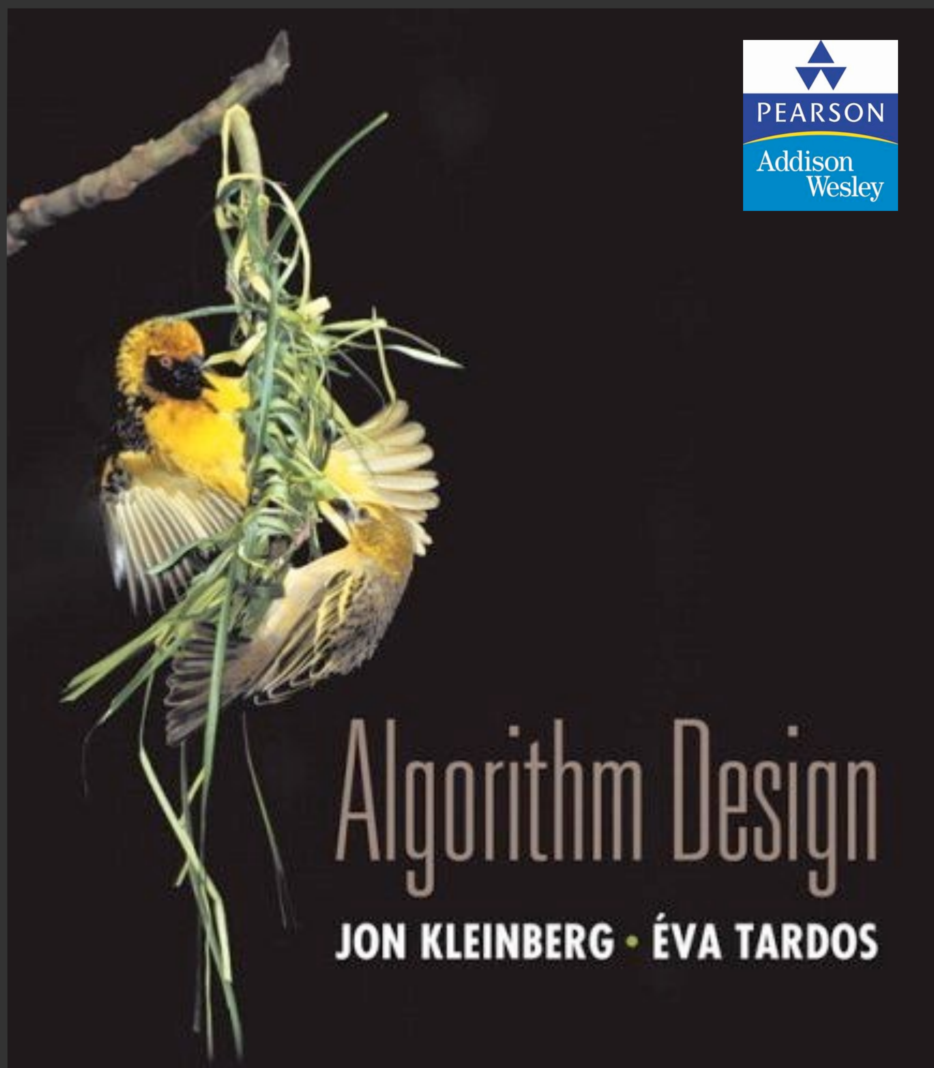


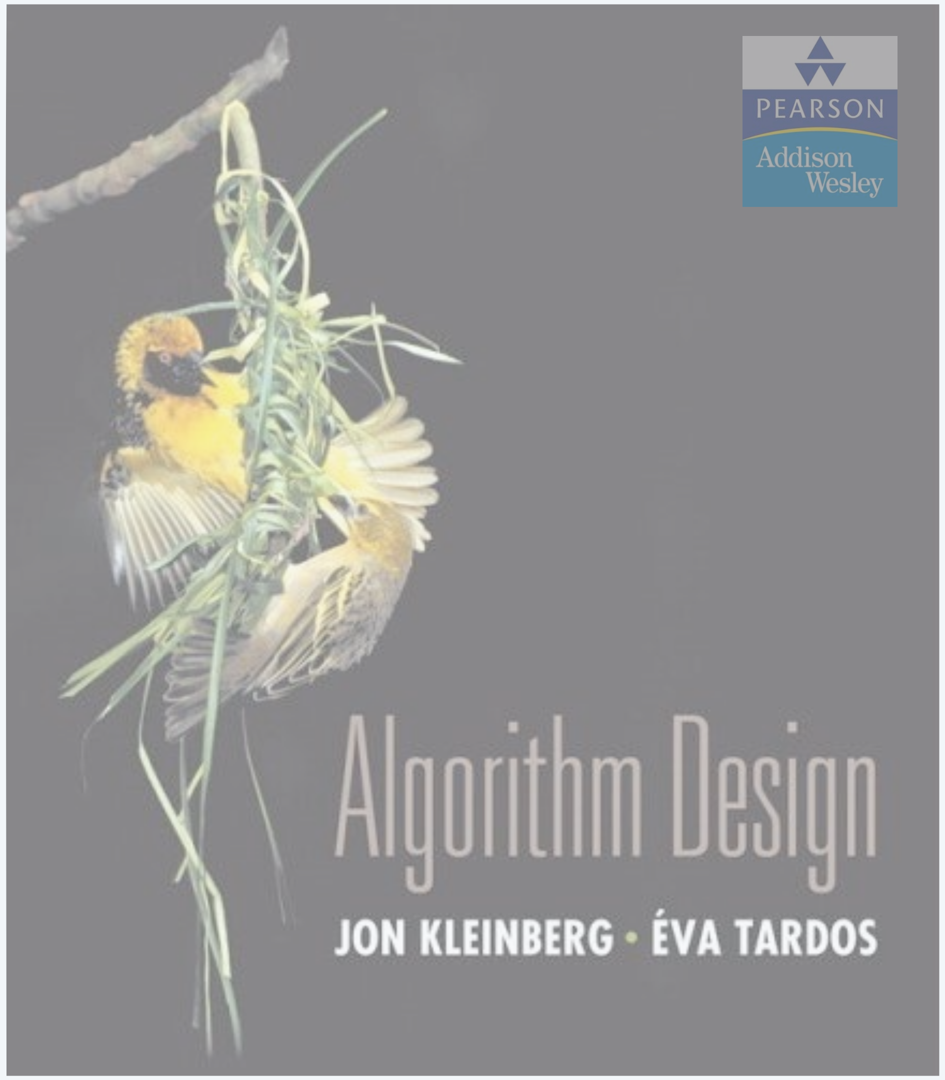
7. Redes de Flujo

- *flujo máximo y corte mínimo*
- *algoritmo de Ford–Fulkerson*
- *teorema flujo máximo - corte mínimo*
- *tiempo de ejecución*
- *emparejamiento bipartito*

Lecture slides by Kevin Wayne

http://www.cs.princeton.edu/~wayne/kleinberg-tardos

Last updated on 17/10/23 10:19



SECTION 7.1

7. REDES DE FLUJO

- *flujo máximo y corte mínimo*
- *algoritmo de Ford–Fulkerson*
- *teorema flujo máximo - corte mínimo*
- *tiempo de ejecución*
- *emparejamiento bipartito*

Red de flujo

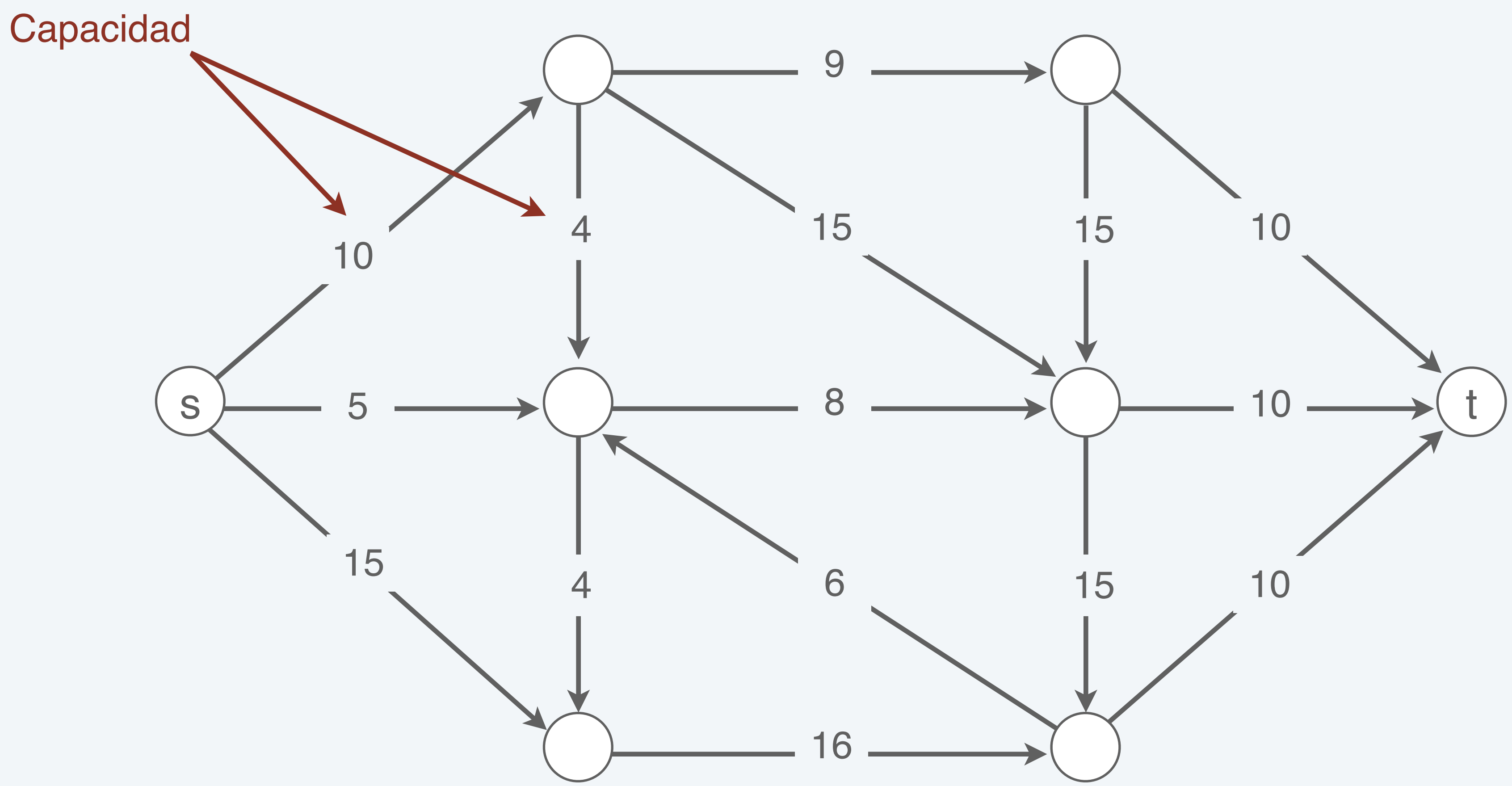
Una red de flujo se define como $G = (V, E, s, t, c)$, donde

- (V, E) es un grafo dirigido, $s \in V$ se denomina *fuente* y $t \in V$ *destino*.
- $c(e)$ es una *capacidad* no negativa para cada $e \in E$.

Asumimos:

- No hay aristas entrantes a s y no hay aristas salientes de t.

Intuición. Material se origina en la fuente y se envía al destino.

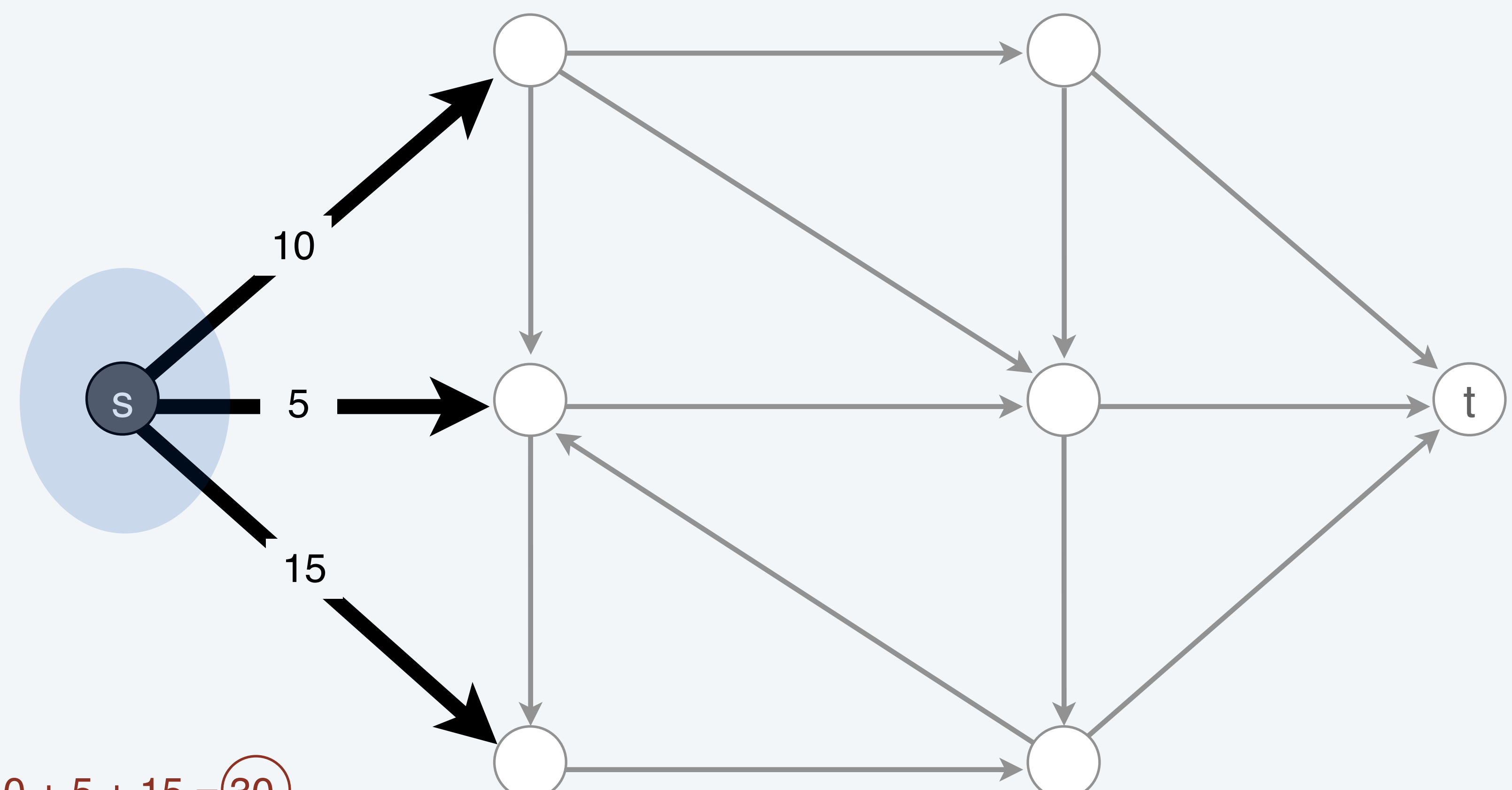


Problema de corte mínimo

Def. Un corte-st (corte) es una partición (A, B) de V con $s \in A$ y $t \in B$.

Def. Su capacidad es la suma de las capacidades de las aristas de A a B.

$$cap(A, B) = \sum_{e \text{ saliente de } A} c(e)$$



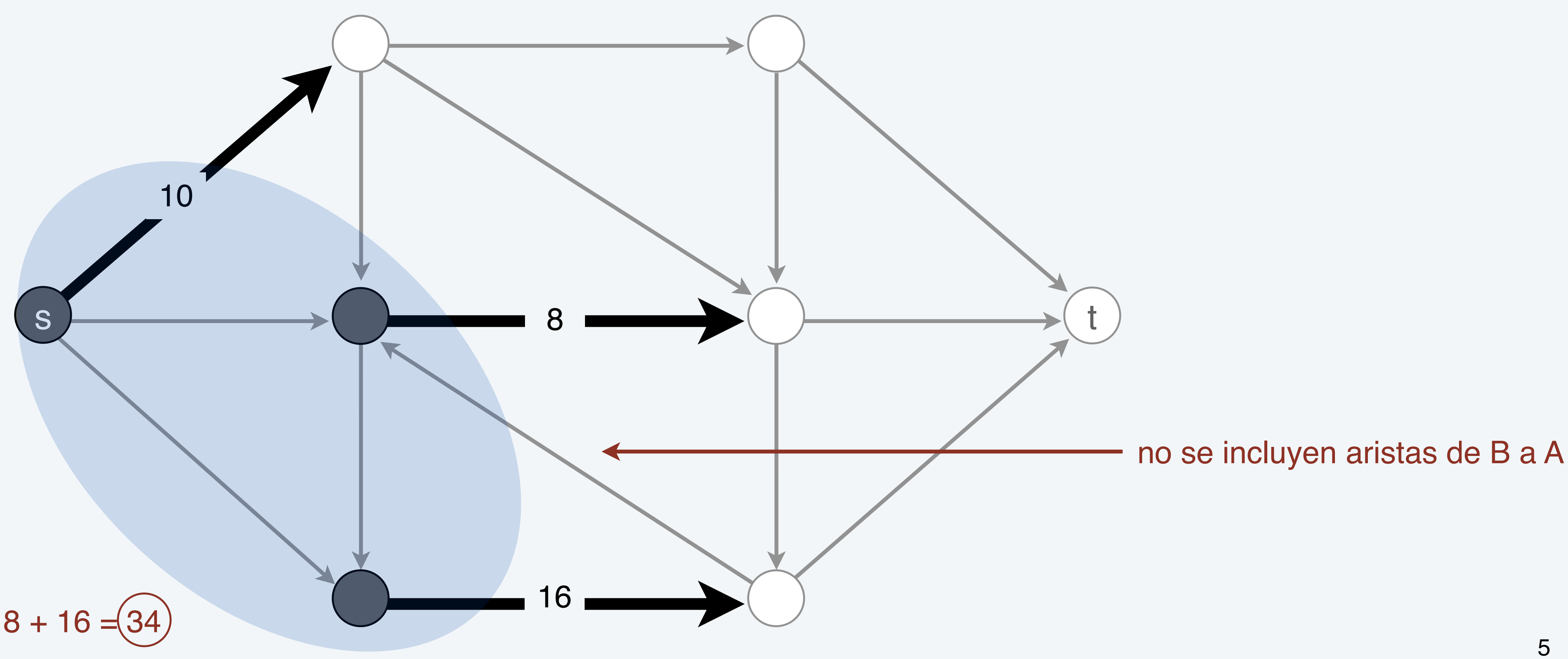
Capacidad = 10 + 5 + 15 = 30

Problema de corte mínimo

Def. Un corte-st (corte) es una partición (A, B) de V con $s \in A$ y $t \in B$.

