

Propiedades del MST

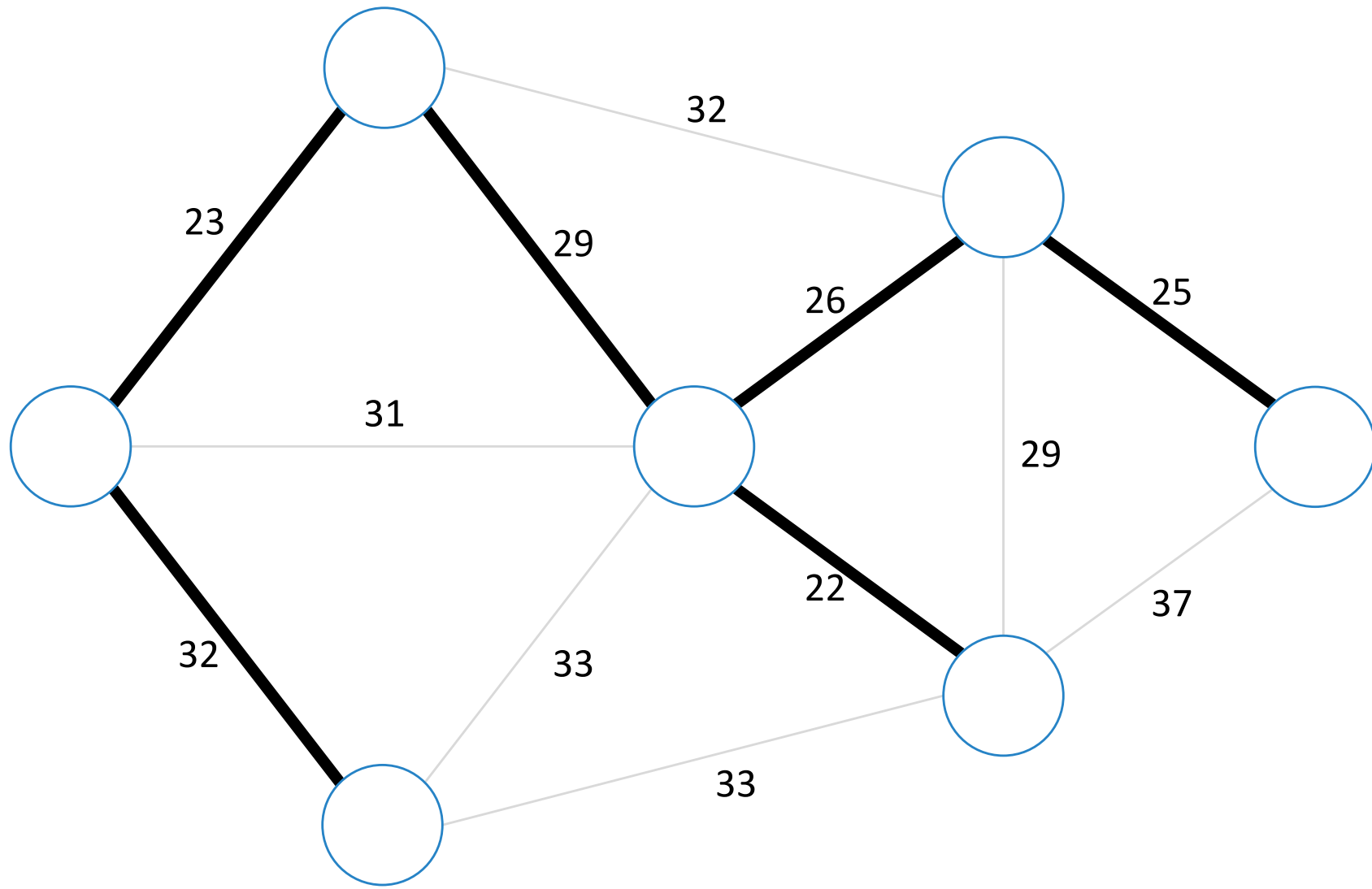


¿Hay alguna arista que **siempre** pertenezca a un **MST**?

¿Se cumple esto recursivamente? ¿En qué casos?

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

Ejemplo de un MST para un grafo



El algoritmo de Kruskal

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

Ordenar E por costo, de menor a mayor

$T \leftarrow \emptyset$

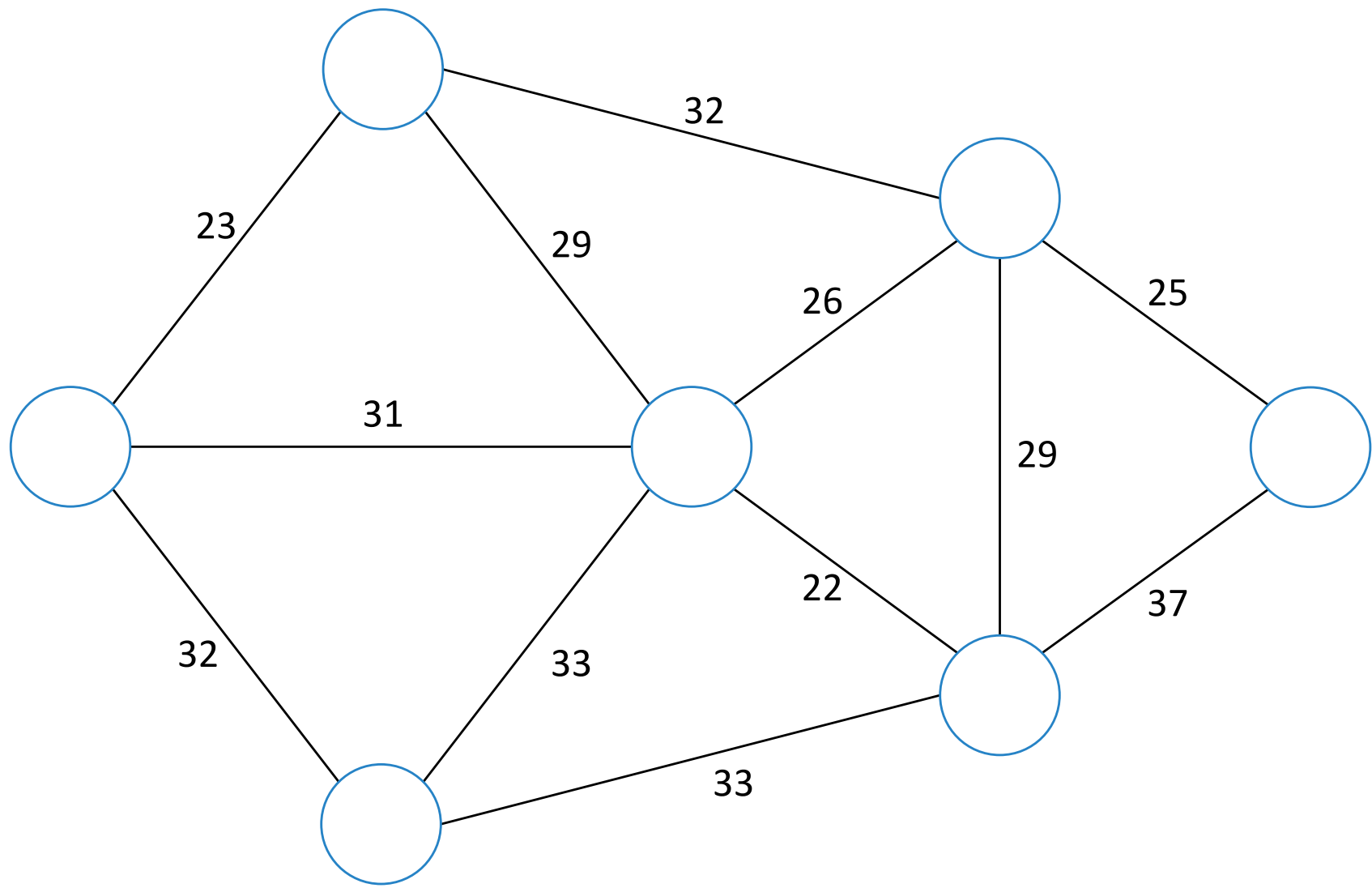
foreach $e \in E$:

if agregar e a T no forma un ciclo:

