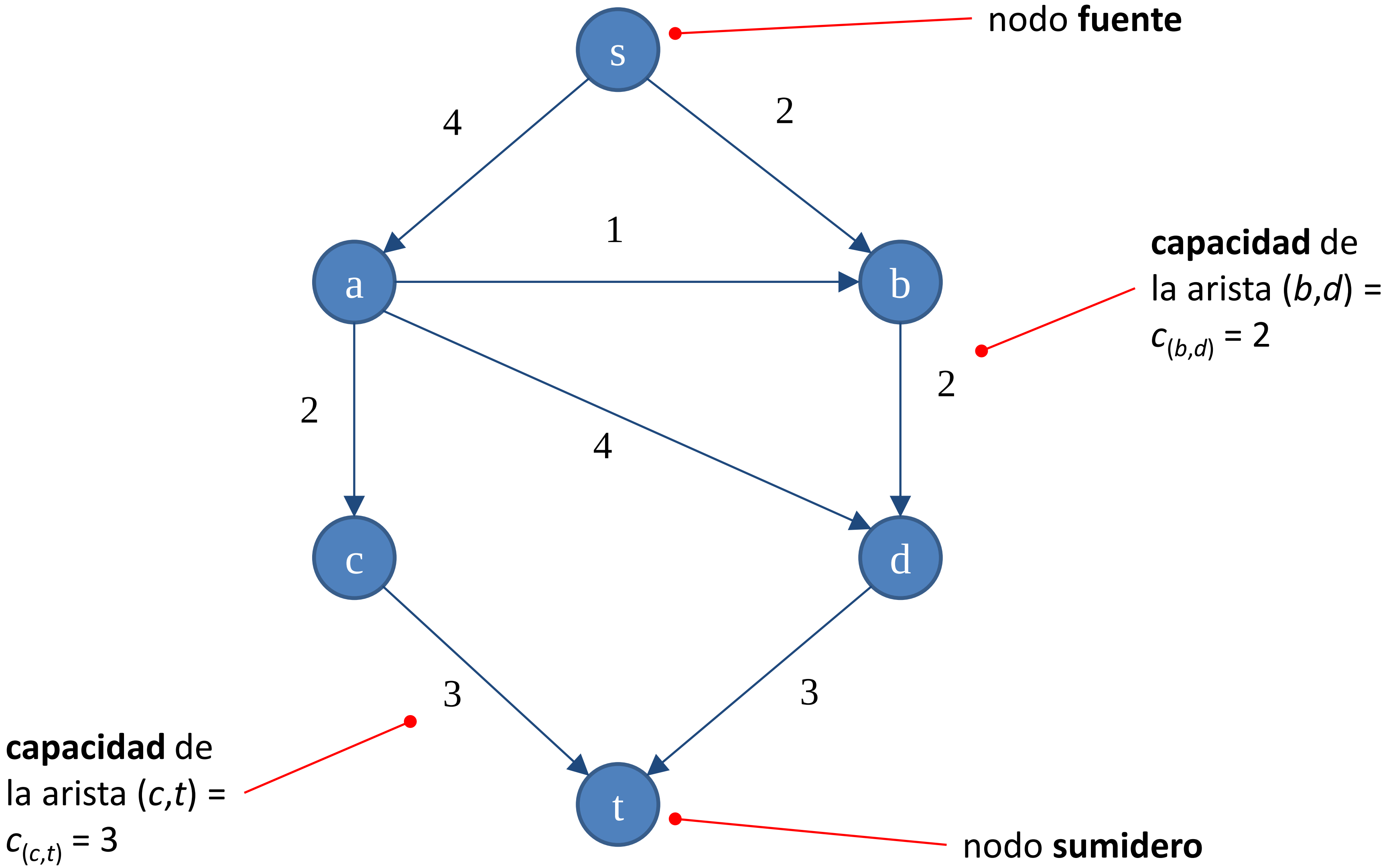
Flujo máximo en redes

IIC2133

Yadran Eterovic

1-2019


s
a
b
c
d
t
4
2
1
2
4
2
3
3
nodo **fuente**
capacidad de la arista (b,d) = $c_{(b,d)} = 2$
capacidad de la arista (c,t) = $c_{(c,t)} = 3$
nodo **sumidero**

Una **red de flujo** es un grafo direccional $G = (V, E)$:

- cada arista e tiene una **capacidad**, un número entero no negativo c_e
- hay un único **nodo fuente** s —suponemos que ninguna arista llega a s
- hay un único **nodo sumidero** t —… y que ninguna arista sale de t

Un **flujo *s*-*t*** es una función f que asigna a cada arista e un número real no negativo $f(e)$

… que satisface dos propiedades:

- para cada arista e, $0 \leq f(e) \leq c_e$
- para cada nodo v distinto de s y t,

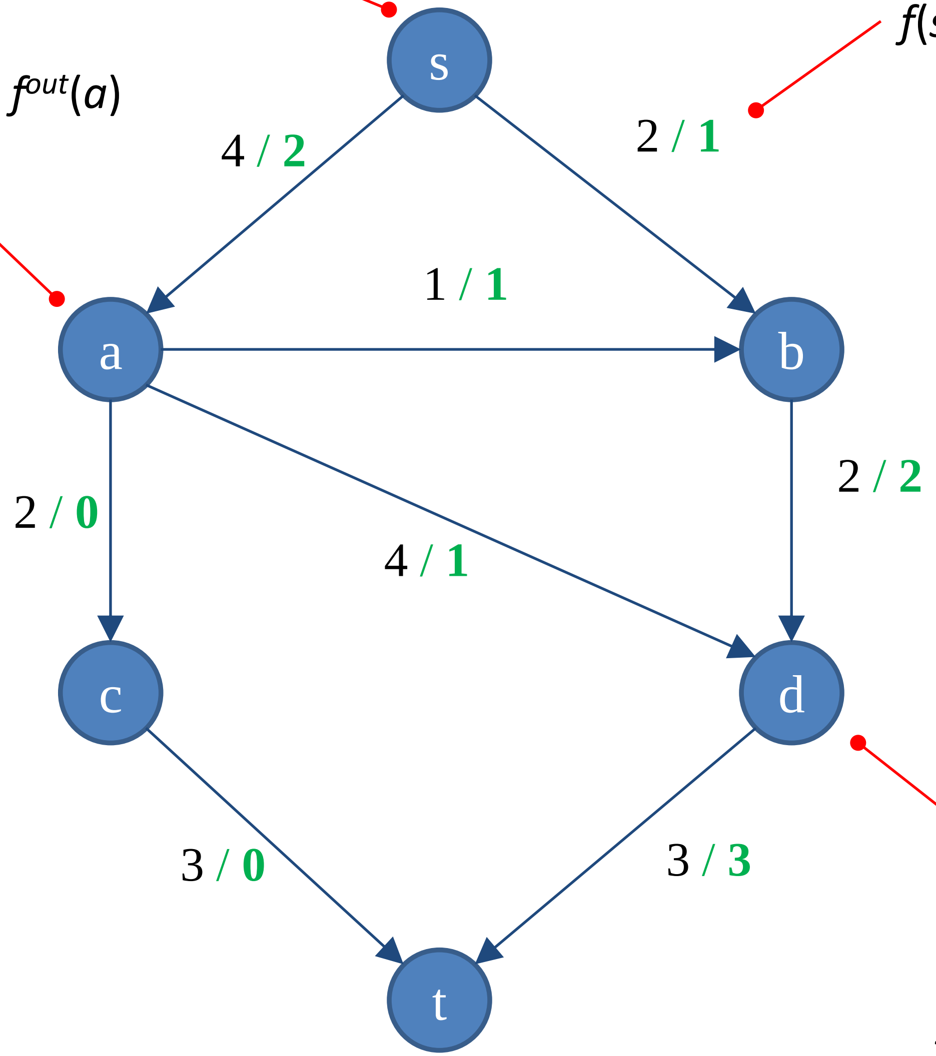
  … la suma de los flujos en las aristas que llegan a v, $f^{in}(v)$

  … debe ser igual a la suma de los flujos en las aristas que salen de v, $f^{out}(v)$

$v(f) = f^{out}(s) = 2 + 1 = 3$

$f(s,b) = 1$

$f^{in}(a) = 2 = 1+1+0 = f^{out}(a)$



$f(a,c) = 0$

$f^{in}(d) = 2+1 = 3 = f^{out}(d)$

Solo la fuente *s* puede generar flujo

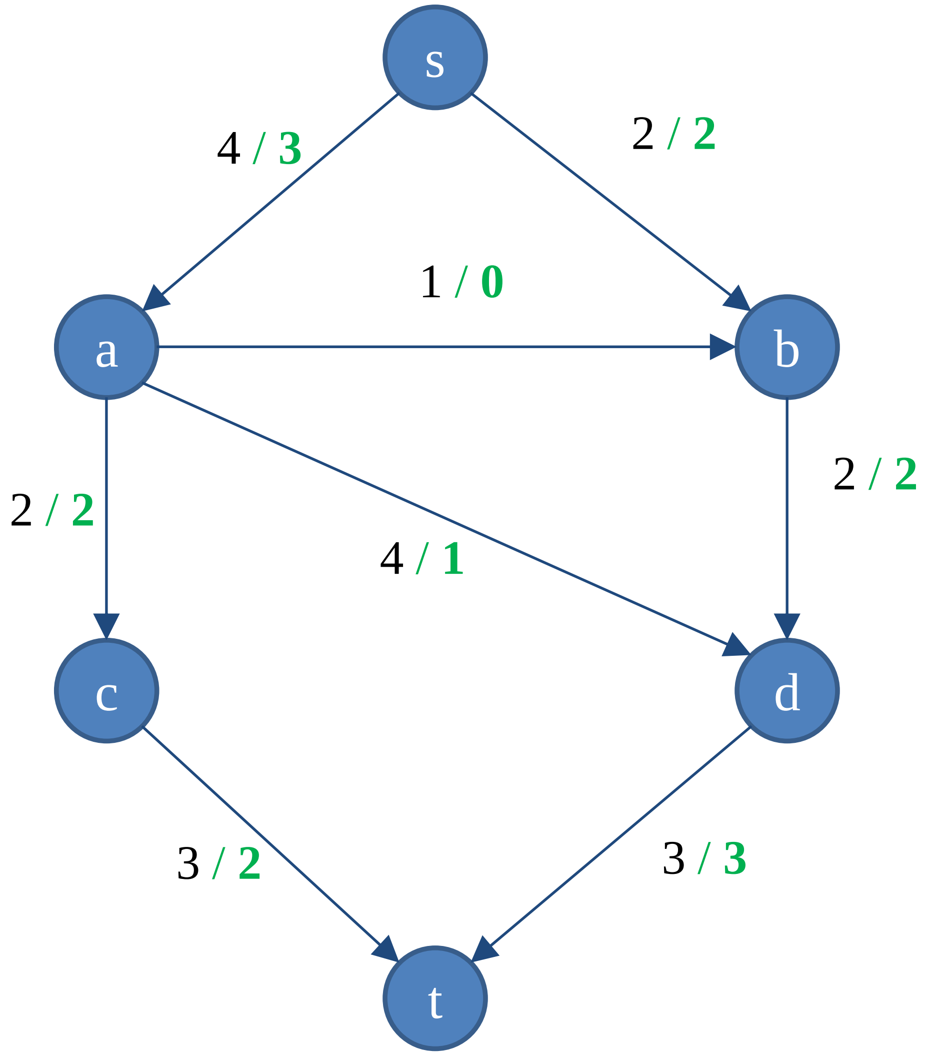
… y solo el sumidero *t* puede consumir flujo