Def. Su capacidad es la suma de las capacidades de las aristas de A a B.

$$cap(A, B) = \sum_{e \text{ saliente de } A} c(e)$$



Capacidad = 10 + 8 + 16 = 34

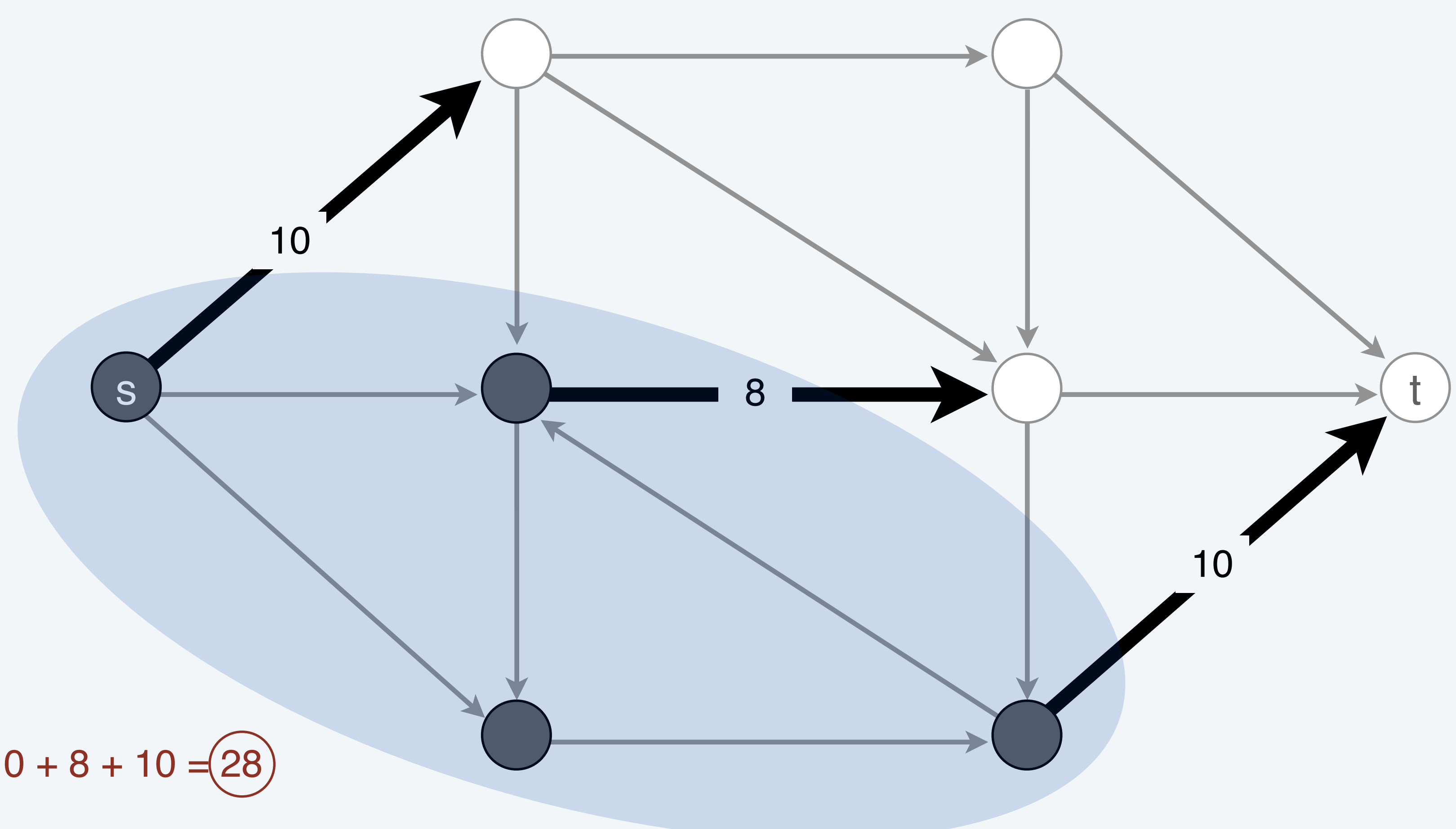
Problema de corte mínimo

Def. Un corte-st (corte) es una partición (A, B) de V con $s \in A$ y $t \in B$.

Def. Su capacidad es la suma de las capacidades de las aristas de A a B.

$$cap(A, B) = \sum_{e \text{ saliente de } A} c(e)$$

Problema de corte mínimo. Encontrar un corte de capacidad mínima.

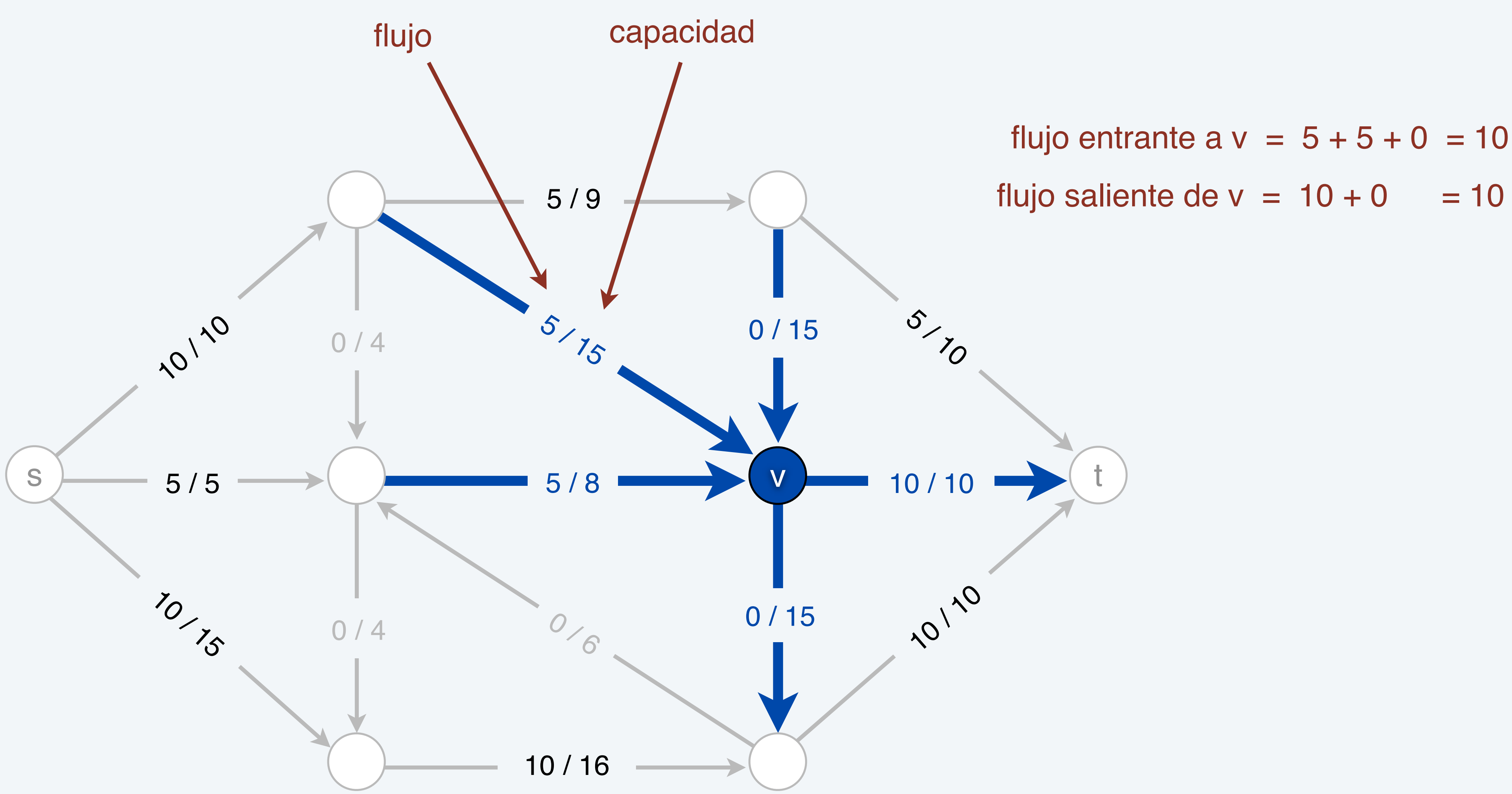


capacidad = 10 + 8 + 10 = 28

Problema de flujo máximo

Def. Un flujo-st (flujo) f es una función que satisface las siguientes restricciones:

- Para cada $e \in E$: $\quad 0 \leq f(e) \leq c(e) \quad$ [capacidad]
- Para cada $v \in V - \{s, t\}$: $\quad \sum_{e \text{ entrante a } v} f(e) = \sum_{e \text{ saliente de } v} f(e) \quad$ [conservación de flujo]

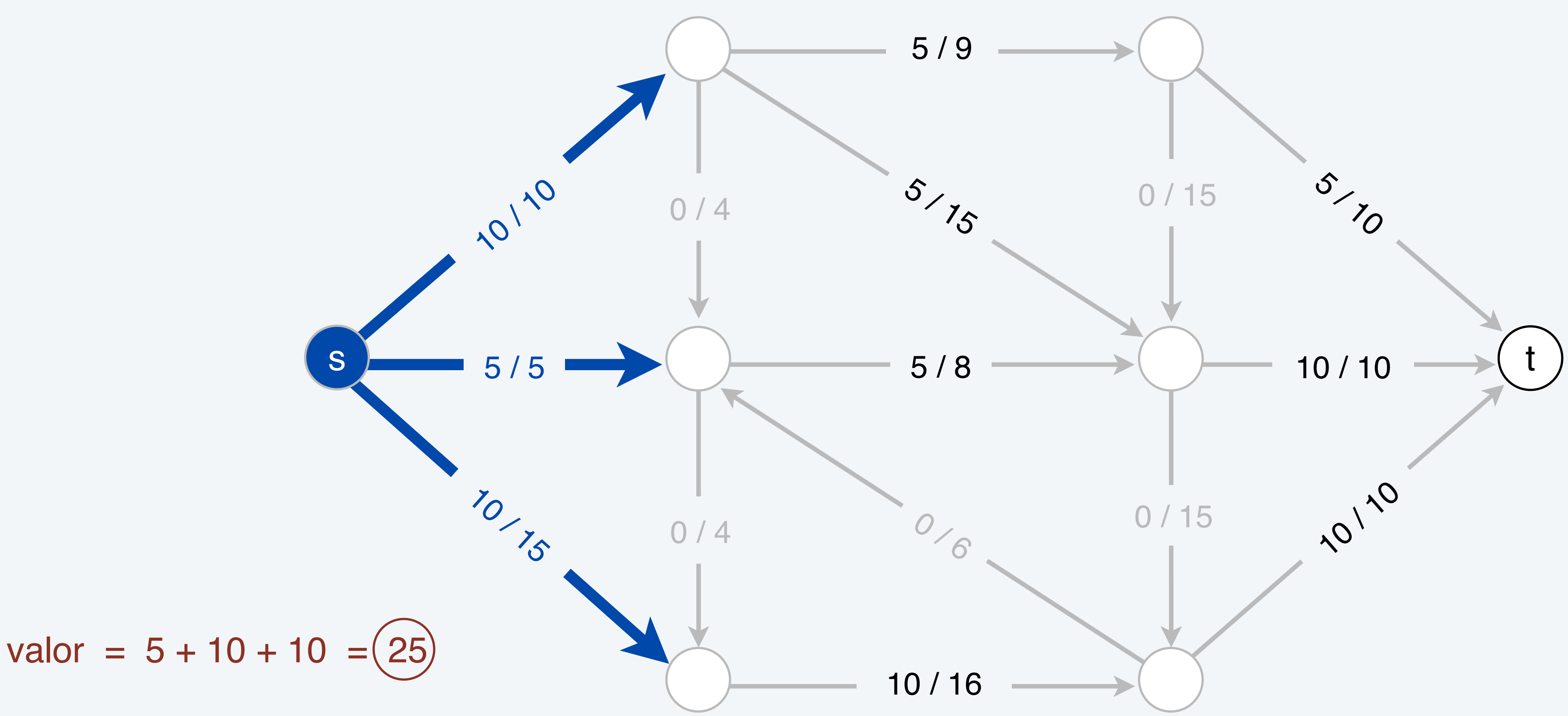


Problema de flujo máximo

Def. Un flujo-st (flujo) f es una función que satisface las siguientes restricciones:

- Para cada $e \in E$: $\quad 0 \leq f(e) \leq c(e)$ [capacidad]
- Para cada $v \in V - \{s, t\}$: $\displaystyle\sum_{e \text{ entrante a } v} f(e) = \sum_{e \text{ saliente de } v} f(e)$ [conservación de flujo]

Def. El valor de un flujo f es: $\displaystyle val(f) = \sum_{e \text{ saliente de } s} f(e)$



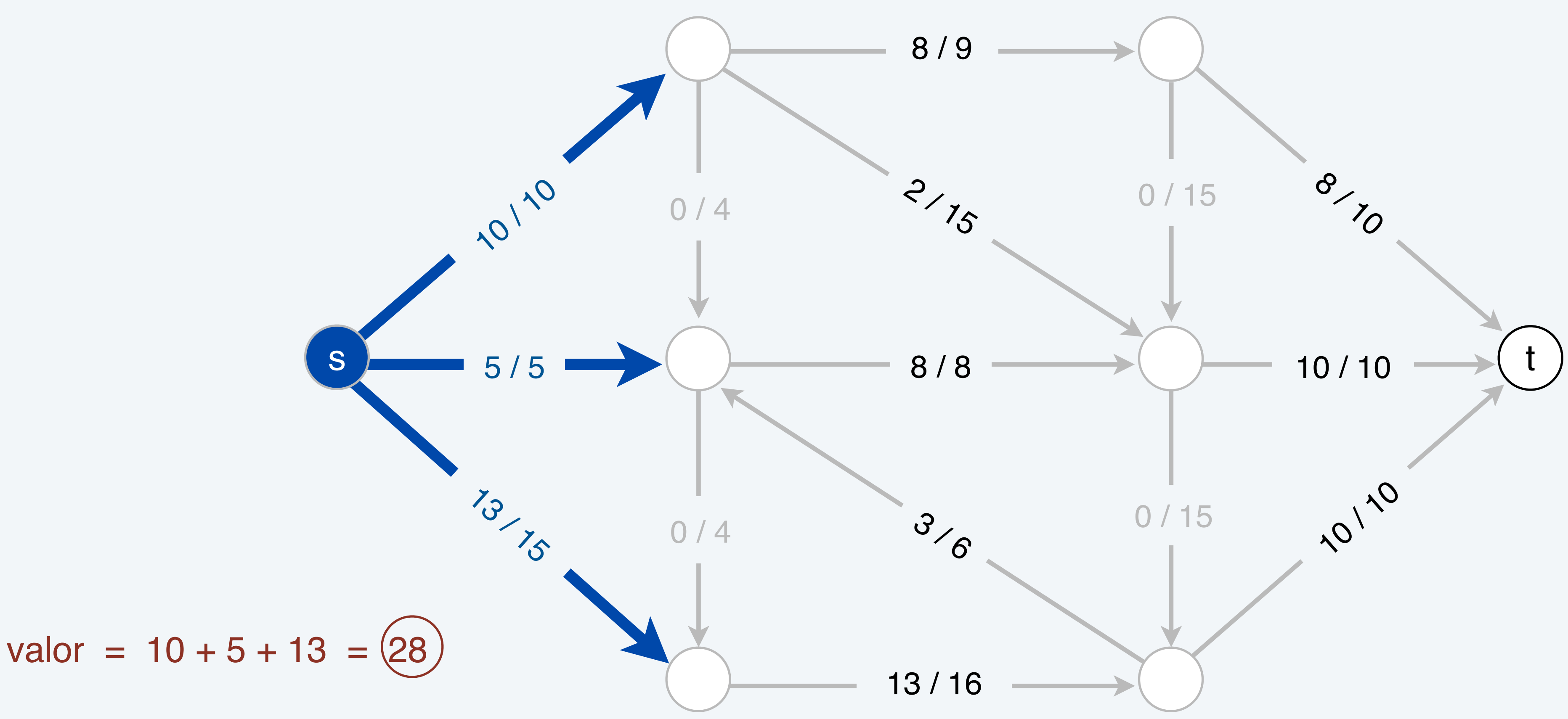
Problema de flujo máximo

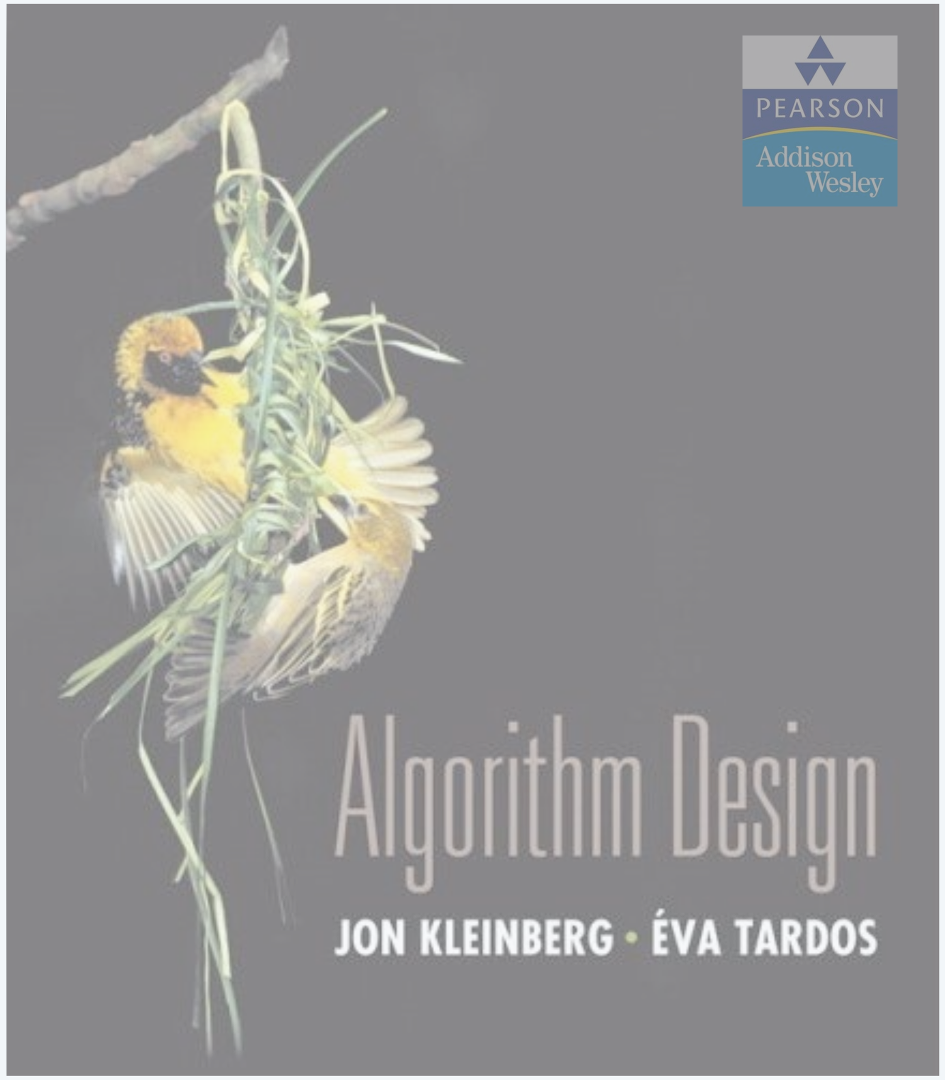
Def. Un flujo-st (flujo) f es una función que satisface las siguientes restricciones:

- Para cada $e \in E$: $\quad 0 \leq f(e) \leq c(e)$ [capacidad]
- Para cada $v \in V - \{s, t\}$: $\quad \sum_{e \text{ entrante a } v} f(e) = \sum_{e \text{ saliente de } v} f(e)$ [conservación de flujo]

Def. El valor de un flujo f es: $\quad val(f) = \sum_{e \text{ saliente de } s} f(e)$

Problema de flujo máximo. Encontrar un flujo de valor máximo.