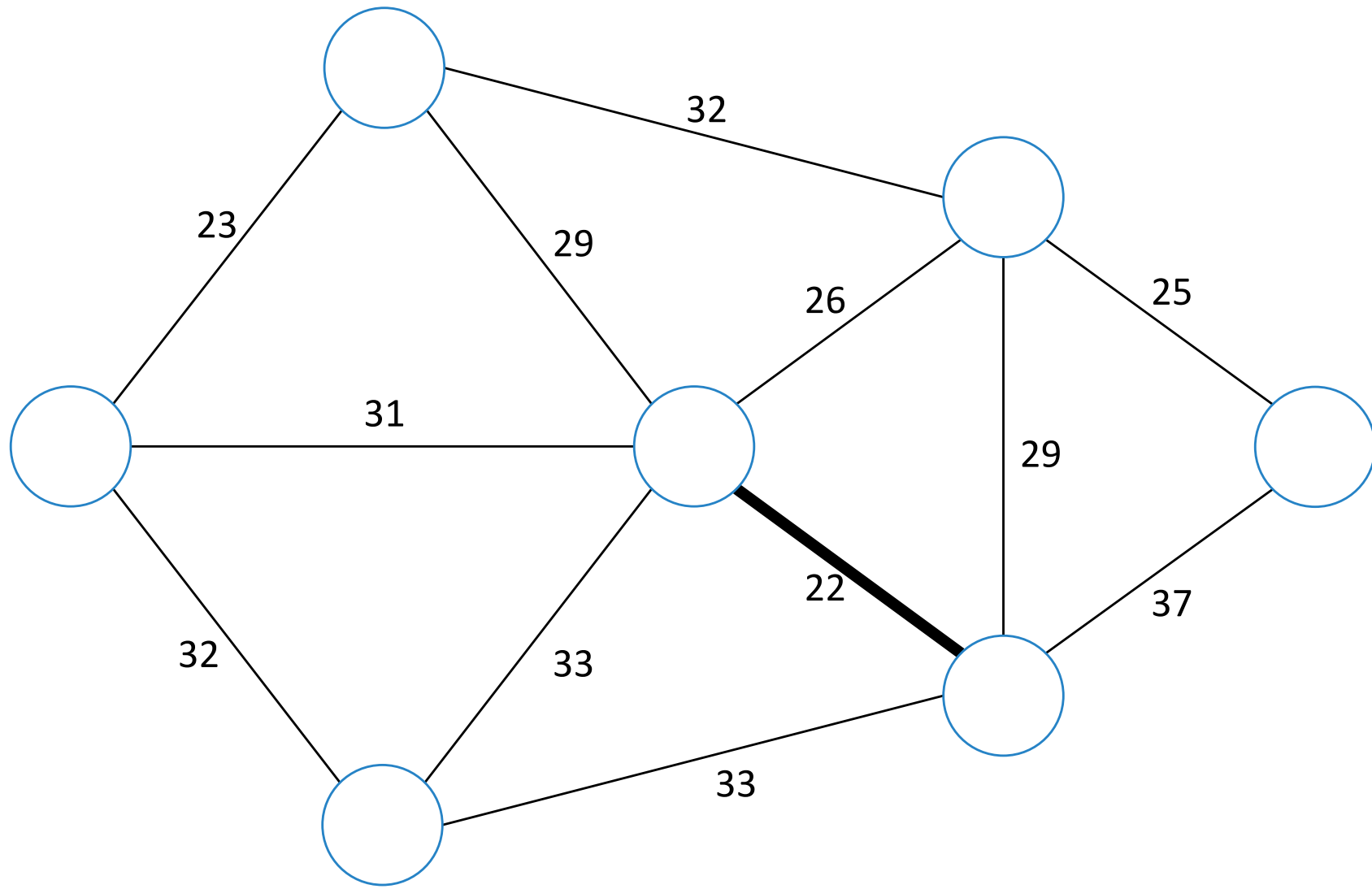
Agregar e a T

return T

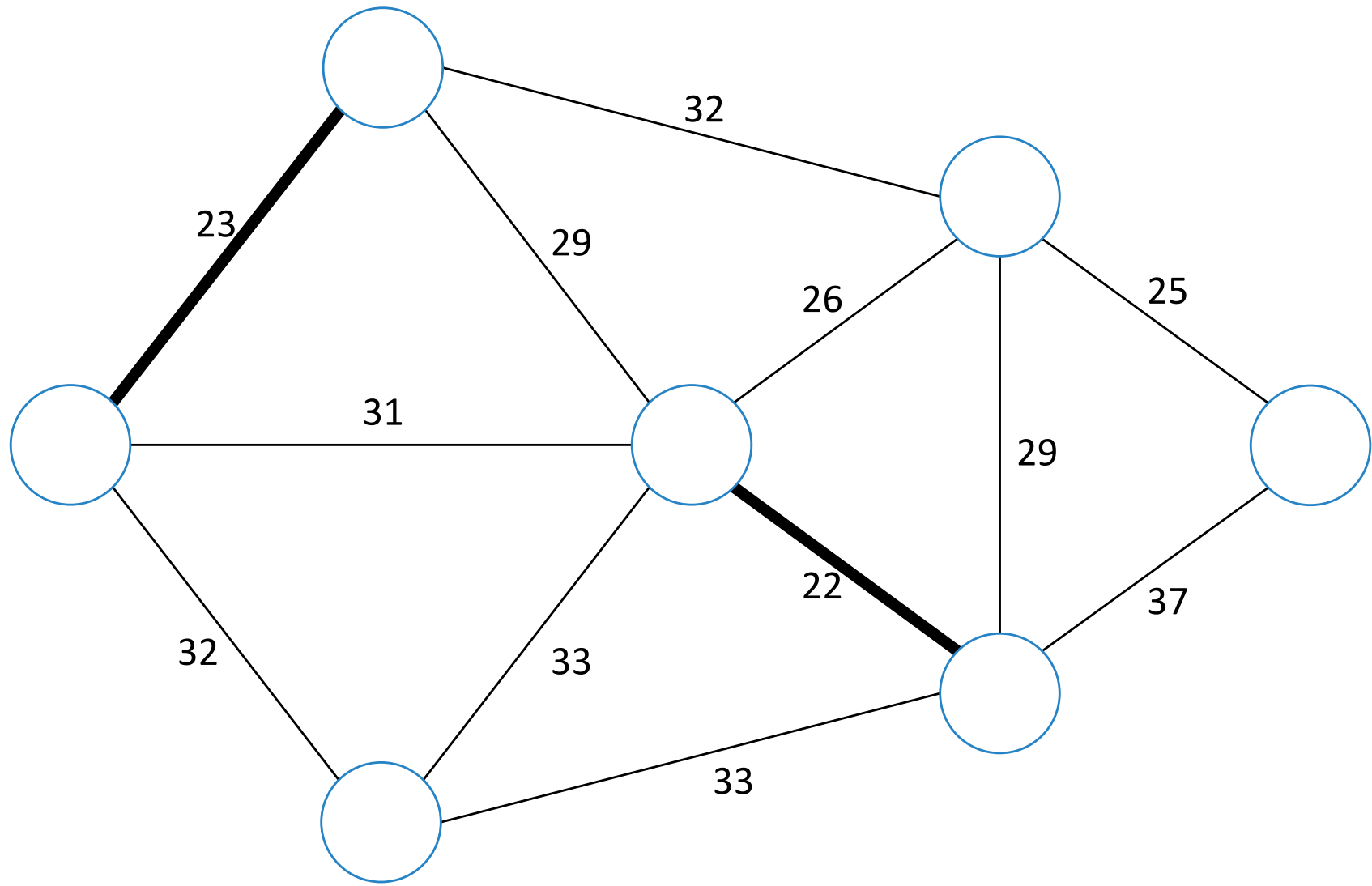
kruskal en acción



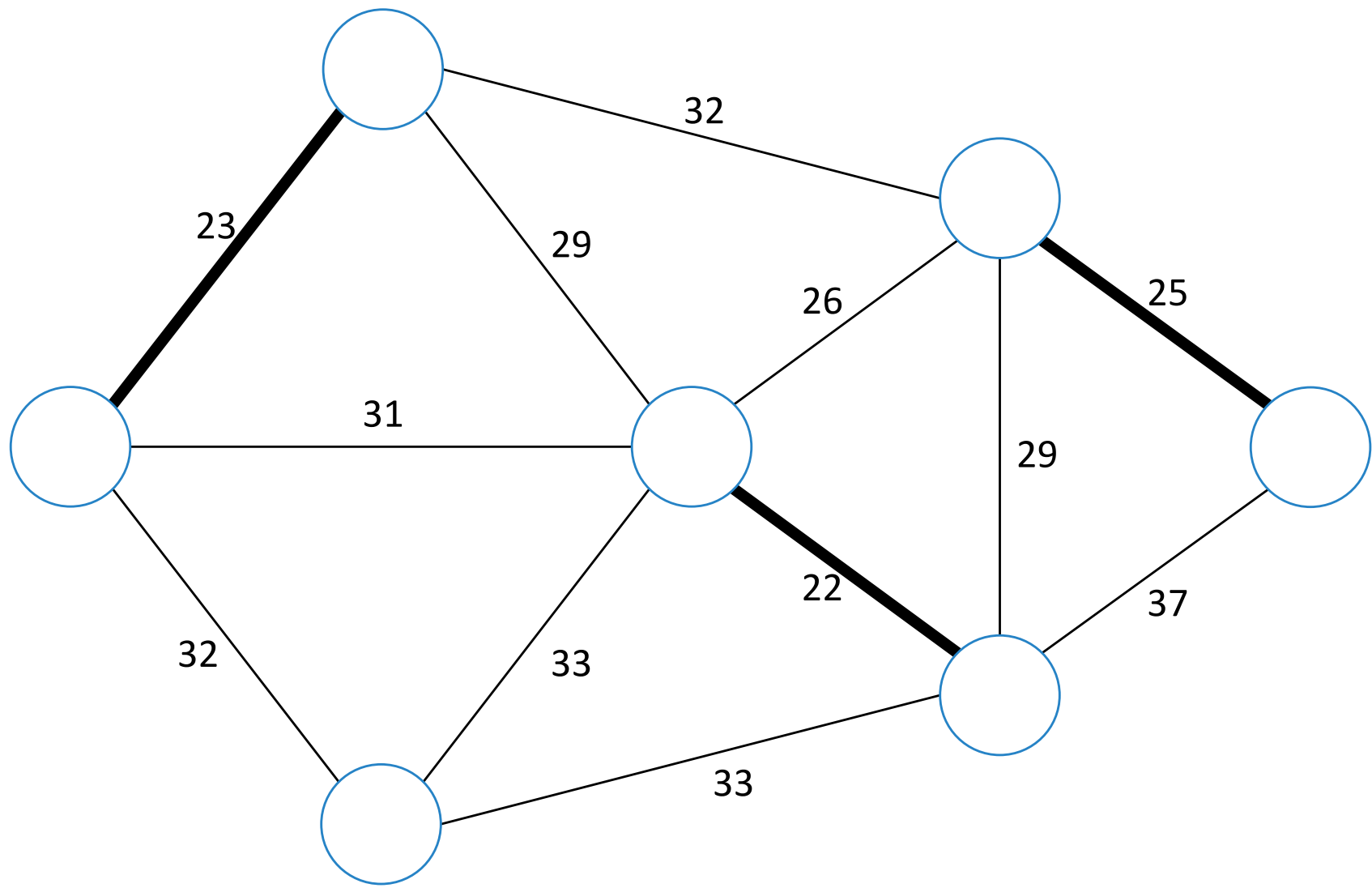
Partimos probando la arista de menor costo



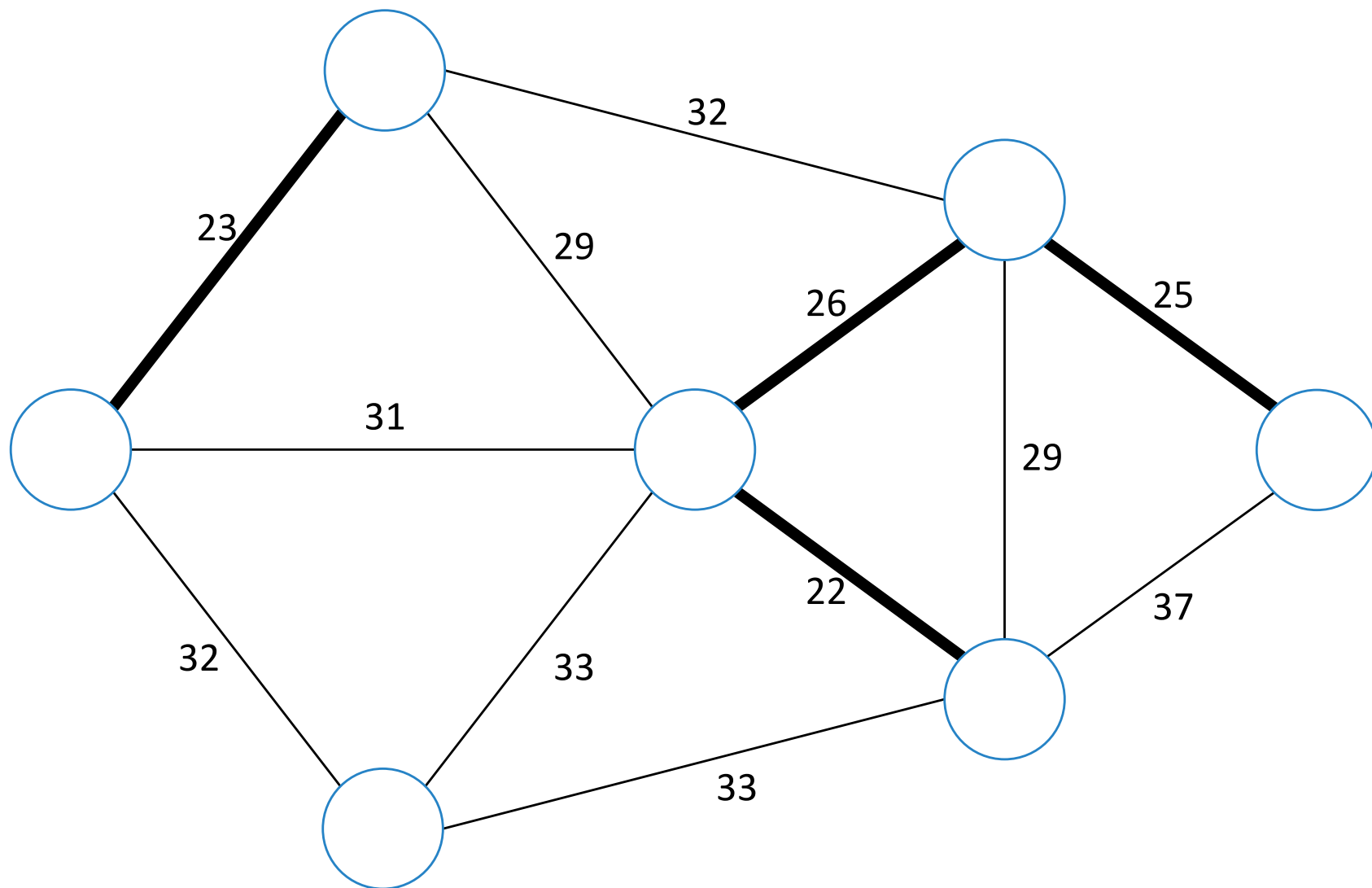
... y así sucesivamente



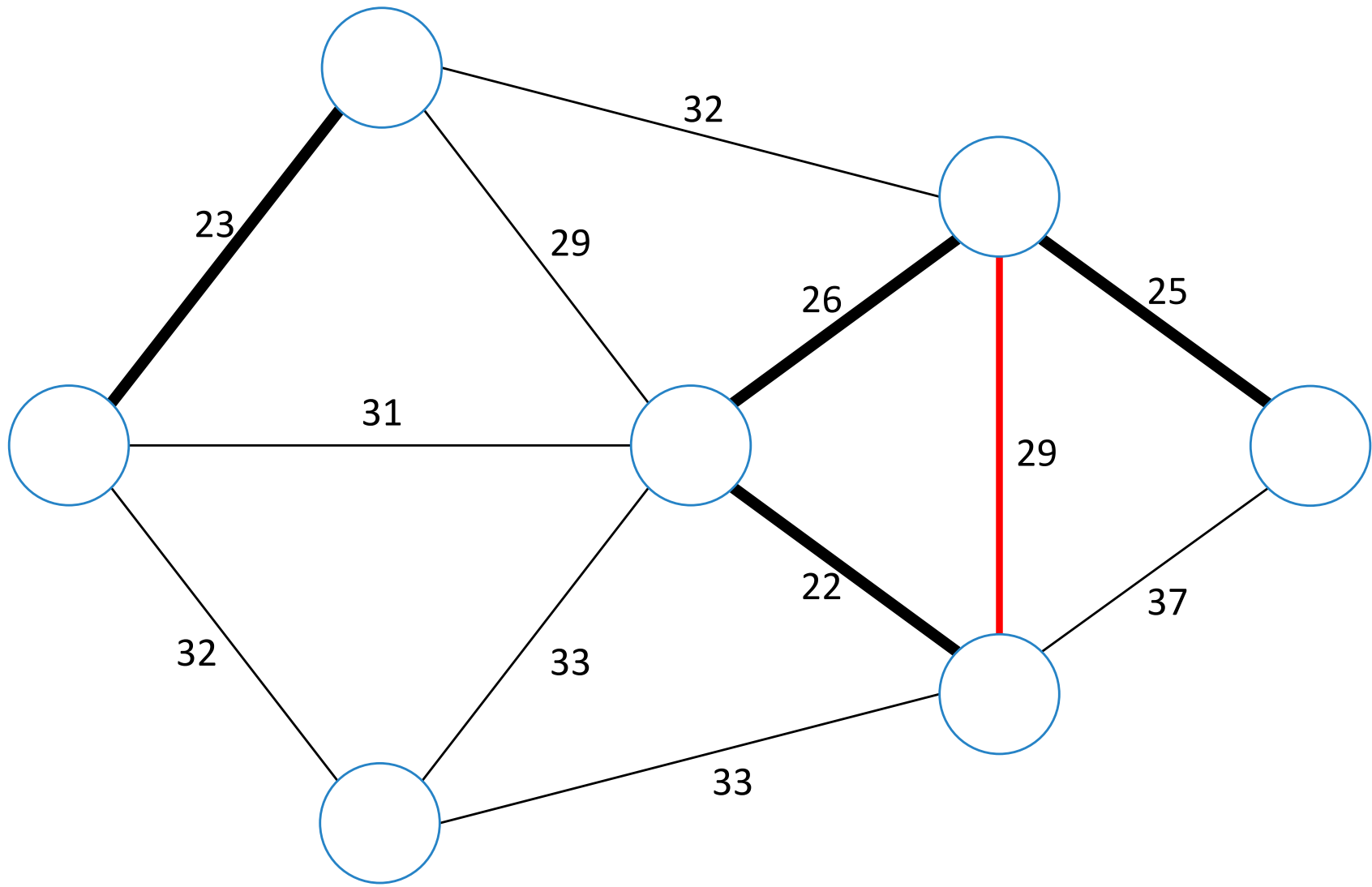
...



