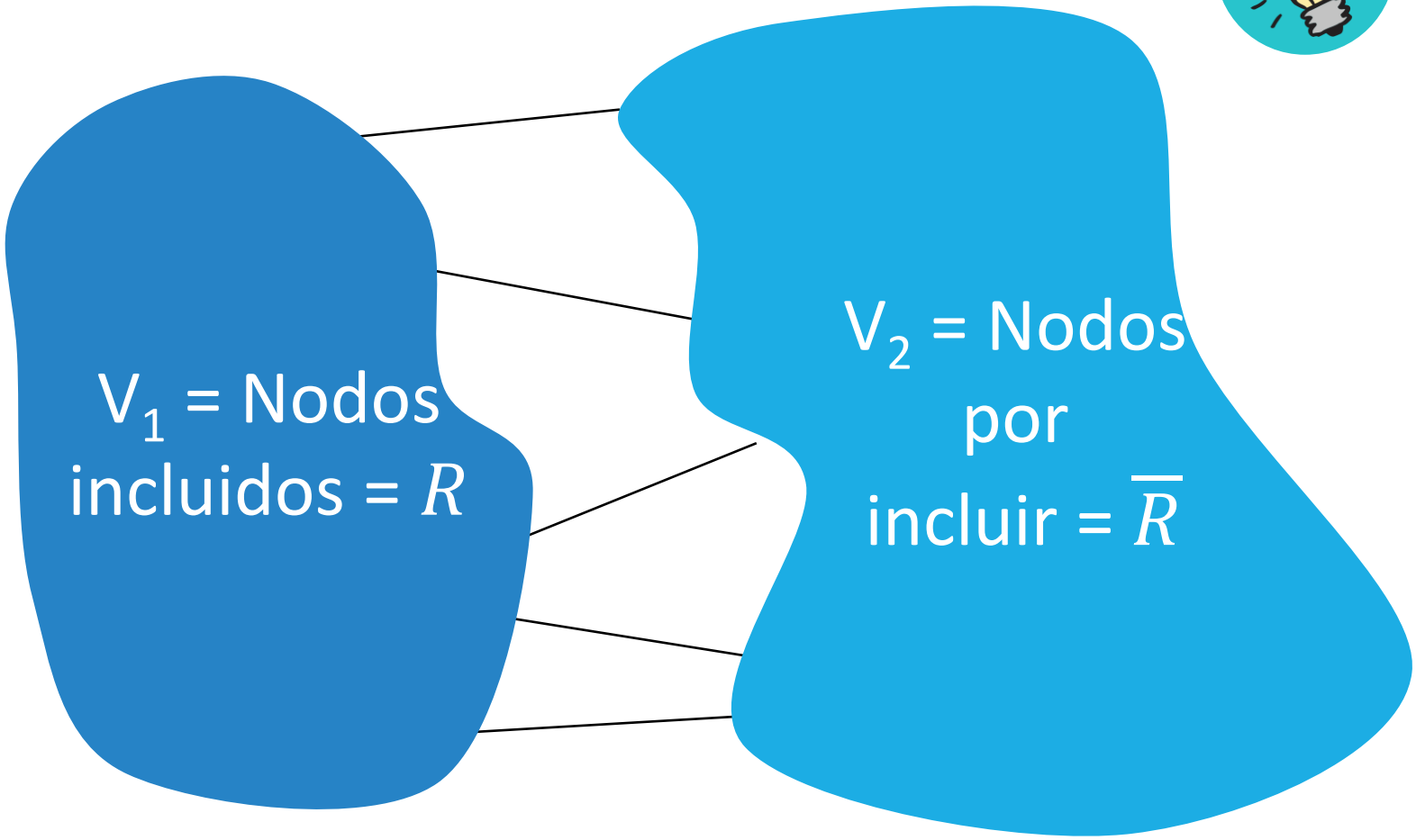
… ¿podremos construirlo una arista a la vez?

Algoritmo de Prim

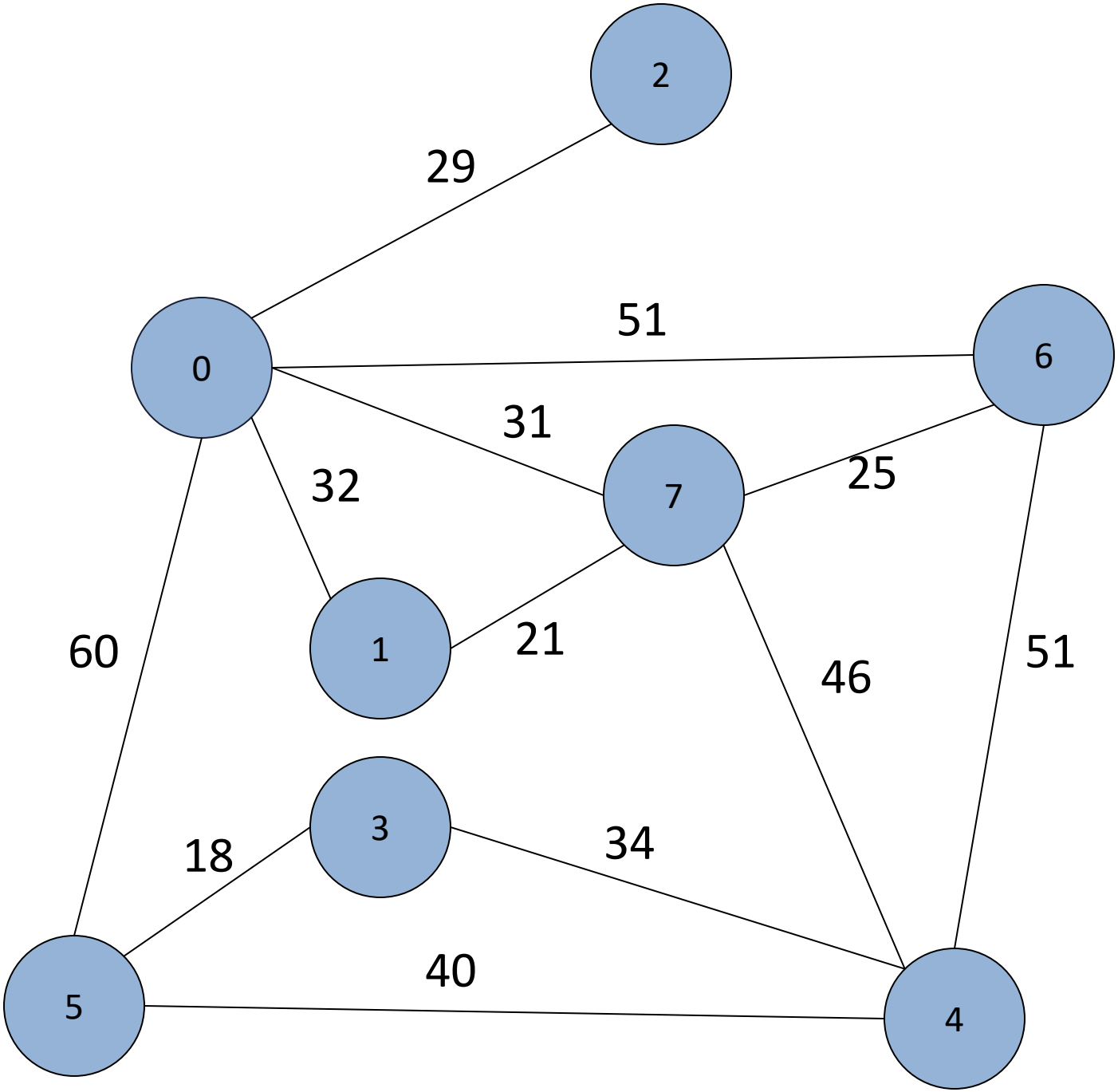
Para un grafo $G(V, E)$, y un nodo inicial x