SECTION 7.1

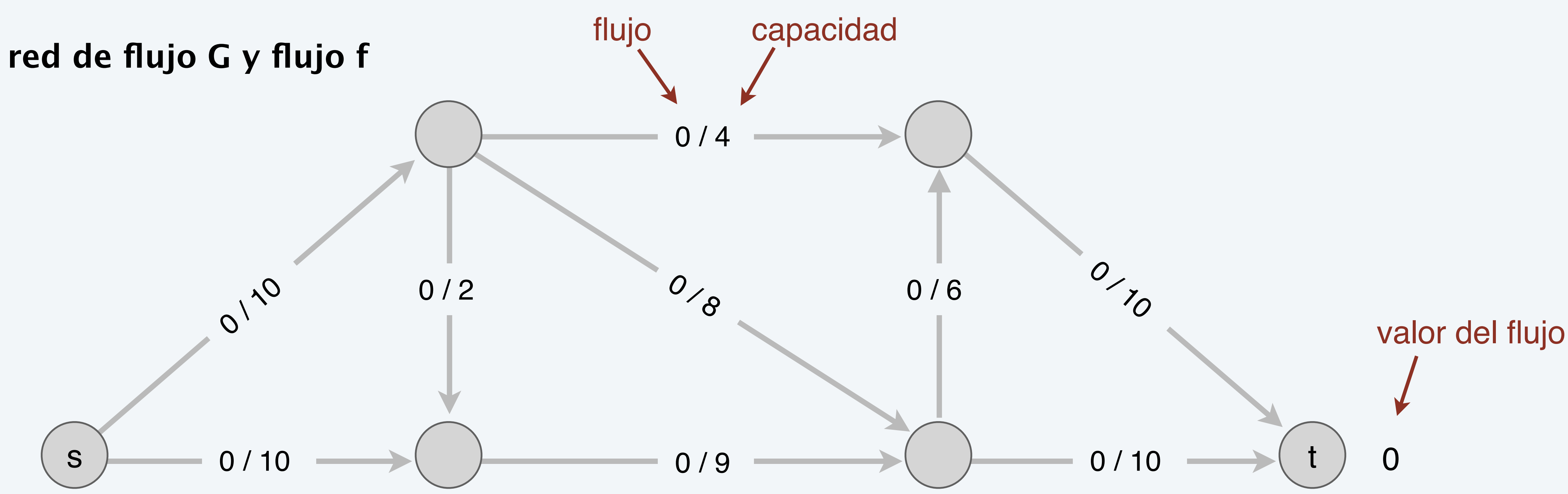
7. Redes de Flujo

- *flujo máximo y corte mínimo*
- *algoritmo de Ford–Fulkerson*
- *teorema flujo máximo - corte mínimo*
- *tiempo de ejecución*
- *emparejamiento bipartito*

Primer intento de solución

Algoritmo greedy.

- Empezar con $f(e) = 0$ para toda arista $e \in E$.
- Buscar un camino $s \leadsto t$, P, donde toda arista e satisface $f(e) < c(e)$.
- Si tal camino existe, aumentar el flujo a lo largo de P.
- Repetir hasta que ya no exista tal camino.

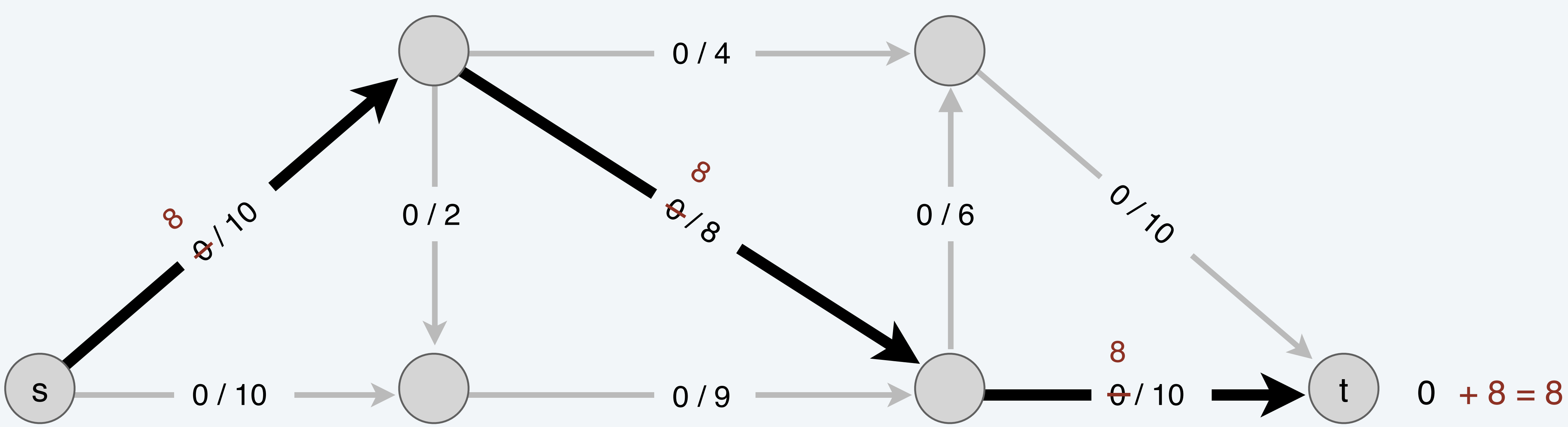


Primer intento de solución

Algoritmo greedy.

- Empezar con $f(e) = 0$ para toda arista $e \in E$.
- Buscar un camino $s \rightsquigarrow t$, P, donde toda arista e satisface $f(e) < c(e)$.
- Si tal camino existe, aumentar el flujo a lo largo de P.
- Repetir hasta que ya no exista tal camino.

red de flujo G y flujo f

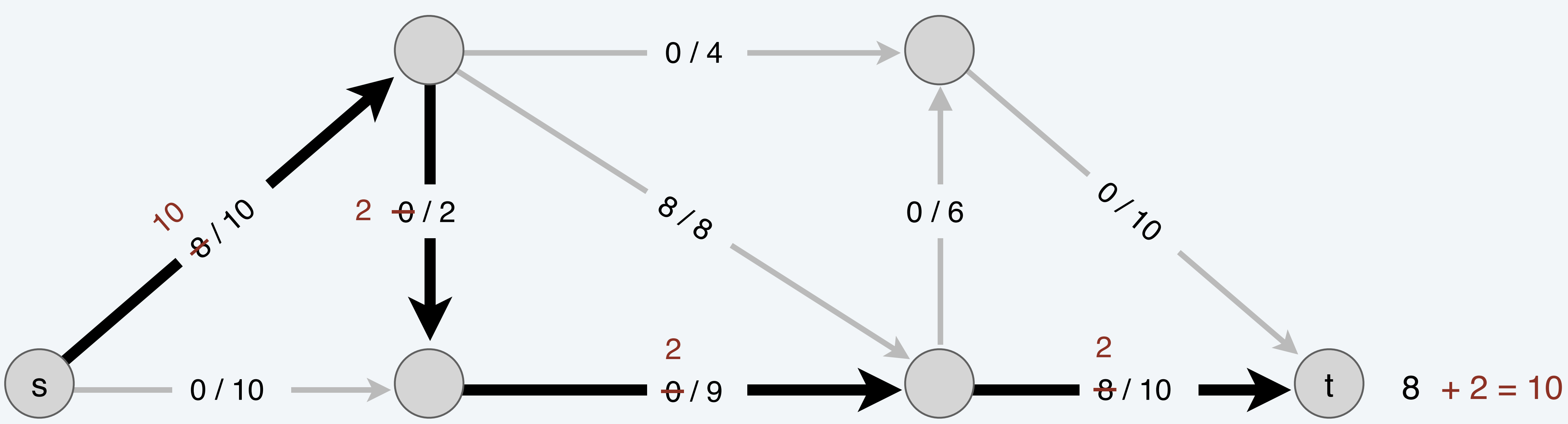


Primer intento de solución

Algoritmo greedy.

- Empezar con $f(e) = 0$ para toda arista $e \in E$.
- Buscar un camino $s \leadsto t$, P, donde toda arista e satisface $f(e) < c(e)$.
- Si tal camino existe, aumentar el flujo a lo largo de P.
- Repetir hasta que ya no exista tal camino.

red de flujo G y flujo f

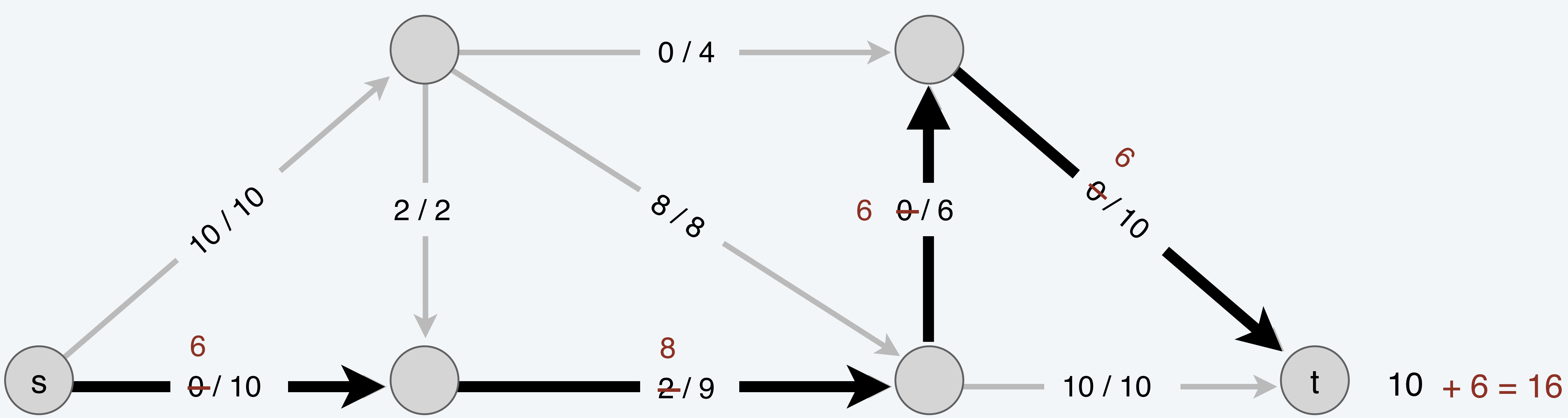


Primer intento de solución

Algoritmo greedy.

- Empezar con $f(e) = 0$ para toda arista $e \in E$.
- Buscar un camino $s \rightsquigarrow t$, P, donde toda arista e satisface $f(e) < c(e)$.
- Si tal camino existe, aumentar el flujo a lo largo de P.
- Repetir hasta que ya no exista tal camino.

red de flujo G y flujo f



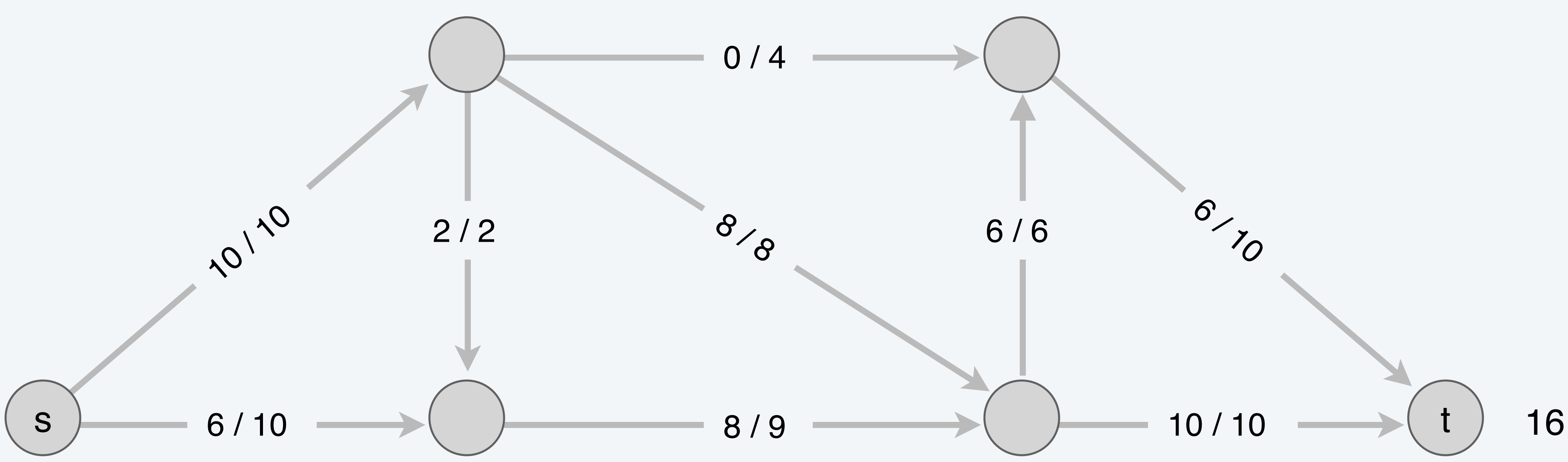
Primer intento de solución

Algoritmo greedy.

- Empezar con $f(e) = 0$ para toda arista $e \in E$.
- Buscar un camino $s \rightsquigarrow t$, P, donde toda arista e satisface $f(e) < c(e)$.
- Si tal camino existe, aumentar el flujo a lo largo de P.
- Repetir hasta que ya no exista tal camino.

valor final del flujo = 16

red de flujo G y flujo f



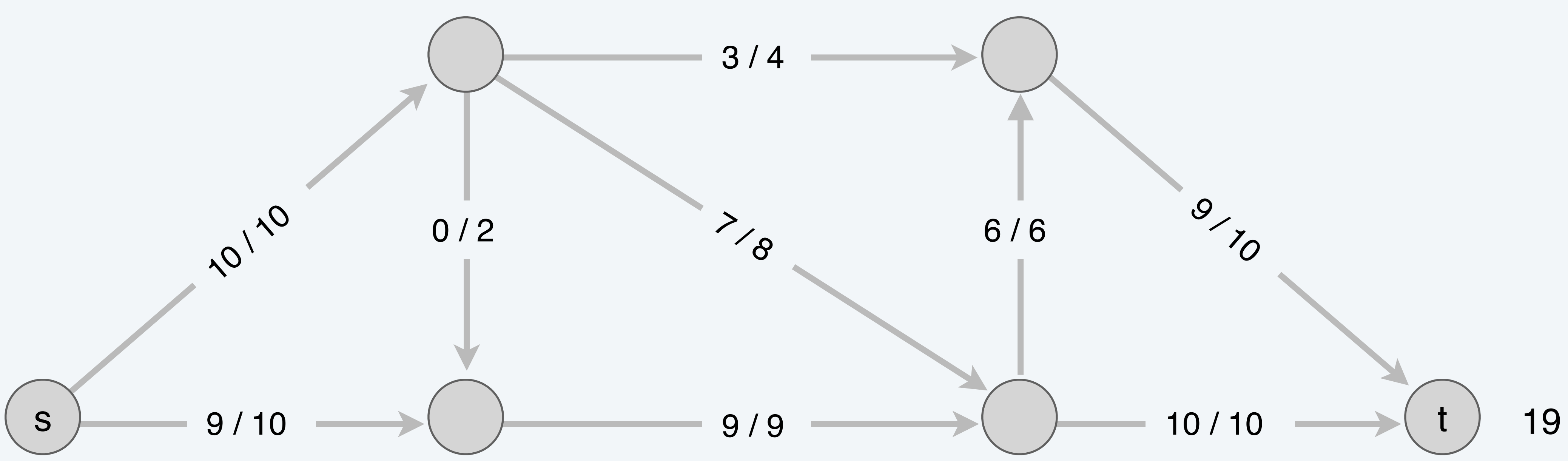
Primer intento de solución

Algoritmo greedy.