El valor de un flujo *f*, denotado por $v(f)$, es la cantidad de flujo generado en la fuente:

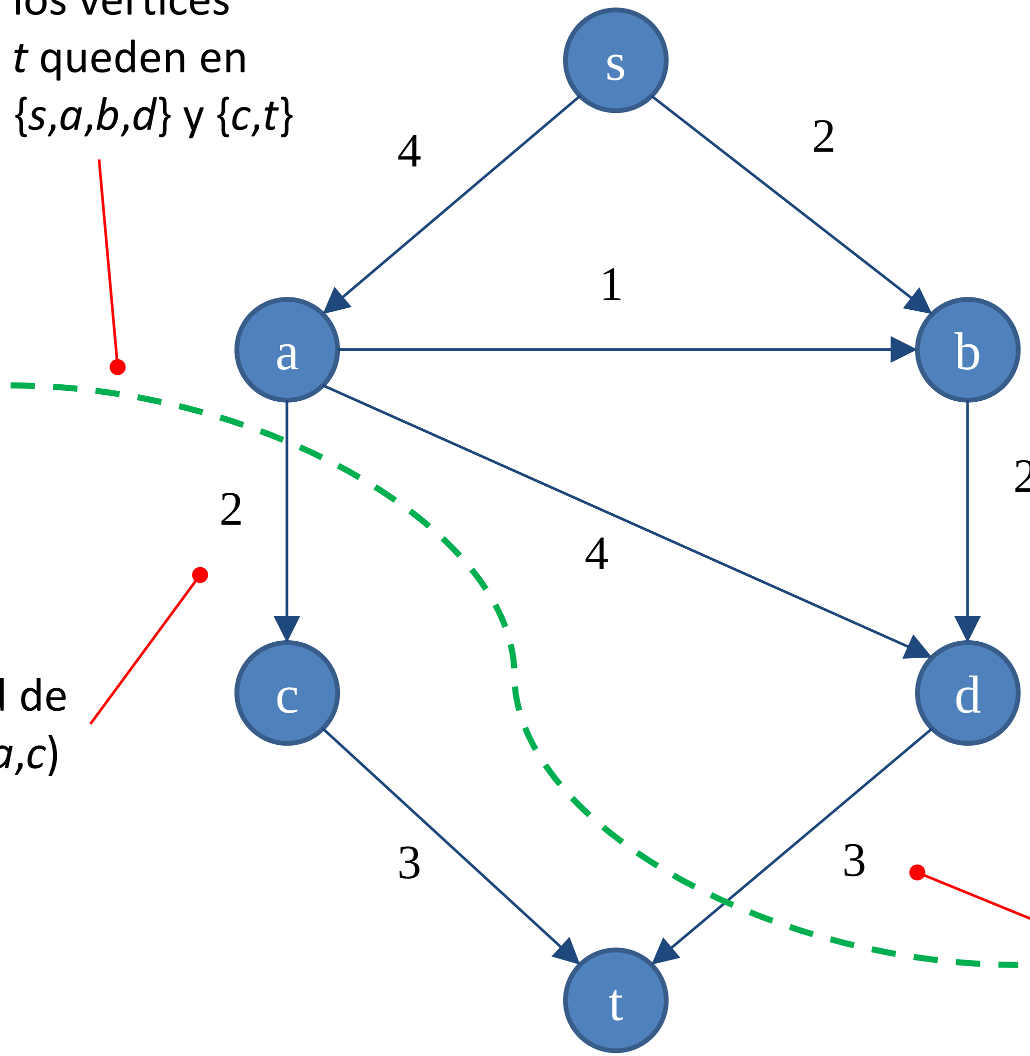
- $v(f) = f^{out}(s)$

El **problema del flujo máximo** consiste en que dada una red de flujo,

… hay que encontrar un flujo de valor máximo

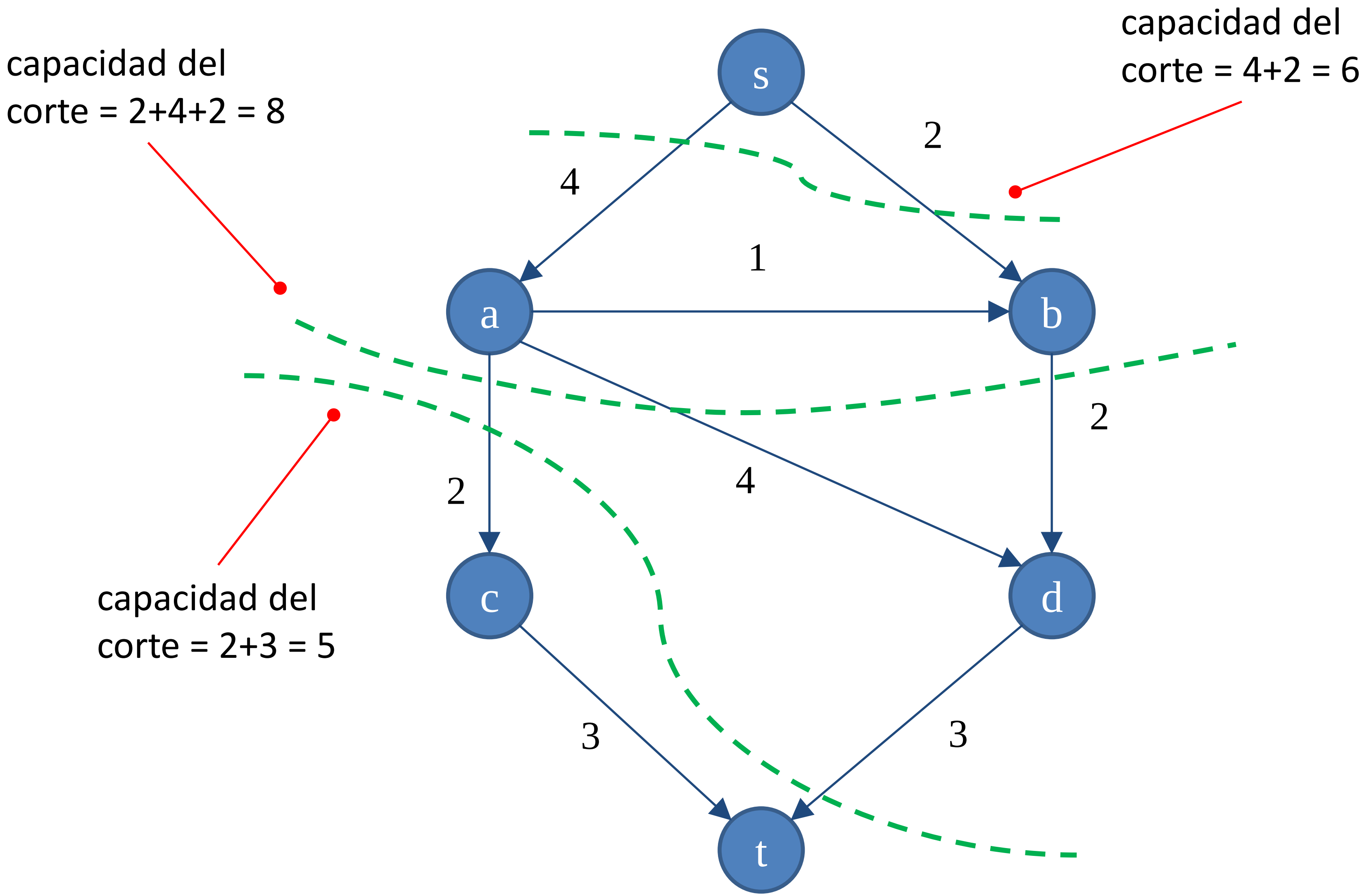

s
4 / 3
2 / 2
1 / 0
a
b
2 / 2
2 / 2
4 / 1
c
d
3 / 2
3 / 3
t

corte: particiona los vértices de modo que *s* y *t* queden en grupos distintos: {*s*,*a*,*b*,*d*} y {*c*,*t*}



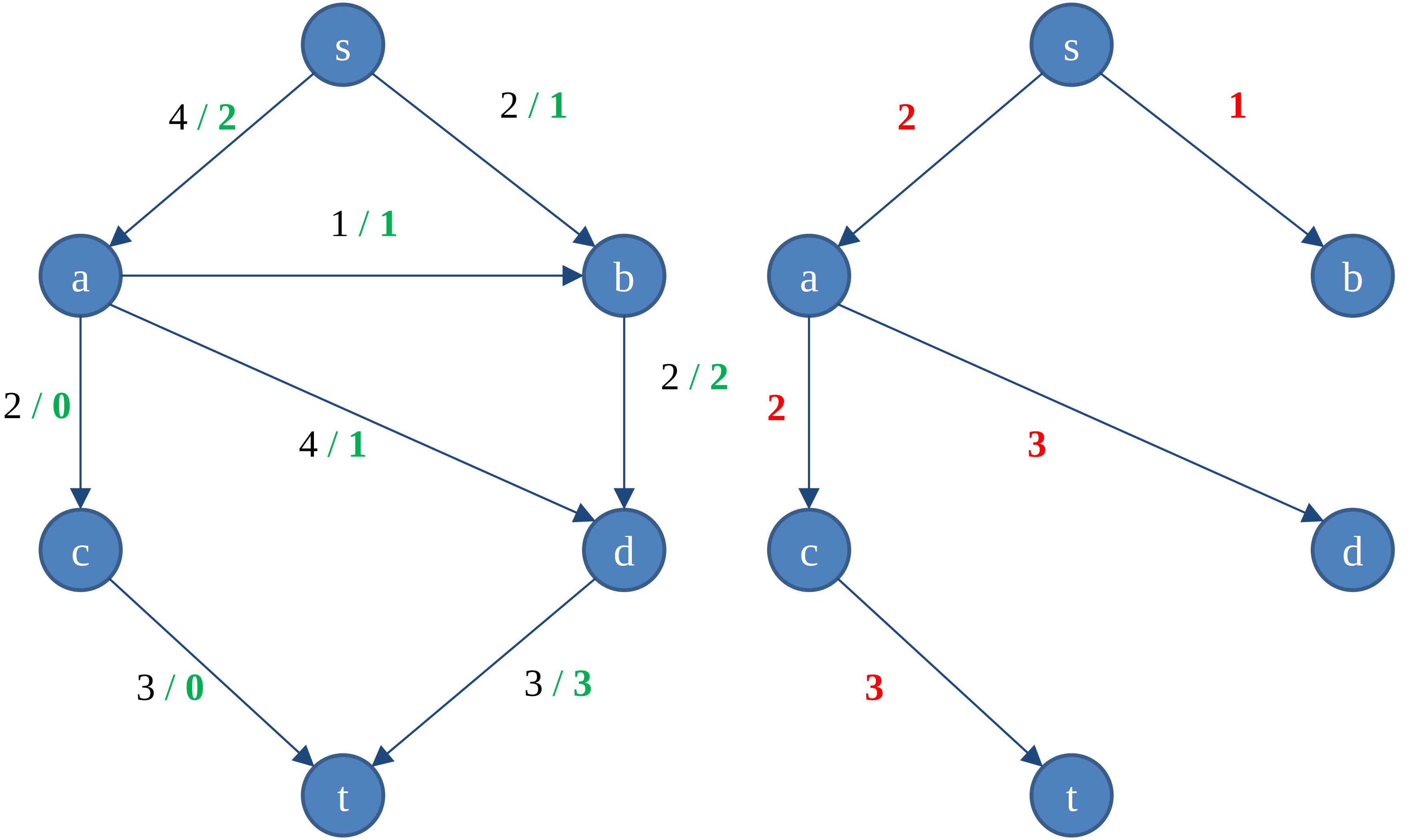
capacidad de la arista (*a*,*c*)