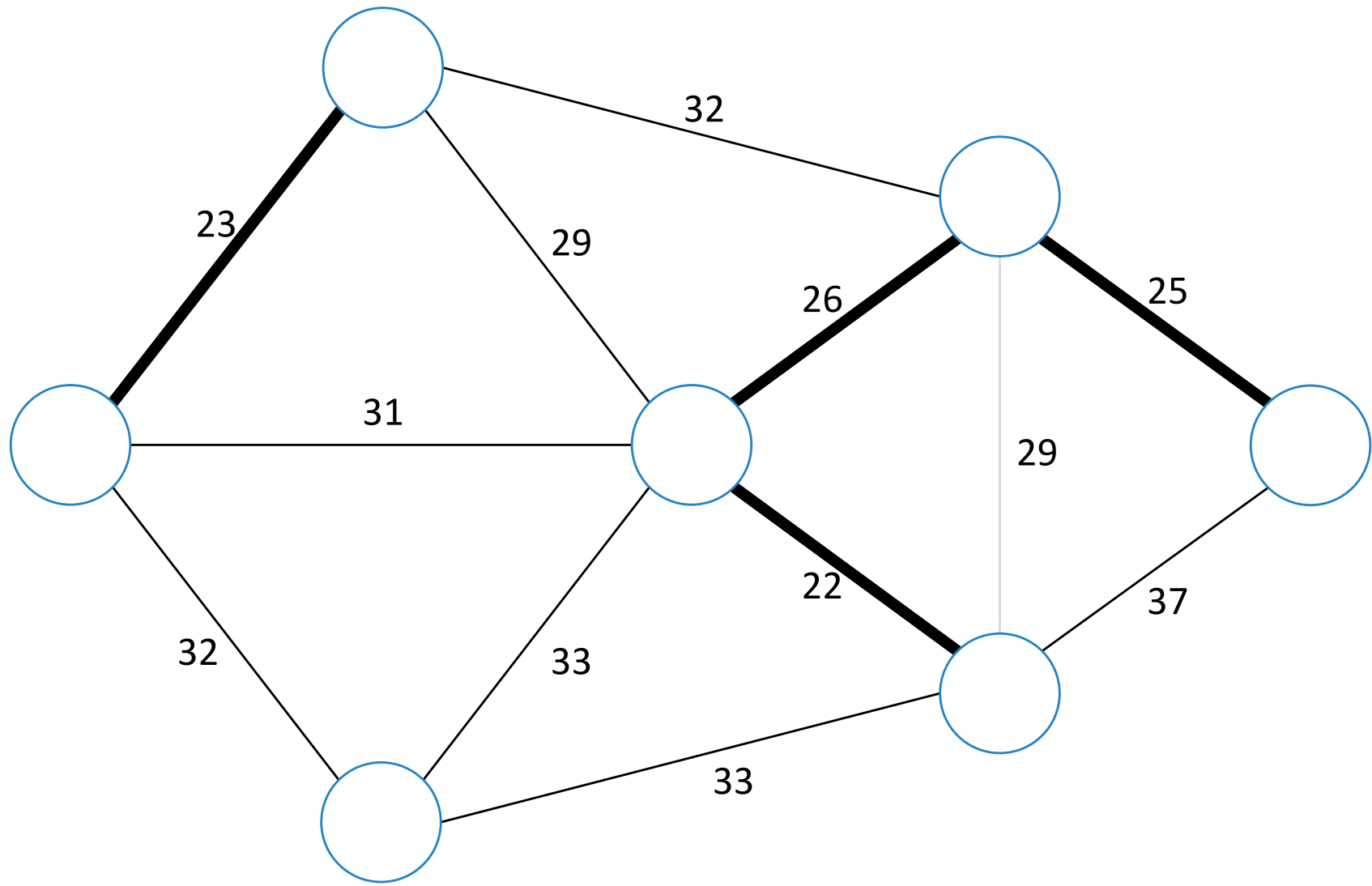
...



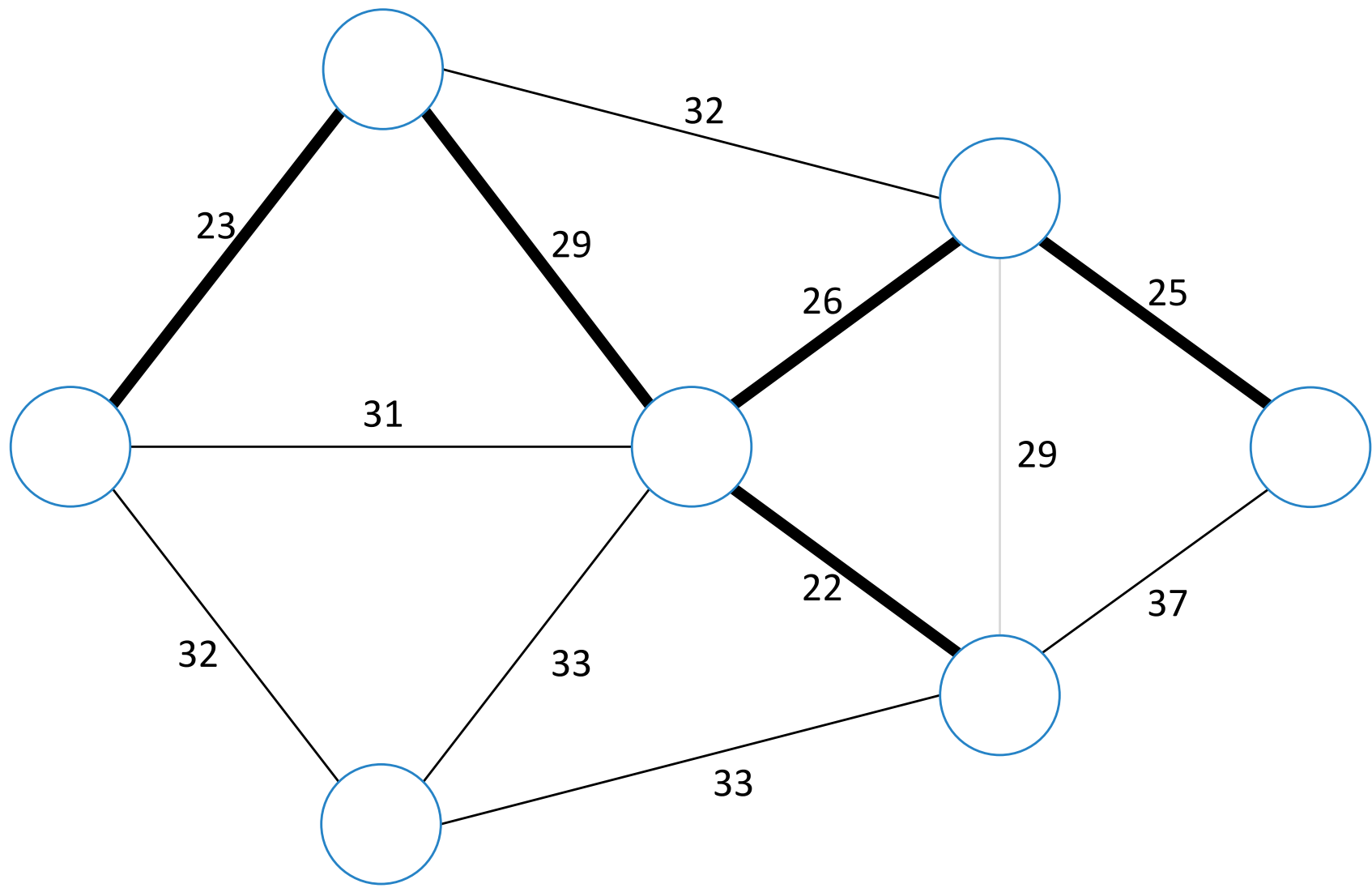
“probar” significa asegurarse de que la nueva arista no forme un ciclo



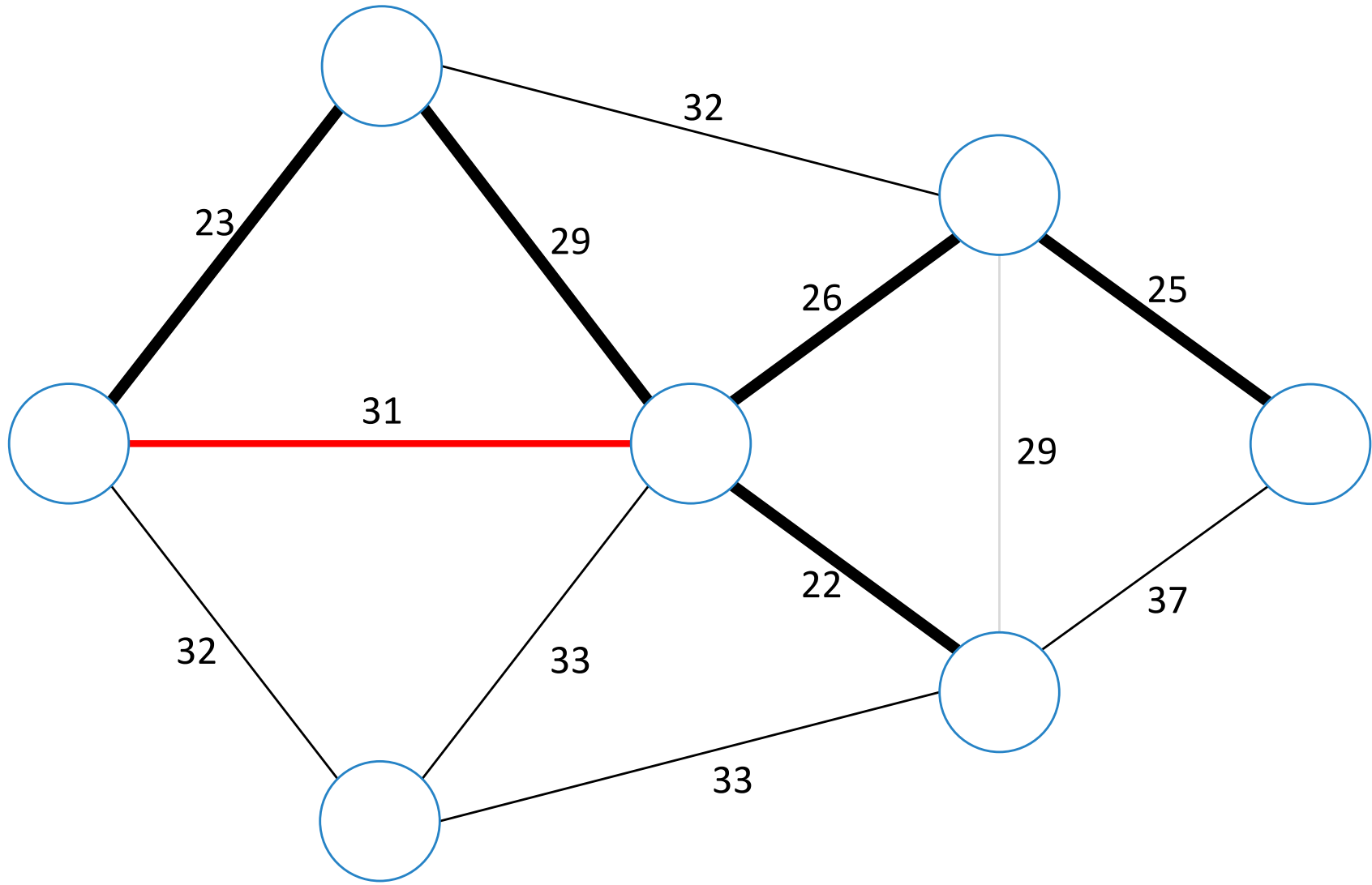
...