- Empezar con $f(e) = 0$ para toda arista $e \in E$.
- Buscar un camino $s \rightsquigarrow t$, P, donde toda arista e satisface $f(e) < c(e)$.
- Si tal camino existe, aumentar el flujo a lo largo de P.
- Repetir hasta que ya no exista tal camino.

valor de flujo máximo = 19

red de flujo G y flujo f



El algoritmo greedy falla

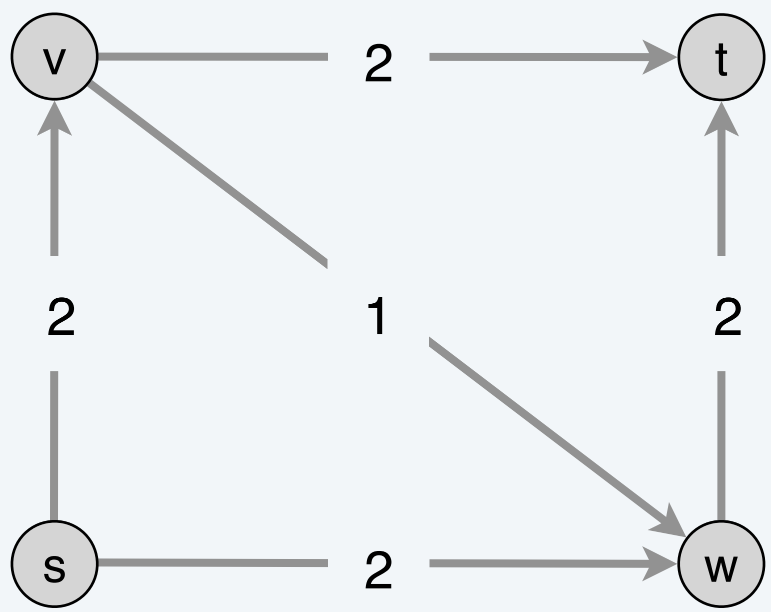
¿Por qué falla el algoritmo greedy?

Una vez que incrementa el flujo en una arista, nunca lo decrementa.

Ejemplo.

- El flujo máximo es único; el flujo en (v, w) es cero.
- El algoritmo podría elegir $s \rightarrow v \rightarrow w \rightarrow t$ como camino para el primer incremento.

red de flujo G



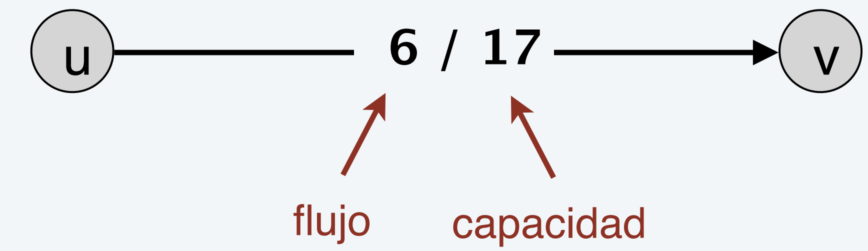
Conclusión. Se necesita algún mecanismo para revertir malas decisiones.

Grafo residual (con respecto al flujo f)

Arista original. $e = (u, v) \in E$.

- Flujo $f(e)$.
- Capacidad $c(e)$.

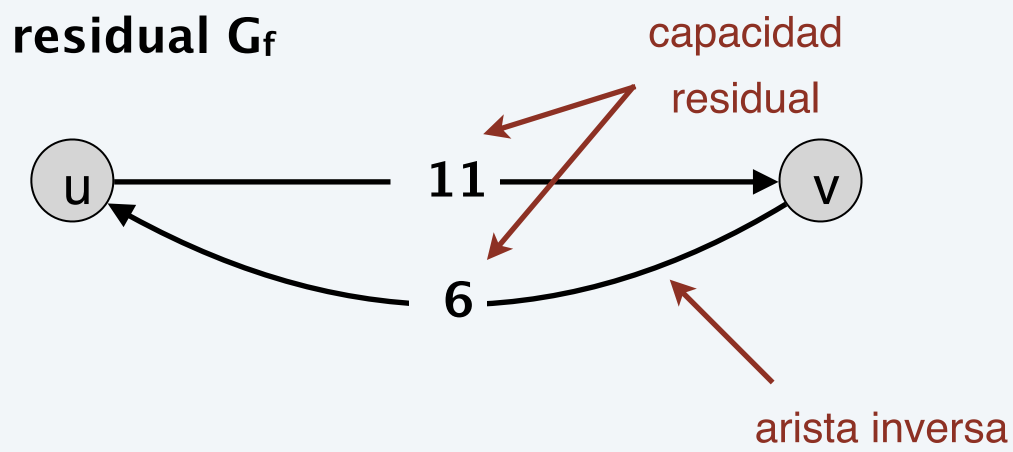
red de flujo G



Arista inversa. $e^{inversa} = (v, u)$.

- "Revierte" flujo ya enviado.

grafo residual Gf



Capacidad residual.

$$c_f(e) = \begin{cases} c(e) - f(e) & \text{if } e \in E \\ f(e) & \text{if } e^{inversa} \in E \end{cases}$$

aristas con
capacidad residual positiva

Grafo residual. $G_f = (V, E_f, s, t, c_f)$.

- $E_f = \{e : f(e) < c(e)\} \cup \{e^{inversa} : f(e) > 0\}$.

Camino de aumento

Def. Un camino de aumento es un camino $s \rightsquigarrow t$ simple en el grafo residual G_f.

Def. El cuello de botella de un camino de aumento P es el mínimo de las capacidades residuales de las aristas de P.

Propiedad. Sea f un flujo, P un camino de aumento en G_f, y sea b el cuello de botella de P. AUGMENT(f, c, P) devuelve un flujo f' con valor $val(f') = val(f) + b$.

AUGMENT (f, c, P)

$b \leftarrow$ cuello de botella del camino P.

FOREACH arista $e \in P$

  IF ($e \in E$) $f(e) \leftarrow f(e) + b$.

  ELSE $f(e^{\text{inversa}}) \leftarrow f(e^{\text{inversa}}) - b$.

RETURN f.

Algoritmo de Ford–Fulkerson

Algoritmo de Ford–Fulkerson.

- Hacer $f(e) = 0$ para toda arista $e \in E$.
- Buscar un camino $s \leadsto t$ P en el grafo residual G_f.
- Si tal camino existe, aumentar el flujo a lo largo de P.
- Repetir hasta que ya no exista tal camino.

FORD–FULKERSON (G)

FOREACH edge $e \in E$: $f(e) \leftarrow 0$.

$G_f \leftarrow$ grafo residual con respecto a f.

WHILE (existe un camino P de s a t en G_f)

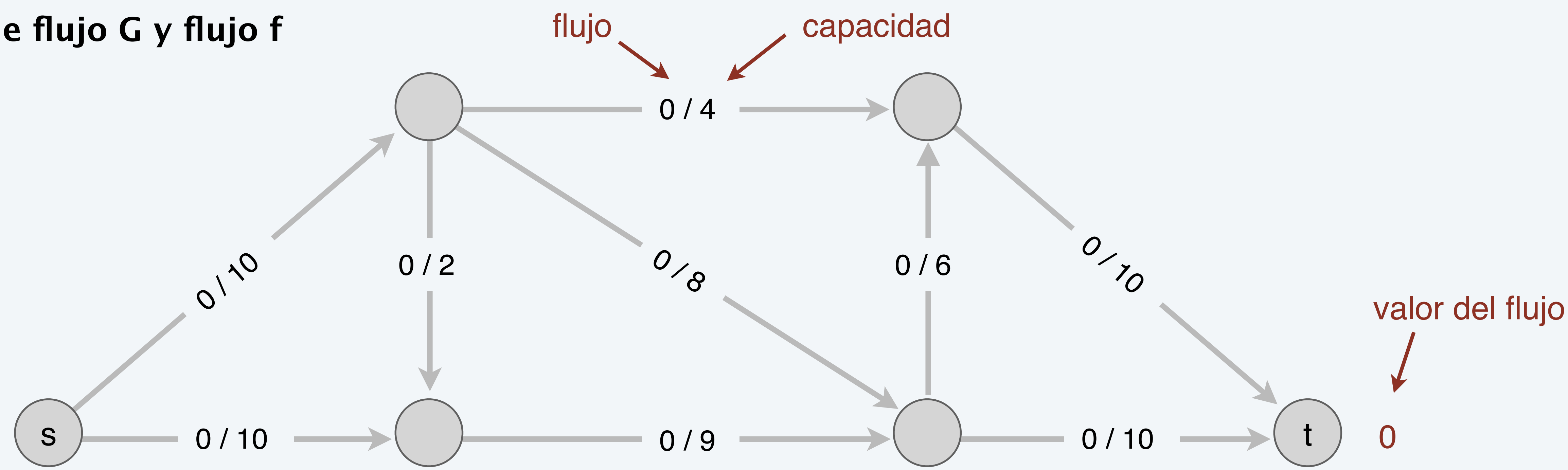
    $f \leftarrow$ AUGMENT (f, c, P).   Camino de aumento

    Actualizar G_f con respecto al nuevo flujo f.

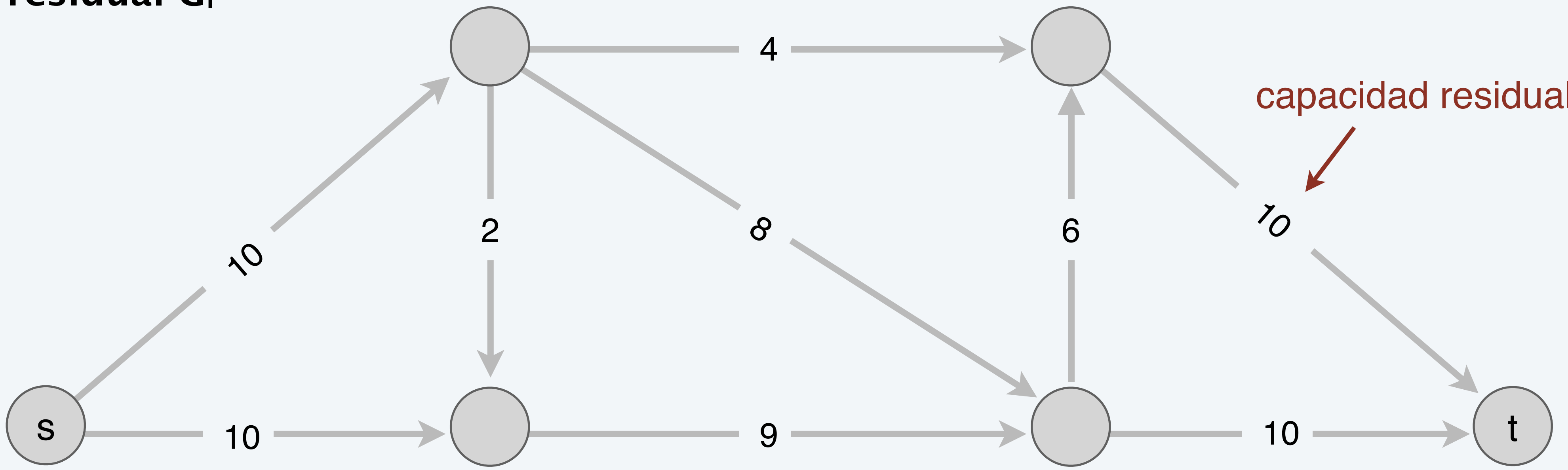
RETURN f.