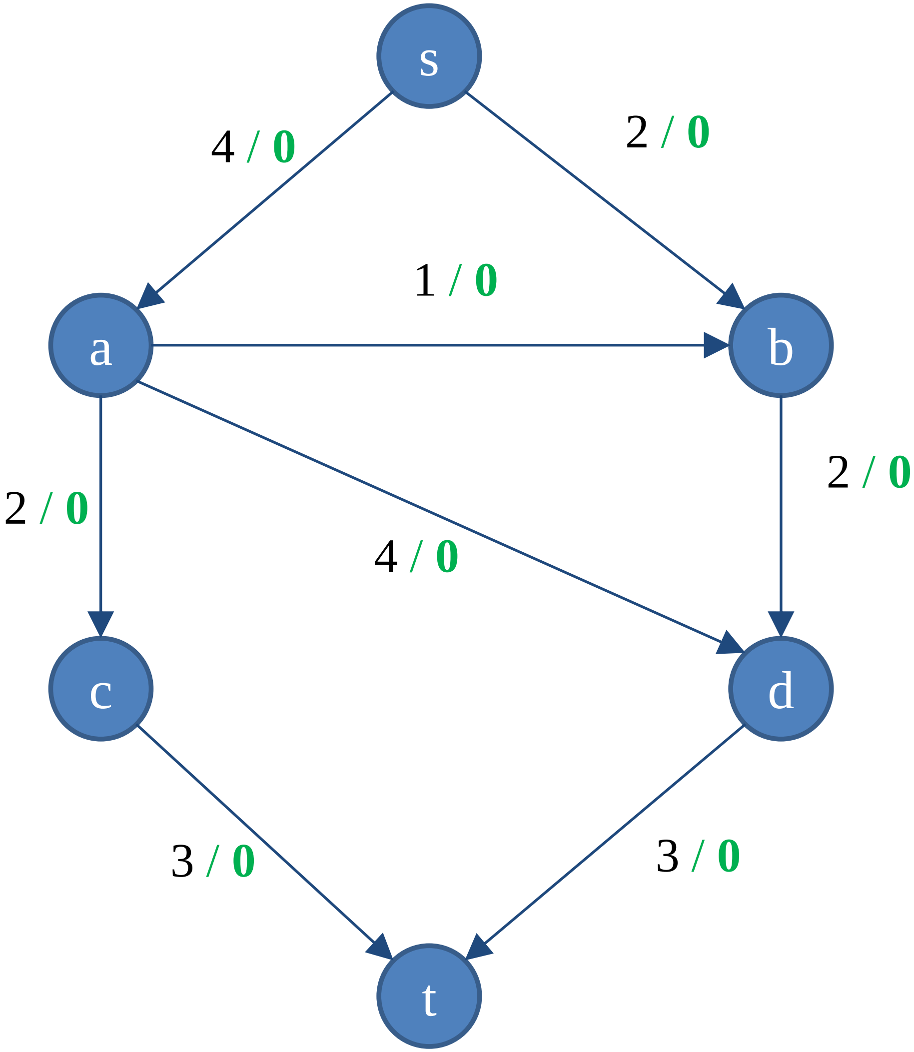
capacidad de la arista (*d*,*t*)

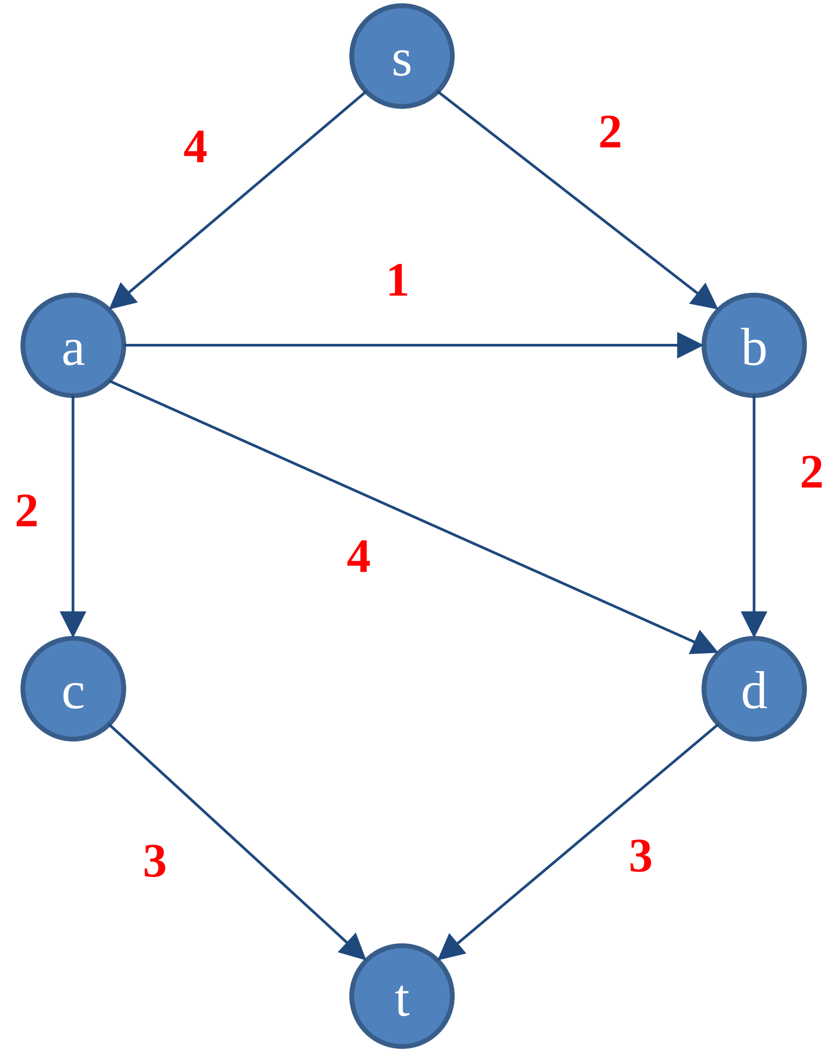

capacidad del
corte = 2+4+2 = 8
capacidad del
corte = 4+2 = 6
capacidad del
corte = 2+3 = 5
s
a
b
c
d
t
4
2
1
2
4
2
3
3

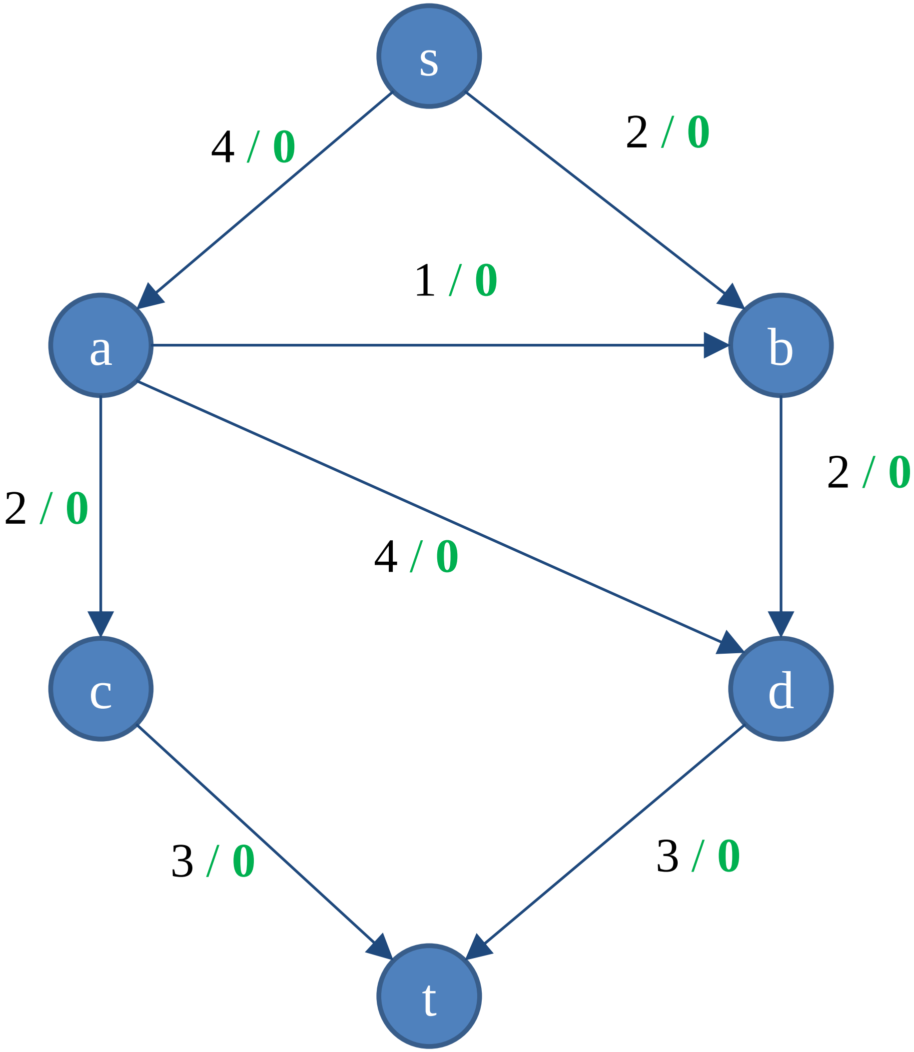
Dada una red G y un flujo f, definimos (preliminarmente) la **red residual G_f** :