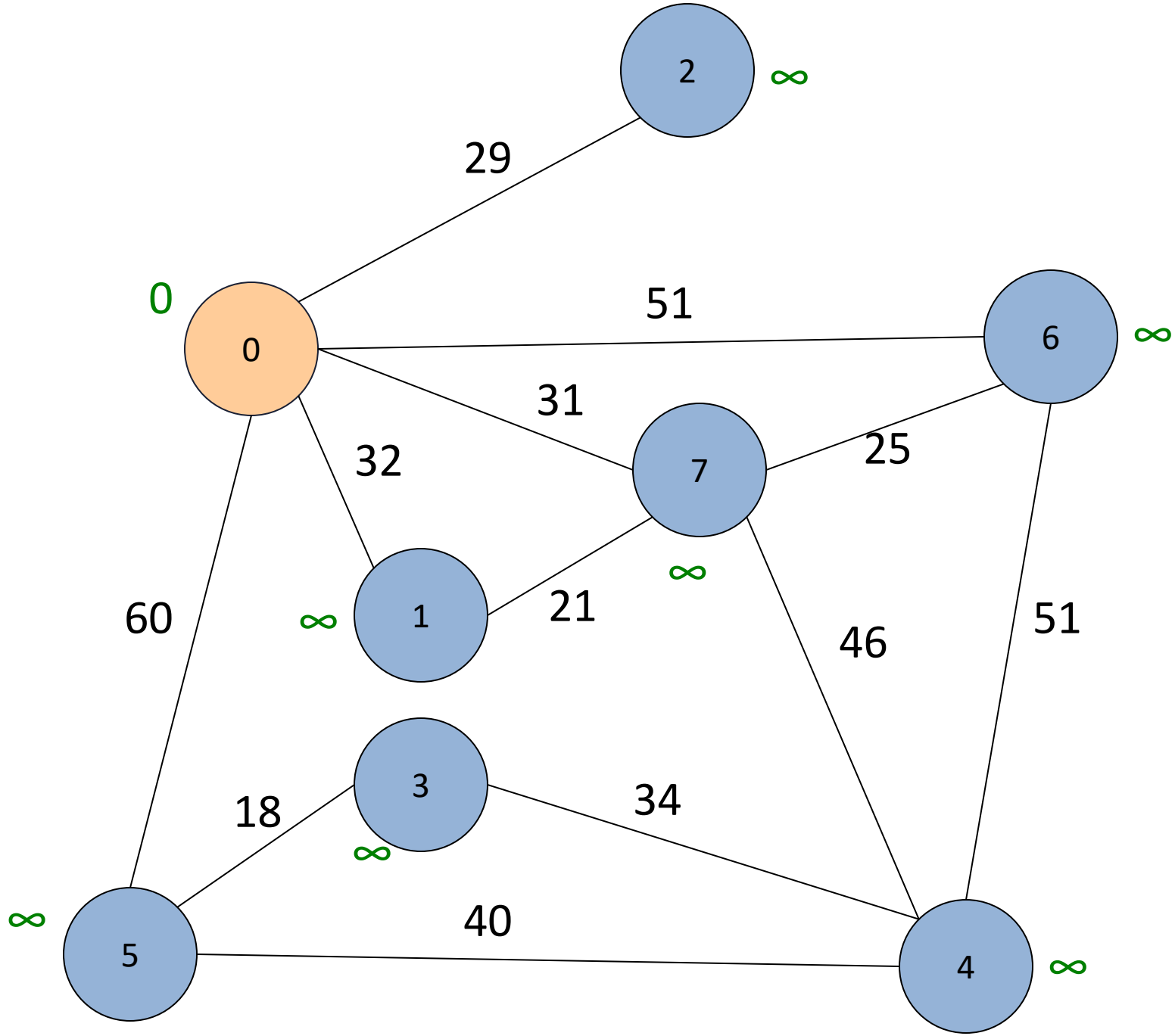
1. Sean $R = \{x\}$, $\overline{R} = V - R$, los nodos incluidos y los que no
2. Sea e la arista de menor costo que cruza de R a $\overline{R}$
3. Sea u el nodo de e que pertenece a $\overline{R}$
4. Agregar e al MST. Eliminar u de $\overline{R}$ y agregarlo a R
5. Si quedan elementos en $\overline{R}$, volver a 2.