Ejemplo de ejecución del algoritmo de Ford–Fulkerson

red de flujo G y flujo f

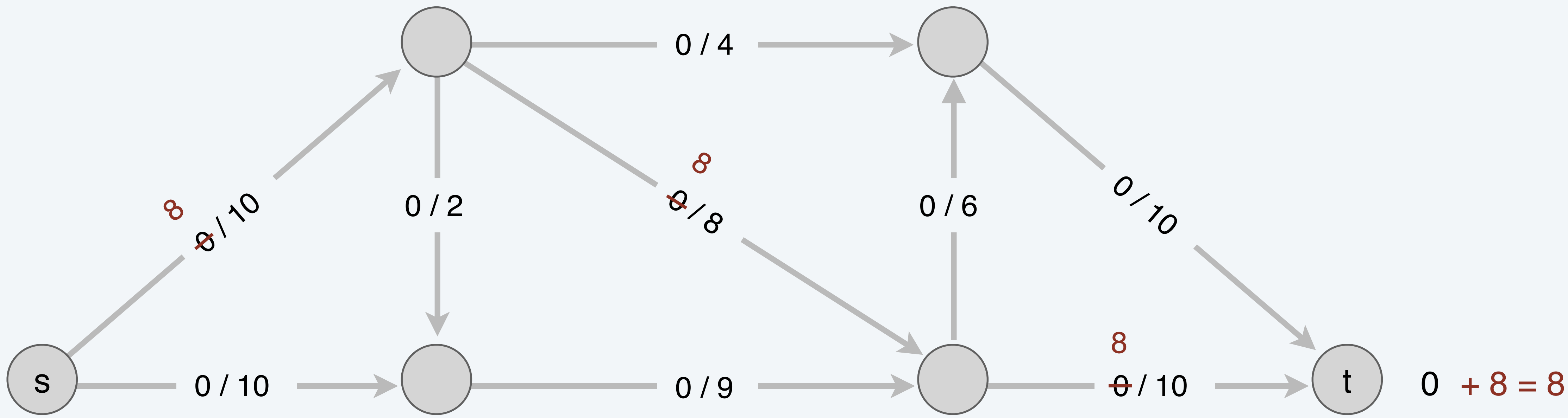


grafo residual G_f

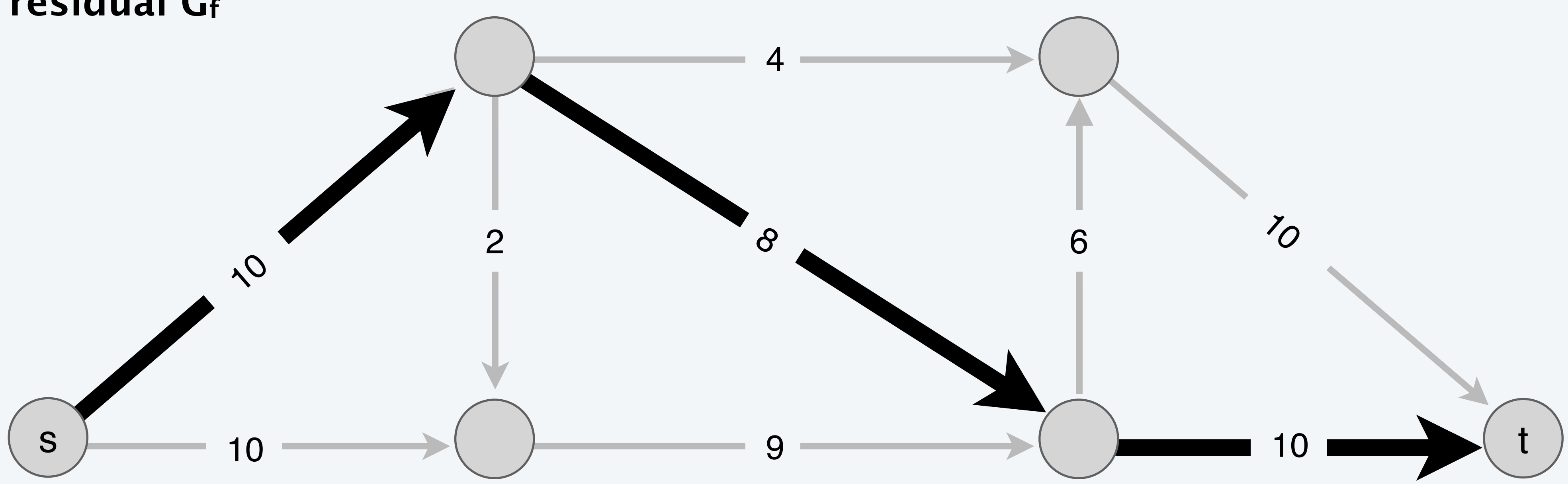


Ejemplo de ejecución del algoritmo de Ford–Fulkerson

red de flujo G y flujo f

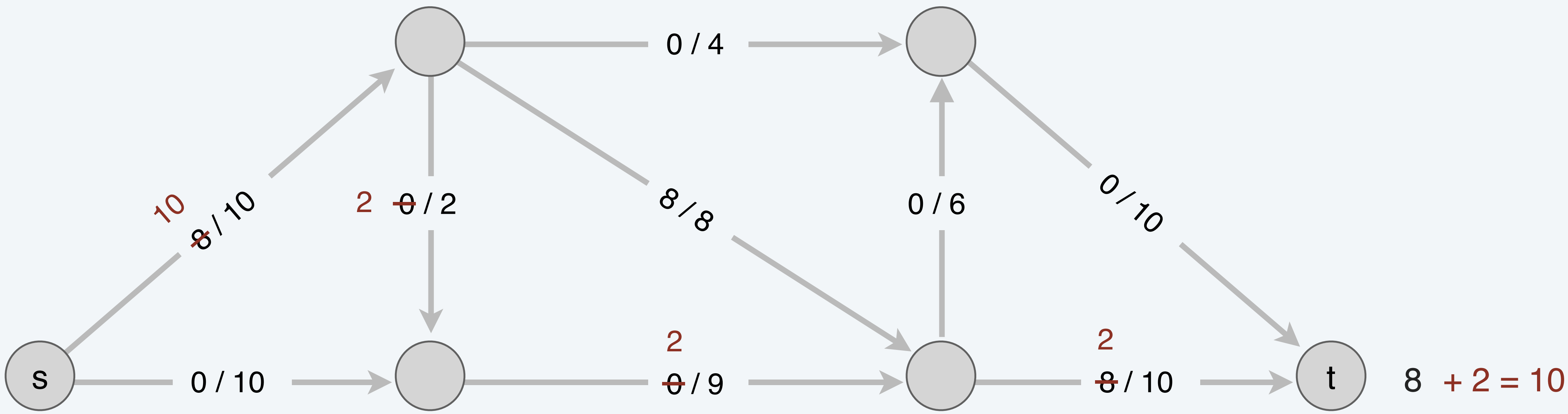


grafo residual G_f

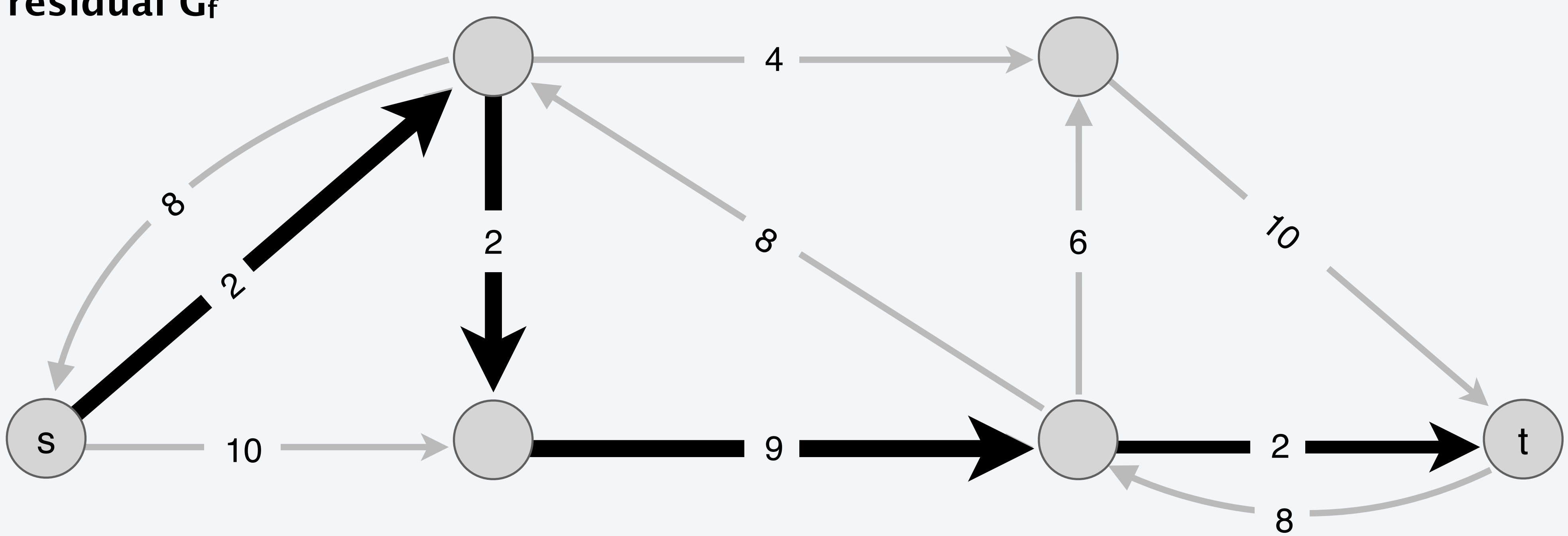


Ejemplo de ejecución del algoritmo de Ford–Fulkerson

red de flujo G y flujo f

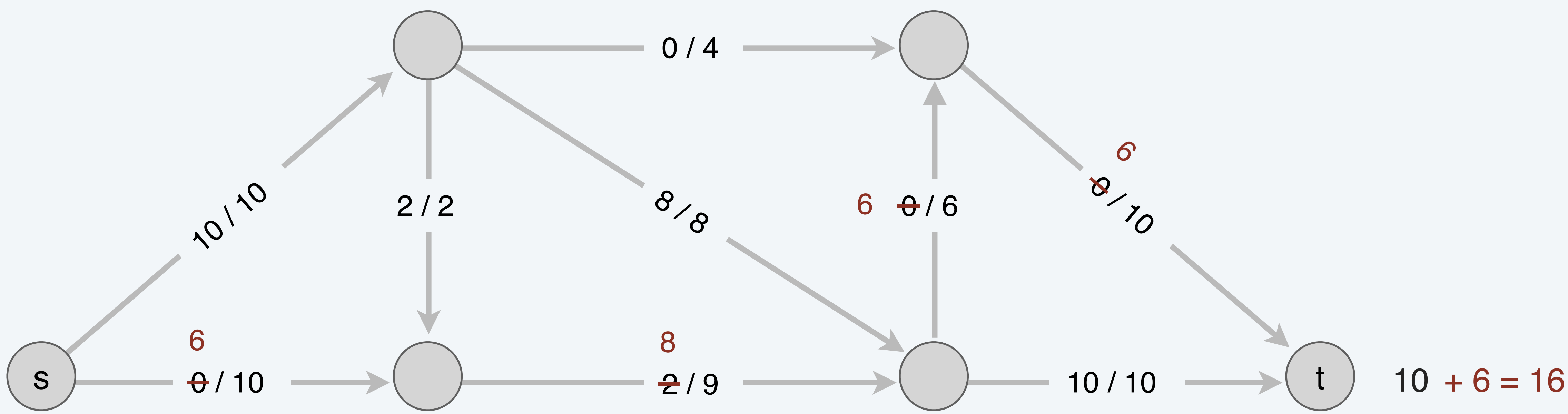


grafo residual G_f

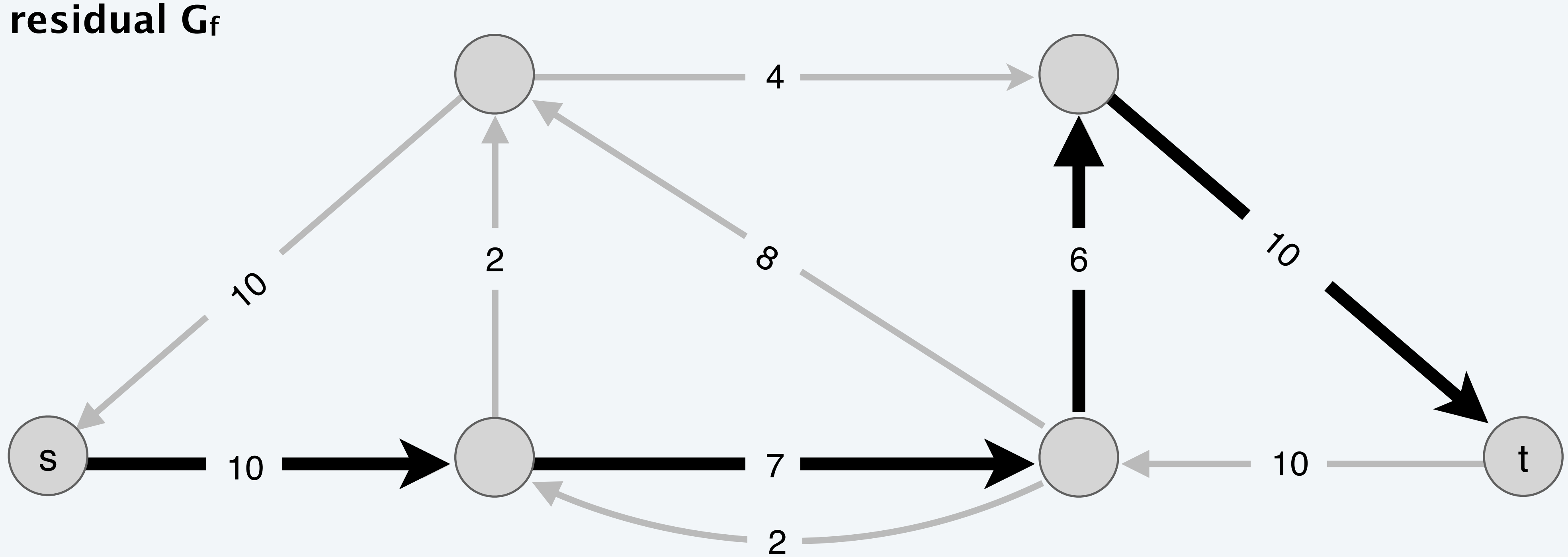


Ejemplo de ejecución del algoritmo de Ford–Fulkerson

red de flujo G y flujo f

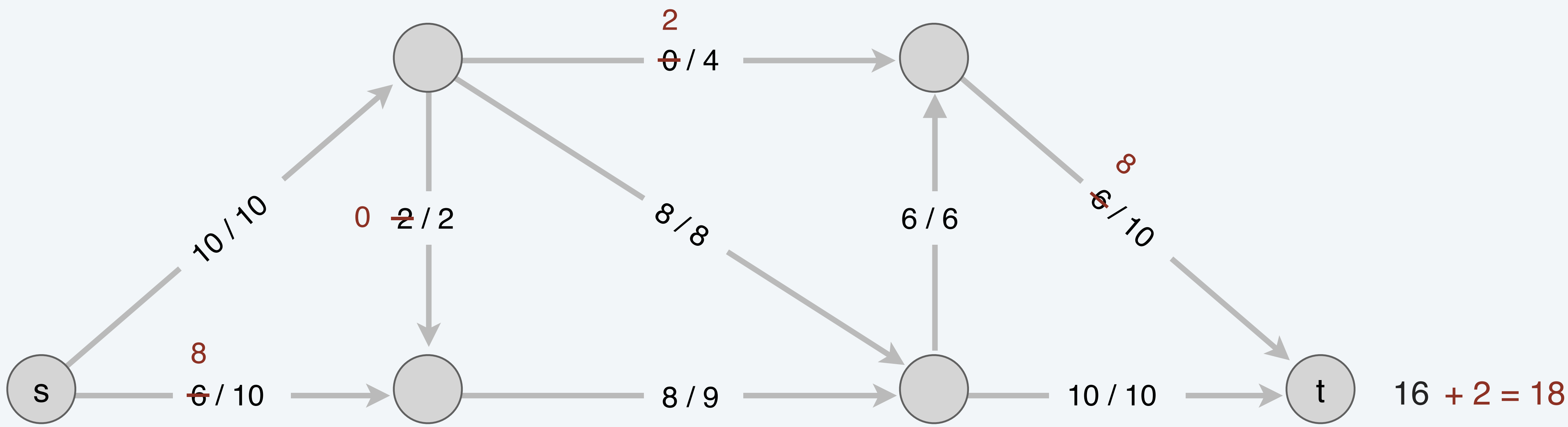


grafo residual G_f

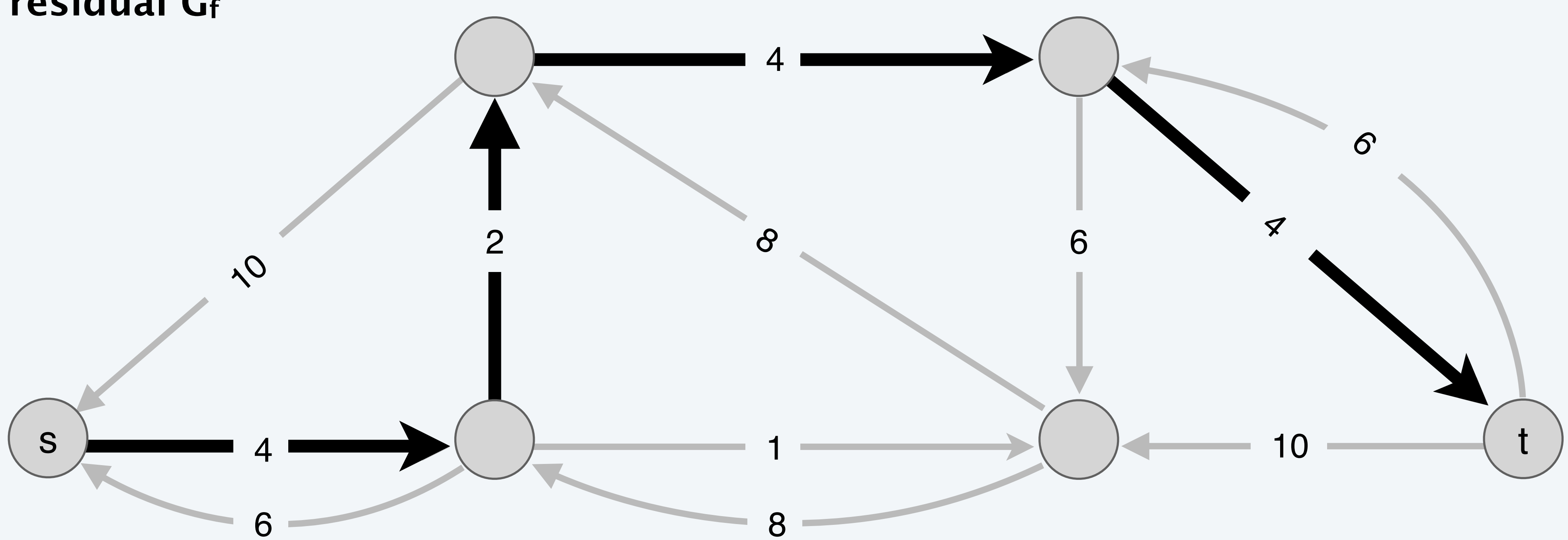


Ejemplo de ejecución del algoritmo de Ford–Fulkerson

red de flujo G y flujo f

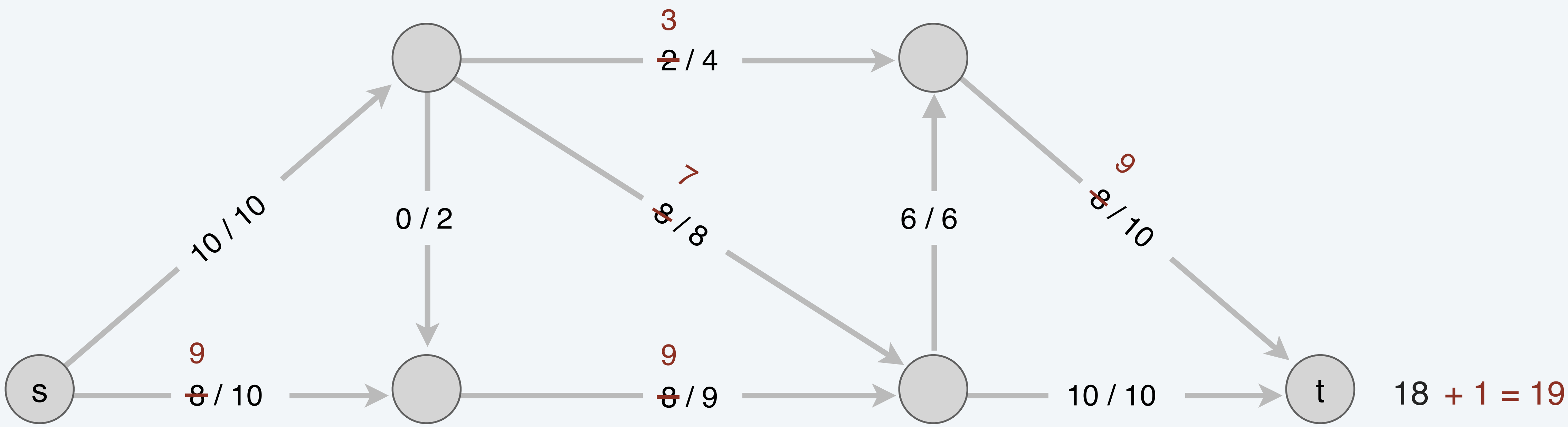


grafo residual G_f

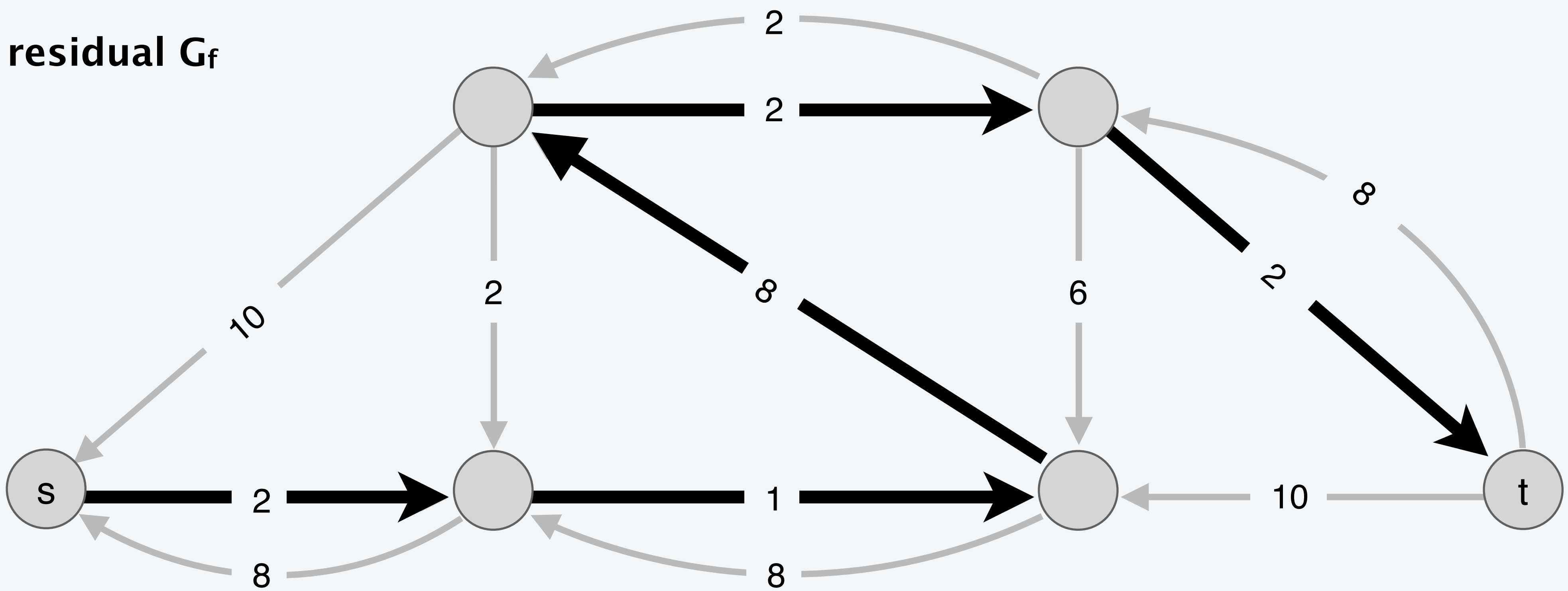


Ejemplo de ejecución del algoritmo de Ford–Fulkerson

red de flujo G y flujo f

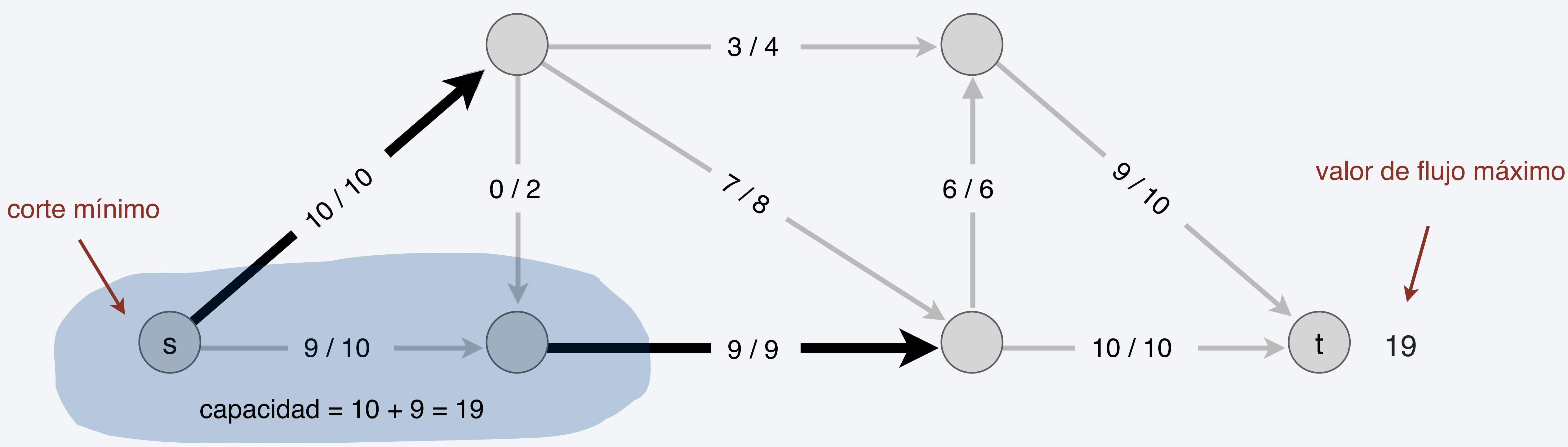


grafo residual G_f

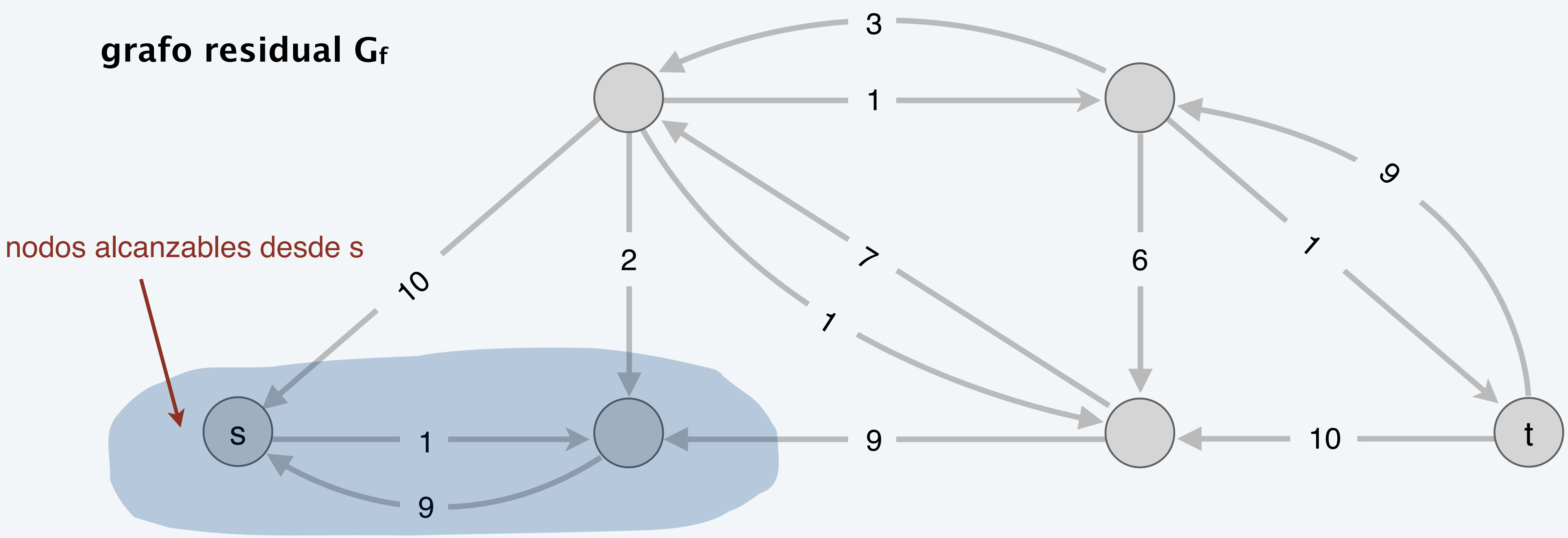


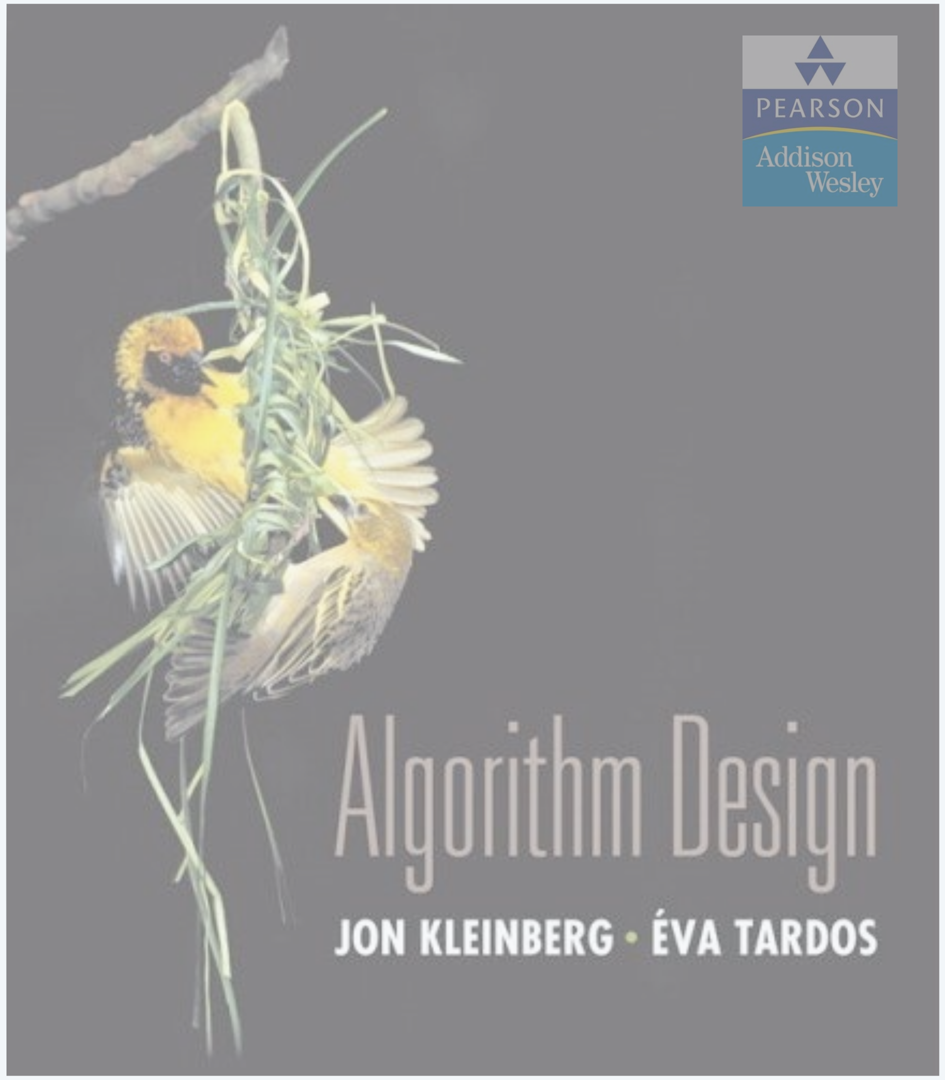