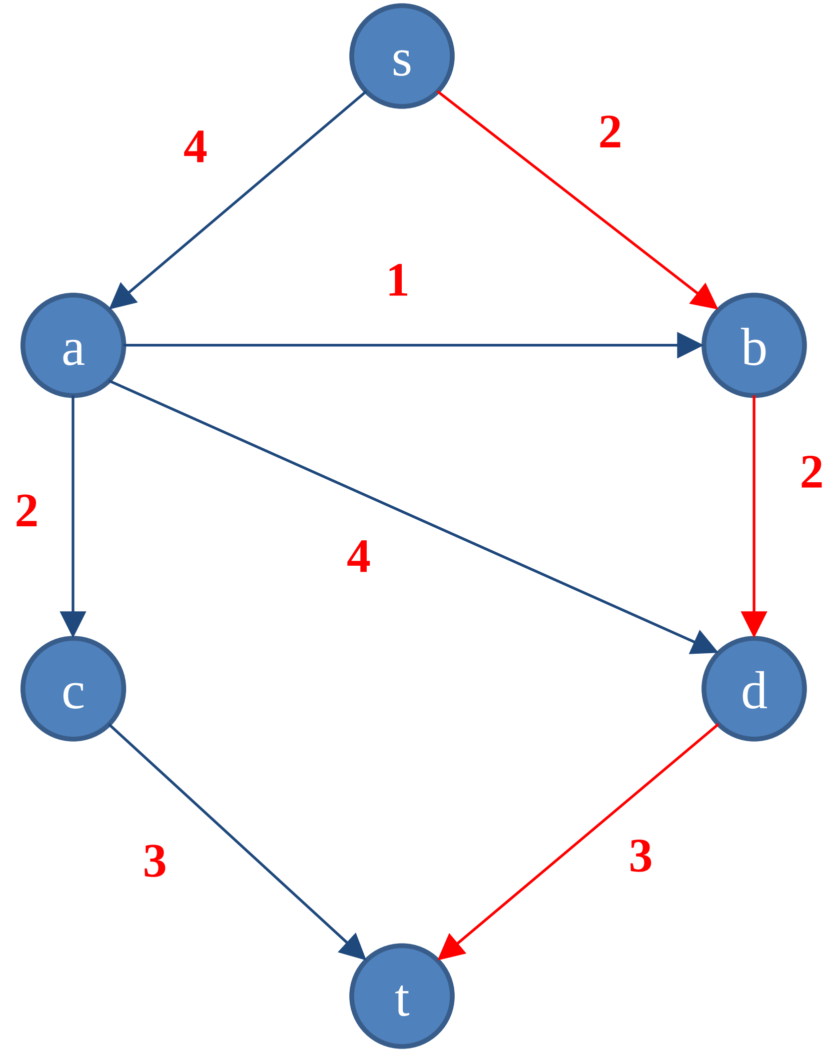
- el conjunto de nodos de G_f es el mismo que el de G
- para cada arista $e = (u, v)$ en que $f(e) < c_e$, incluimos la arista $e = (u, v)$ en G_f , con capacidad $c_e - f(e)$

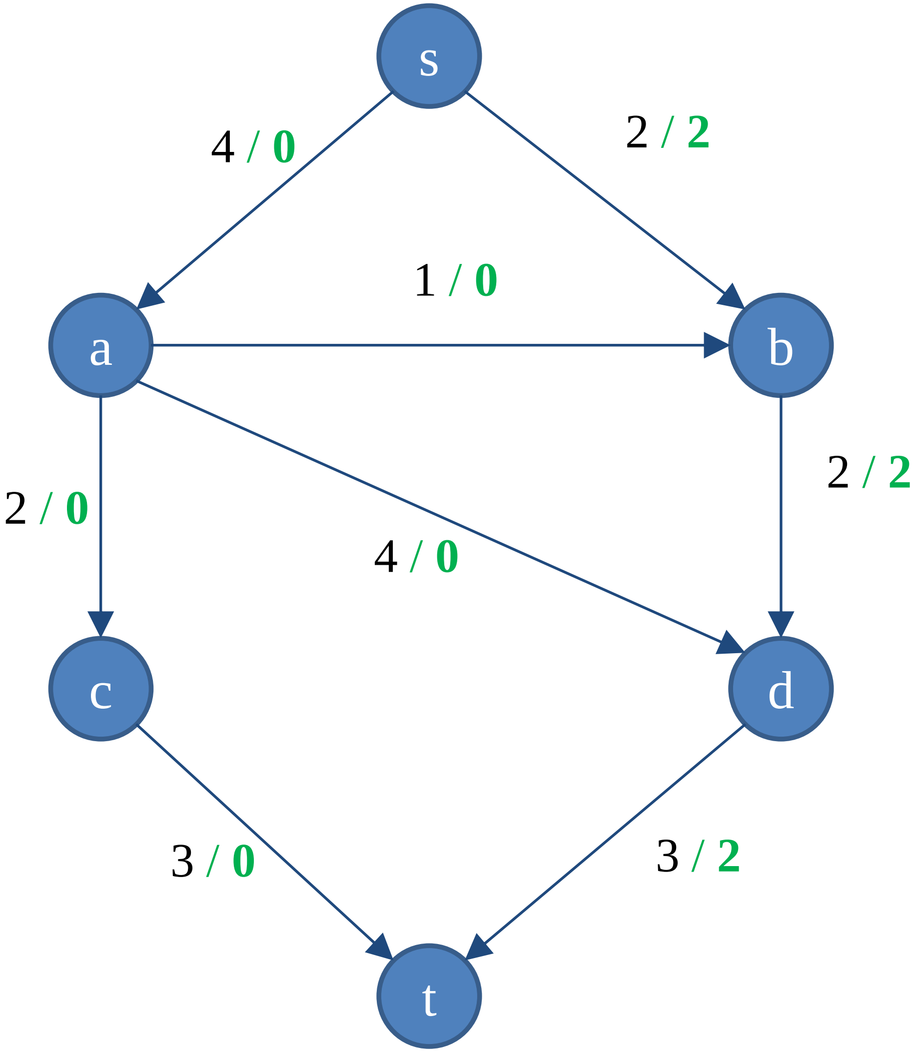

s
a
b
c
d
t
4 / 2
2 / 1
1 / 1
2 / 0
4 / 1
2 / 2
3 / 0
3 / 3
s
a
b
c
d
t
2
1
2
3
3

