

Tres problemas en grafos *con costos*

a) Grafos no direccionales:

- encontrar el *árbol de cobertura de costo mínimo*

b) Grafos direccionales:

- encontrar la *ruta más corta desde un vértice a todos los otros*

c) Grafos direccionales:

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

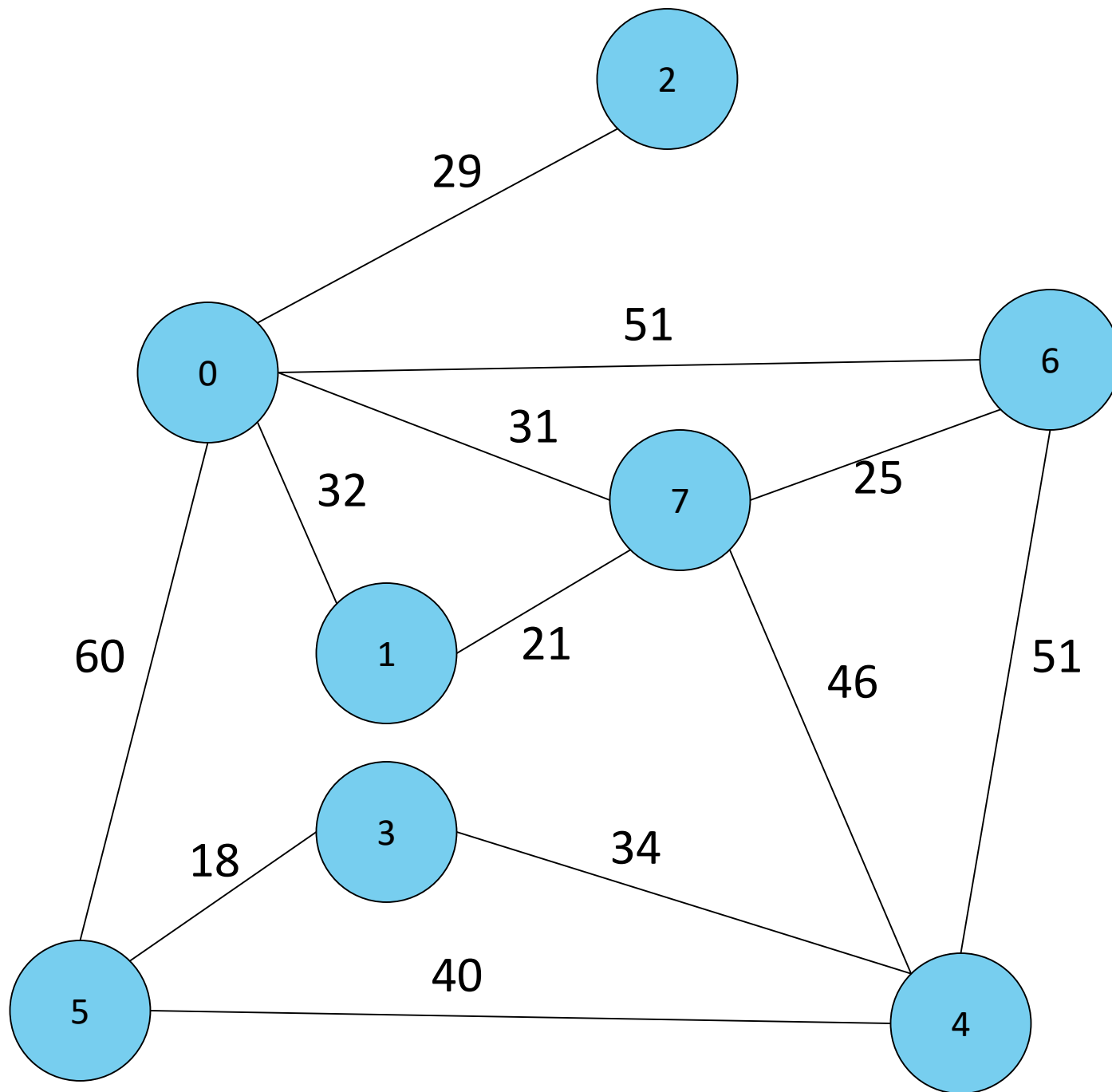
El desastre



- Un terremoto ha devastado la Región del Maule
- Se han caído puentes y destruido caminos enteros
- Hay demasiado que reparar para hacerlo todo de una vez
- Lo prioritario es restaurar la conectividad vial

¿Cuál es la forma más barata de hacer esto?