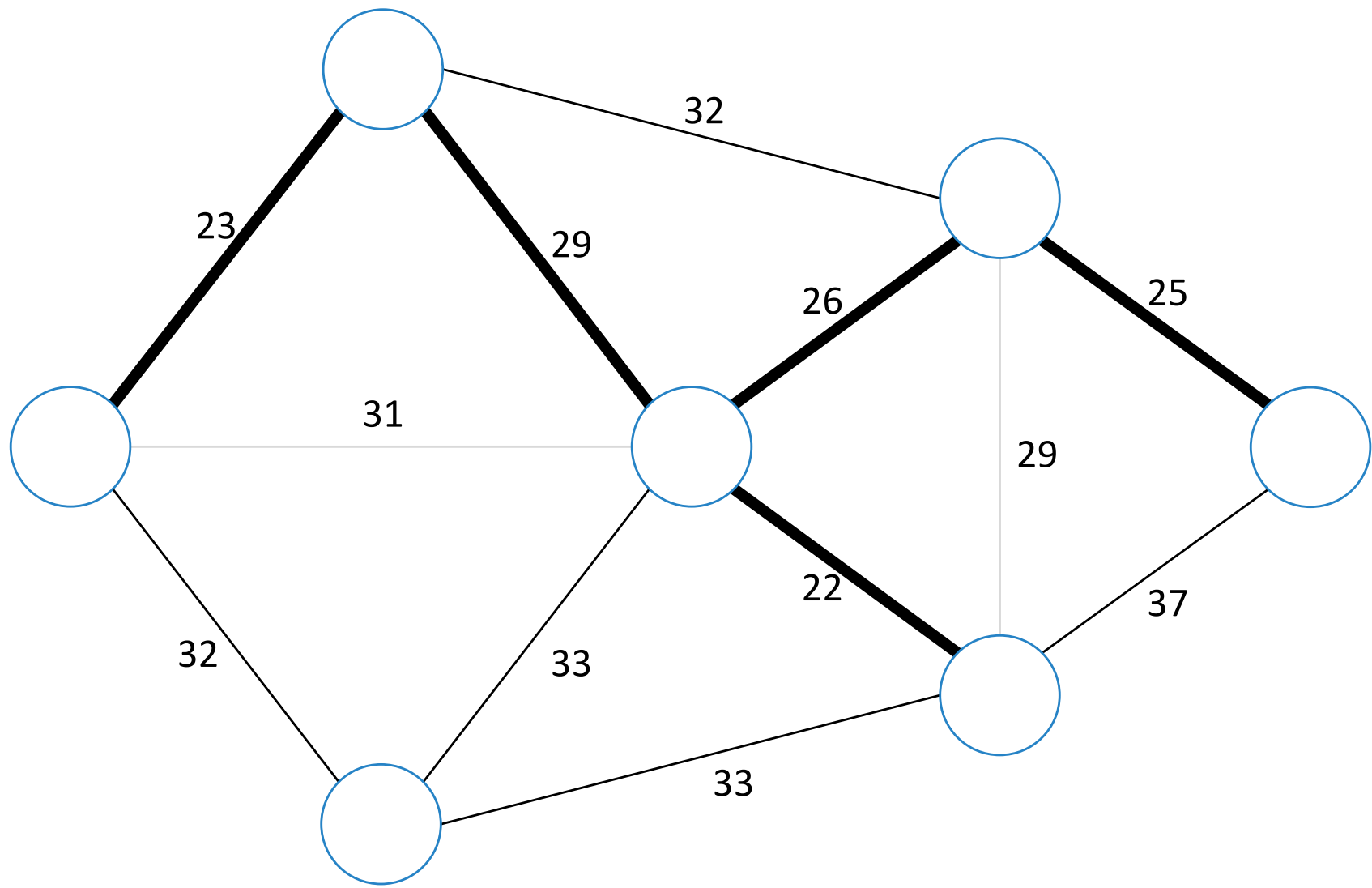
...



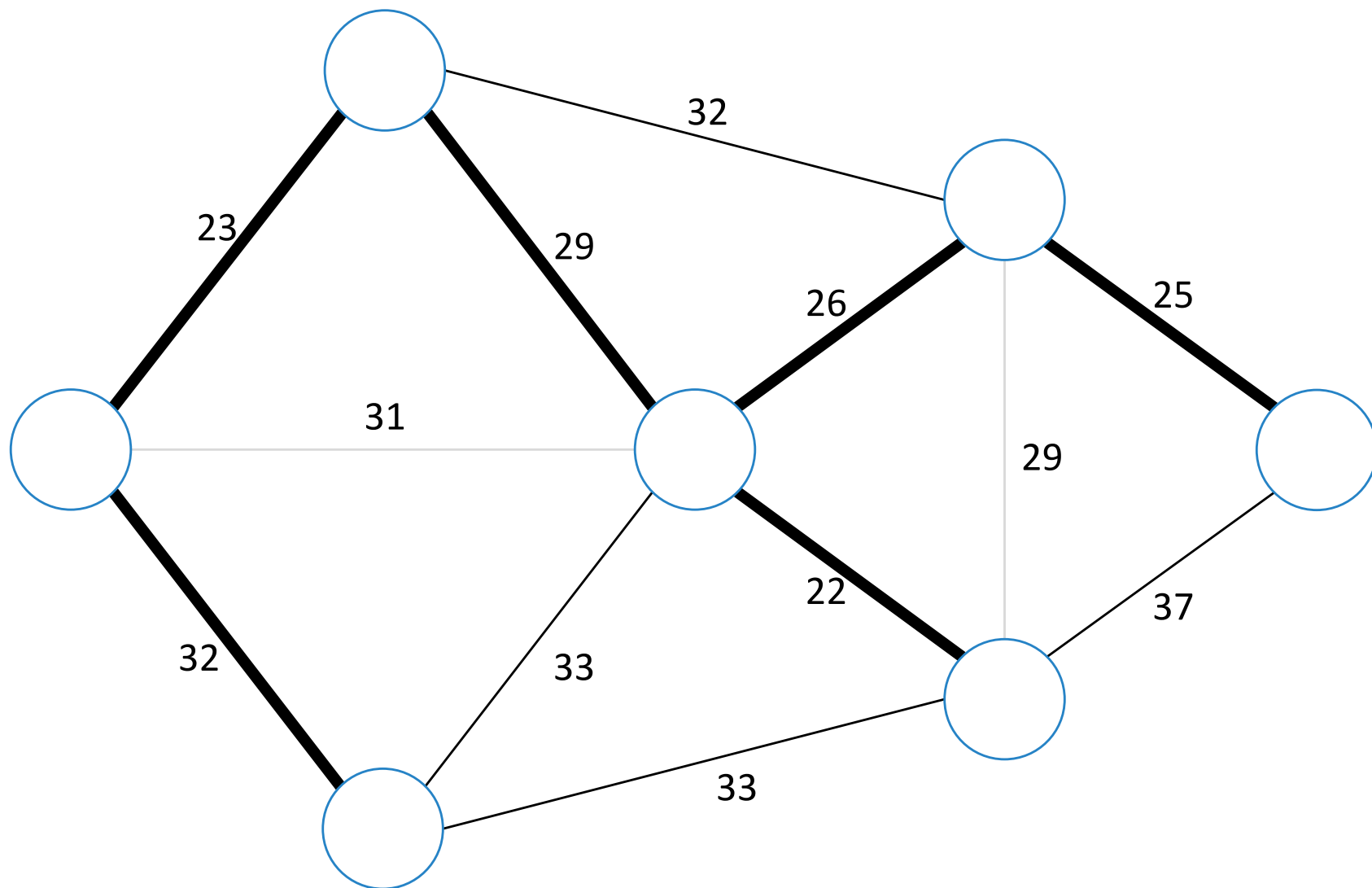
Si la nueva arista forma un ciclo,
entonces la descartamos



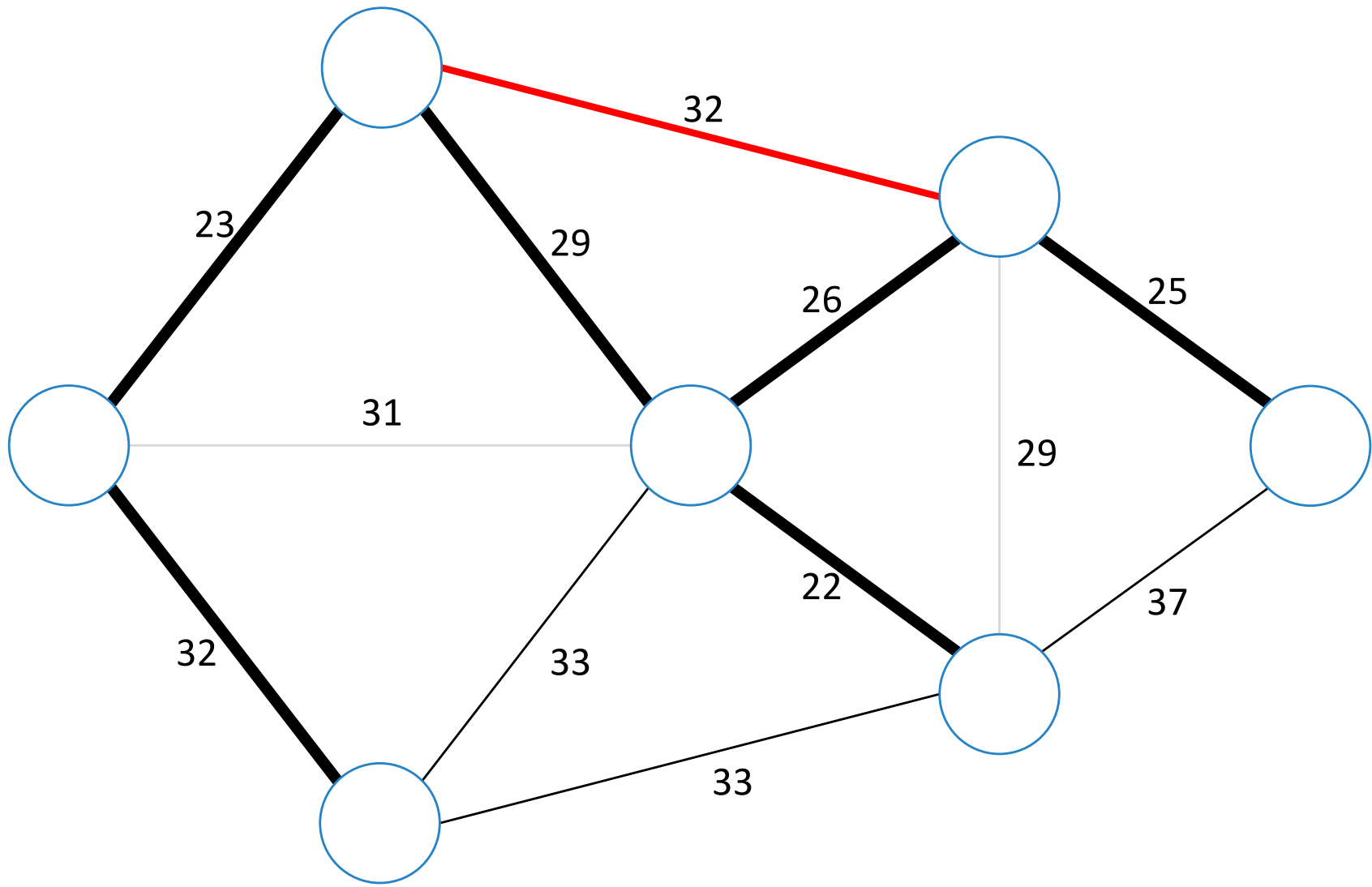
...



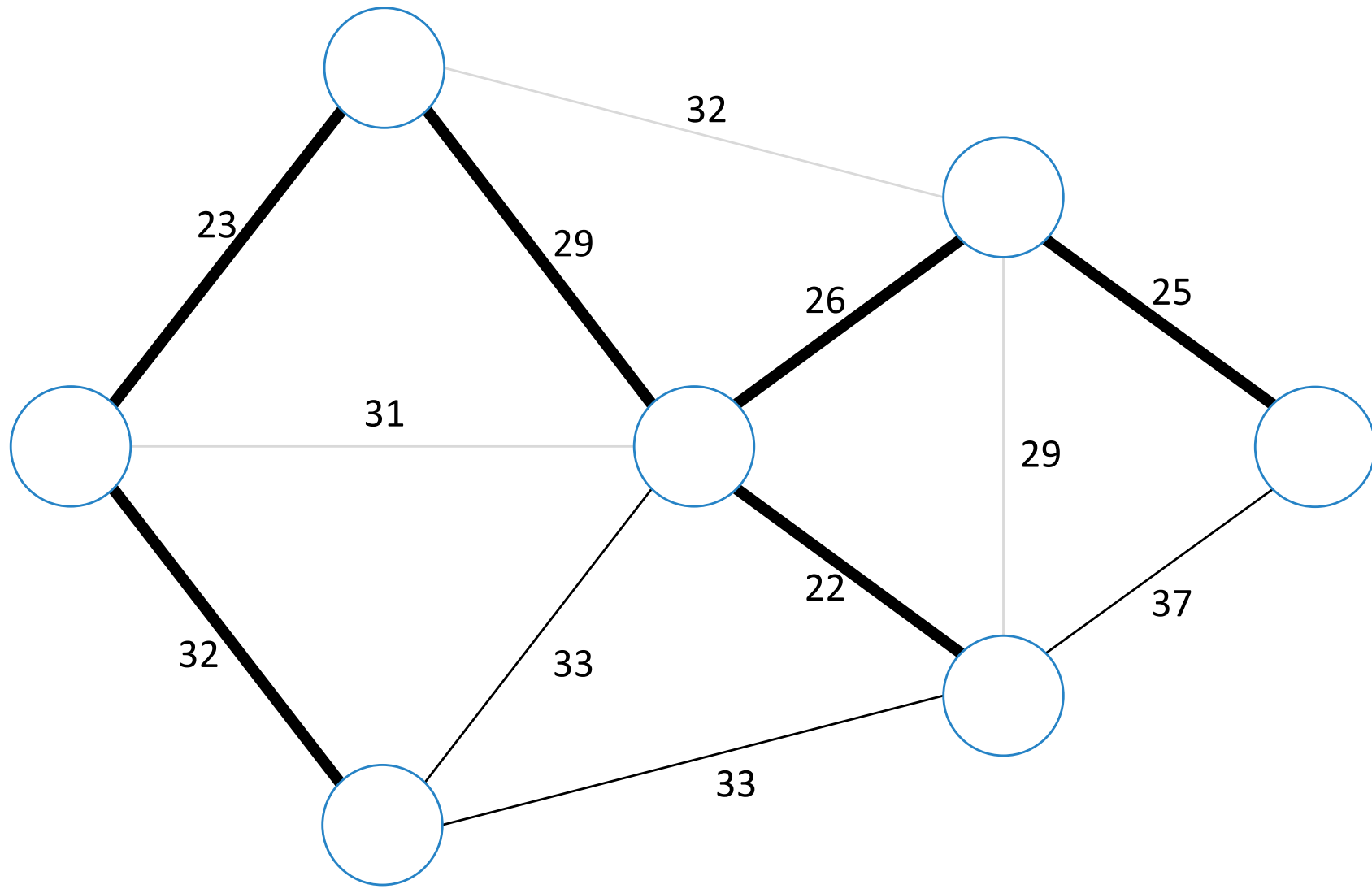
...