Ejemplo de ejecución del algoritmo de Ford–Fulkerson

red de flujo G y flujo f



grafo residual G_f





SECTION 7.1

7. REDES DE FLUJO

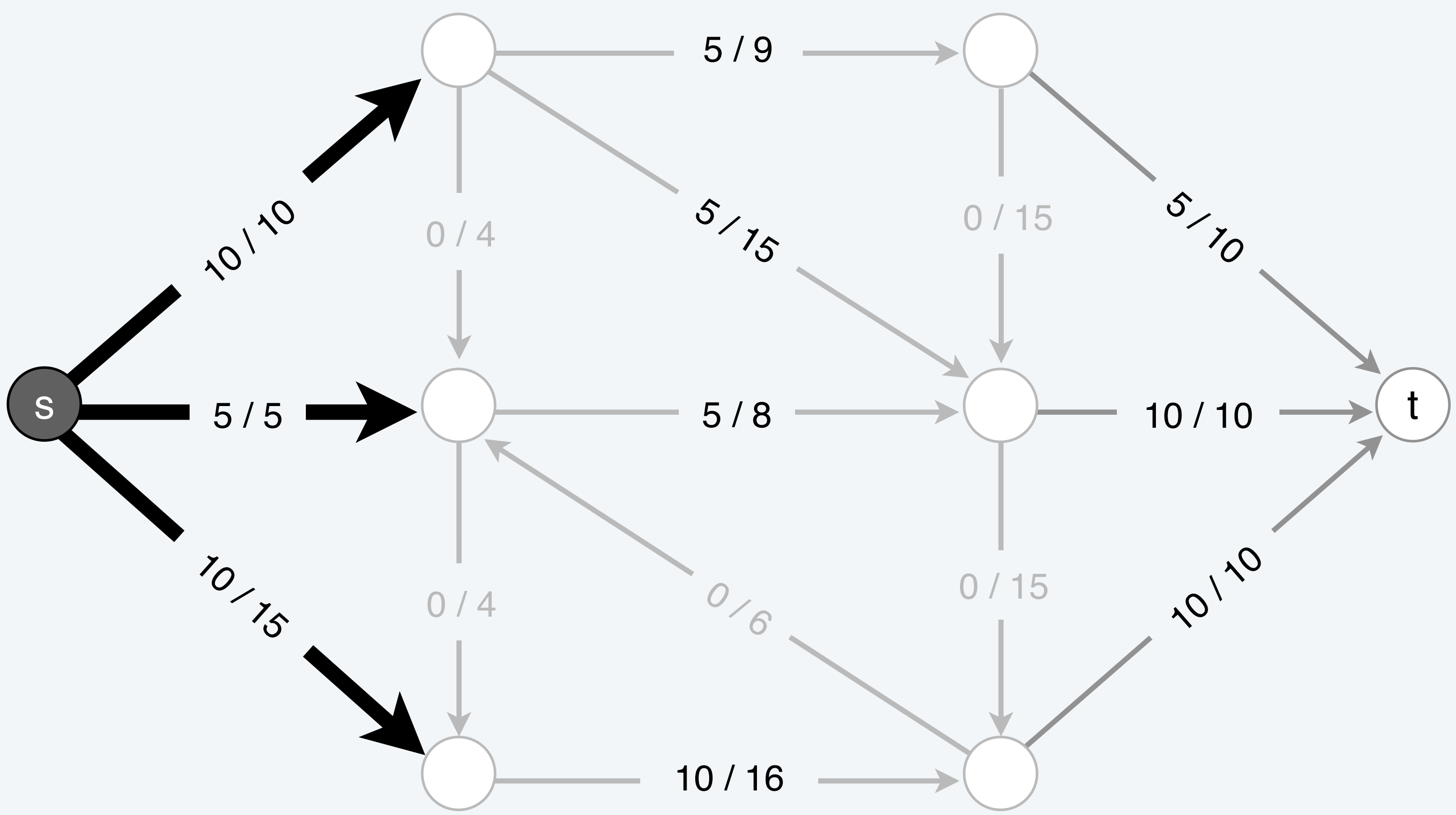
- *flujo máximo y corte mínimo*
- *algoritmo de Ford–Fulkerson*
- *teorema flujo máximo - corte mínimo*
- *tiempo de ejecución*
- *emparejamiento bipartito*

Relación entre flujos y cortes

Lema (valor de flujo). Sea f un flujo y (A, B) un corte.
El valor de f es igual al flujo neto a través del corte (A, B).

$$val(f) = \sum_{e \text{ saliente de } A} f(e) \; - \sum_{e \text{ entrante a } A} f(e)$$

flujo neto a través del corte = 10 + 5 + 10 = 25



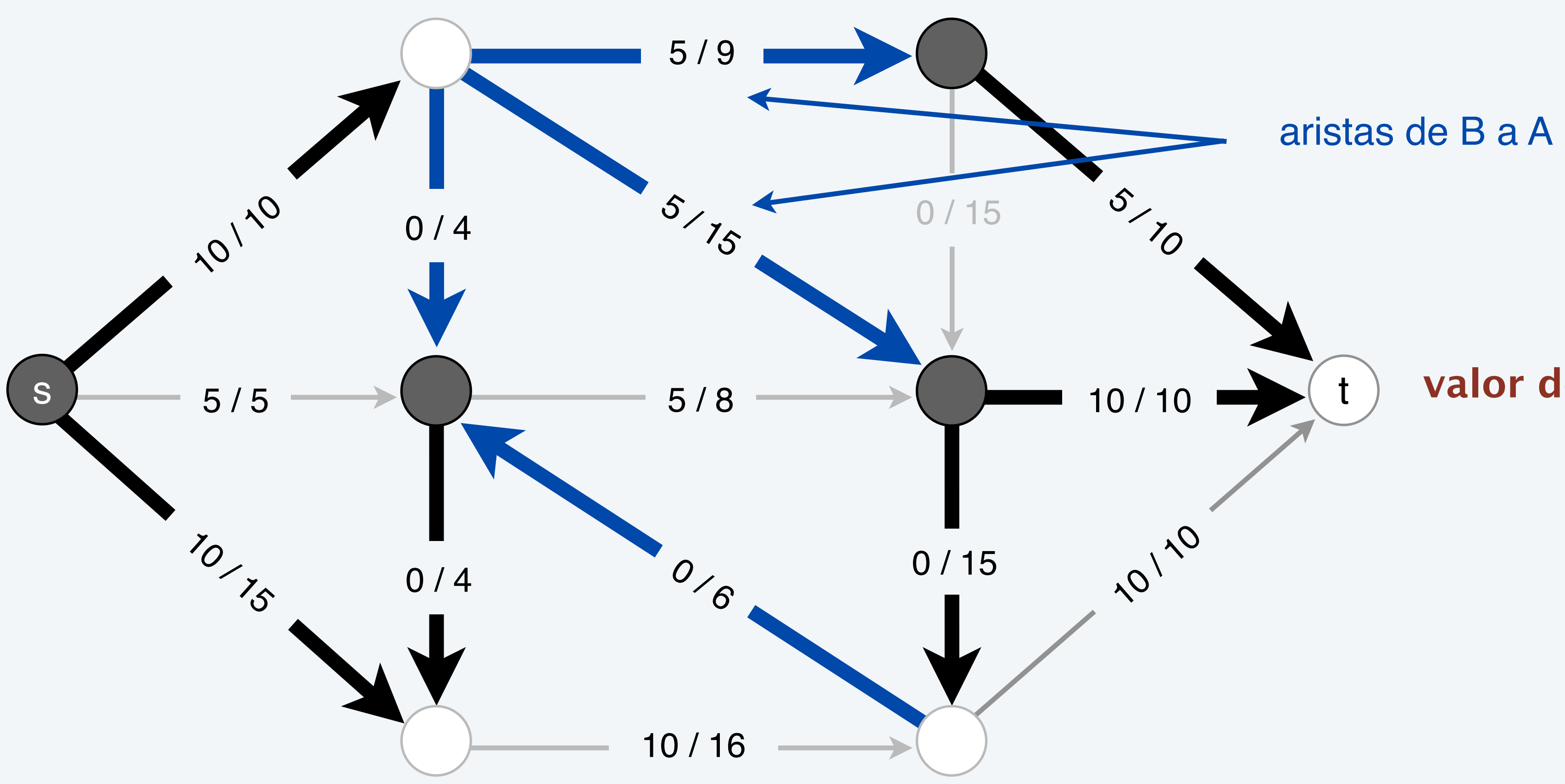
valor del flujo = 25

Relación entre flujos y cortes

Lema (valor de flujo). Sea f un flujo y (A, B) un corte.
El valor de f es igual al flujo neto a través del corte (A, B).

$$val(f) = \sum_{e \text{ saliente de } A} f(e) \; - \sum_{e \text{ entrante a } A} f(e)$$

flujo neto a través del corte = (10 + 10 + 5 + 10 + 0 + 0) – (5 + 5 + 0 + 0) = 25



valor del flujo = 25

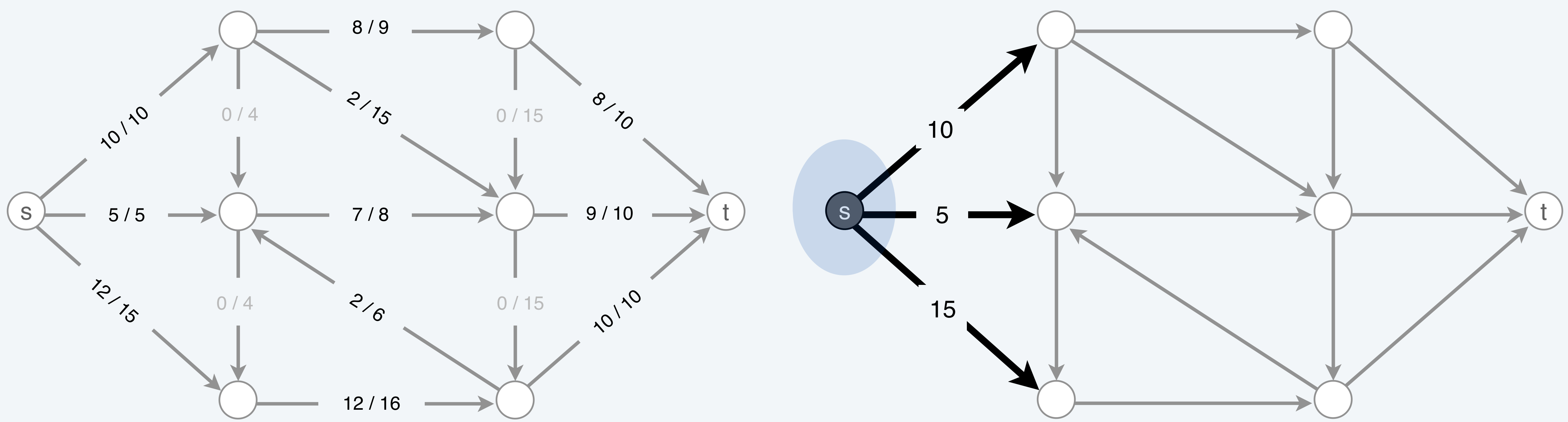
Relación entre flujos y cortes

Dualidad débil. Sea f un flujo y (A, B) un corte. Se cumple $v(f) \leq cap(A, B)$.