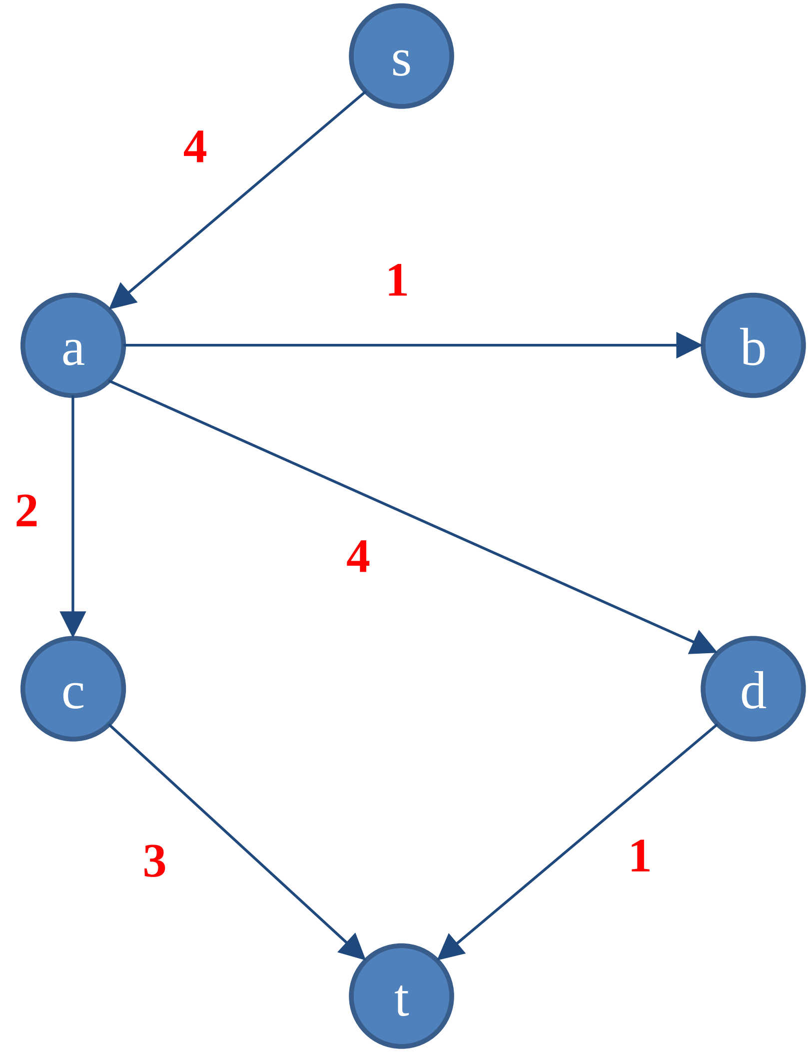

s
4 / 0
2 / 0
a
1 / 0
b
2 / 0
4 / 0
2 / 0
c
d
3 / 0
3 / 0
t

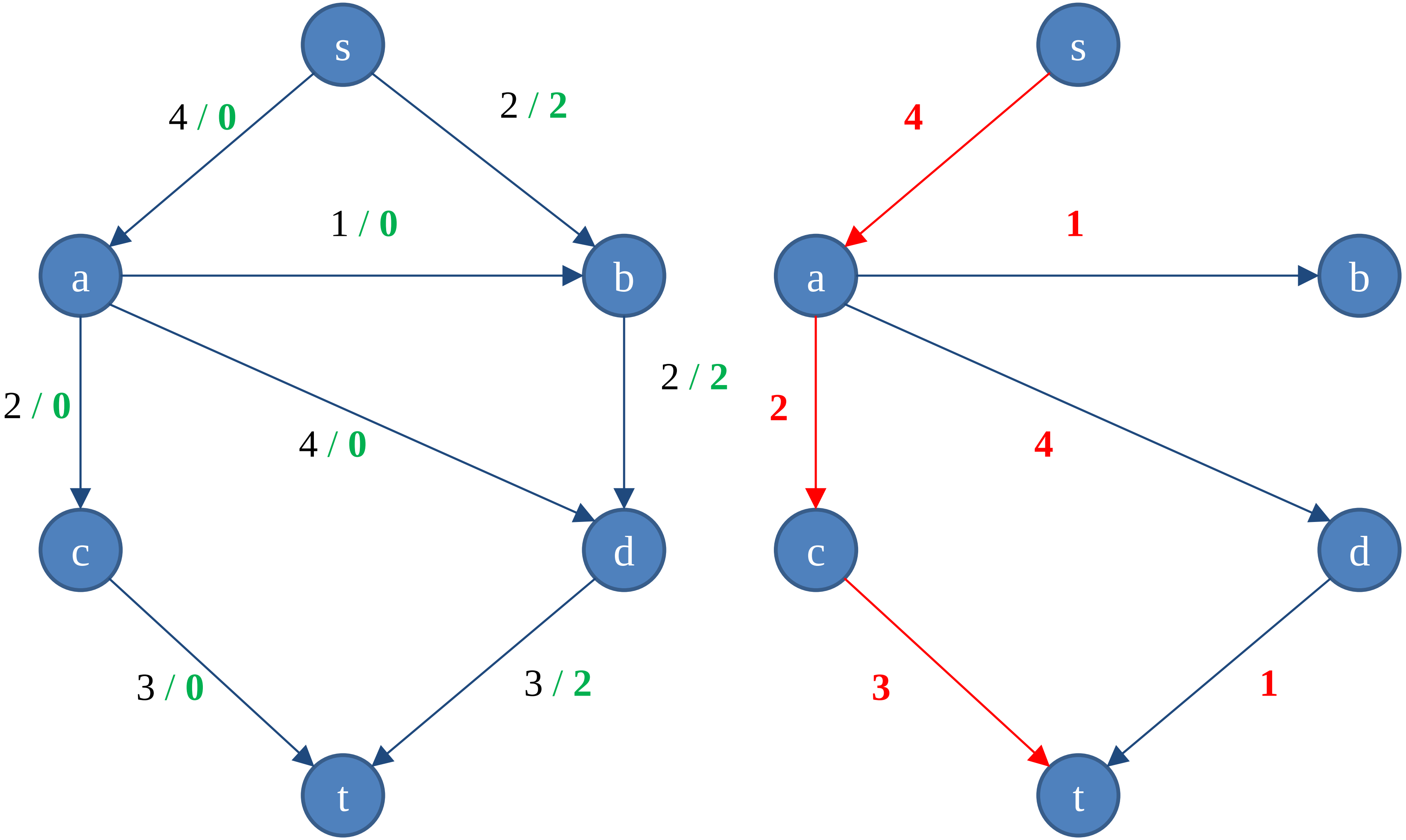

s
4
2
a
1
b
2
4
2
c
d
3
3
t

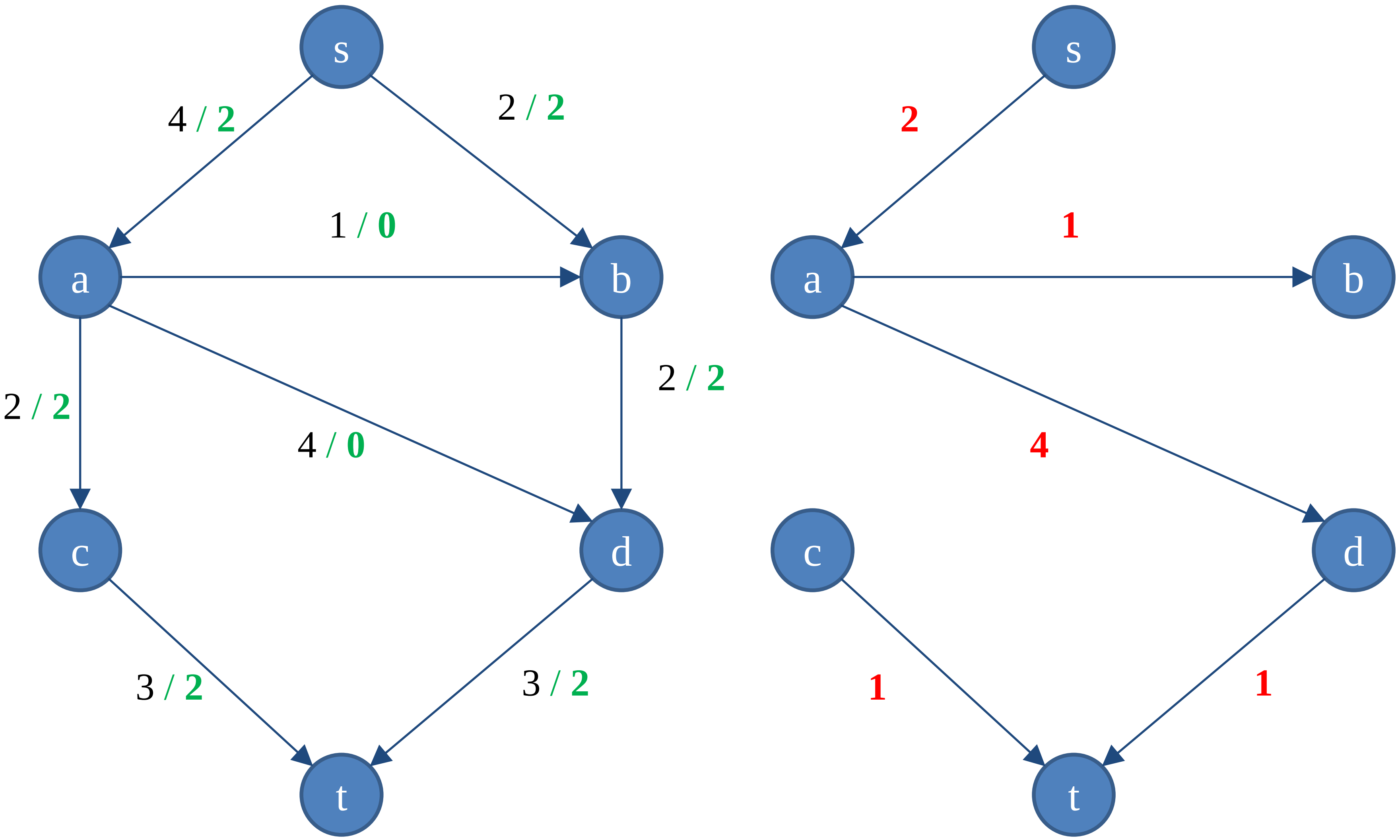

s
a
b
c
d
t
4 / 0
2 / 0
1 / 0
2 / 0
2 / 0
4 / 0
3 / 0
3 / 0

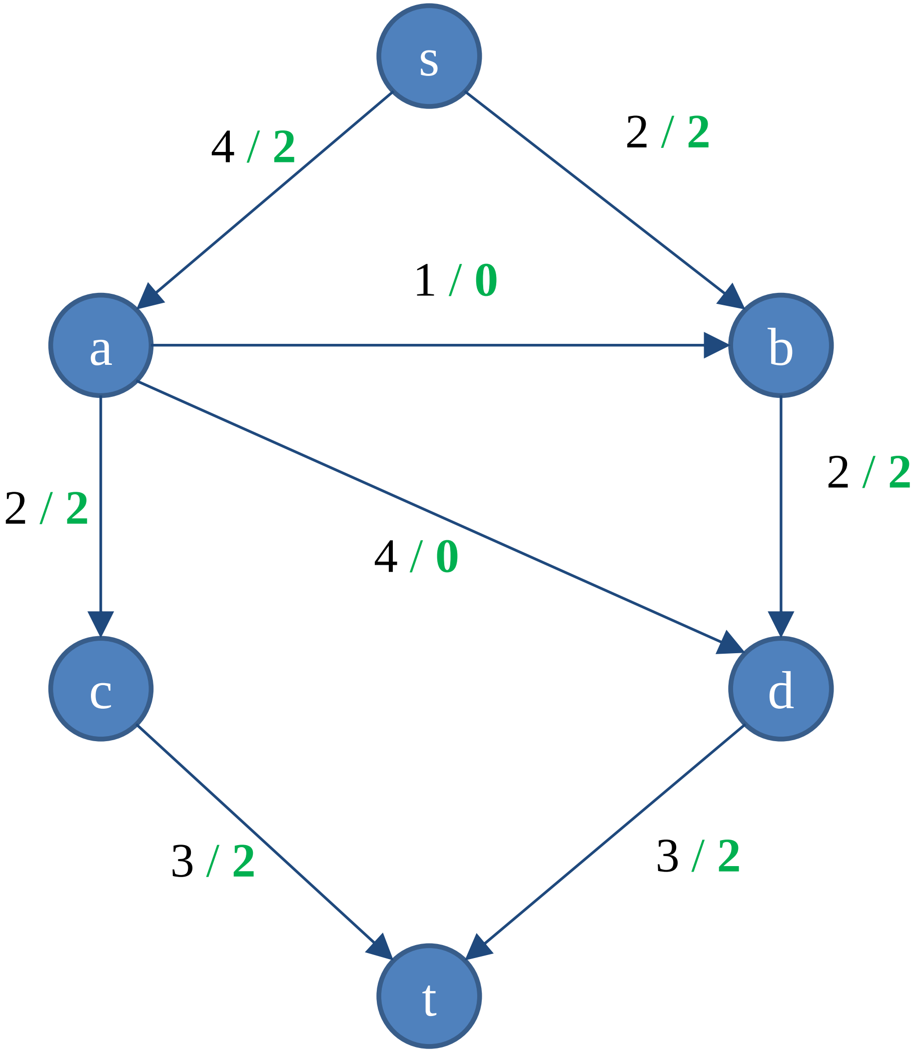

s
a
b
c
d
t
4
2
1
2
2
4
3
3