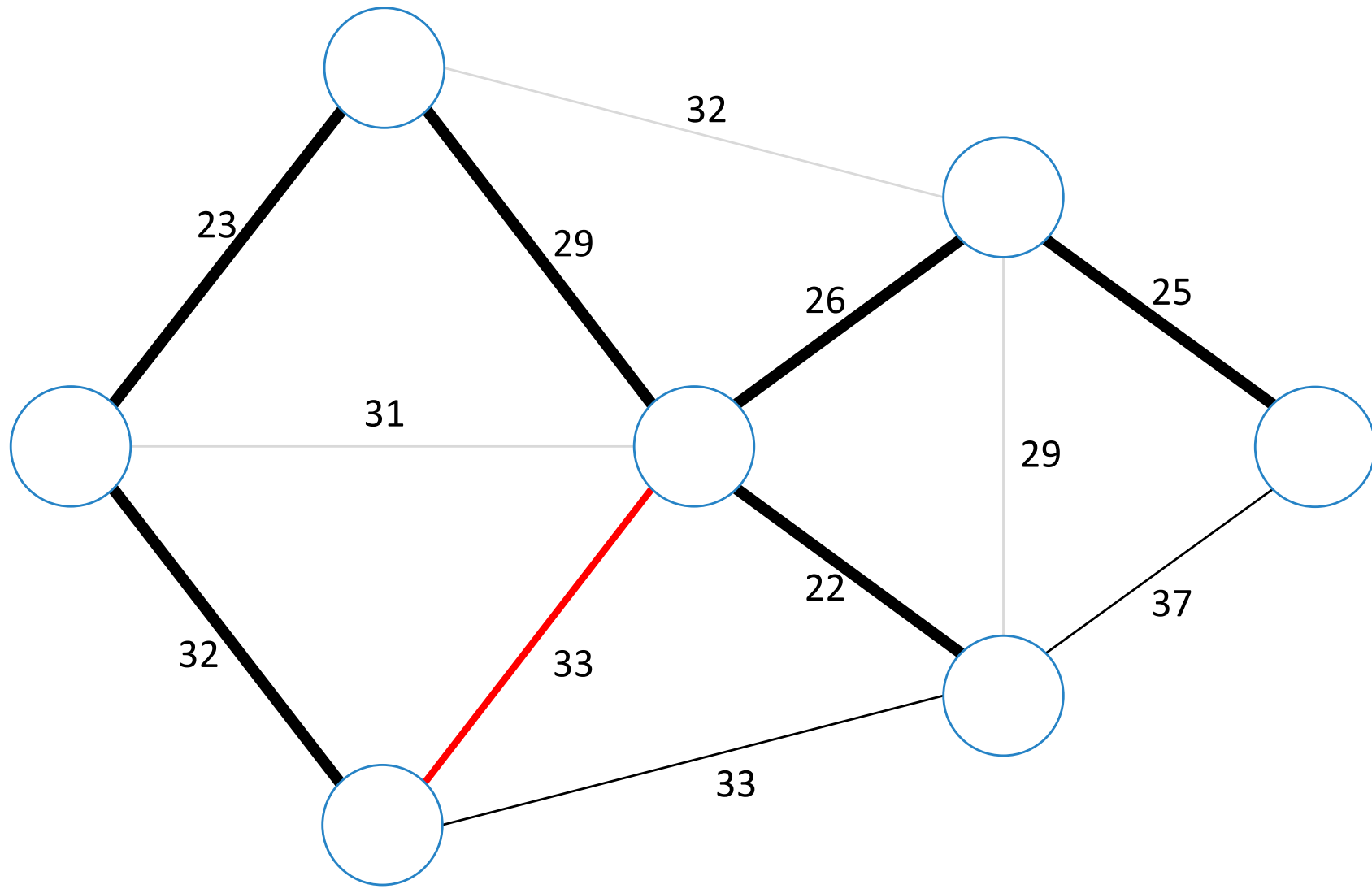
Si el grafo tiene $|V|$ vértices, entonces el MST tiene $|V|-1$ aristas



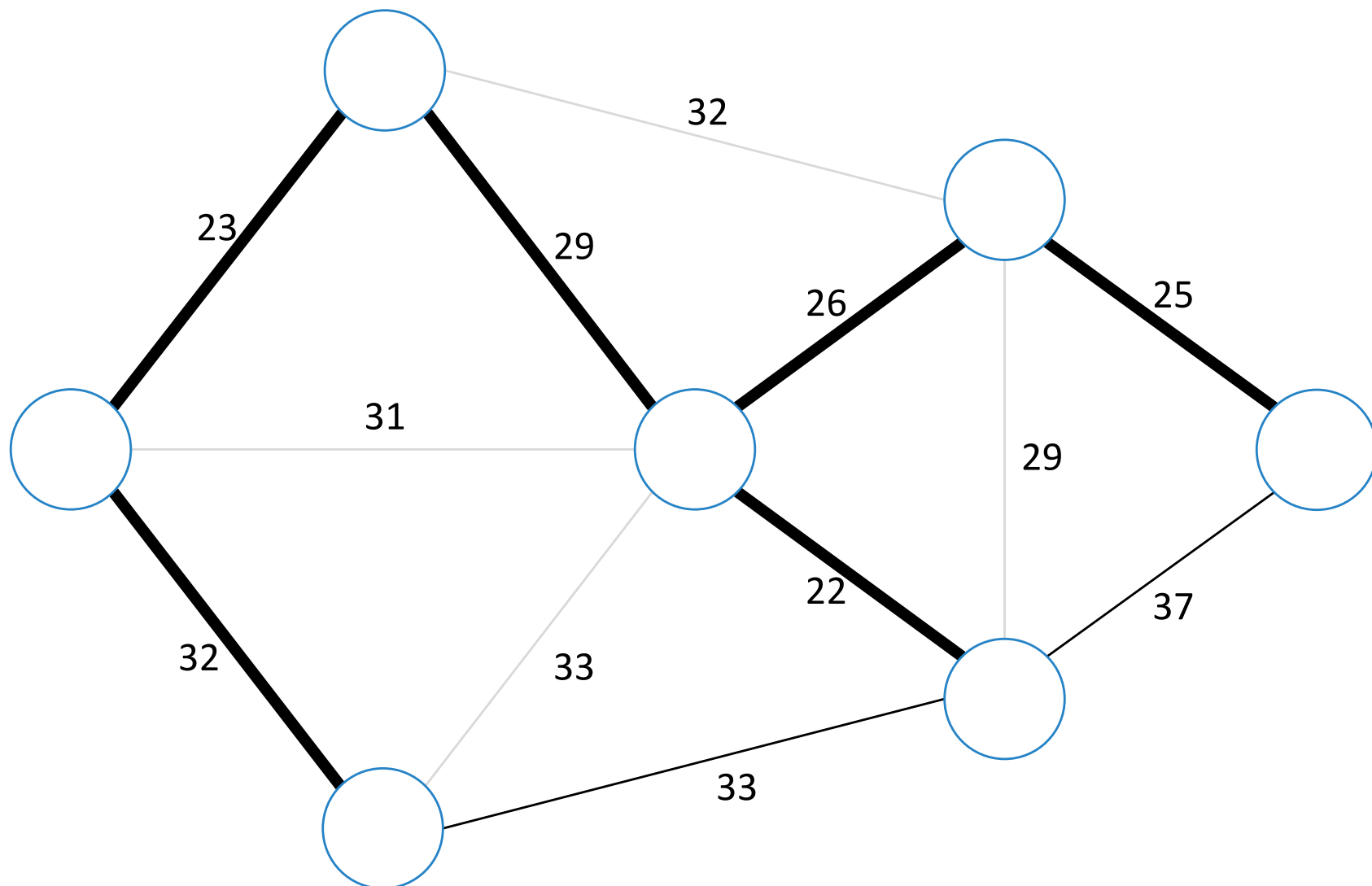
A partir de ahí,
todas las aristas restantes forman ciclos



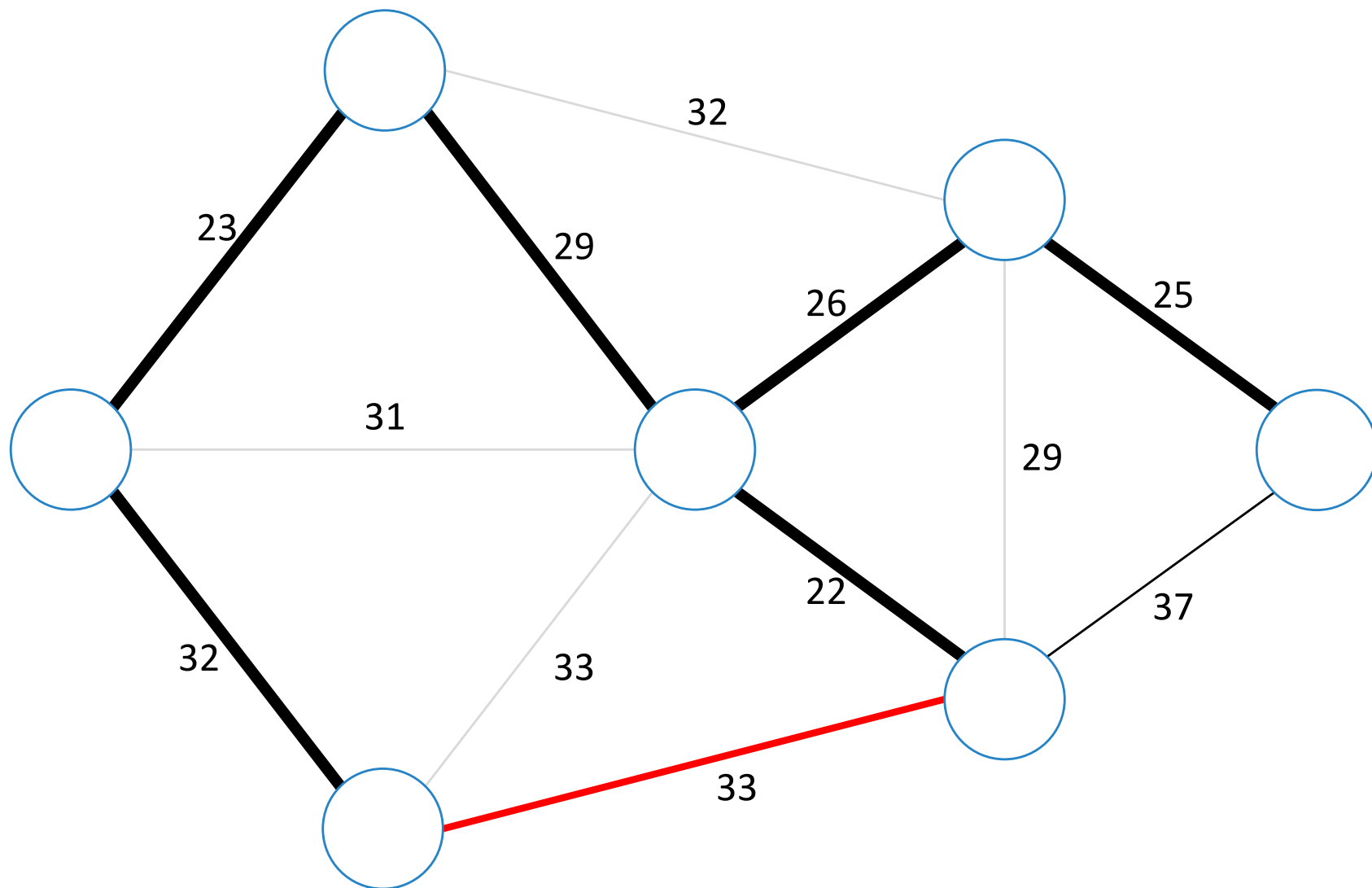
... y en la práctica
no es necesario revisarlas