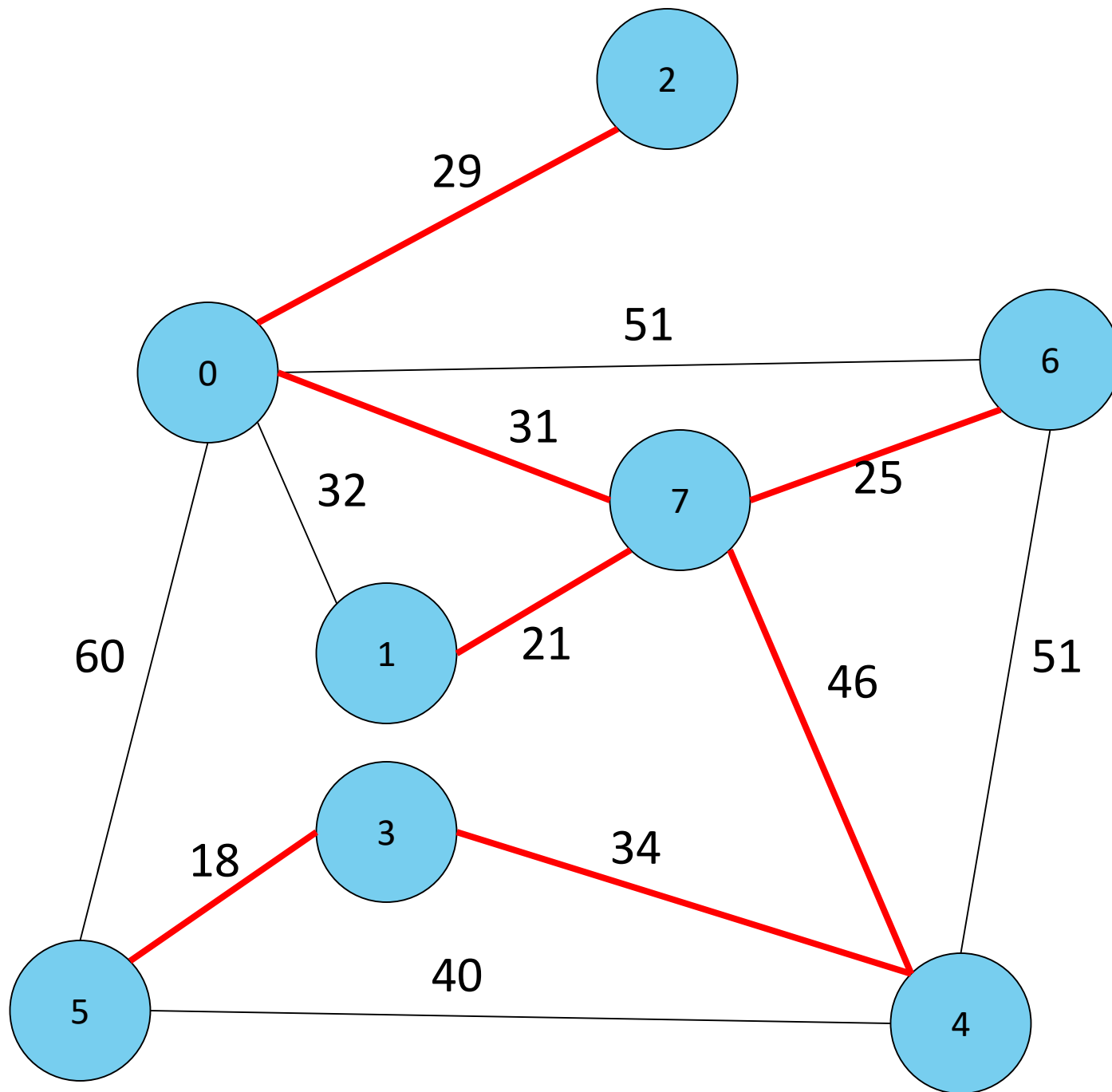


MST: *Minimum spanning tree*

Es un **árbol**: sus aristas no forman ciclos

Es de **cobertura**: el grafo es conexo

Es **mínimo**: no existe otro árbol de cobertura con menor costo total



MSTs, cortes y aristas que cruzan el corte



Cortemos (particionemos) el grafo en dos conjuntos de vértices V_1 y V_2

Una arista **cruza** el corte si un extremo está en V_1 y el otro en V_2

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

Buscando un MST

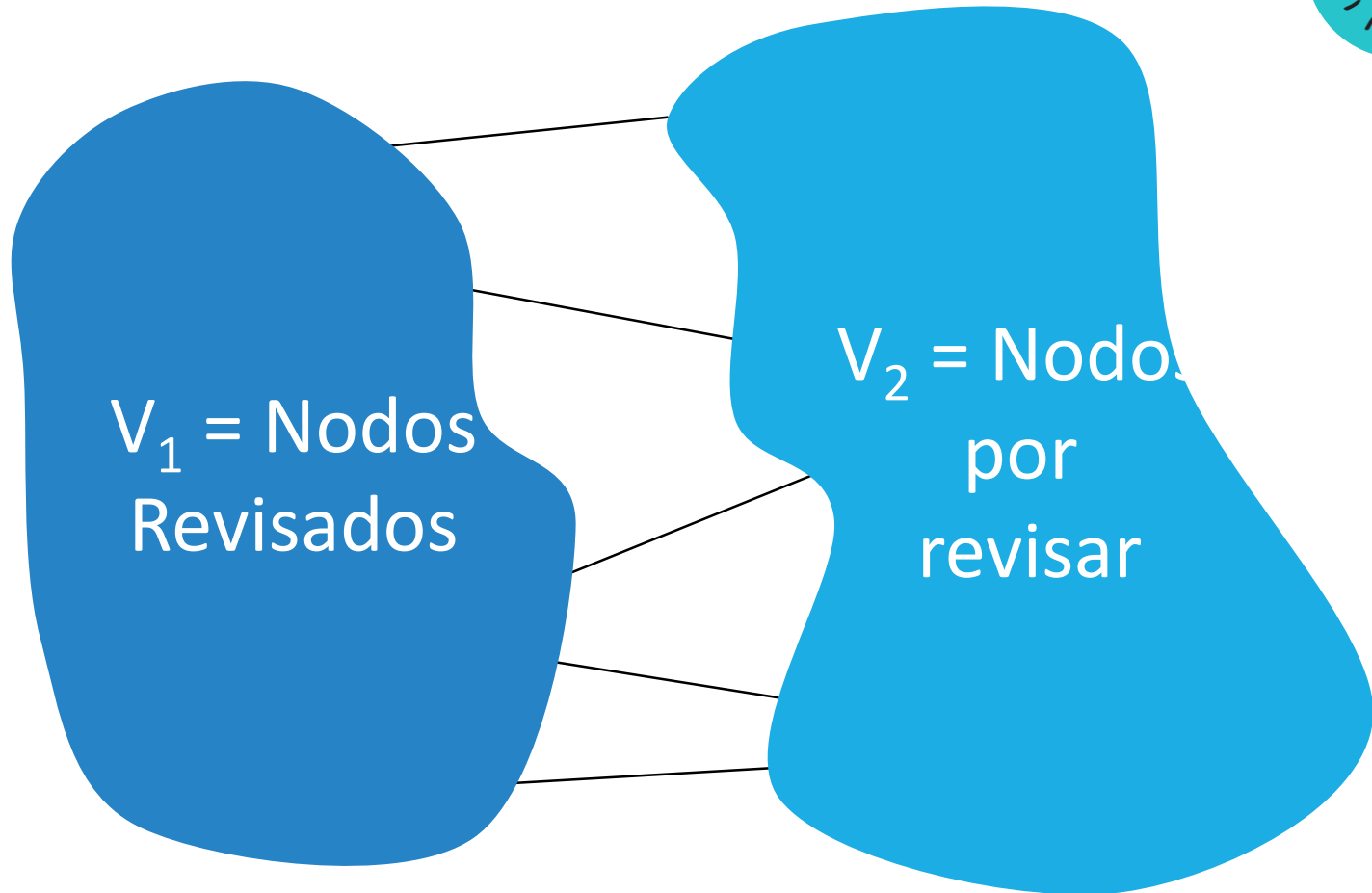


Si para cada corte la arista de menor costo está en un **MST**

... ¿cómo podemos encontrar un **MST**?

... ¿podremos construirlo una arista a la vez?

El plan general



¿Cuál debería ser el siguiente nodo a revisar?

Algoritmo de Prim

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

1. Sean $R = \{x\}$, $\bar{R} = V - R$, los nodos revisados y los que no.
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 a 2.

prim($G(V, E), x$):

$T \leftarrow \emptyset, \quad H \leftarrow$ una **cola de prioridades** únicamente con x

$x.key \leftarrow 0, \quad x.parent \leftarrow \emptyset,$

while $H \neq \emptyset$:

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

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

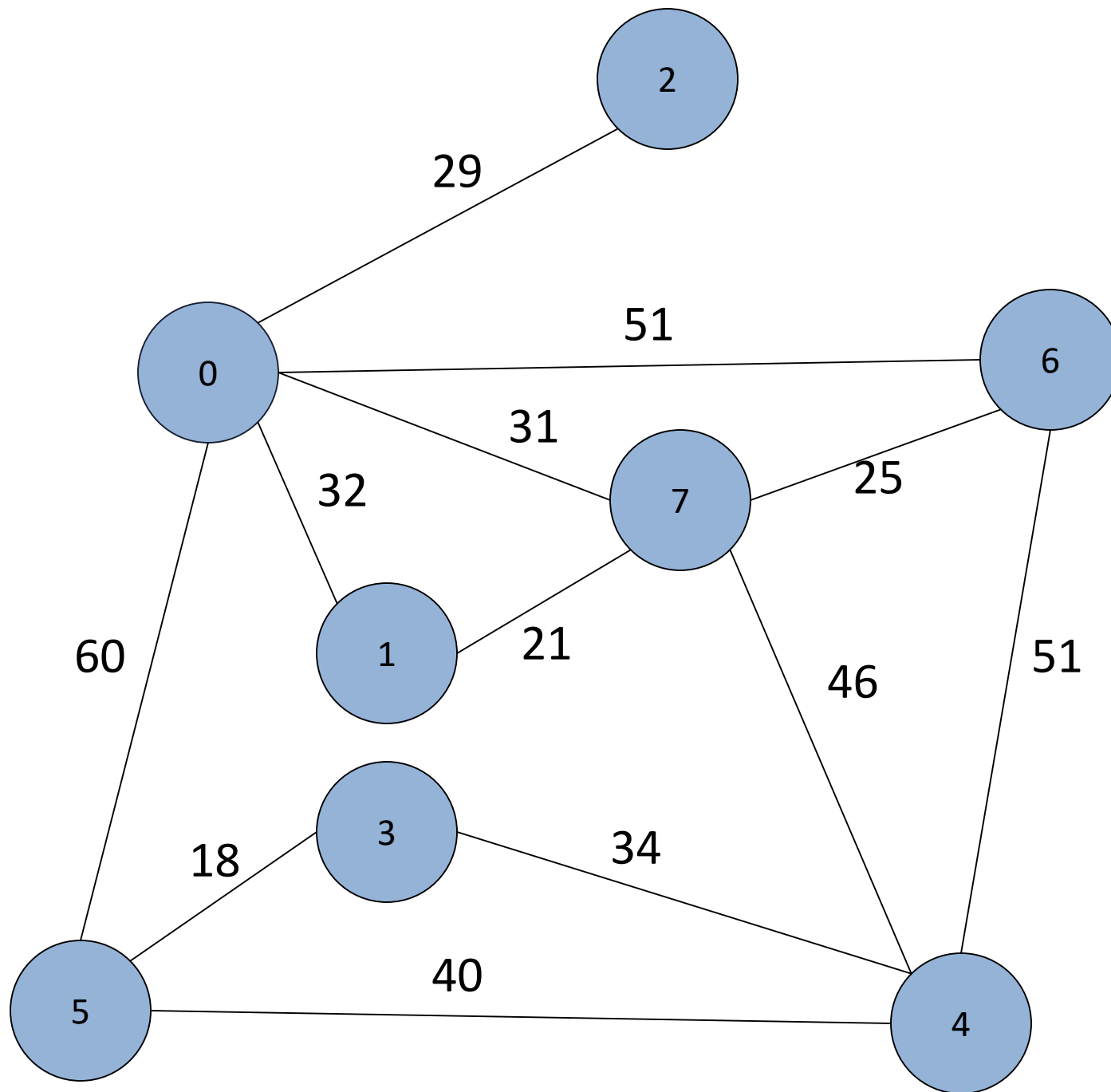
foreach vecino no pintado v de u :