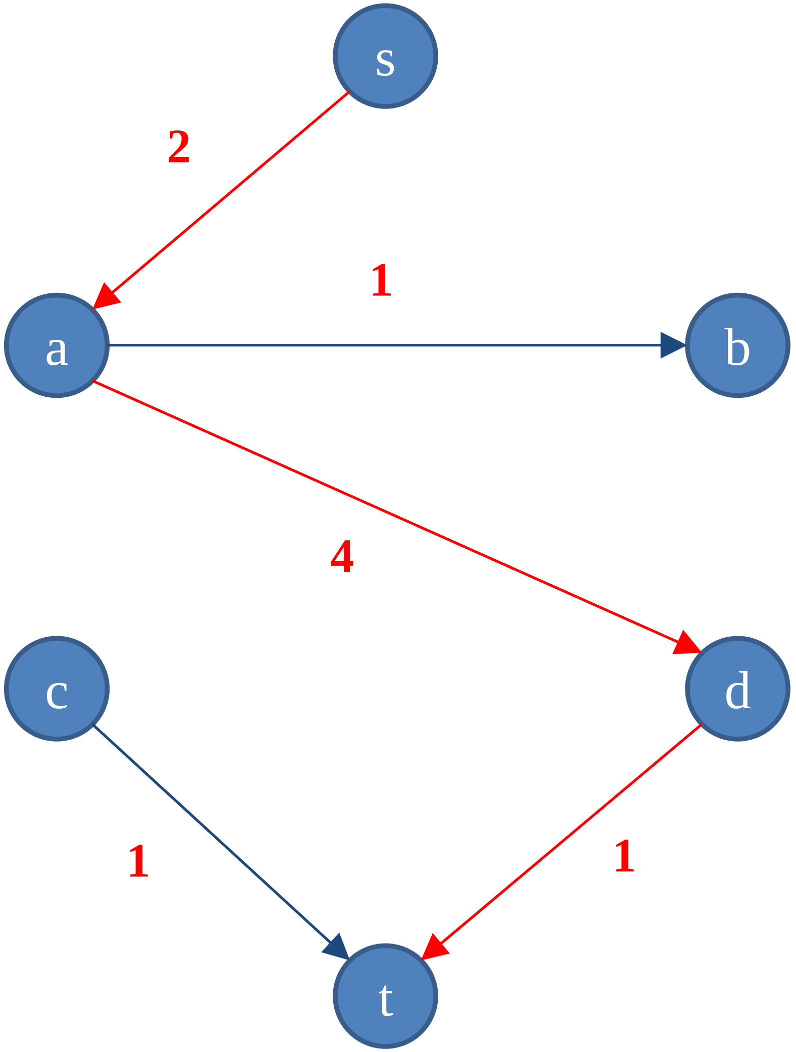

s
4 / 0
2 / 2
1 / 0
a
b
2 / 2
2 / 0
4 / 0
c
d
3 / 0
3 / 2
t

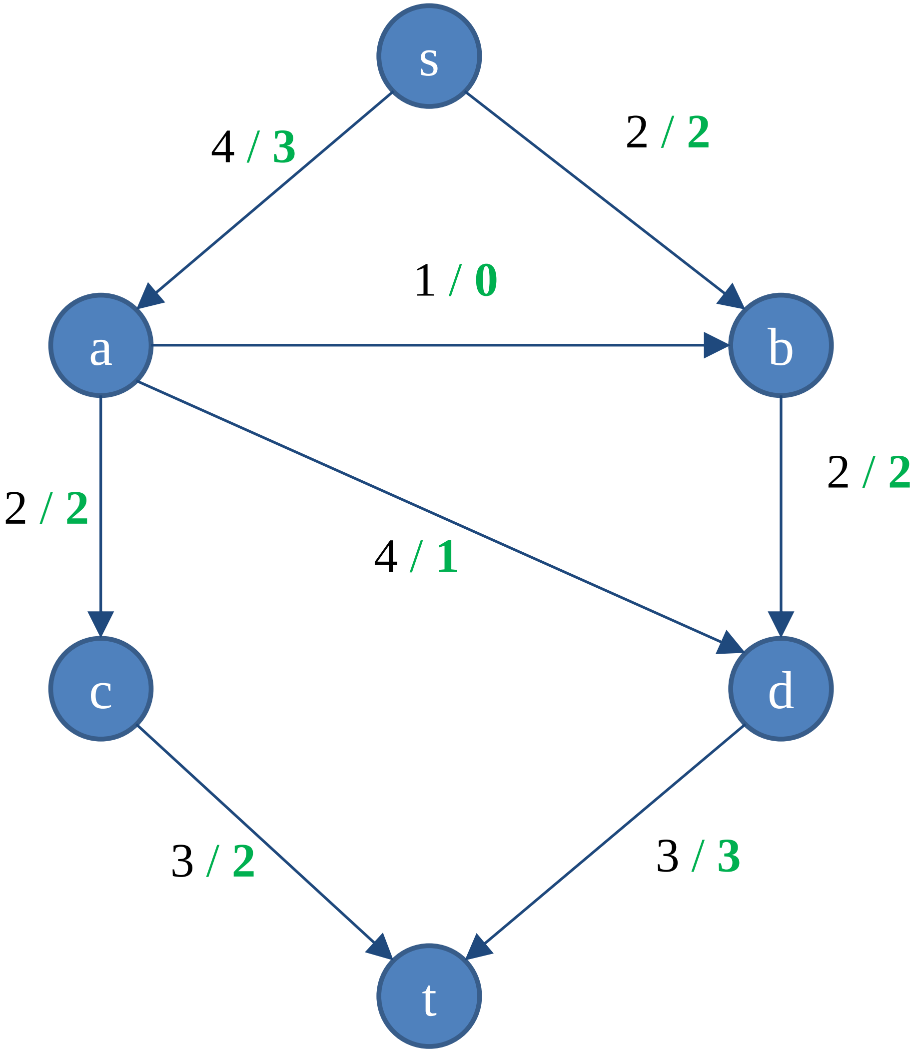

s
4
1
a
b
2
4
c
d
3
1
t

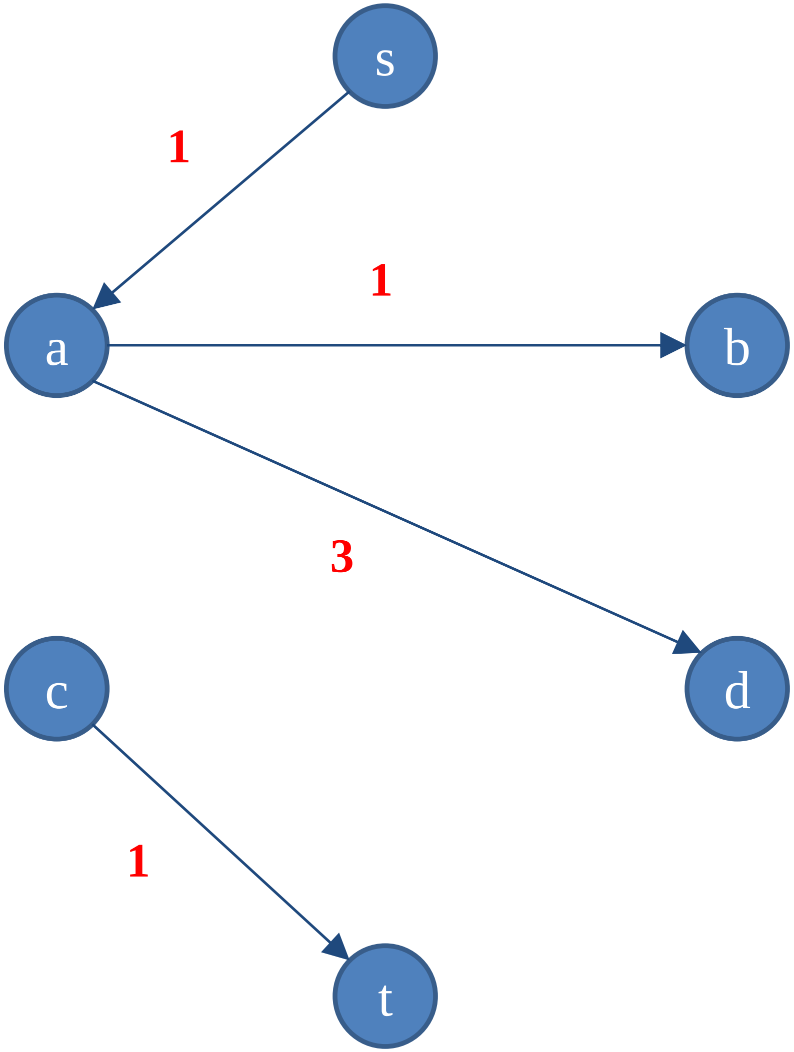

s
a
b
c
d
t
4 / 0
2 / 2
1 / 0
2 / 0
4 / 0
2 / 2
3 / 0
3 / 2
s
a
b
c
d
t
4
1
2
4
3
1


s
4 / 2
2 / 2
1 / 0
a
b
2 / 2
2 / 2
4 / 0
c
d
3 / 2
3 / 2
t
s
2
1
a
b
4
c
d
1
1
t