Prueba.

$$
\begin{aligned}
val(f) &= \sum_{e \text{ saliente de } A} f(e) \; - \sum_{e \text{ entrante a } A} f(e) \\
&\leq \sum_{e \text{ saliente de } A} f(e) \\
&\leq \sum_{e \text{ saliente de } A} c(e) \\
&= cap(A, B) \quad \blacksquare
\end{aligned}
$$

lema de valor de flujo

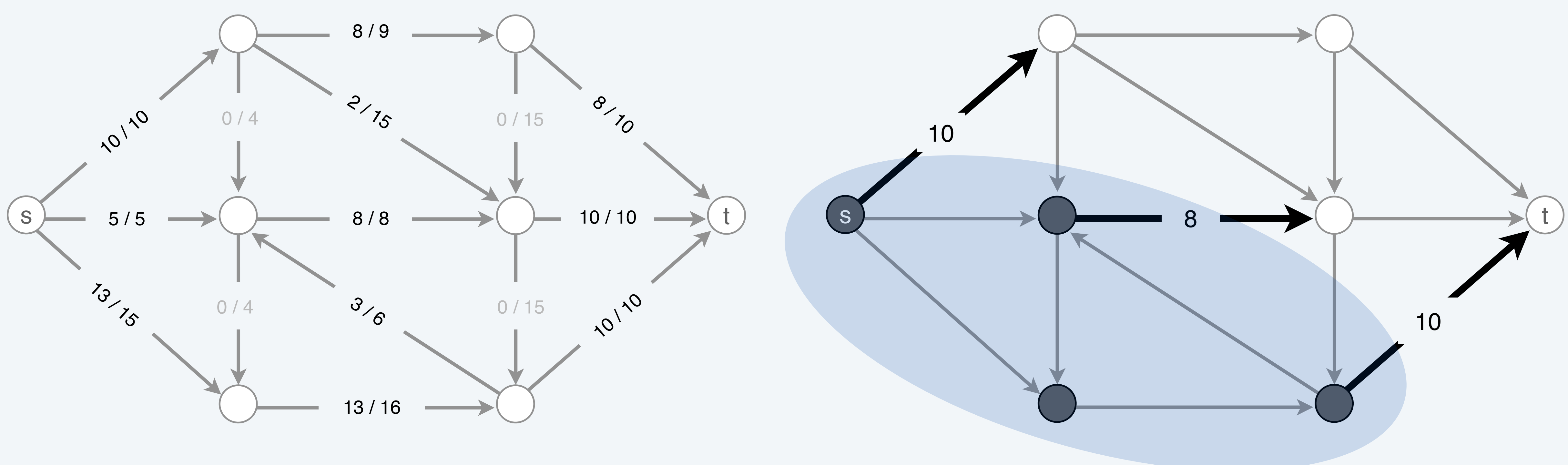


valor del flujo = 27 $\leq$ **capacidad del corte = 30**

Certificado de optimalidad

Corolario. Sea f un flujo y (A, B) un corte.

Si $val(f) = cap(A, B)$, entonces f es un flujo de valor máximo y (A, B) es un corte de capacidad mínima.



valor del flujo = 28 = **capacidad del corte = 28**

Teorema de flujo máximo - corte mínimo

Teorema del camino de aumento. Un flujo f es de valor máximo si y solo si no existe ningún camino de aumento con respecto a f.

Teorema de flujo máximo - corte mínimo. El máximo valor de flujo es igual a la mínima capacidad de corte.

Prueba. Las siguientes condiciones son equivalentes para cualquier flujo f:

i. Existe un corte (A, B) tal que $cap(A, B) = val(f)$.

ii. f es un flujo de valor máximo.

iii. No existe ningún camino de aumento con respecto a f.

Si el algoritmo de Ford–Fulkerson termina, entonces f tiene valor máximo

[i ⇒ ii]

- Es consecuencia del corolario anterior.

Teorema de flujo máximo - corte mínimo

Teorema del camino de aumento. Un flujo f es de valor máximo si y solo si no existe ningún camino de aumento con respecto a f.

Teorema de flujo máximo - corte mínimo. El máximo valor de flujo es igual a la mínima capacidad de corte.

Prueba. Las siguientes condiciones son equivalentes para cualquier flujo f :

i. Existe un corte (A, B) tal que $cap(A, B) = val(f)$.

ii. f es un flujo de valor máximo.

iii. No existe ningún camino de aumento con respecto a f.

[ii ⇒ iii] Contrarecíproco: ~iii ⇒ ~ii.

- Si existiera un camino de aumento, podría aumentarse el valor de f enviando flujo a lo largo de ese camino.

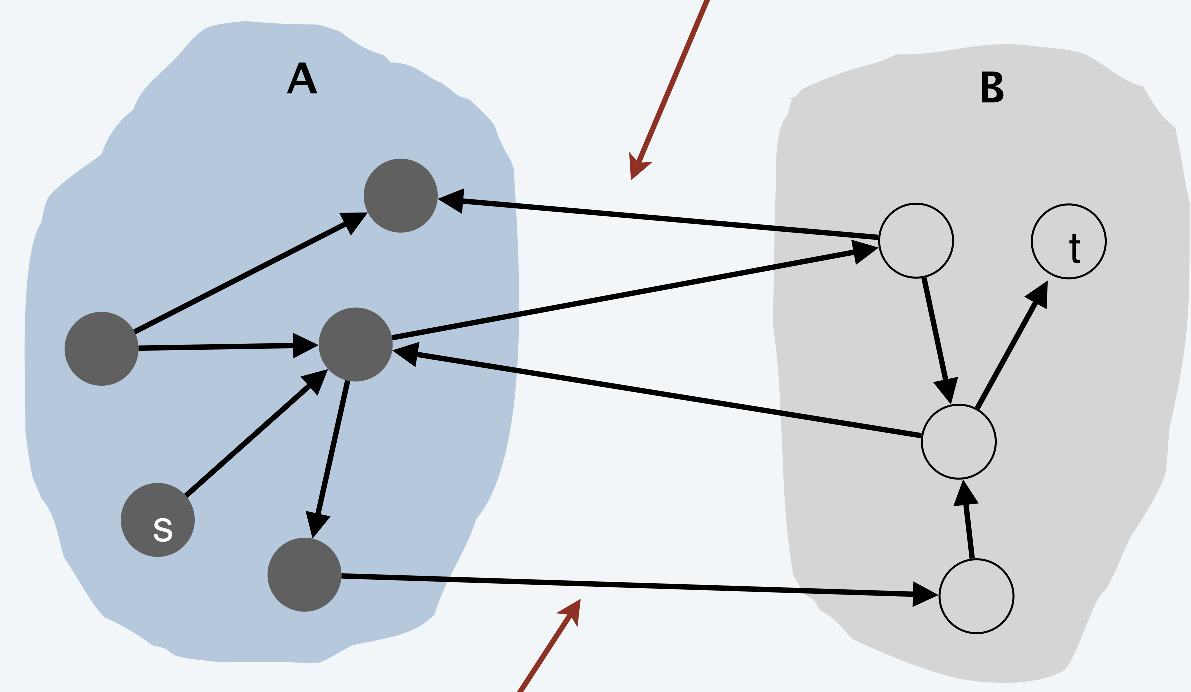
Teorema de flujo máximo - corte mínimo

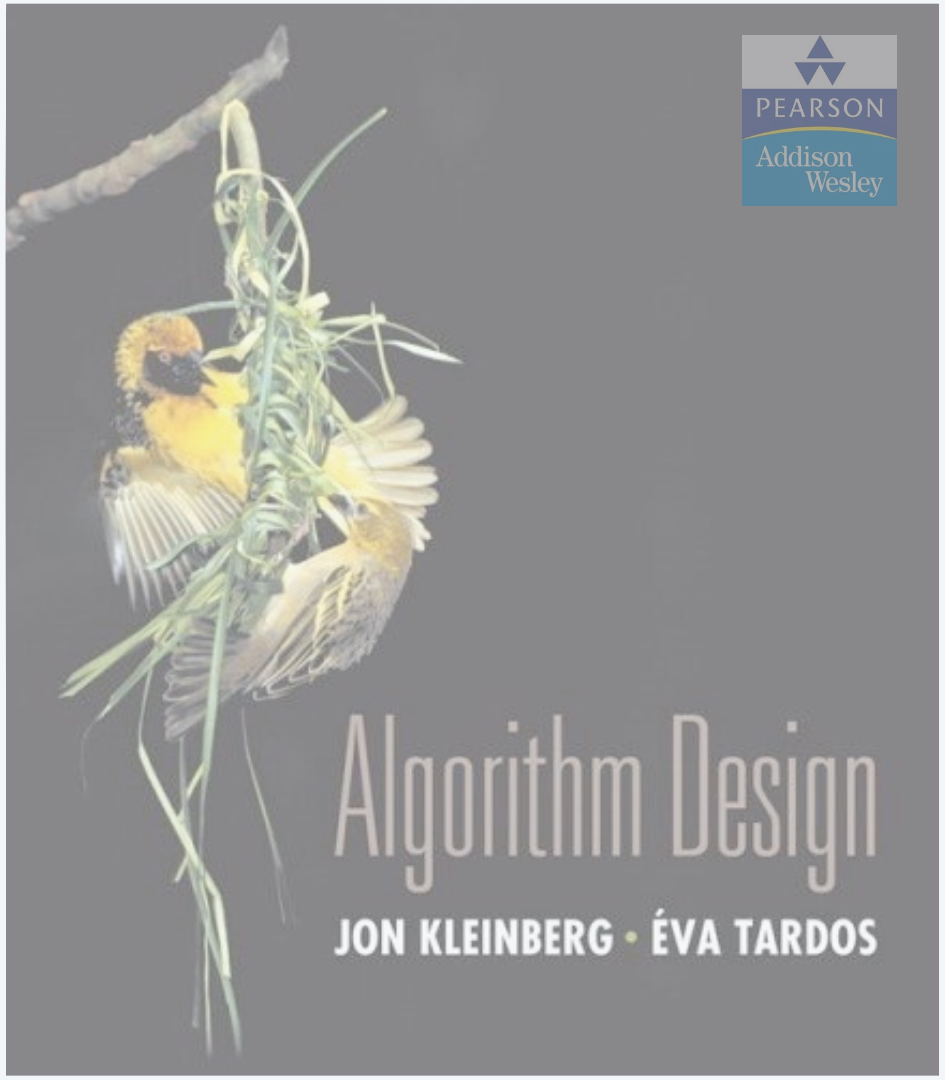
[iii ⇒ i]

- Sea f un flujo tal que no existe camino de aumento con respecto a f.
- Sea A el conjunto de nodos alcanzables desde s en el grafo residual G_f.
- Por definición de A, $s \in A$.
- Como no existe camino de aumento, $t \notin A$.
- Entonces (A, B) es un corte, con $B = V - A$.

$$
\begin{aligned}
val(f) &= \sum_{e \text{ saliente de } A} f(e) - \sum_{e \text{ entrante a } A} f(e) \\
&= \sum_{e \text{ saliente de } A} c(e) \\
&= cap(A, B)
\end{aligned}
$$

lema de
valor de flujo





SECTION 7.1

7. REDES DE FLUJO

- *flujo máximo y corte mínimo*
- *algoritmo de Ford–Fulkerson*
- *teorema flujo máximo - corte mínimo*
- *tiempo de ejecución*
- *emparejamiento bipartito*

Tiempo de ejecución

Asumimos.

- Todo vértice está conectado a al menos una arista.
- Las capacidades son números enteros.

Propiedad de integridad. A lo largo de la ejecución del algoritmo, $f(e)$ y las capacidades residuales $c_f(e)$ son enteros para toda arista e. Por la tanto el valor de f es entero.

Teorema. Sea C una cota superior para el valor de flujo máximo de una red G. El algoritmo de Ford-Fulkerson ejecutado en G termina en a lo sumo C iteraciones.
Prueba. En cada iteración el valor del flujo aumenta en al menos 1. ▪

Corolario. El tiempo de ejecución del algoritmo de Ford–Fulkerson es $O(mC)$.

Teorema. El algoritmo devuelve un flujo de valor máximo f^* tal que $f^*(e)$ es entero para toda arista e.
Prueba. Como el algoritmo termina, la tesis surge de la propiedad de integridad. ▪

Tiempo de ejecución

Observación. Una posible cota superior para el valor de flujo máximo es

$$C = \sum_{e \text{ saliente de } s} c(e)$$

Polinomial vs. pseudo-polinomial.

- Como el tiempo de ejecución es $O(mC)$, si C es polinomial en m, entonces el tiempo de ejecución es polinomial en el tamaño de la entrada.
- Pero en general, todo lo que podemos decir es que el tiempo de ejecución es pseudo-polinomial, porque depende del **valor** de C, no del **tamaño de su representación**.
- ¿Se puede implementar en tiempo estrictamente polinomial?

Algoritmo con escalamiento de capacidades

CAPACITY-SCALING (G)

FOREACH arista $e \in E : f(e) \leftarrow 0$.

$\Delta \leftarrow$ mayor potencia de 2 que no supera

max{c(e), e saliente de s}.

WHILE $(\Delta \geq 1)$

$G_f(\Delta) \leftarrow$ Descartar aristas de capacidad menor que Δ en G_f.

WHILE (existe un camino P de s a t en $G_f(\Delta)$)

$f \leftarrow$ AUGMENT (f, c, P).

Actualizar $G_f(\Delta)$ con respecto al nuevo flujo f.

$\Delta \leftarrow \Delta / 2$.

RETURN f.

Algoritmo con escalamiento de capacidades: tiempo de ejecución.

Lema 1. El ciclo externo se repite a lo sumo $1 + \lceil \log_2 C \rceil$ veces.

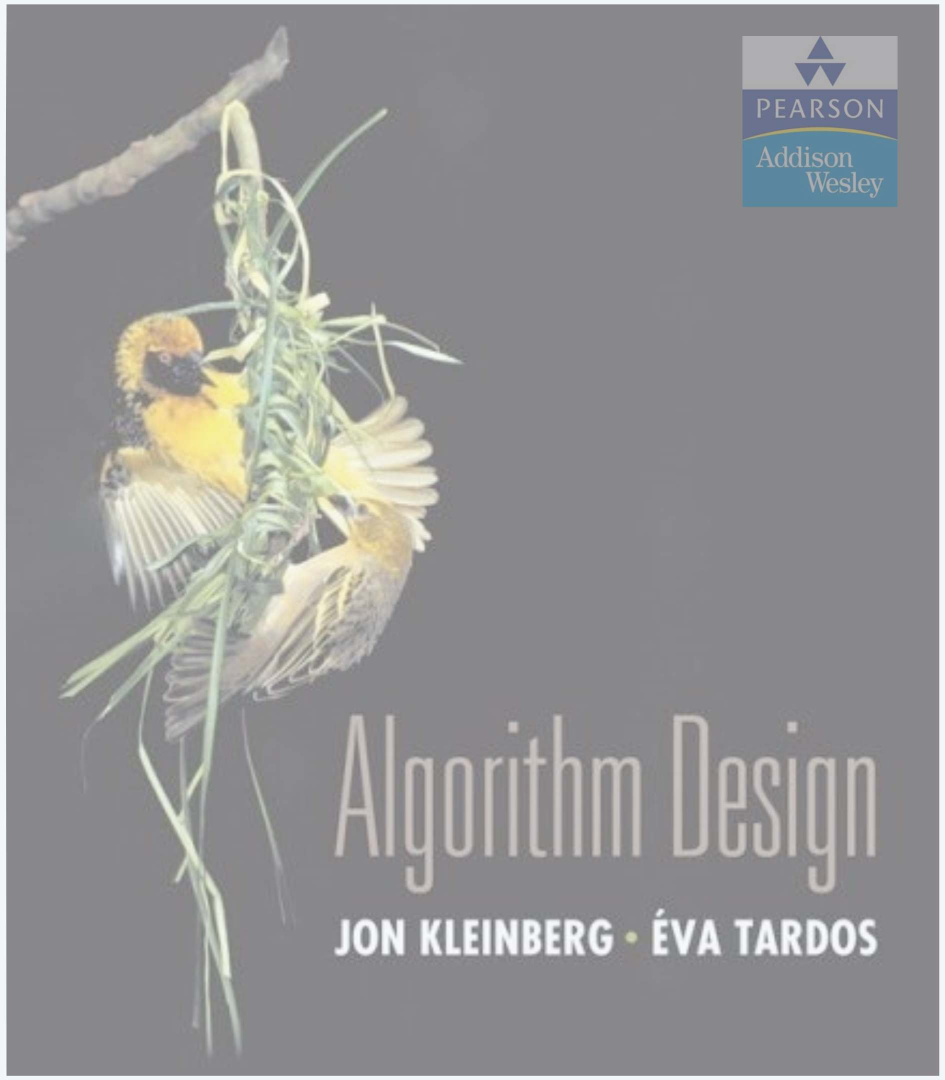
Lema 2. Cada ejecución del ciclo interno se repite a lo sumo $2m$ veces.

Idea.

- Cada camino de aumento incrementa $val(f)$ en al menos Δ.
- $\Rightarrow$ No puede haber "demasiados" aumentos.

Teorema. El algoritmo de Ford-Fulkerson con escalamiento de capacidades admite una implementación con tiempo de ejecución $O(m^2 \log C)$.

Observación. Es polinomial en el tamaño de la entrada!