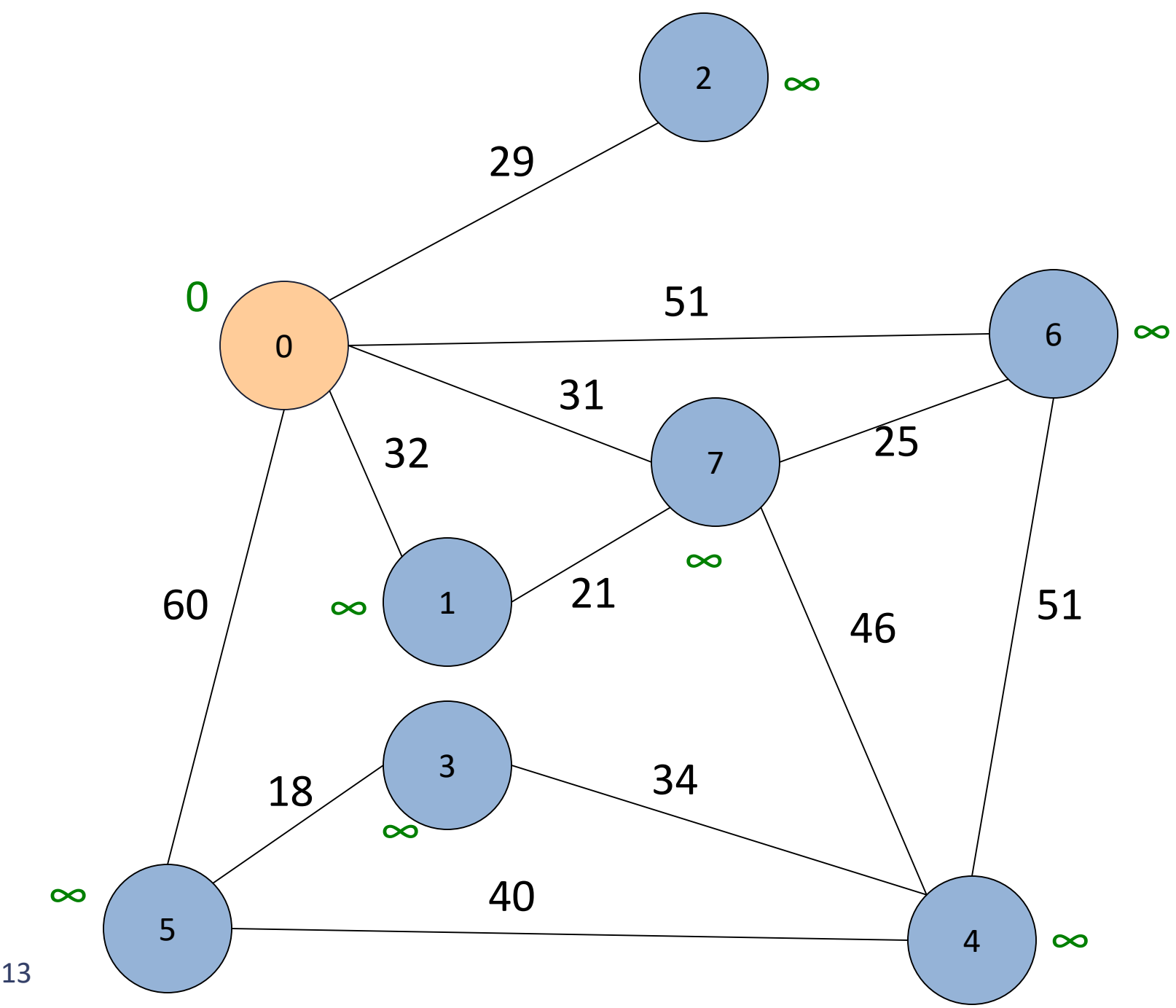
if $v \notin H$, insertar v en H

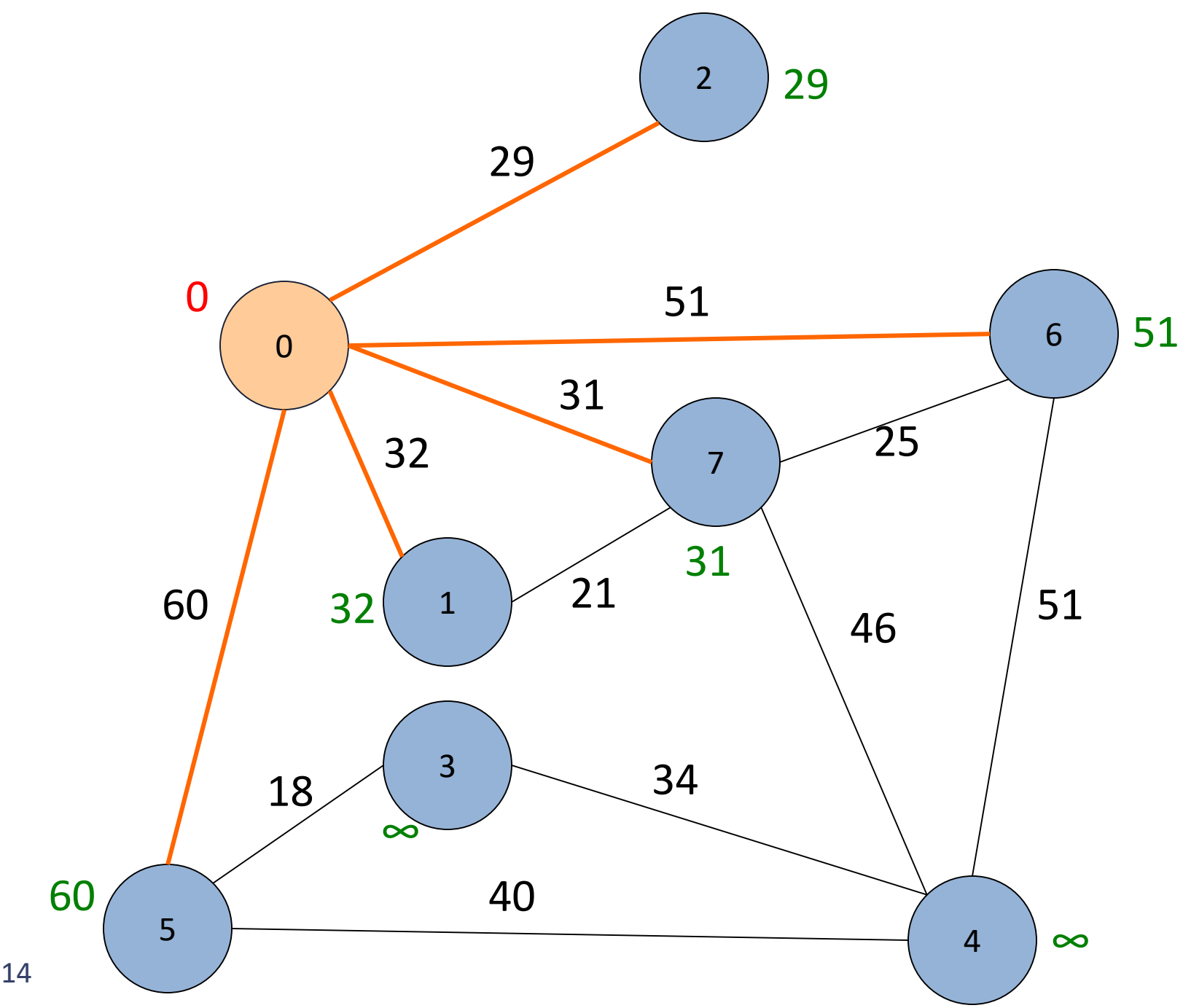
if $w(u, v) < v.key$:

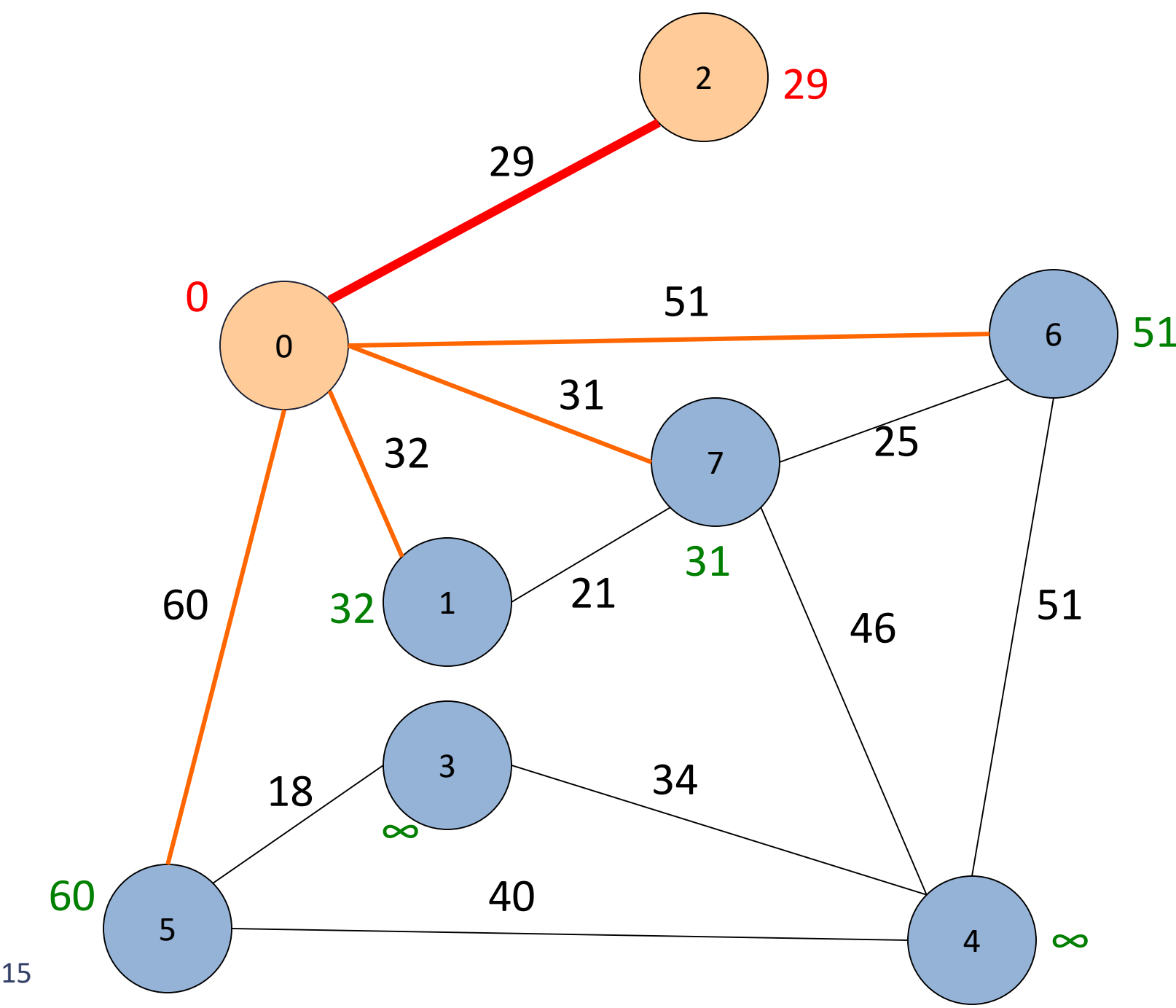
$v.key \leftarrow w(u, v), \quad v.parent \leftarrow u$