s
4 / 2
2 / 2
a
1 / 0
b
2 / 2
4 / 0
2 / 2
c
d
3 / 2
3 / 2
t

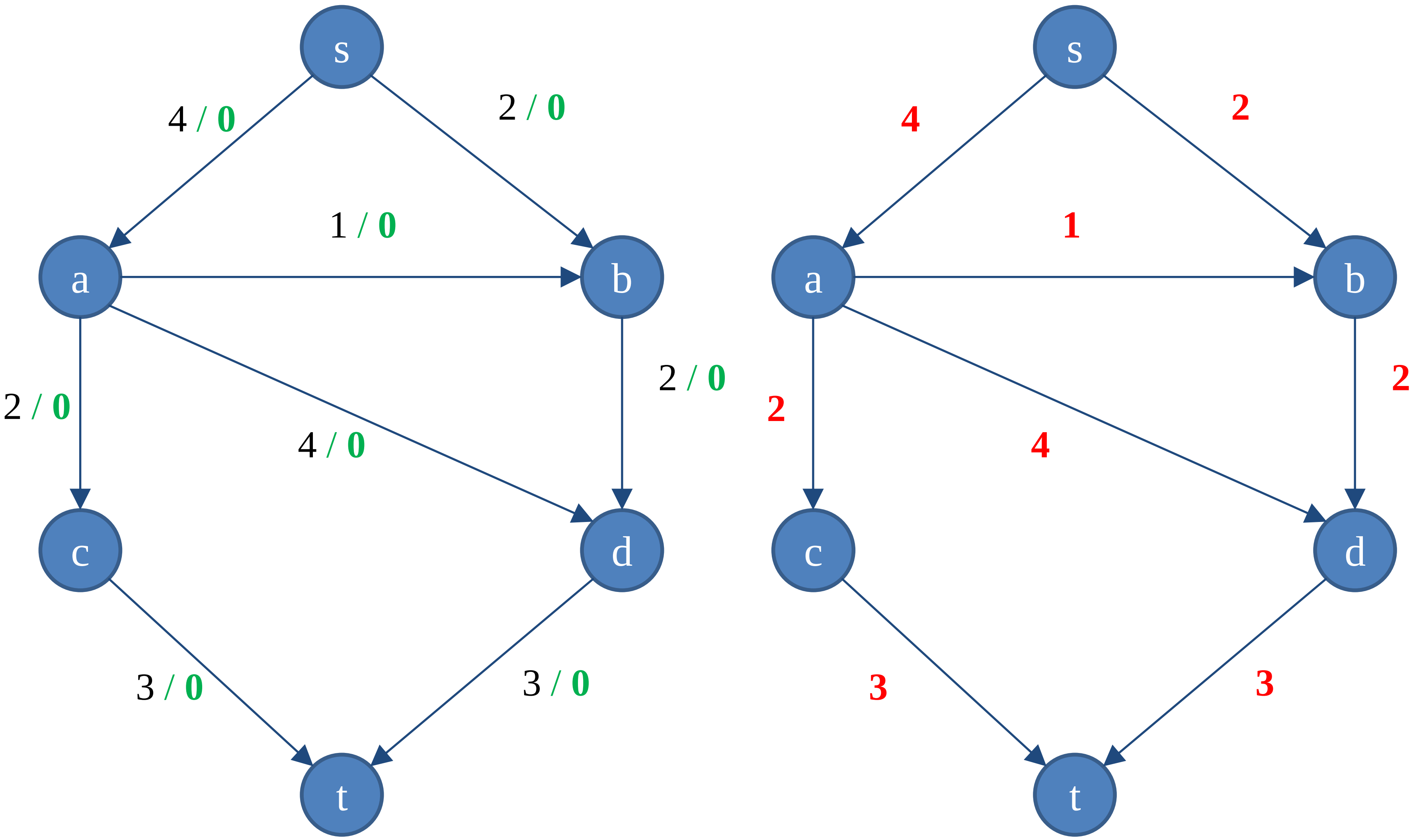

s
2
a
1
b
4
c
d
1
1
t

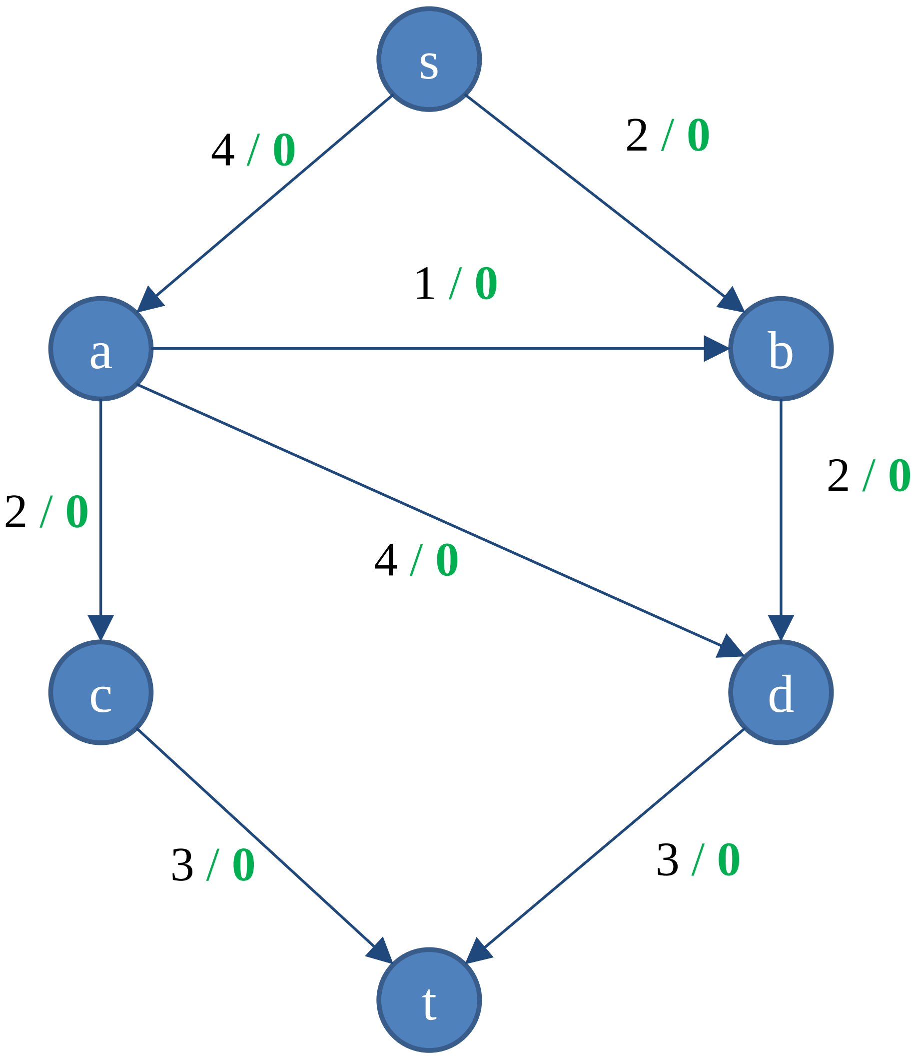

s
a
b
c
d
t
4 / 3
2 / 2
1 / 0
2 / 2
4 / 1
2 / 2
3 / 2
3 / 3

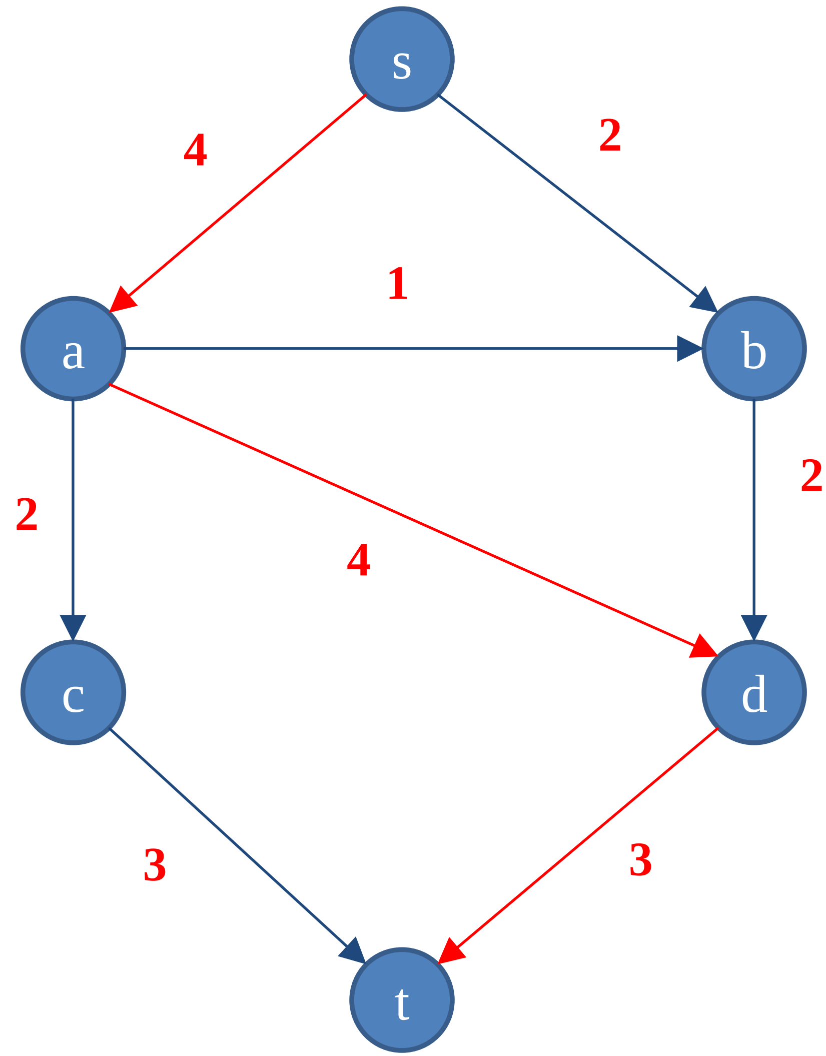

s
a
b
c
d
t
1
1
3
1

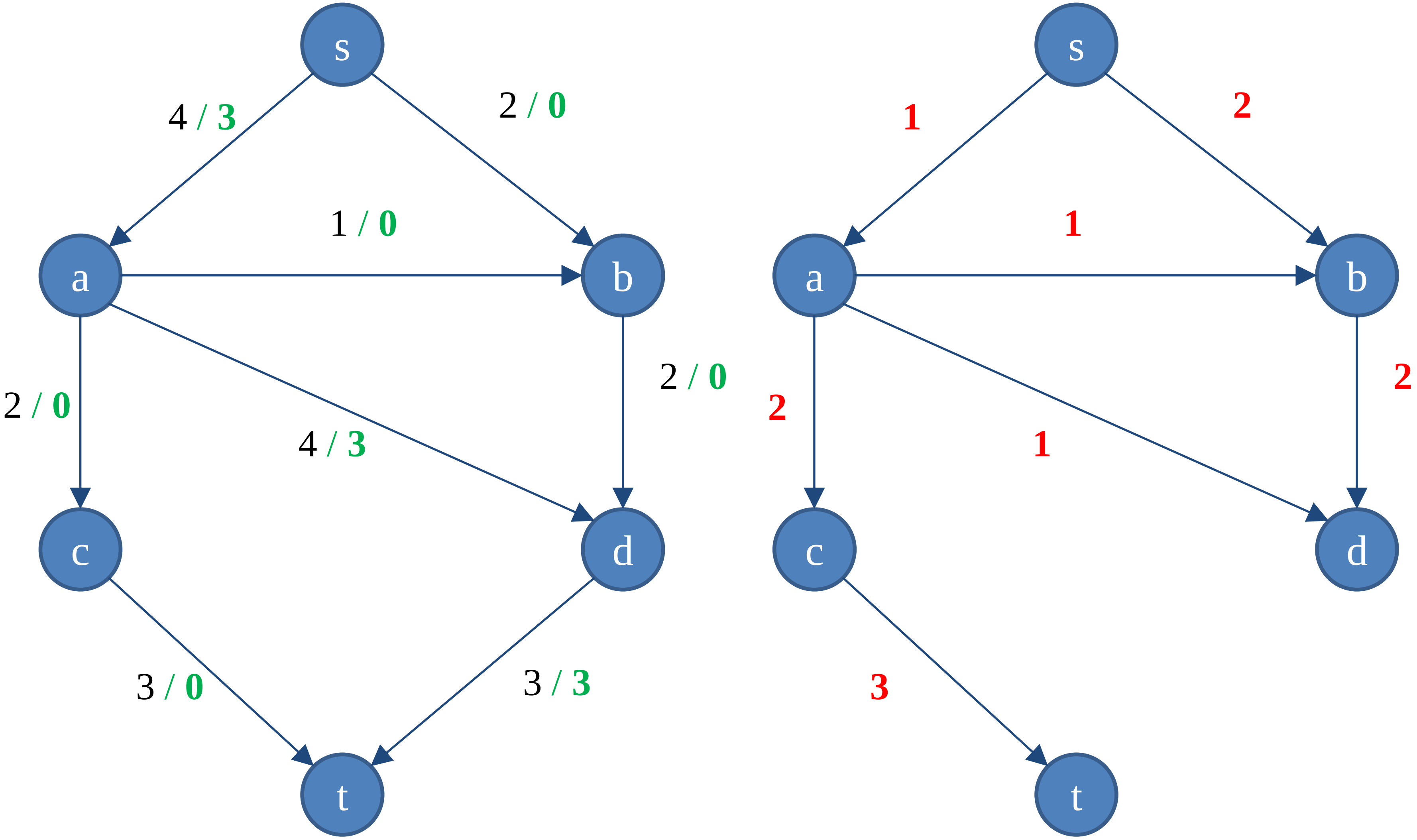
Lo que hemos hecho es aplicar una estrategia codiciosa, que nunca se arrepiente de las decisiones tomadas