SECTION 7.1

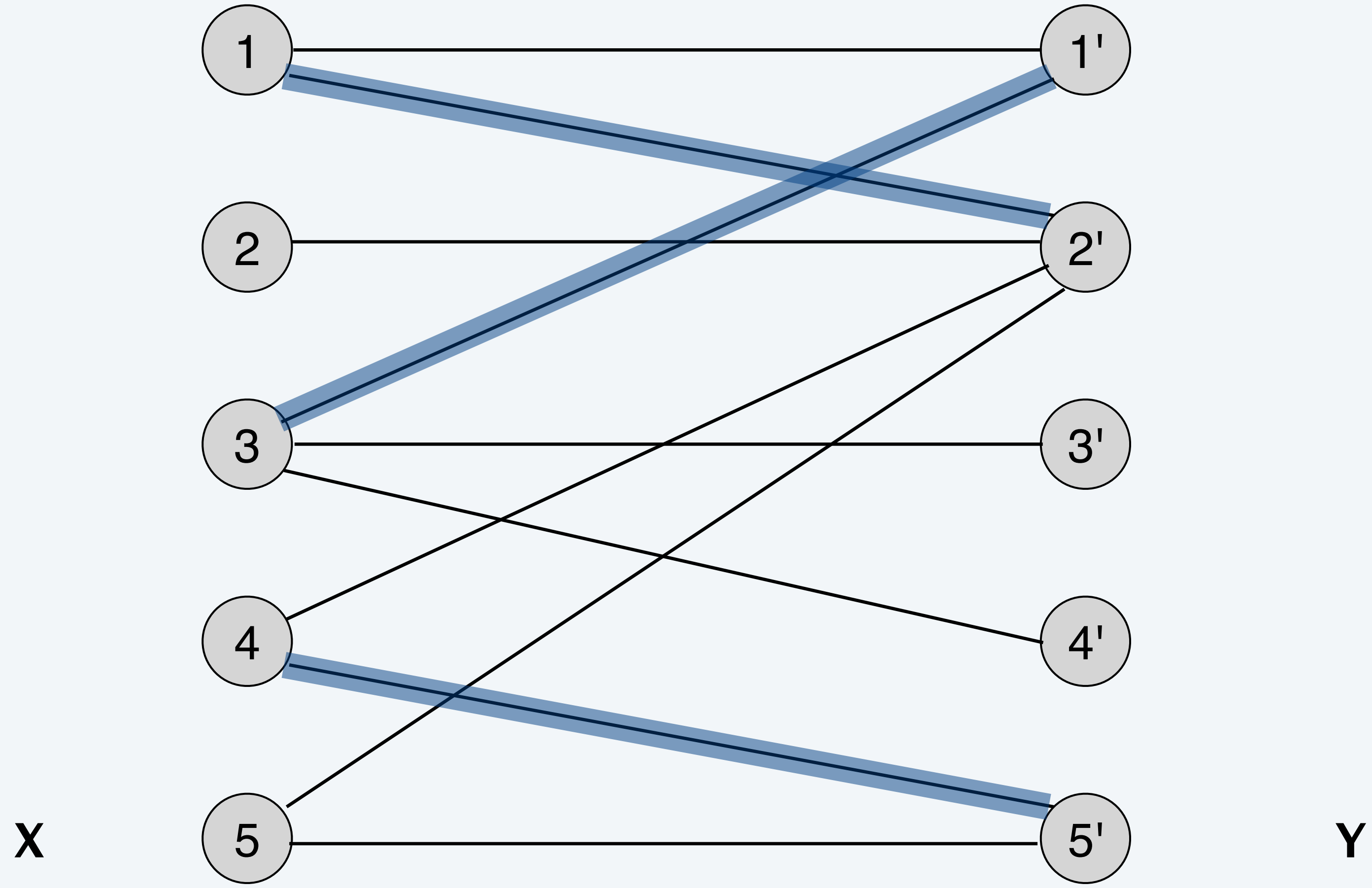
7. Redes de Flujo

- *flujo máximo y corte mínimo*
- *algoritmo de Ford–Fulkerson*
- *teorema flujo máximo - corte mínimo*
- *tiempo de ejecución*
- *emparejamiento bipartito*

Emparejamiento bipartito

Def. Un grafo G es bipartito si existe una partición de los vértices en dos subconjuntos X, Y, tal que toda arista conecta un vértice de X con uno de Y.

Emparejamiento bipartito. Dado un grafo bipartito $G = (X \cup Y, E)$, encontrar un emparejamiento de tamaño máximo.

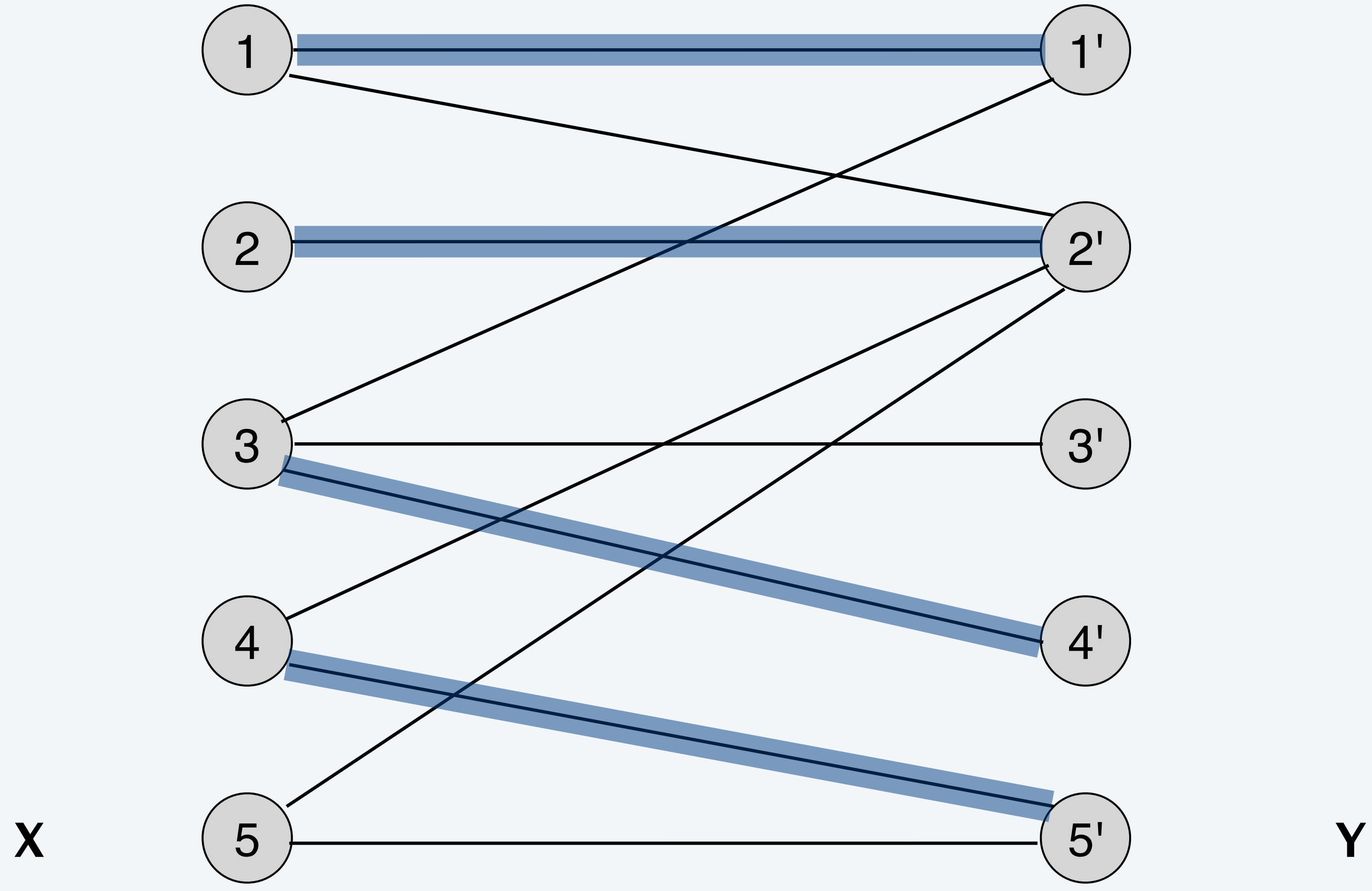


emparejamiento: 1–2', 3–1', 4–5'

Emparejamiento bipartito

Def. Un grafo G es bipartito si existe una partición de los vértices en dos subconjuntos X, Y, tal que toda arista conecta un vértice de X con uno de Y.

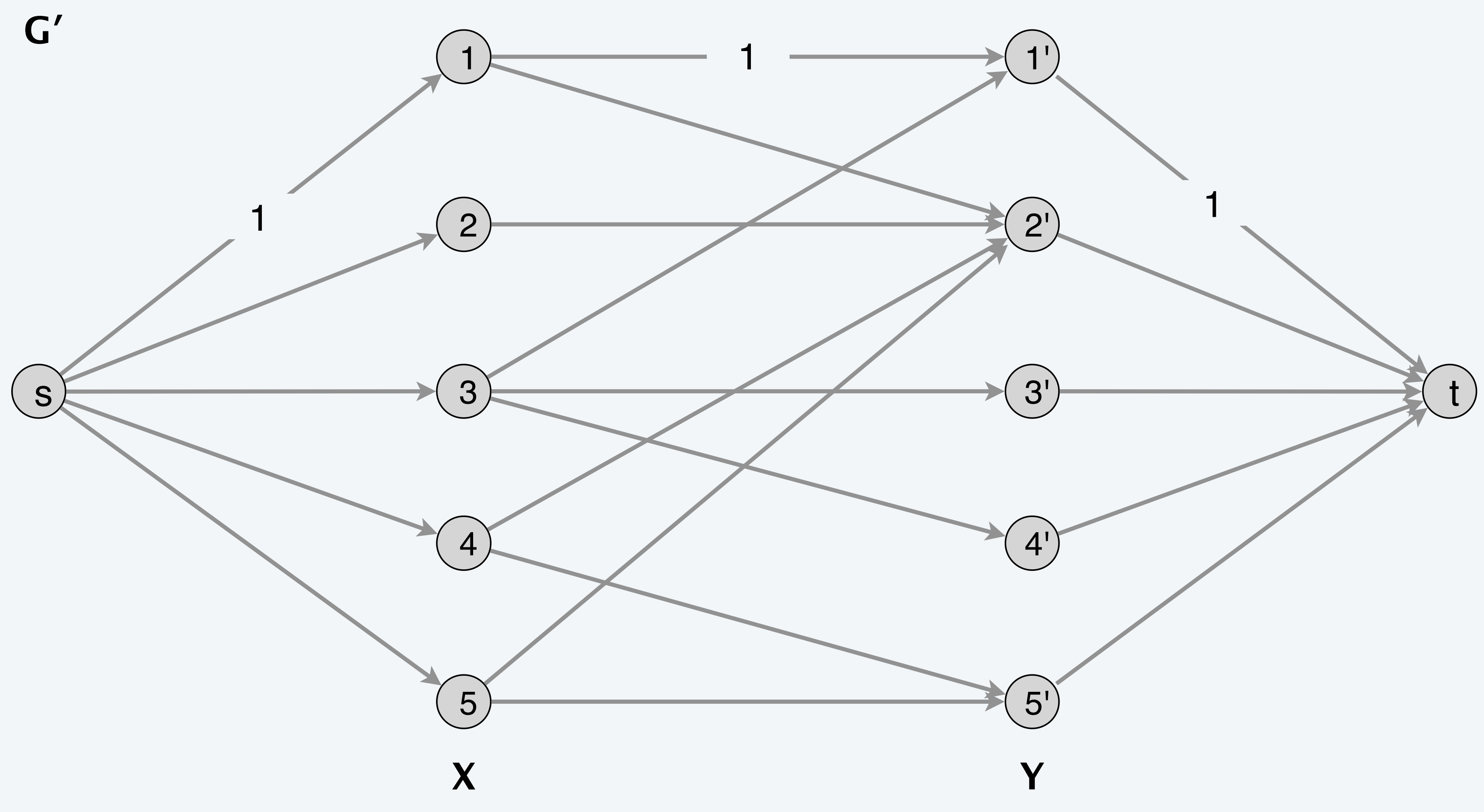
Emparejamiento bipartito. Dado un grafo bipartito $G = (X \cup Y, E)$, encontrar un emparejamiento de tamaño máximo.



emparejamiento: 1–1', 2–2', 3–4', 4–5'

Emparejamiento bipartito: formulación como problema de flujo

- Crear un grafo dirigido $G' = (X \cup Y \cup \{s, t\},\ E')$.
- Dirigir todas las aristas desde X hacia Y, y asignarles capacidad 1.
- Agregar fuente s, y aristas con capacidad 1 de s a cada vértice de X.
- Agregar destino t, y aristas con capacidad 1 de cada vértice de Y a t.

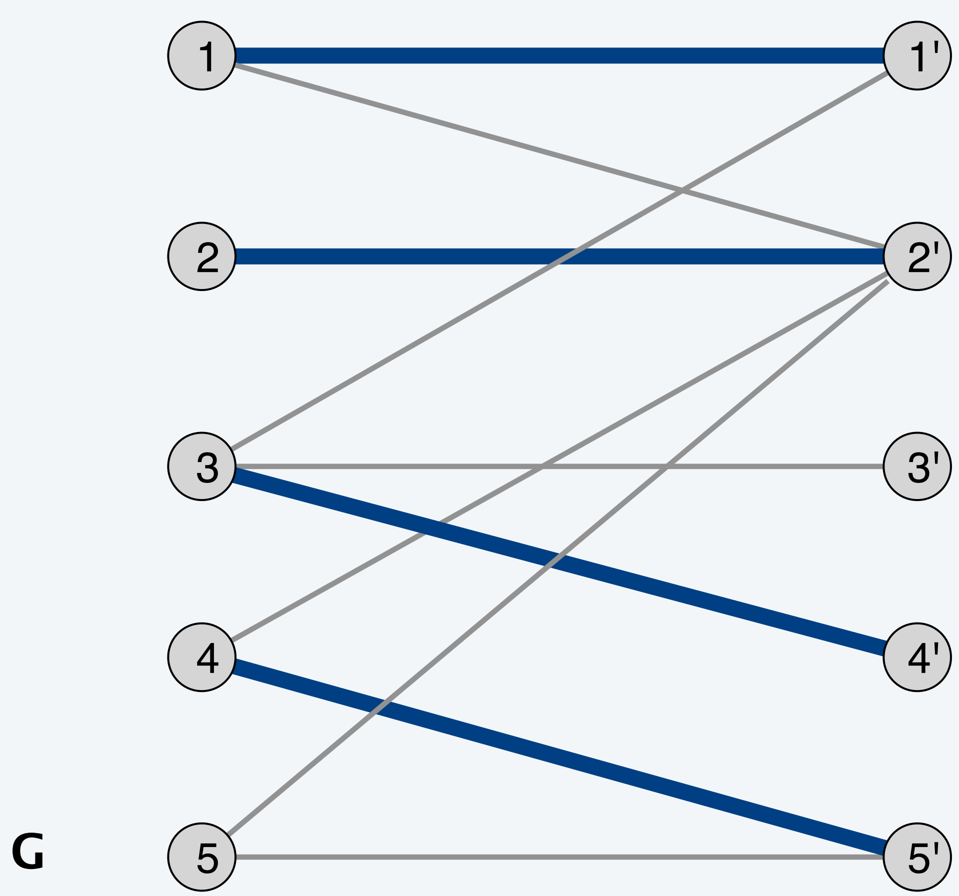


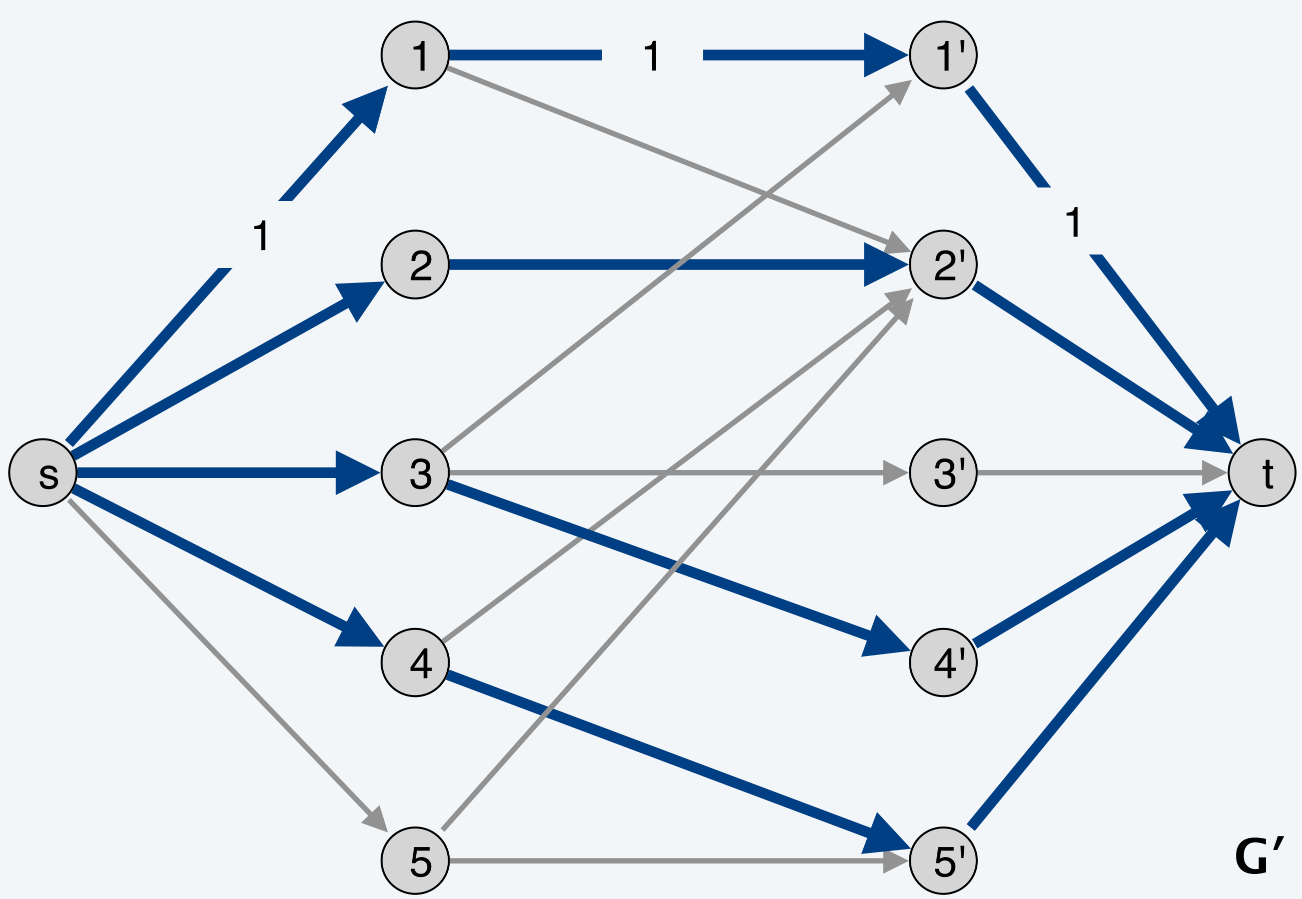
Corrección

Teorema. Tamaño máximo de emparejamiento en G = máximo valor de flujo en G'.

Boceto de prueba. ≤

- Dado un emparejamiento M de tamaño k.
- Considerar el flujo f que envía una unidad a lo largo de cada uno de los k caminos de s a t que pasan por las aristas de M.
- f es un flujo (verificar), y tiene valor k. ▪





Corrección

Teorema. Tamaño máximo de emparejamiento en G = máximo valor de flujo en G'.

Boceto de prueba. $\geq$

propiedad de integridad

- Sea f un flujo **entero** en G' de valor máximo k (también entero).
- $f(e)$ pertenece a $\{0, 1\}$ para toda arista e.
- Considerar M = conjunto de aristas de X a Y con $f(e) = 1$.
  - cada vértice de X y de Y participa en a lo sumo una arista de M (verificar).
  - $|M| = k$: considerar el corte $(X \cup \{s\}, Y \cup \{t\})$ y lema de valor de flujo ▪