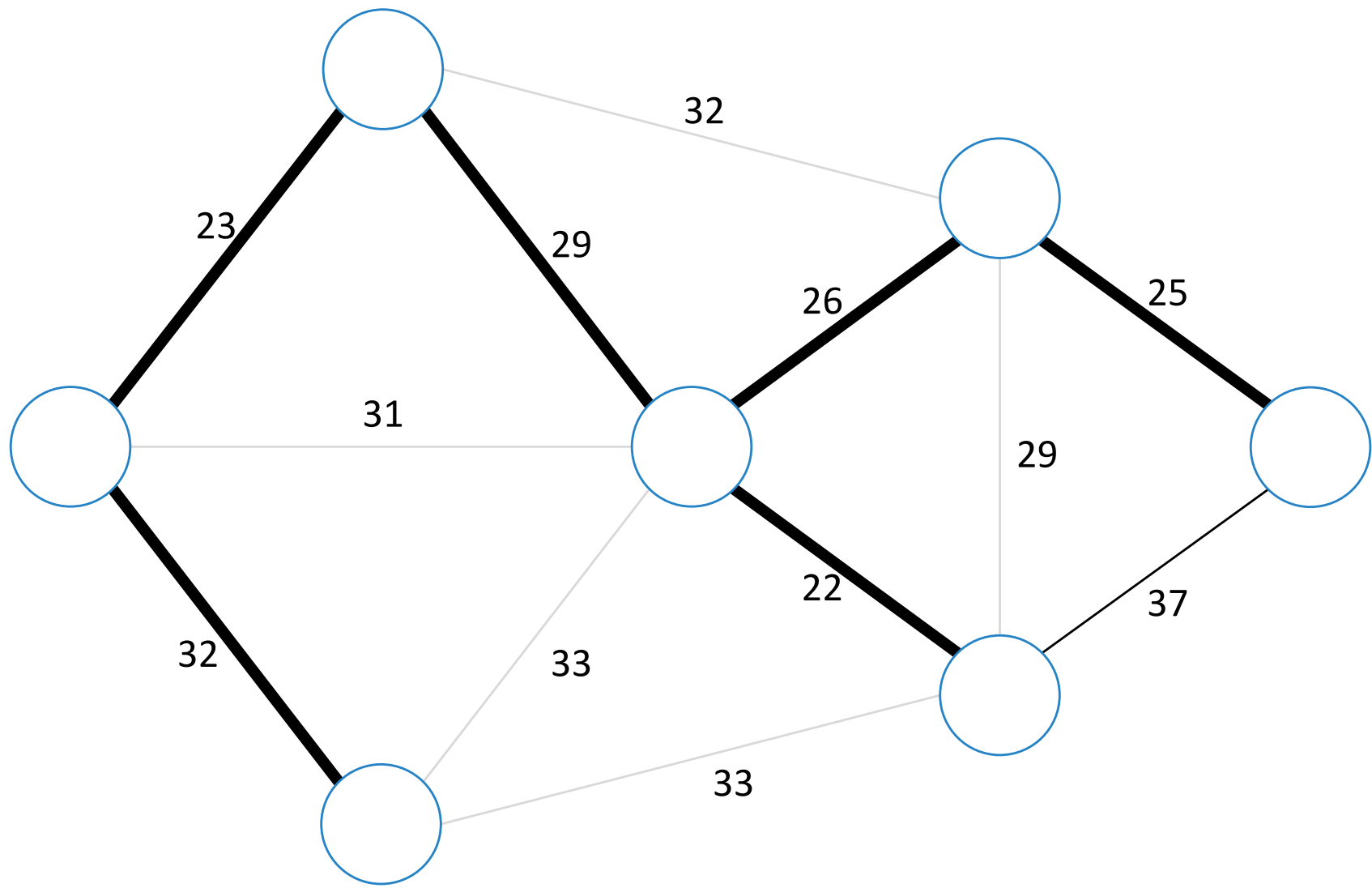
...



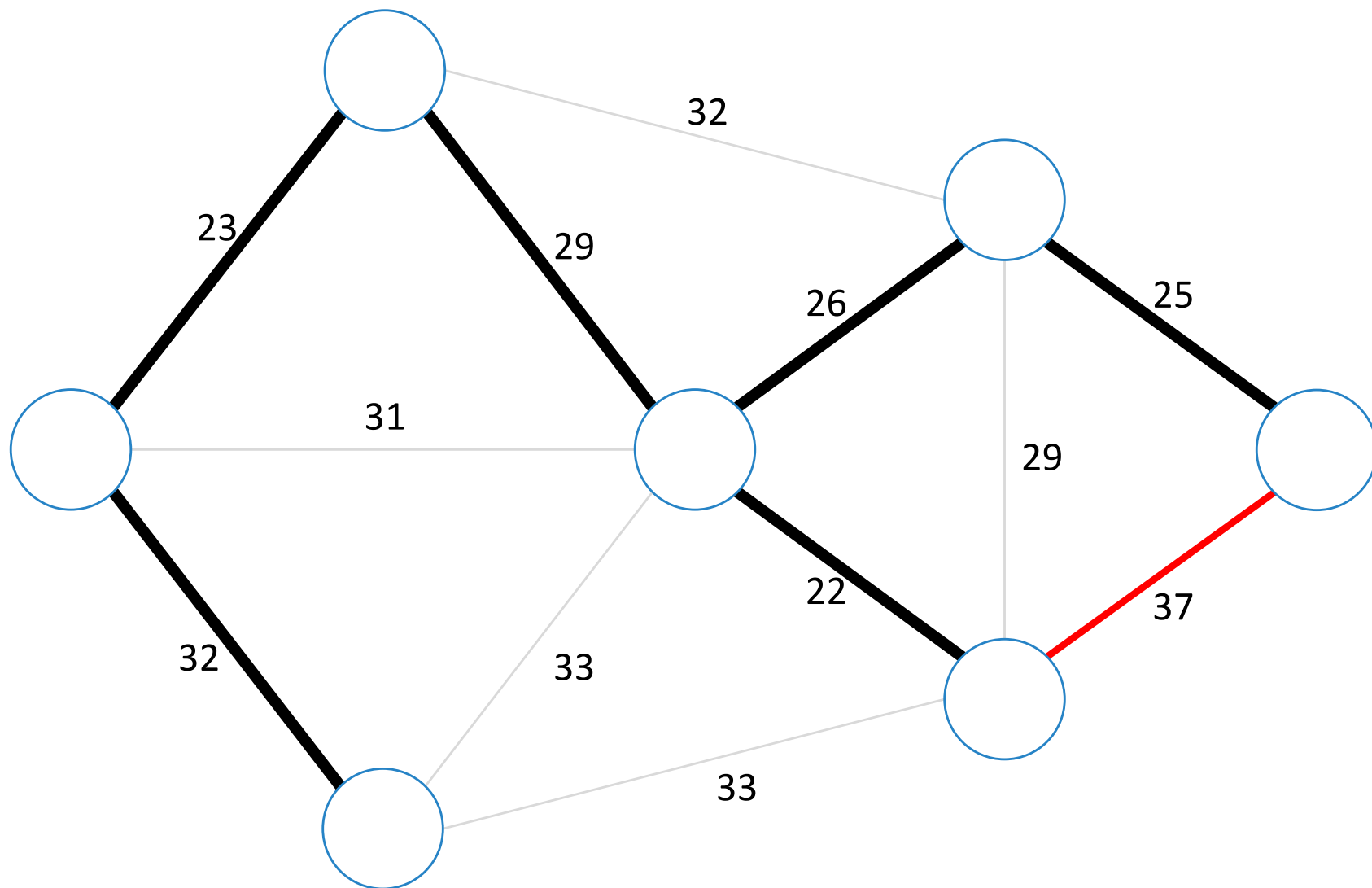
...



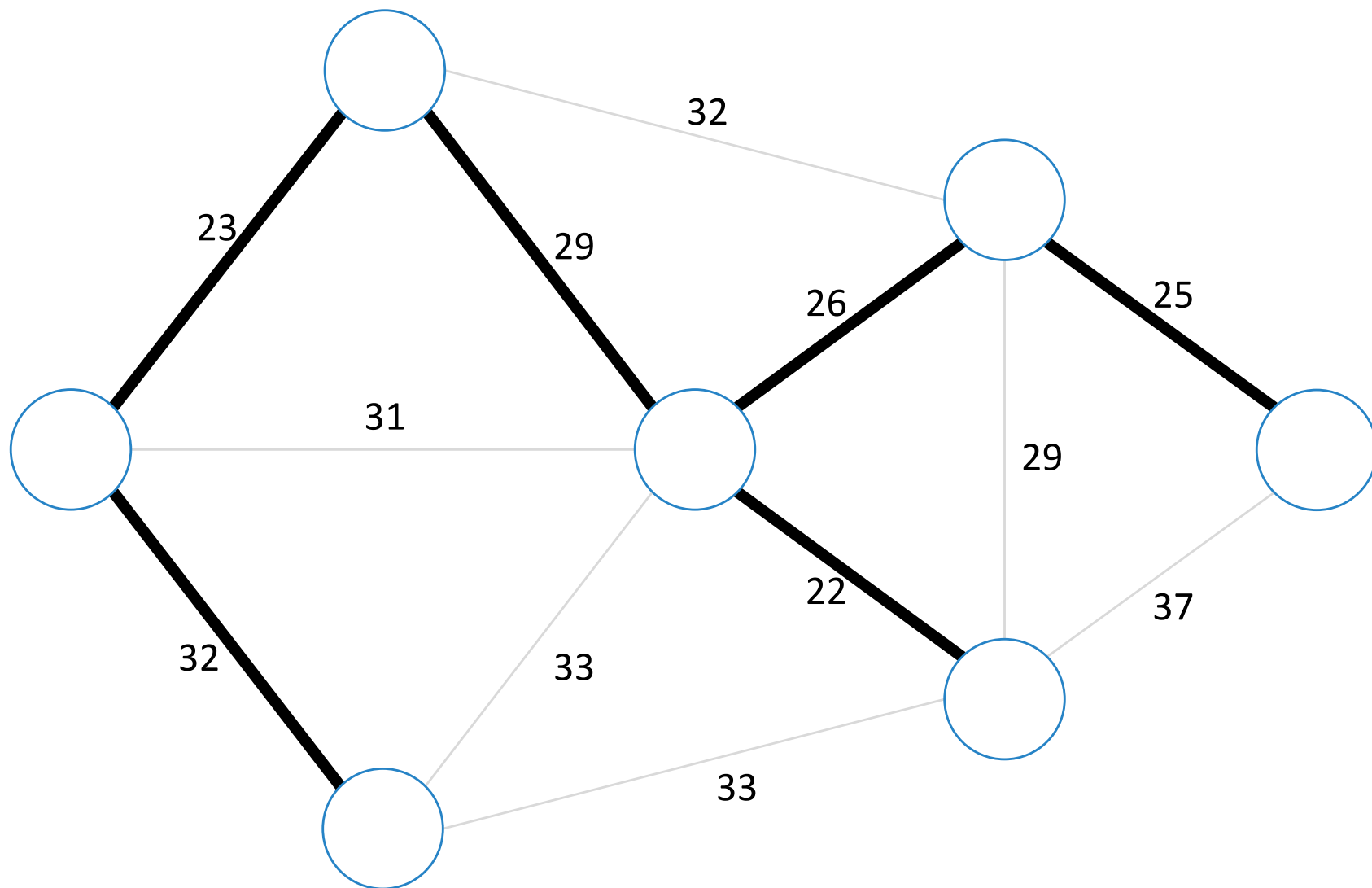
...



...



...



Corrección de *kruskal*



Para demostrar que el algoritmo de Kruskal es correcto

... demostramos por separado que el resultado es

- un árbol
- una cobertura
- de costo total mínimo (entre todos los árboles de cobertura)

Un “detalle” no menor



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

Ordenar E por costo, de menor a mayor

$T \leftarrow \emptyset$

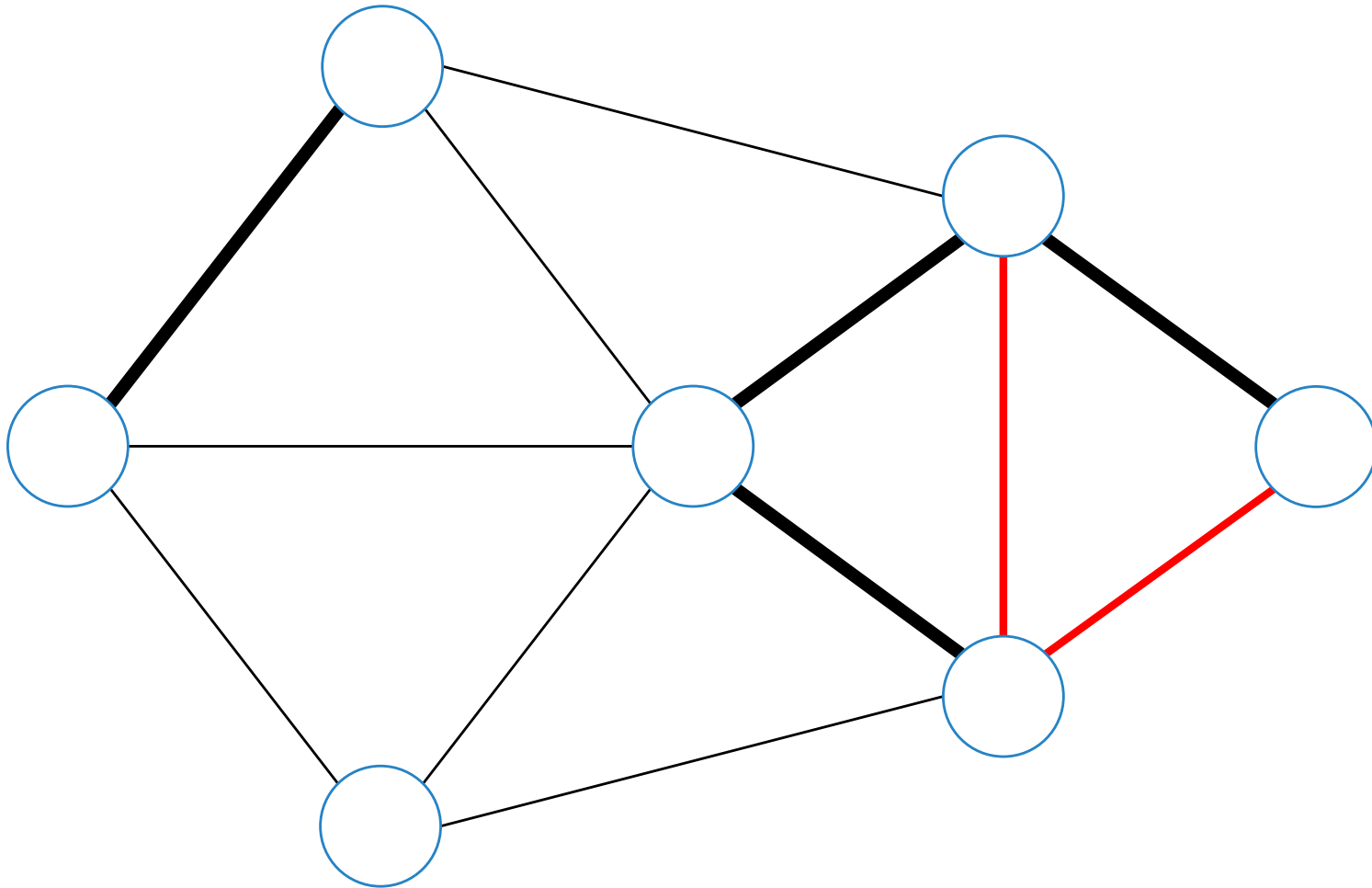
foreach $e \in E$:

if agregar e a T **no forma un ciclo:**

Agregar e a T

return T
¿Cómo revisamos esto de manera eficiente?