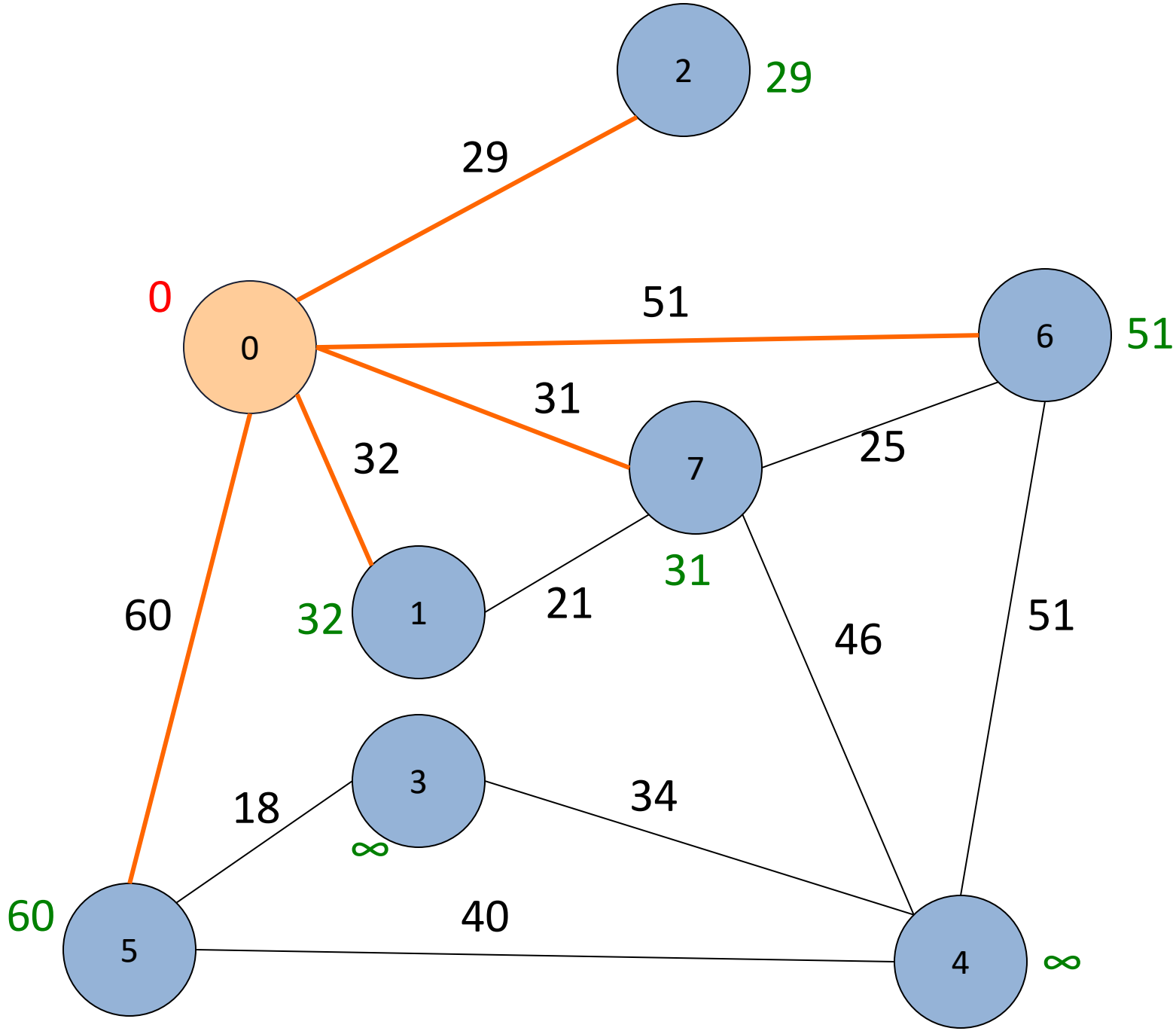
El corte (V_1, V_2) y las aristas que cruzan el corte en el algoritmo de Prim