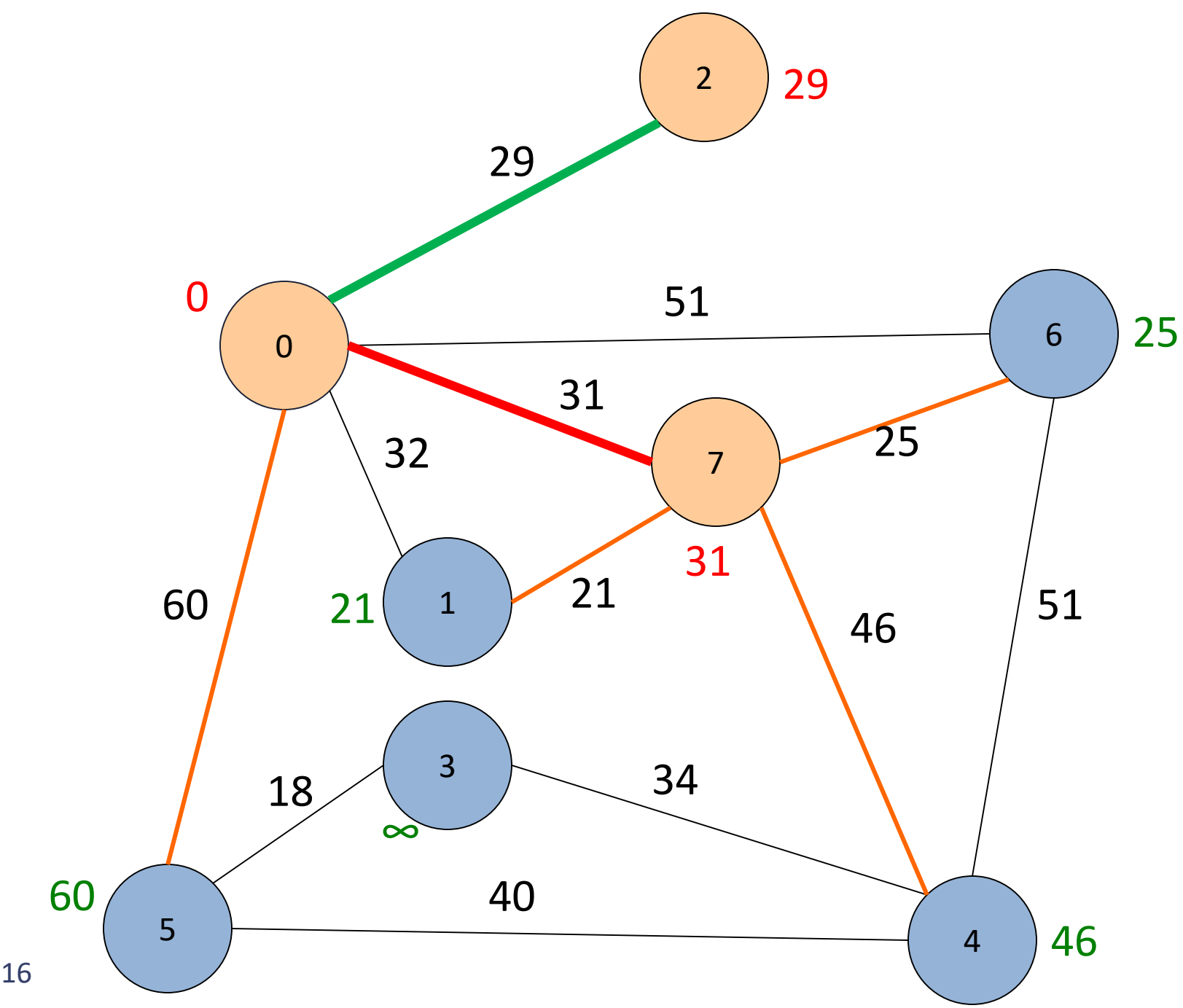
return T

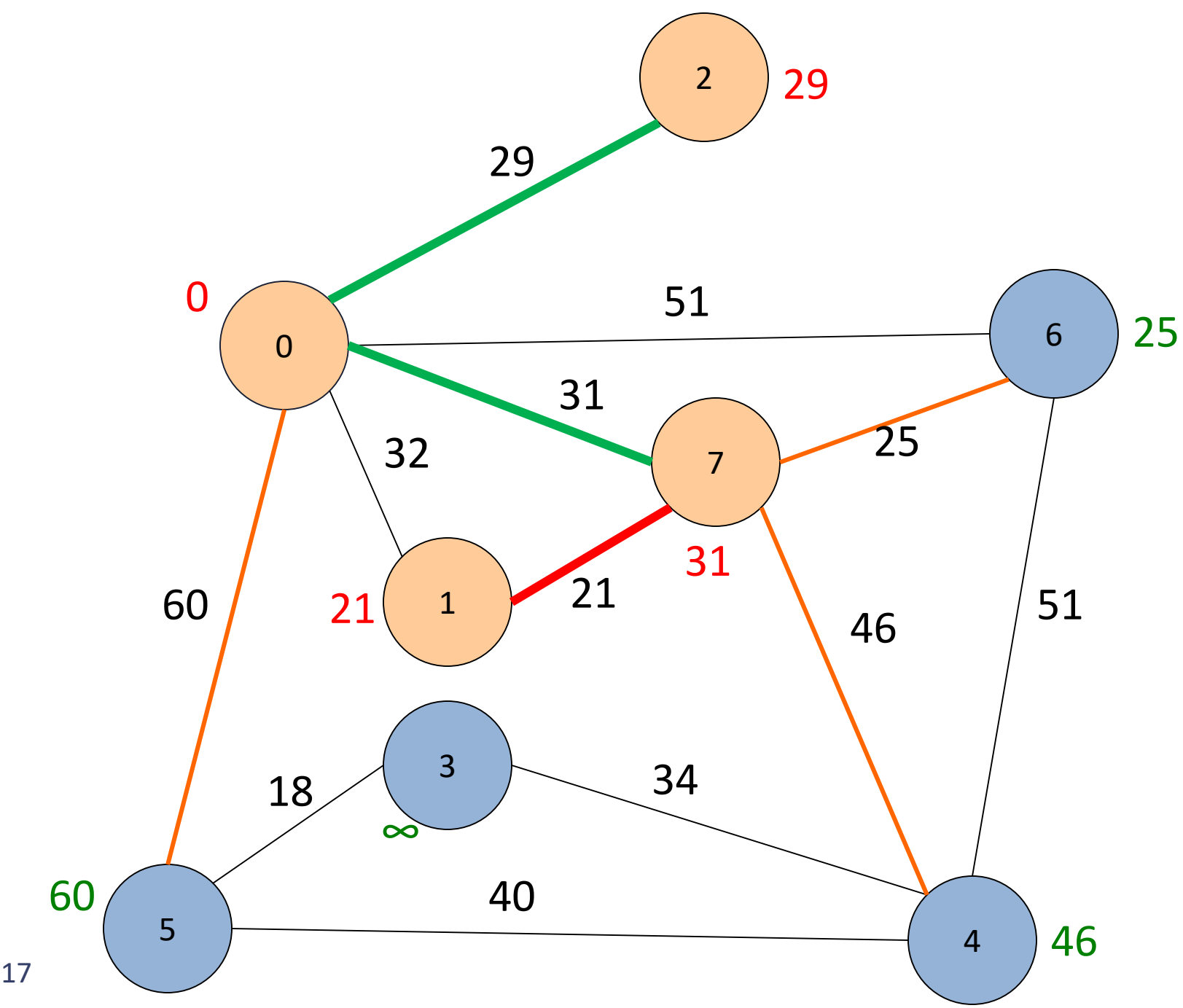


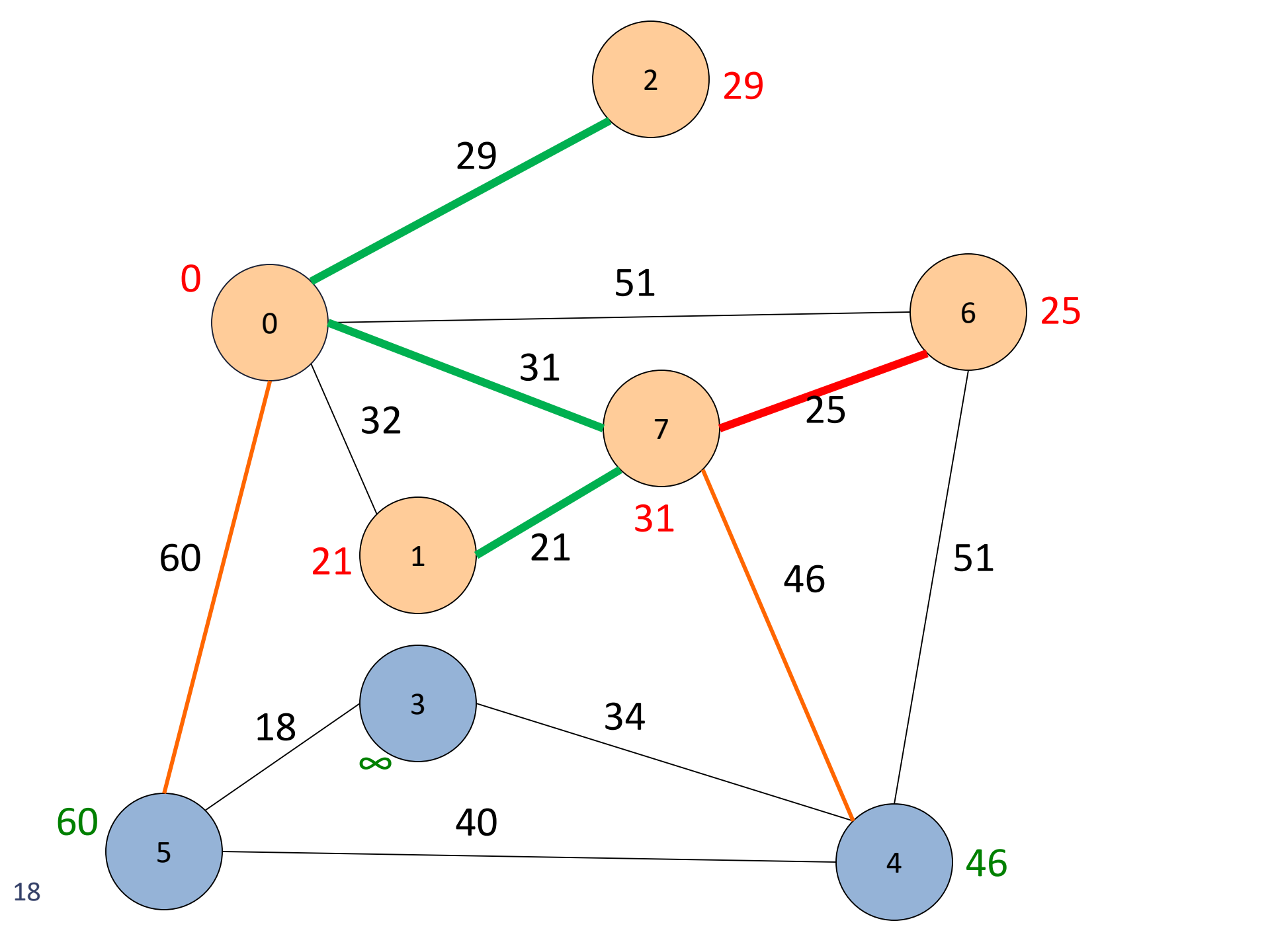


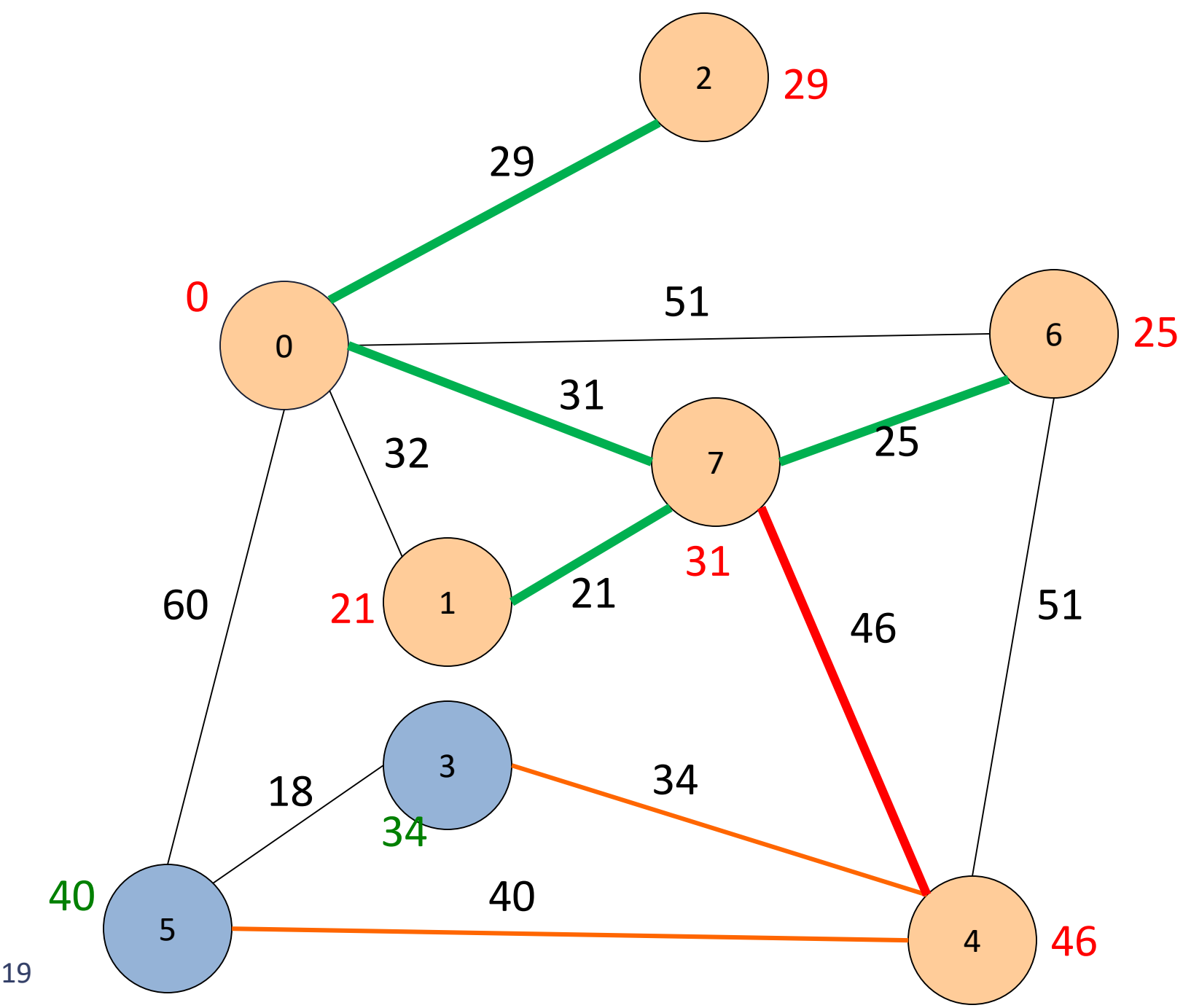