Observación



Agregar (u, v) forma un ciclo **ssi** u y v están en el mismo **sub-árbol**

Conjuntos disjuntos



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 conjuntos disjuntos

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

Ordenar E por costo, de menor a mayor

Considerar cada nodo como formando un conjunto por sí mismo

$T \leftarrow \emptyset$

foreach $(u, v) \in E$:

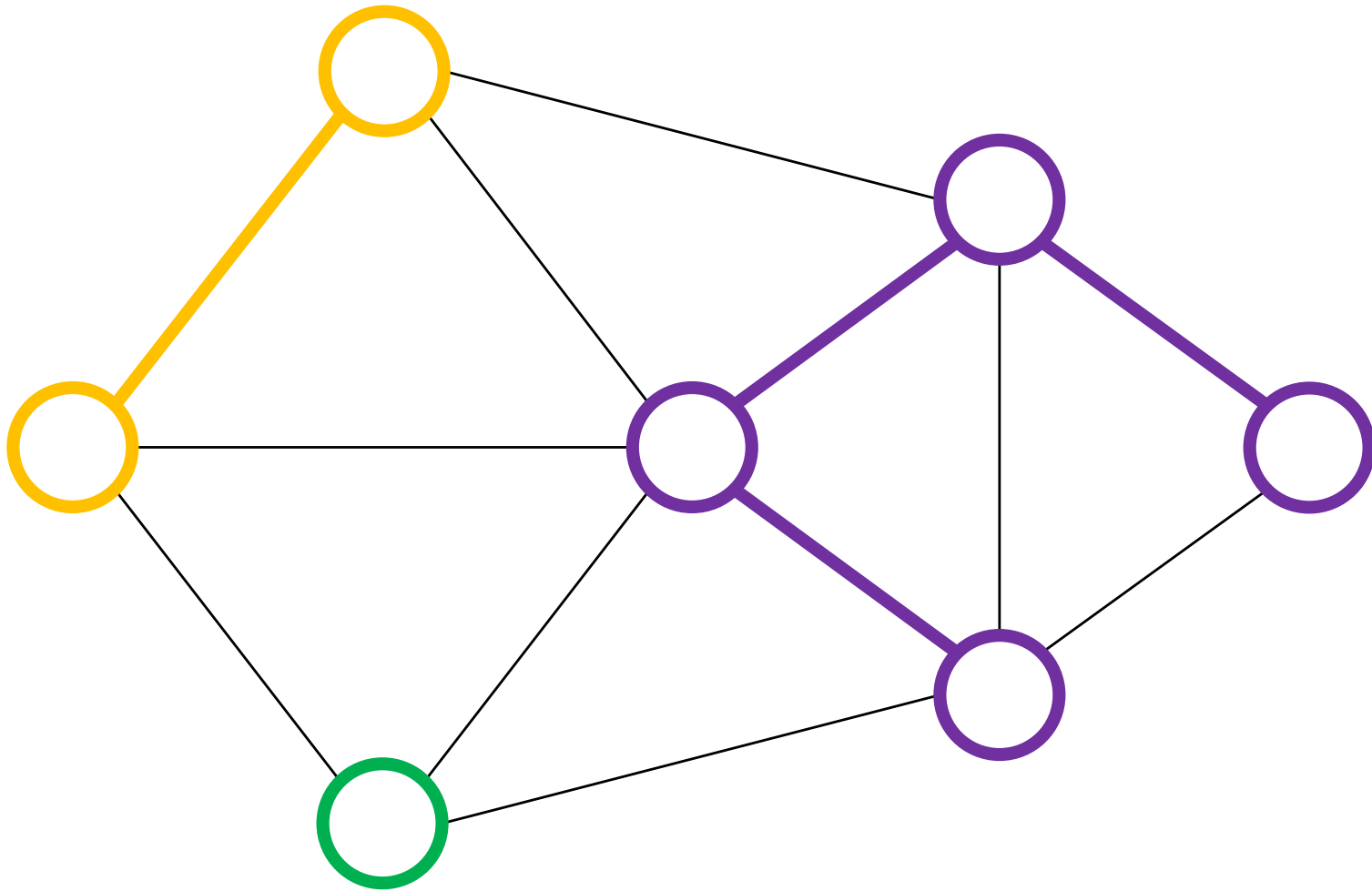
if si u y v no están en el mismo conjunto:

Agregar (u, v) a T

Unir los conjuntos de u y v

return T

Sub-árboles como conjuntos



Agregar una arista significa unir dos conjuntos

Operaciones necesarias sobre conjuntos disjuntos



Nos interesan dos cosas:

- **Identificar** en qué conjunto está un elemento
- **Unir** dos conjuntos (y que sólo quede la unión y no los conjuntos originales)

¿Cómo podemos hacer esto de manera eficiente?

Representación

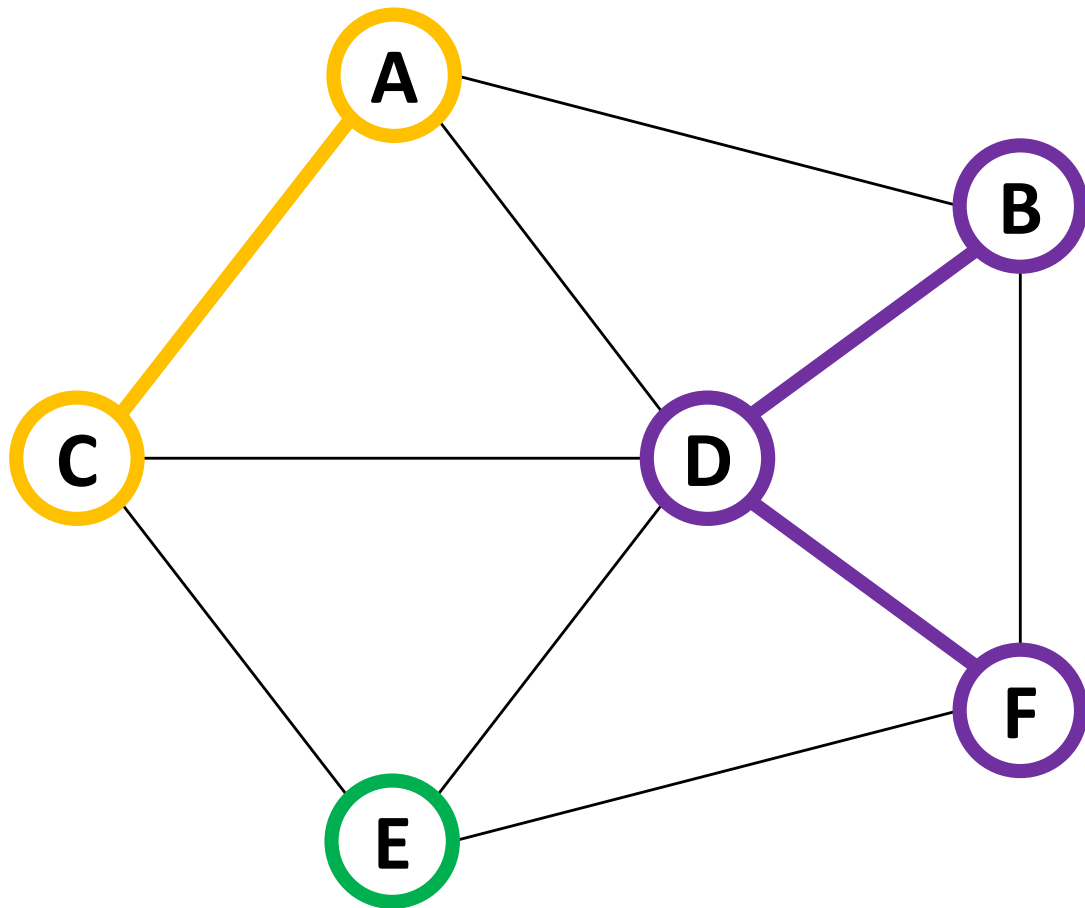
Para cada conjunto, escogemos un **representante** : uno de sus elementos

Cada nodo tiene una **referencia** a su representante, incluyendo el propio representante

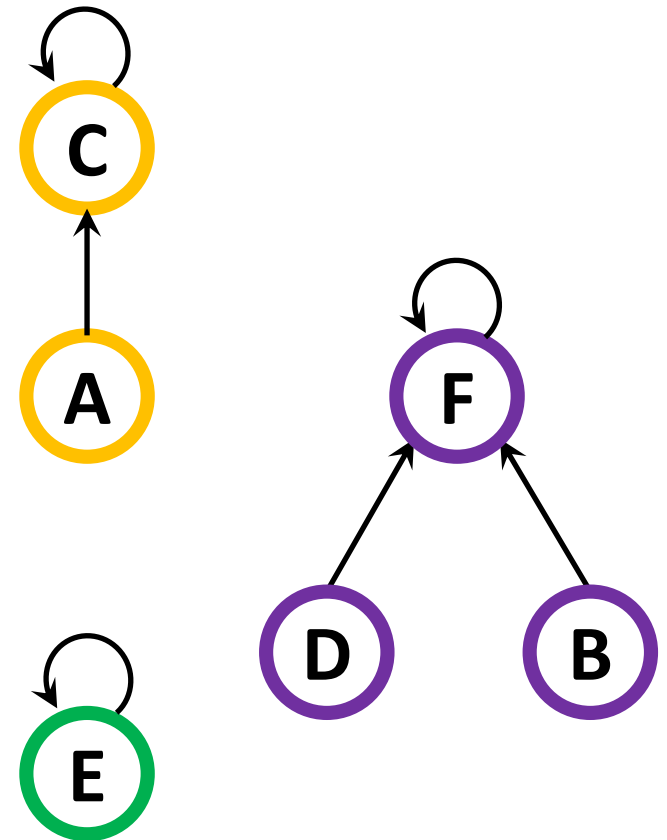
Dos nodos están en el **mismo** conjunto si y sólo si tienen el mismo representante

Conjuntos disjuntos

Sub-árboles



Conjuntos



Operaciones sobre conjuntos disjuntos

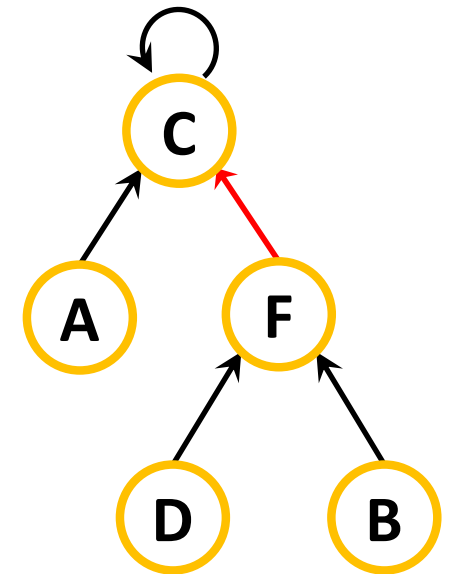
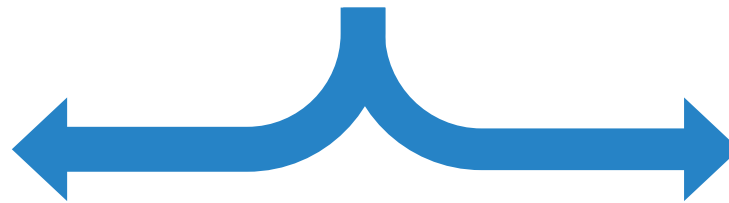
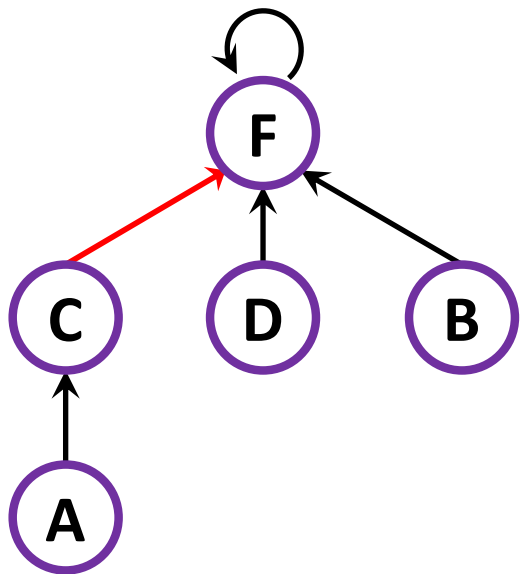
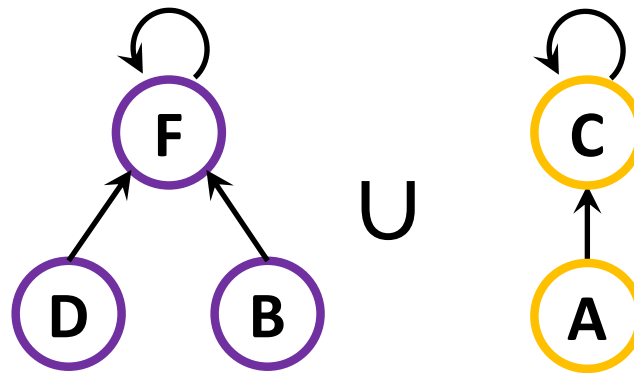