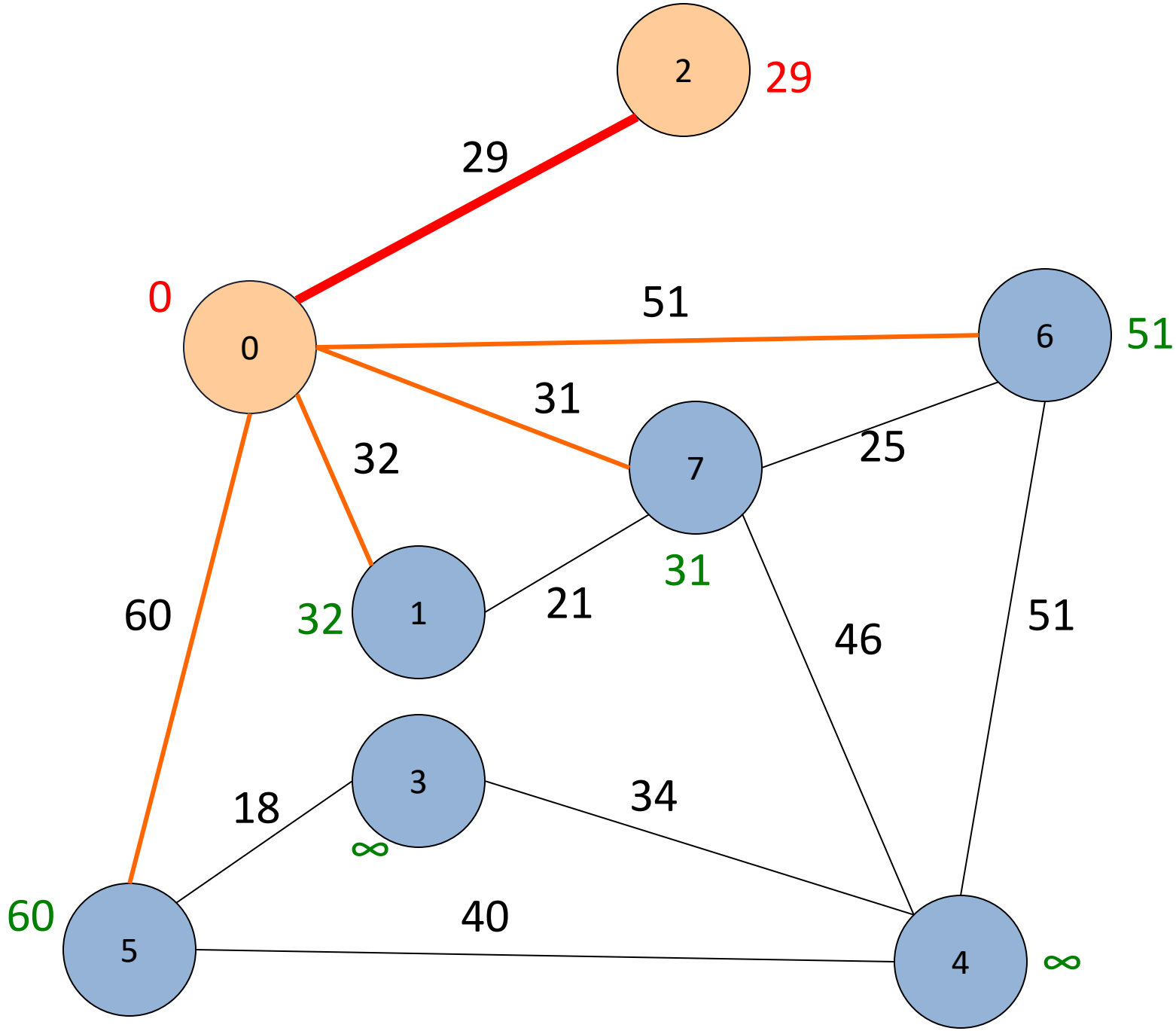


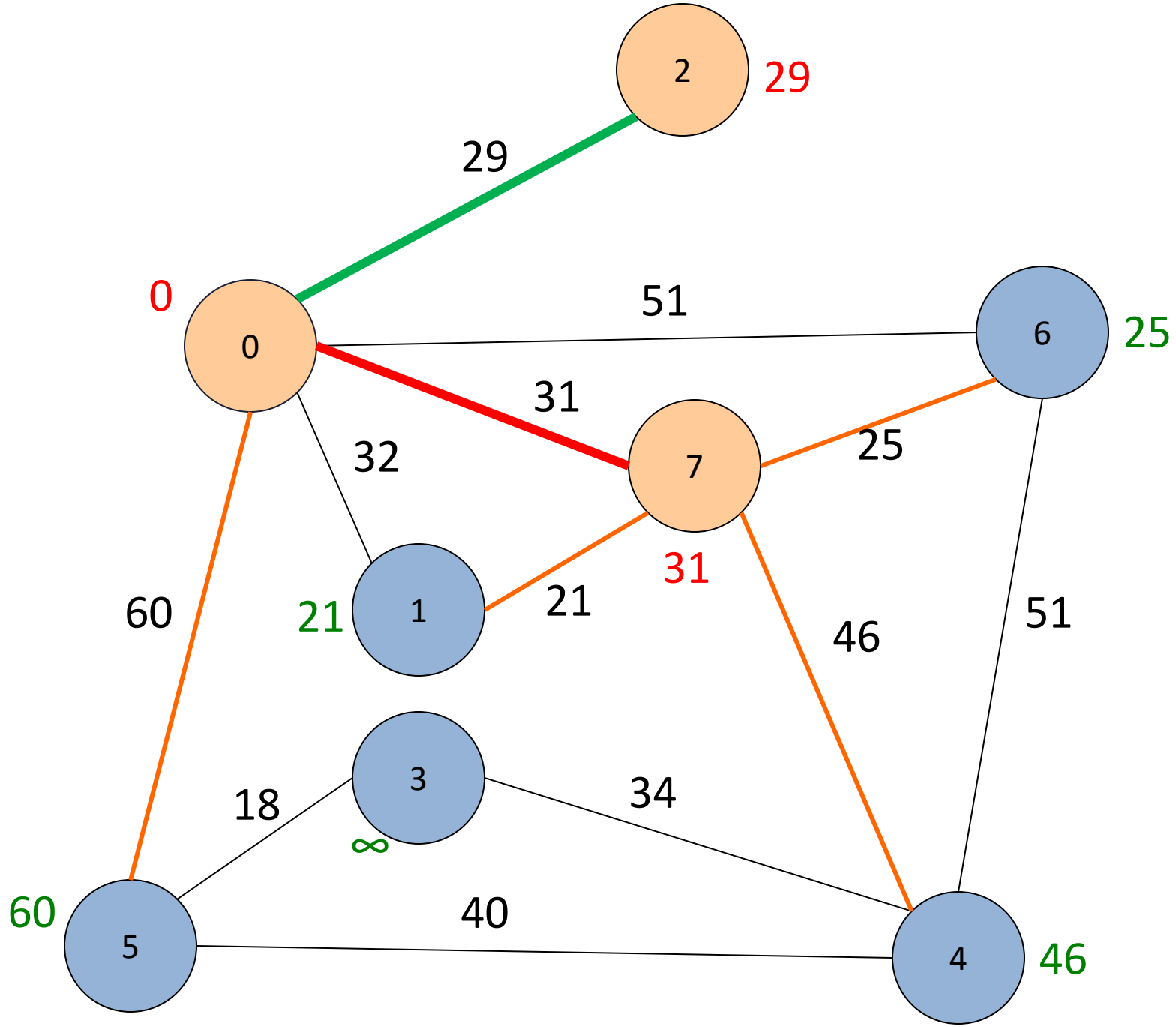
¿Cuál debería ser la siguiente arista, y siguiente nodo, a incluir?