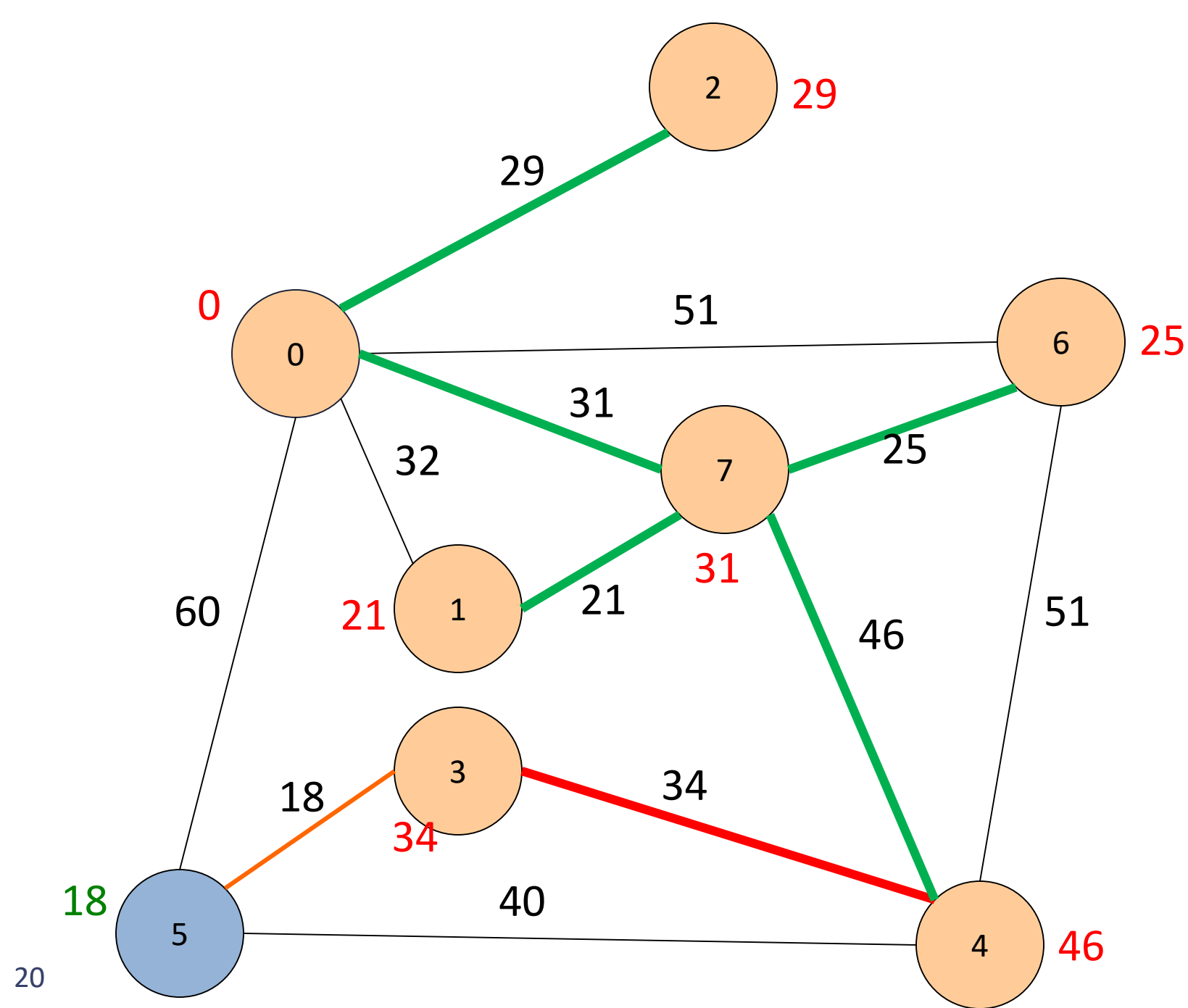


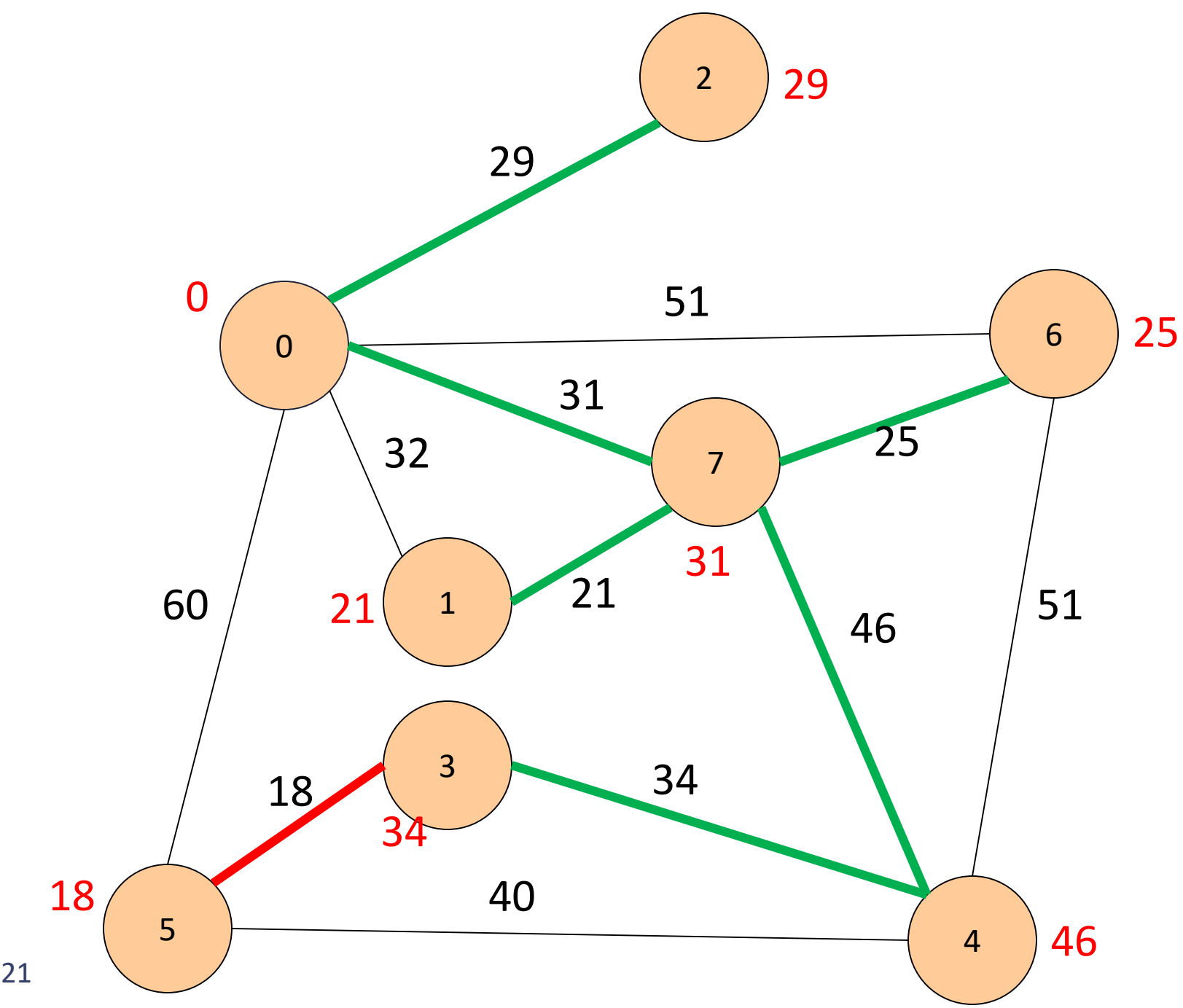


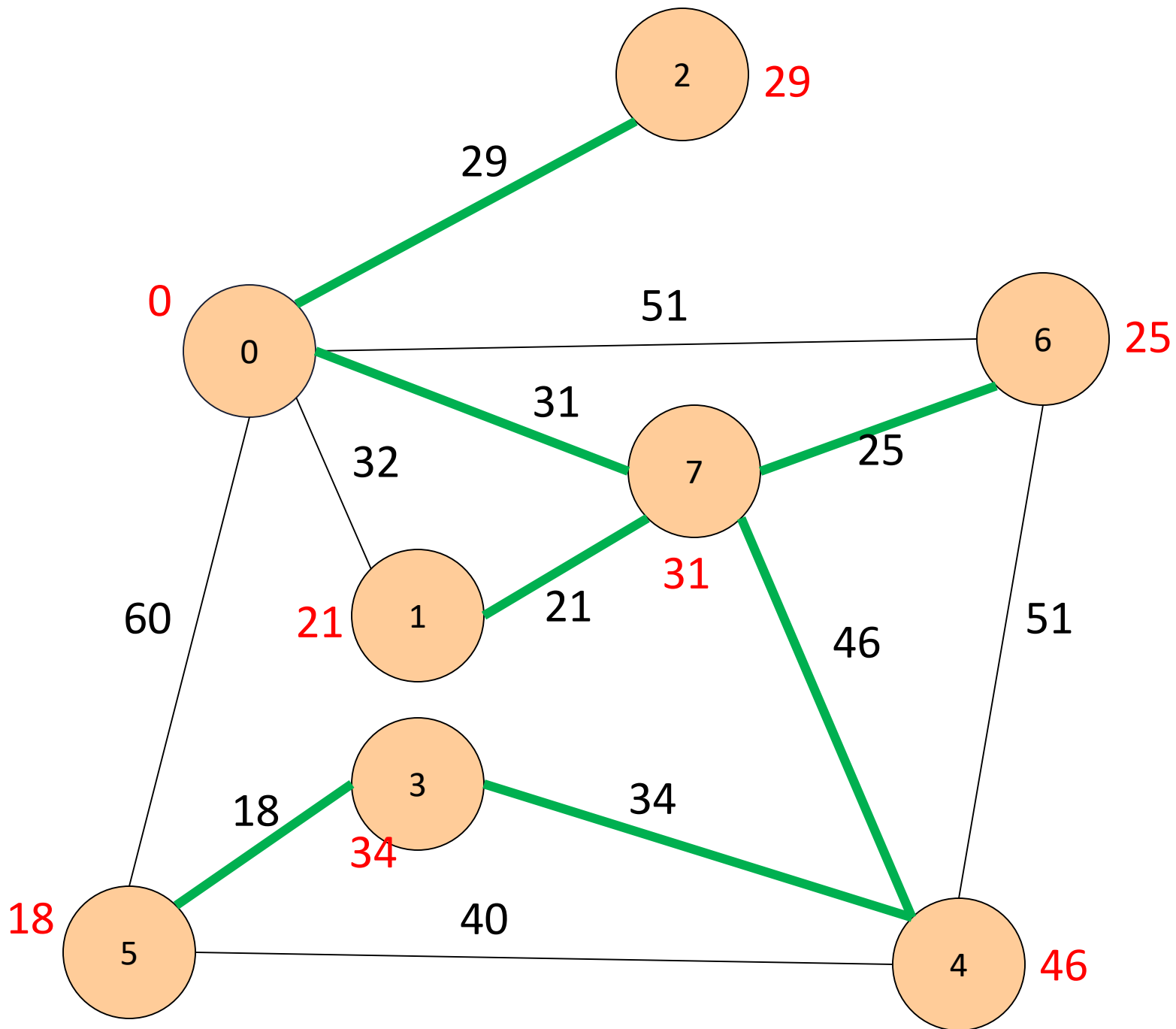












Corrección



¿Cómo demostramos que Prim es correcto?

¿Cuál es su complejidad?

Algoritmos *codiciosos*

Esta estrategia algorítmica es conocida como **codiciosa**

En cada paso, el algoritmo escoge un **óptimo local**

... con la esperanza de llegar al **óptimo global**

Optimalidad *codiciosa*



Los algoritmos *greedy* son muy veloces

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

¿Qué debe cumplirse en un problema para esto?