Definimos 3 funciones para esta estructura:

- *make set*(x): inicializa x como su propio representante —cada x está en un conjunto por sí solo inicialmente
- *find set*(x): retorna el representante del nodo x —el conjunto al que pertenece x
- *union*(x, y): une los conjuntos a los que pertenecen x e y —quedando sólo la unión y desapareciendo los conjuntos originales

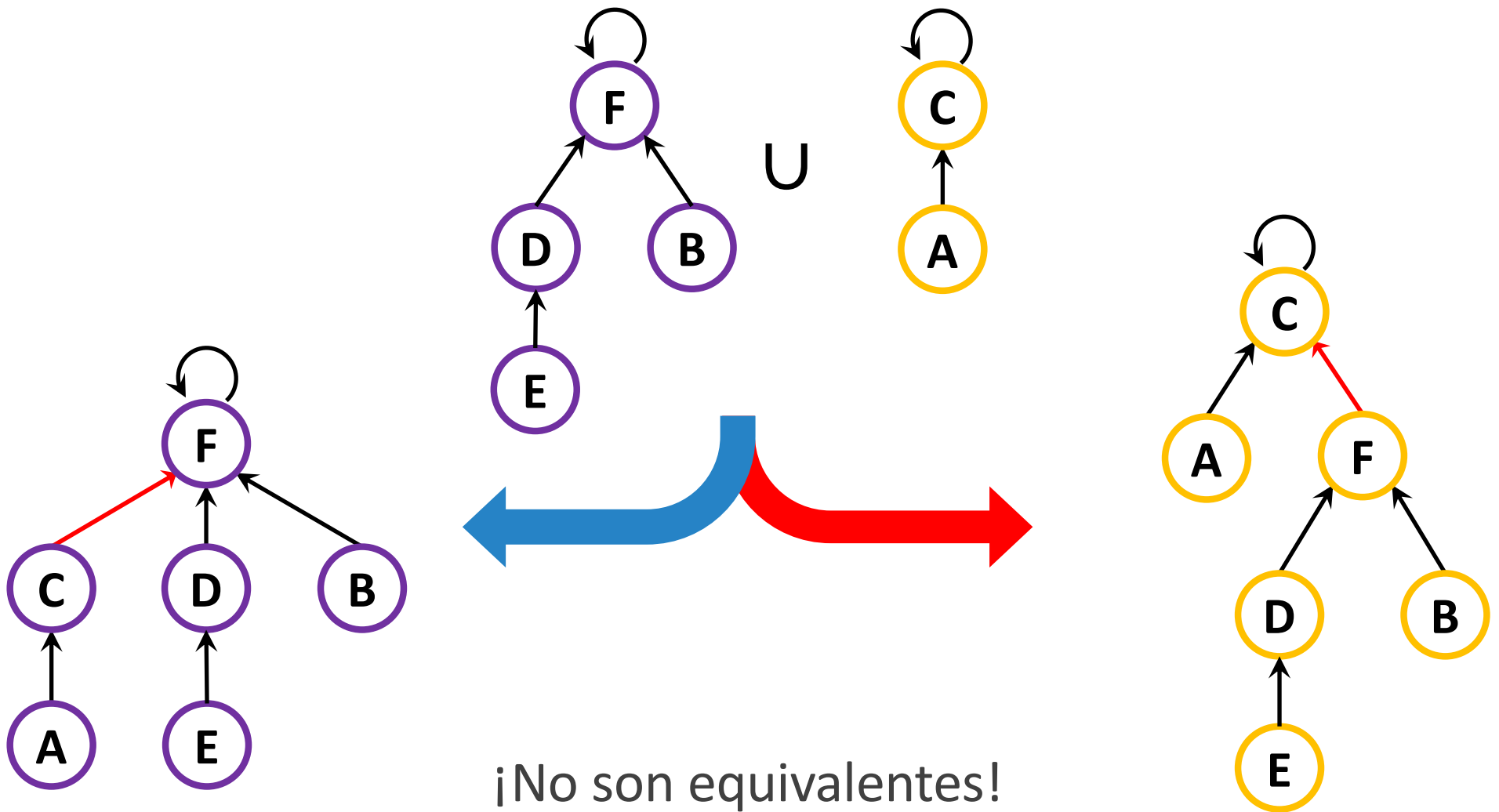
Todas son bastante directas, y pueden implementarse eficientemente; ¿cómo?

Unión

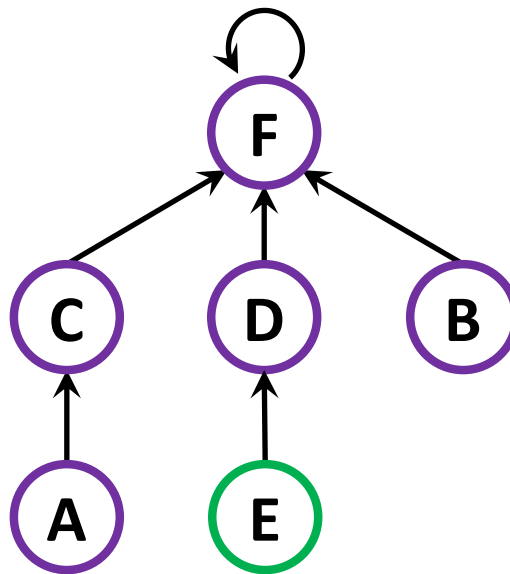


¿Cuál elegimos?

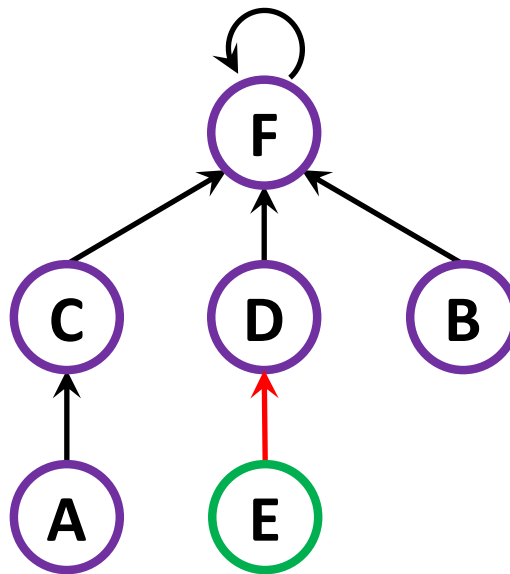
Unión



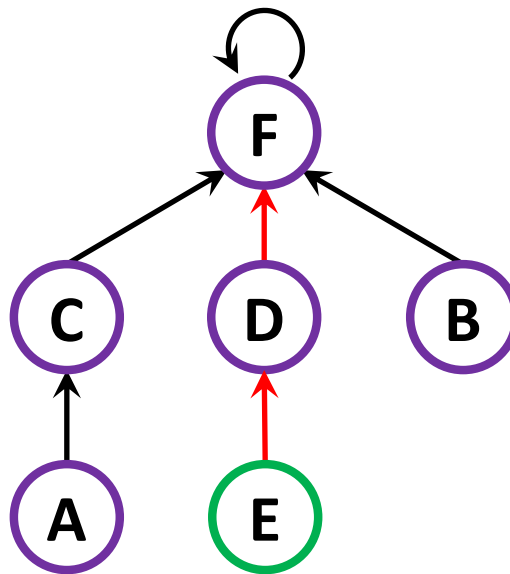
$\text{find-set}(E) = \dots$



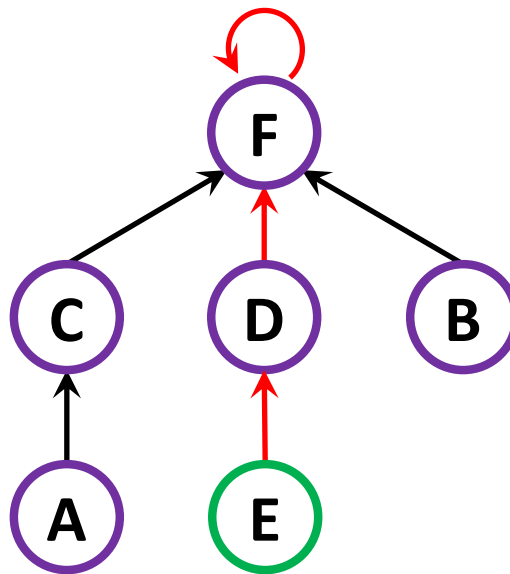
$\text{find-set}(E) = \text{find-set}(D)$



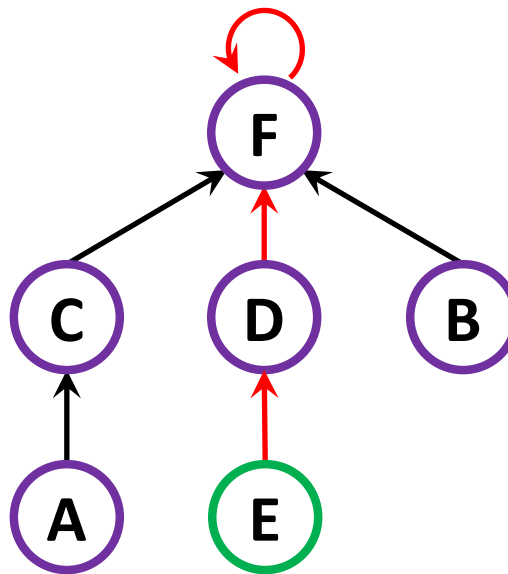
$\text{find-set}(E) = \text{find-set}(F)$



$\text{find-set}(E) = F$

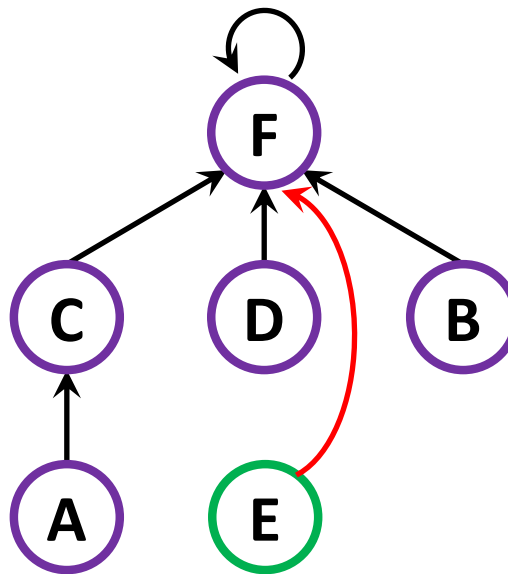


¿Cómo podemos aprovechar esta información una vez que la tenemos?



$$\textit{find set}(E) = \textcircled{F}$$

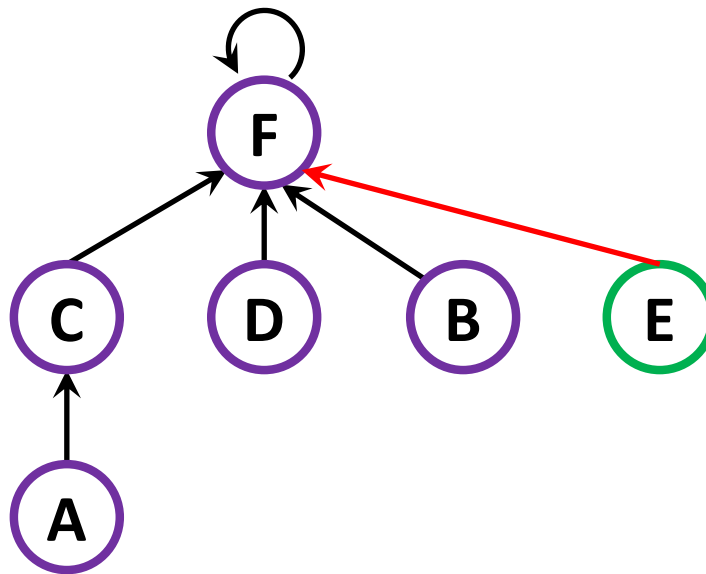
Compresión de caminos



$$\textit{find set}(E) = \textcircled{F}$$

¡Acortando el camino al representante!

Compresión de caminos



$$\textit{find set}(E) = \textcircled{F}$$

Complejidad de *kruskal*



Si pretendemos operar sobre n conjuntos disjuntos

... ¿cuál es la complejidad de estas operaciones?

... ¿y usando las mejoras?

kruskal con conjuntos disjuntos

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

Ordenar E por costo, de menor a mayor

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

$T \leftarrow \emptyset$

foreach $(u, v) \in E$:

if *find set*(u) \neq *find set*(v):

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

union(u, v)

return T

kruskal



¿Considerando esto, cual es la complejidad de Kruskal?