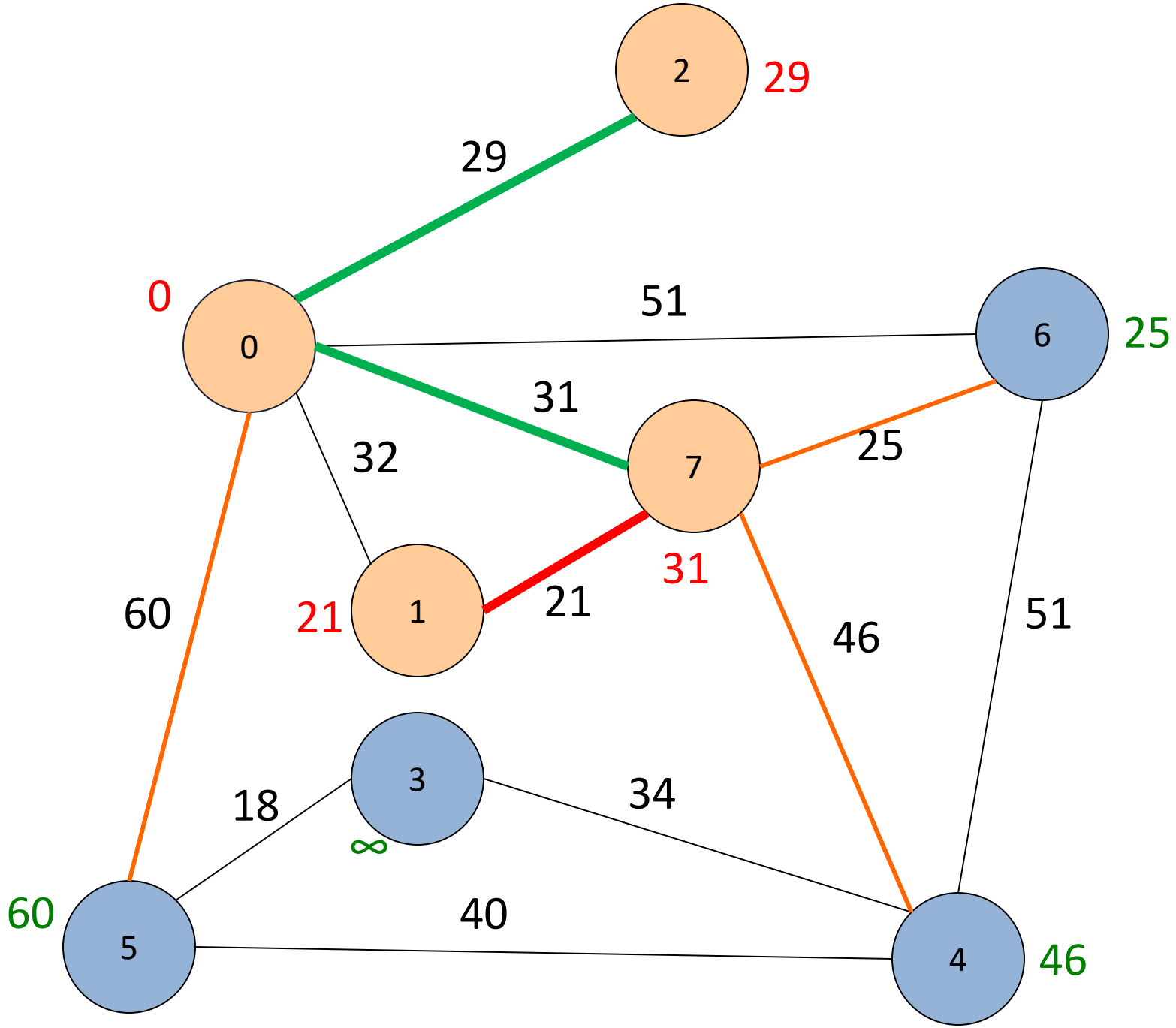

2
29
0
51
6
31
32
7
25
1
21
60
46
51
3
18
34
5
40
4

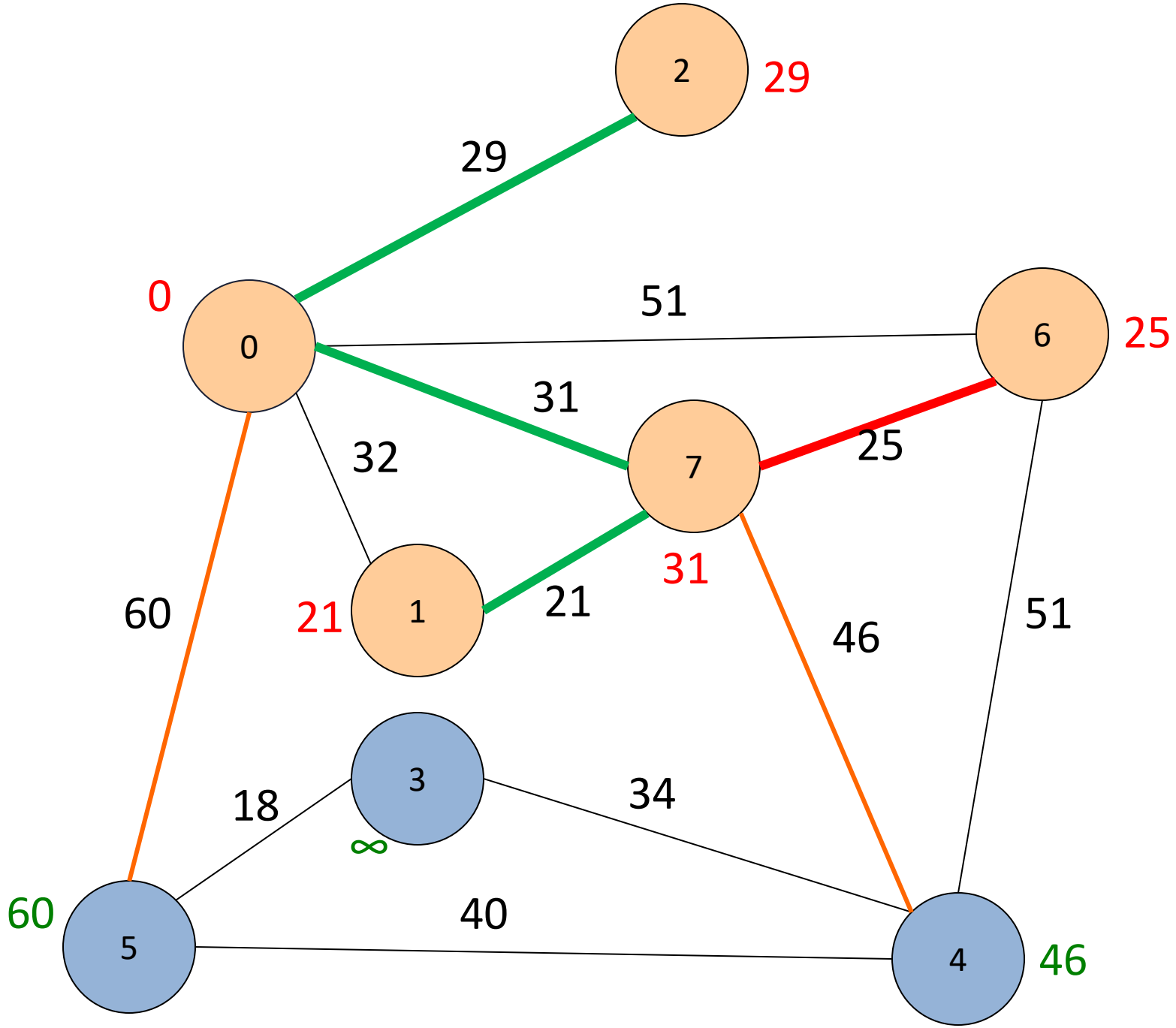

2
∞
29
0
0
51
6
∞
31
32
7
25
∞
60
∞
1
21
46
51
3
18
34
∞
∞
5
40
4
∞

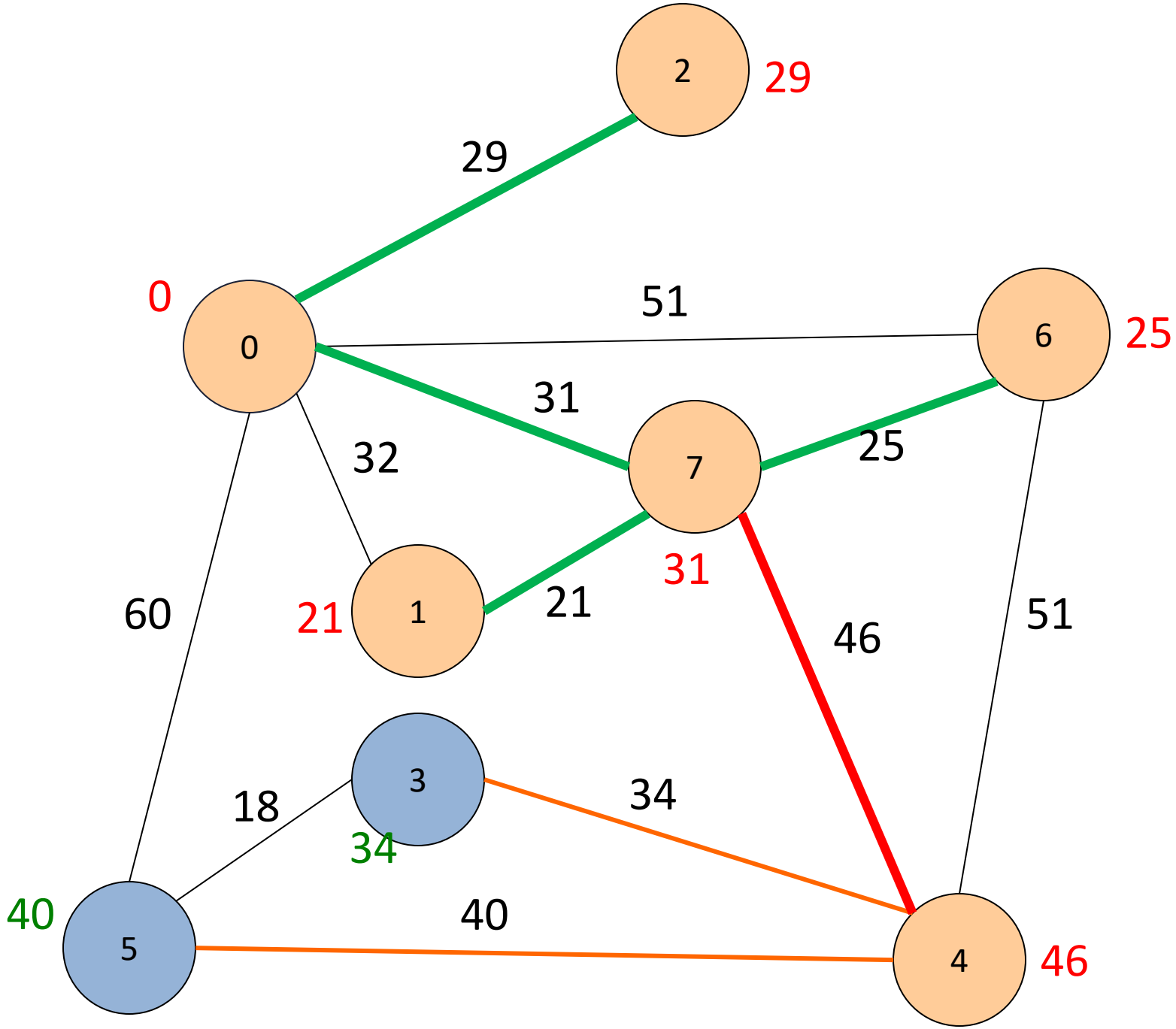

2
29
29
0
0
51
6
51
31
7
25
32
31
60
32
1
21
46
51
3
18
34
∞
60
5
40
4
∞

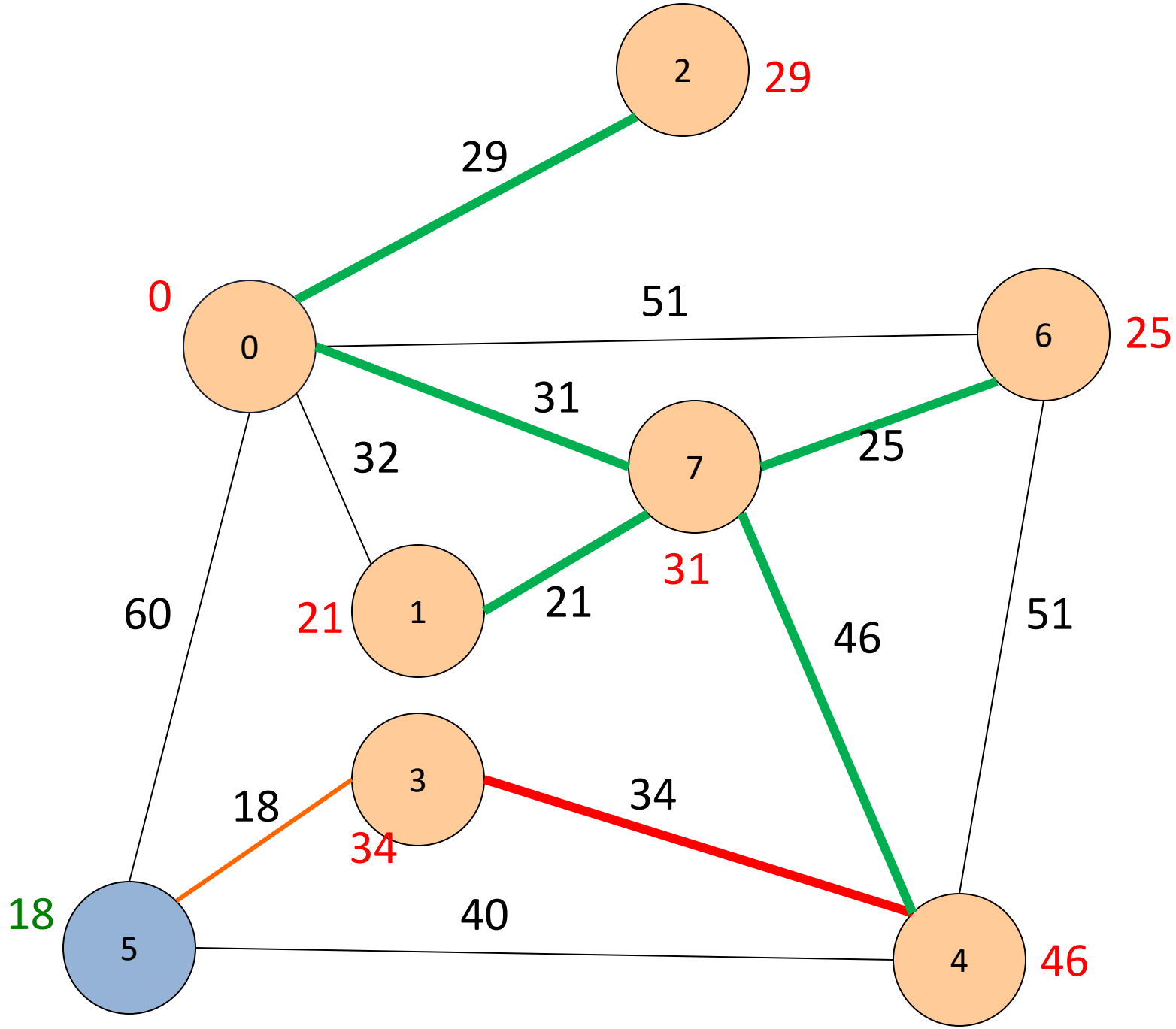

2
29
29
0
0
51
6
51
31
32
7
25
31
60
32
1
21
46
51
3
18
34
∞
60
40
5
4
∞

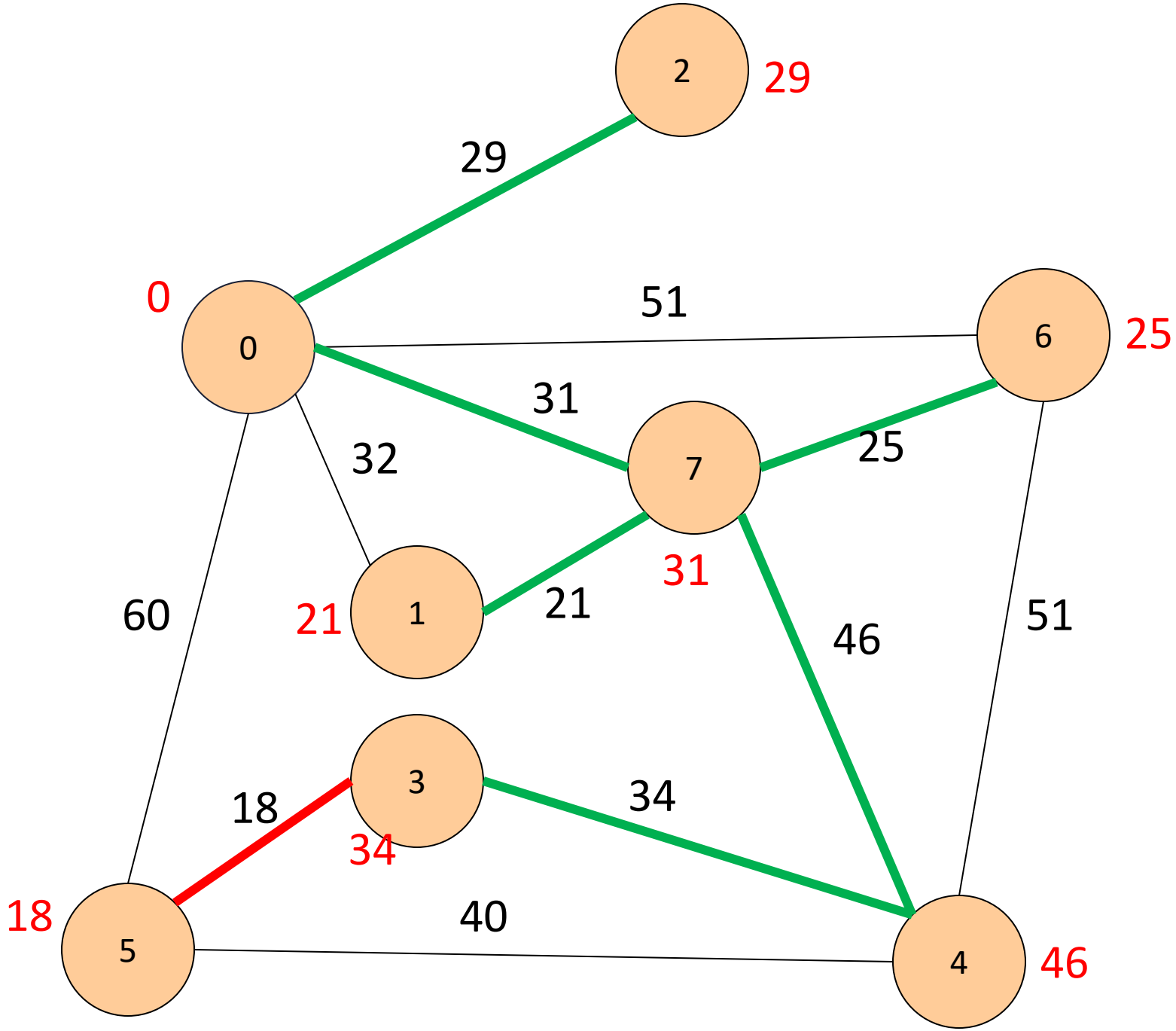

2
29
29
0
0
51
6
25
31
32
7
25
31
60
21
1
21
46
51
3
18
34
∞
60
5
40
4
46

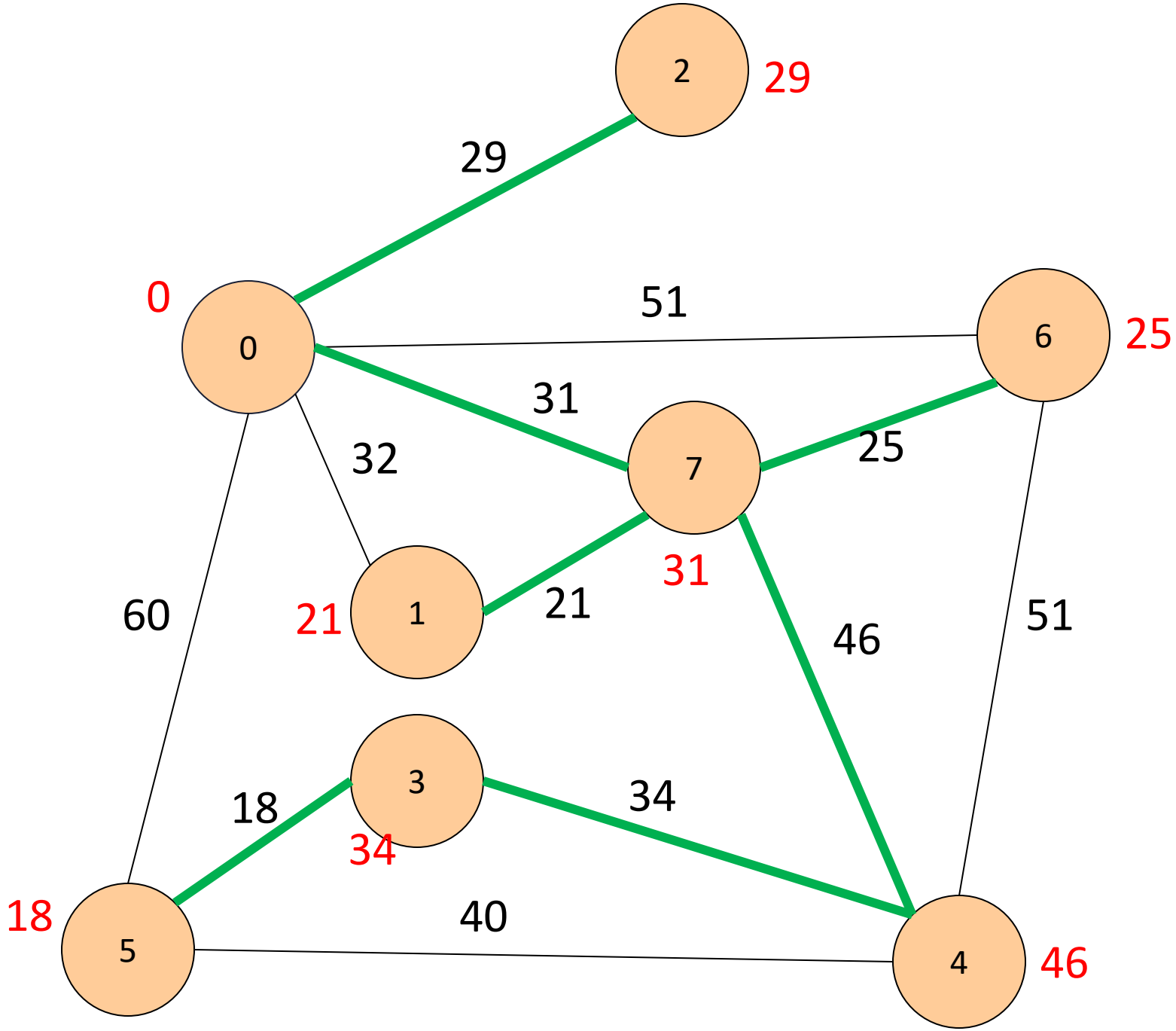

2
29
29
0
0
51
6
25
31
32
7
25
31
60
21
1
21
46
51
3
18
34
∞
60
5
40
4
46


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


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


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


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


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

Prim en pseudo código

```
Prim(s): —s es el vértice de partida
   Q ← cola de prioridades;   T ← ∅
   for each u in V-{s}:
        d[u] ← ∞; π[u] ← null; Q.enqueue(s)
   d[s] ← 0; π[s] ← null; Q.enqueue(s)
   while !Q.empty():
      u ← Q.dequeue();   T ← T ∪ (π[u],u)
      for each v in α[u]:
         if v ∈ Q:
            if d[v] > costo(u,v):
               d[v] ← costo(u,v); π[v] ← u
   return T
```

Corrección de Prim

Para demostrar que Prim es correcto

… basta demostrar que dado cualquier corte, la arista de menor peso que cruza el corte está en el MST (diap. # 15):

- haciendo el supuesto de que todos los pesos son distintos, se puede demostrar (fácilmente) por contradicción

… y luego demostrar que Prim efectivamente implementa esta estrategia —hints:

- ¿cómo define Prim el corte (V_1, V_2) sugerido en la diap. # 14?
- ¿cómo elije Prim la arista de menor peso que cruza el corte anterior?

Complejidad de Prim

La complejidad está dada por la complejidad del ciclo **`while`**

El ciclo ocurre $|V|$ veces, una por cada nodo *u* que se saca de *Q*:

- para cada *u* que sale de *Q* se revisan todas las aristas adyacentes a *u*
- … → el **`while`** revisa cada nodo y cada arista del grafo una vez → O(*V*+*E*)
- … pero en cada revisión hace una actualización → O(*V*+*E*) × *algo*

Q es una cola de prioridades según *d*[*v*]; si la implementamos como un heap binario:

- … *algo* es el tiempo que toma sacar un elemento del heap (no es O(1))
- … y también el tiempo que toma actualizar la posición de un elemento en el heap (la asignación $d[v] \leftarrow costo(u,v)$)

$\boldsymbol{prim}(\boldsymbol{G}(\boldsymbol{V},\boldsymbol{E}),\boldsymbol{x})$:

$\boldsymbol{T} \leftarrow \emptyset$, $\boldsymbol{H} \leftarrow$ una **cola de prioridades** únicamente con $\boldsymbol{x}$

$\boldsymbol{x}.\boldsymbol{key} \leftarrow \boldsymbol{0}$, $\boldsymbol{x}.\boldsymbol{parent} \leftarrow \emptyset$,

while $\boldsymbol{H} \neq \emptyset$:

$\boldsymbol{u} \leftarrow$ extraer el vértice de $\boldsymbol{H}$ con menor clave, y pintarlo

if $\boldsymbol{u}.\boldsymbol{parent} \neq \emptyset$, agregar la arista $(\boldsymbol{u}.\boldsymbol{parent}, \boldsymbol{u})$ a $\boldsymbol{T}$

foreach vecino no pintado $\boldsymbol{v}$ de $\boldsymbol{u}$:

if $\boldsymbol{v} \notin \boldsymbol{H}$, insertar $\boldsymbol{v}$ en $\boldsymbol{H}$

if $\boldsymbol{w}(\boldsymbol{u},\boldsymbol{v}) < \boldsymbol{v}.\boldsymbol{key}$:

$\boldsymbol{v}.\boldsymbol{key} \leftarrow \boldsymbol{w}(\boldsymbol{u},\boldsymbol{v})$, $\boldsymbol{v}.\boldsymbol{parent} \leftarrow \boldsymbol{u}$

return $\boldsymbol{T}$

Actividades que usan un mismo recurso

Sea $S = \{1, 2, ..., n\}$ un conjunto de n actividades que deben usar un mismo recurso para poder ejecutarse; el recurso puede ser usado por sólo una actividad a la vez:

cada actividad i tiene una hora de inicio s_i y una hora de término f_i , $s_i \leq f_i$, y, si se ejecuta, transcurre durante el intervalo de tiempo $[s_i , f_i)$

las actividades i y j son **compatibles** si $[s_i, f_i)$ y $[s_j, f_j)$ no se traslapan, es decir, si $s_i \geq f_j$ o $s_j \geq f_i$

El problema consiste en seleccionar un *subconjunto de tamaño máximo de actividades mutuamente compatibles*

Primero, demostraremos que hay una solución óptima que comienza con la elección codiciosa de la actividad 1 (la que termina más temprano):

sea $A \subseteq S$ una solución óptima

ordenemos las actividades en A por hora de término

sea k la primera actividad en A

si $k = 1$, entonces A comienza con una elección codiciosa (y queda demostrado)

…

…

si $k \neq 1$, probamos que hay otra solución óptima B que sí empieza con la actividad 1:

sea $B = A - \{k\} \cup \{1\}$

… entonces como $f_1 \leq f_k$, las actividades en B son compatibles

… y como B tiene el mismo número de actividades que A, B también es una solución óptima (pero que incluye a la actividad 1)

Elegida la actividad 1, el problema se reduce a encontrar una solución óptima al mismo problema,

… pero sobre las actividades en S que son compatibles con la actividad 1

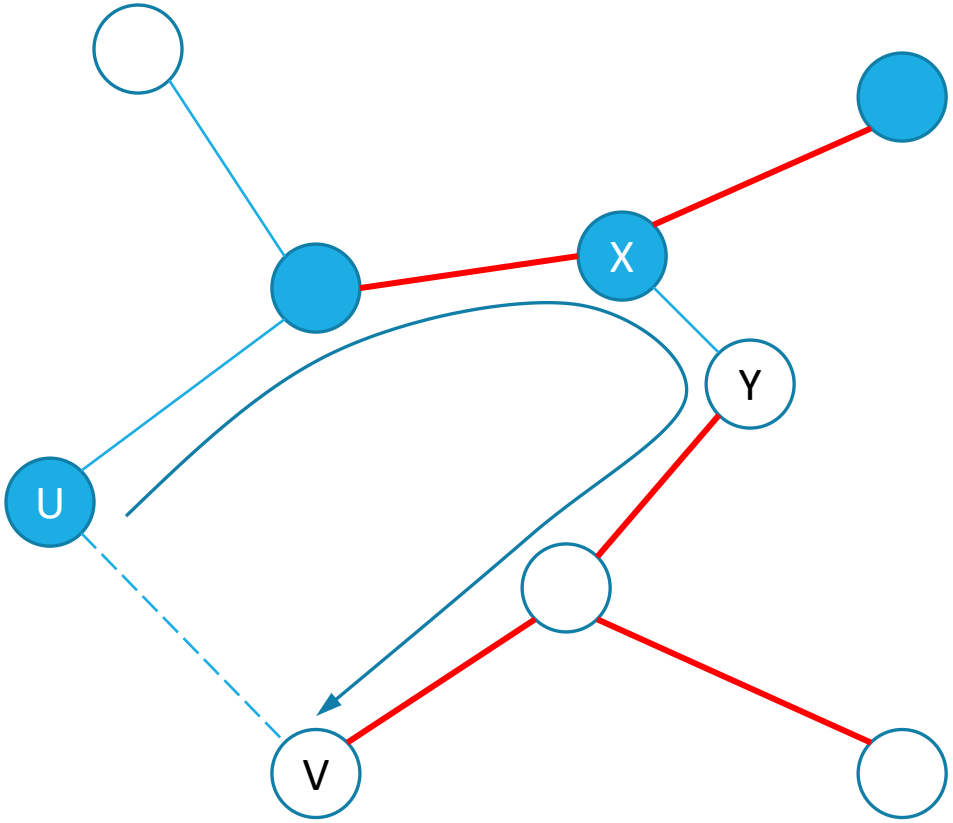
Demostraremos por contradicción que si *A* es una solución óptima a *S*,

… entonces $A' = A - \{1\}$ es una solución óptima a $S' = \{ i \in S : s_i \geq f_1 \}$:

si hubiera una solución *B'* a *S'* con más actividades que *A'*,

… entonces agregando la actividad 1 a *B'* daría una solución *B* a *S* con más actividades que *A*,

… contradiciendo que *A* es óptima


X
Y
U
V

Optimalidad *codiciosa*

Los algoritmos ***greedy*** son muy veloces

Pero no siempre sirven para encontrar el **óptimo**

¿Qué debe cumplirse en un problema para esto?