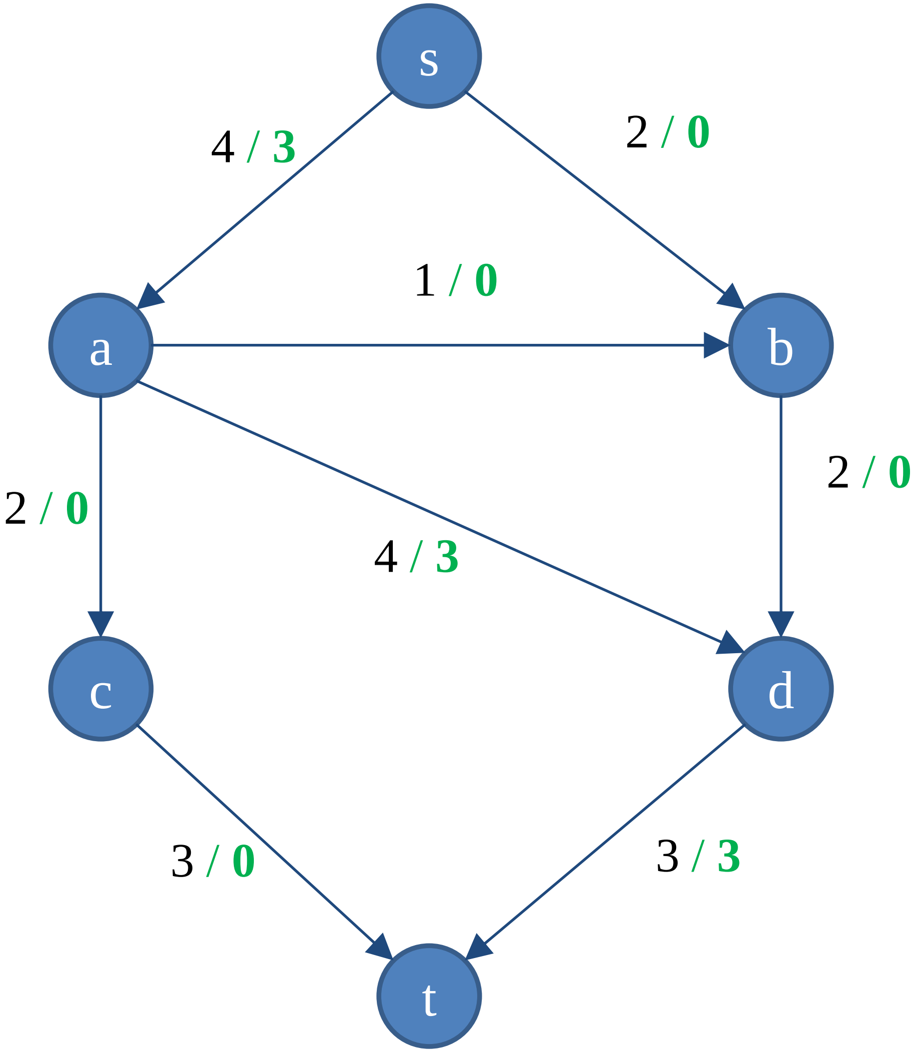
… pero

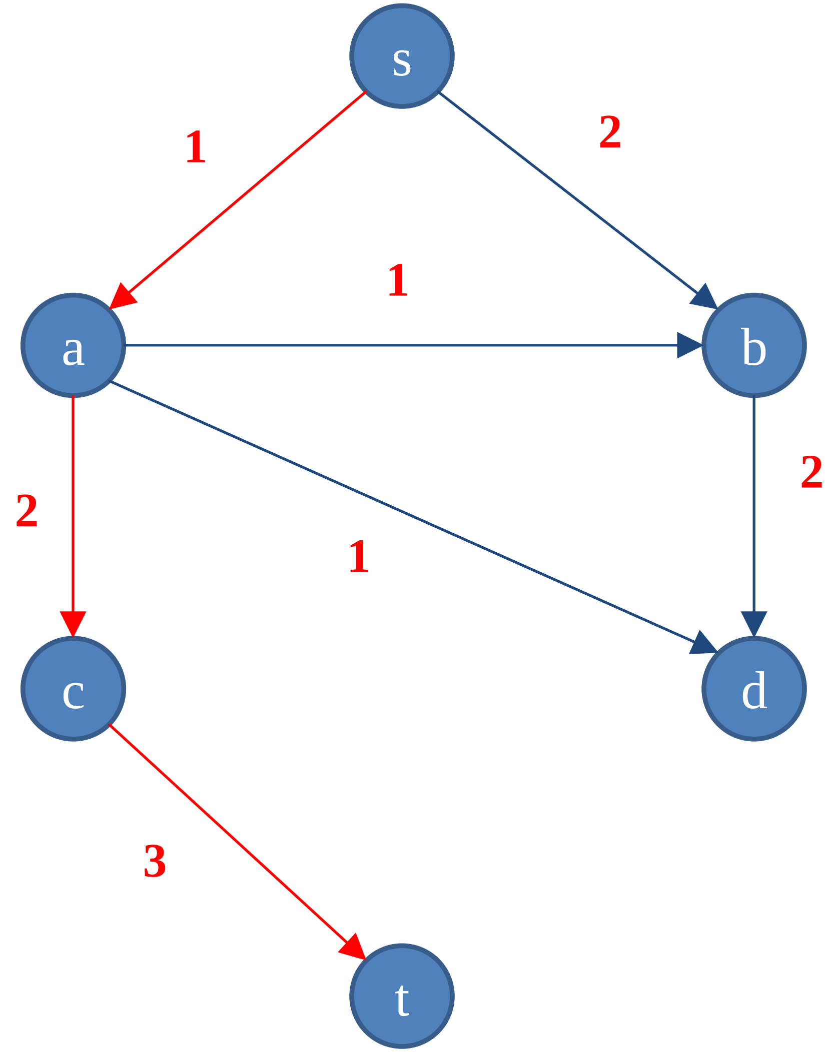

s
4 / 0
2 / 0
1 / 0
a
b
2 / 0
2 / 0
4 / 0
c
d
3 / 0
3 / 0
t
s
4
2
1
a
b
2
2
4
c
d
3
3
t

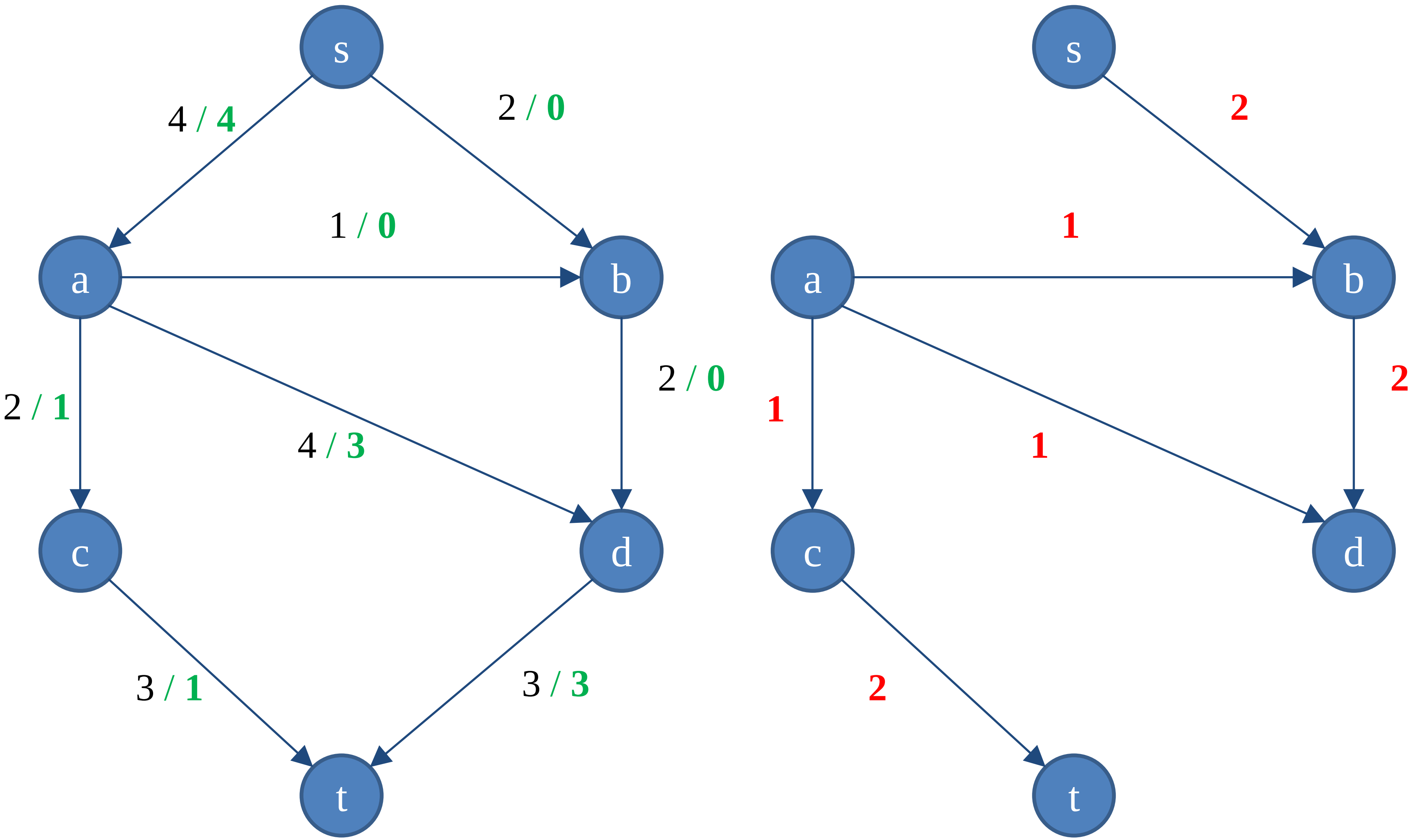

s
4 / 0
2 / 0
a
1 / 0
b
2 / 0
2 / 0
4 / 0
c
d
3 / 0
3 / 0
t


s
4
2
a
1
b
2
2
4
c
d
3
3
t


s
a
b
c
d
t
4 / 3
2 / 0
1 / 0
2 / 0
2 / 0
4 / 3
3 / 0
3 / 3
s
a
b
c
d
t
1
2
1
2
2
1
3


s
4 / 3
2 / 0
1 / 0
a
b
2 / 0
2 / 0
4 / 3
c
d
3 / 0
3 / 3
t

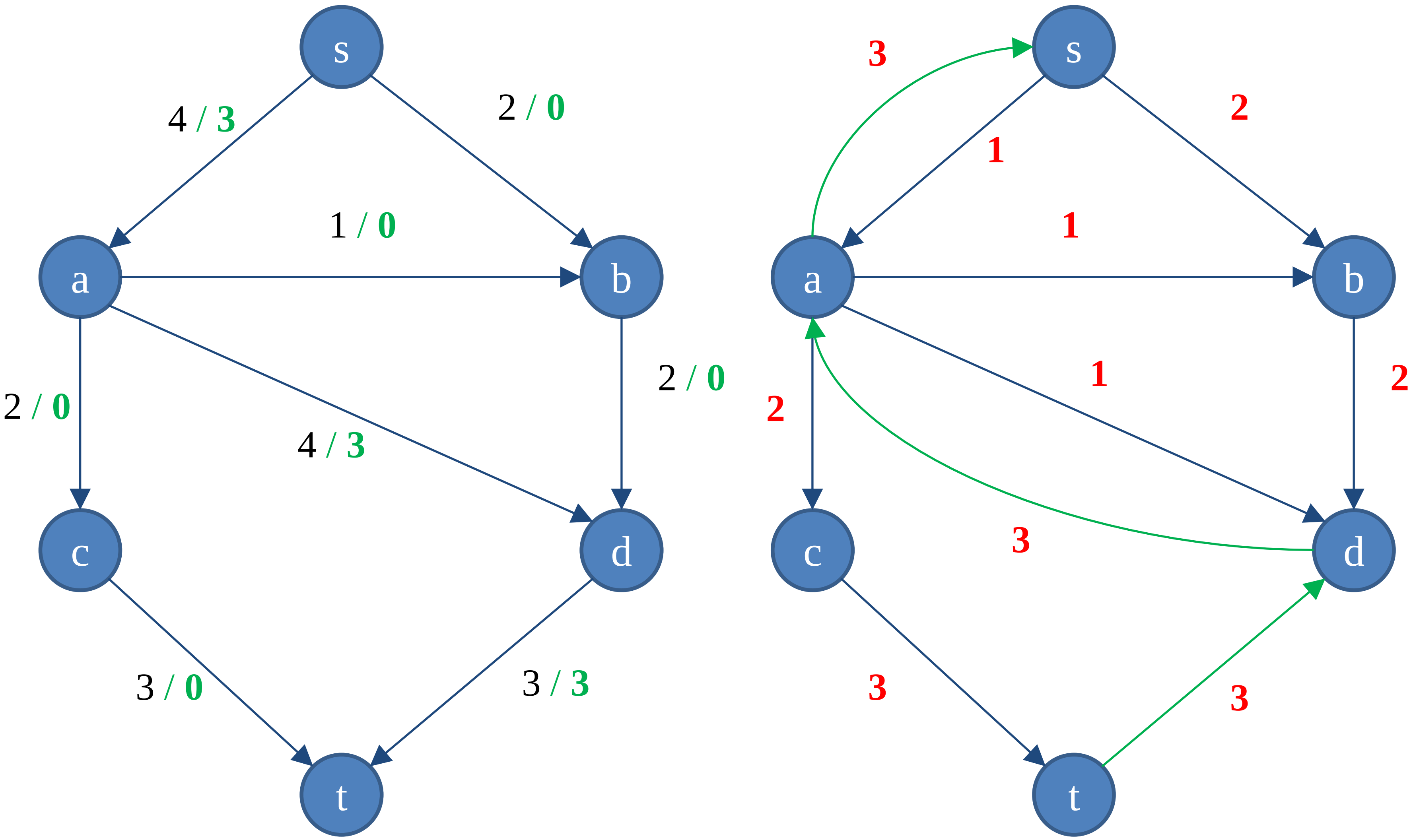

s
1
2
1
a
b
2
2
1
c
d
3
t


s
4 / 4
2 / 0
1 / 0
a
b
2 / 1
4 / 3
2 / 0
c
d
3 / 1
3 / 3
t
s
2
1
a
b
1
1
2
c
d
2
t

Dada una red G y un flujo f, definimos (ahora sí) la

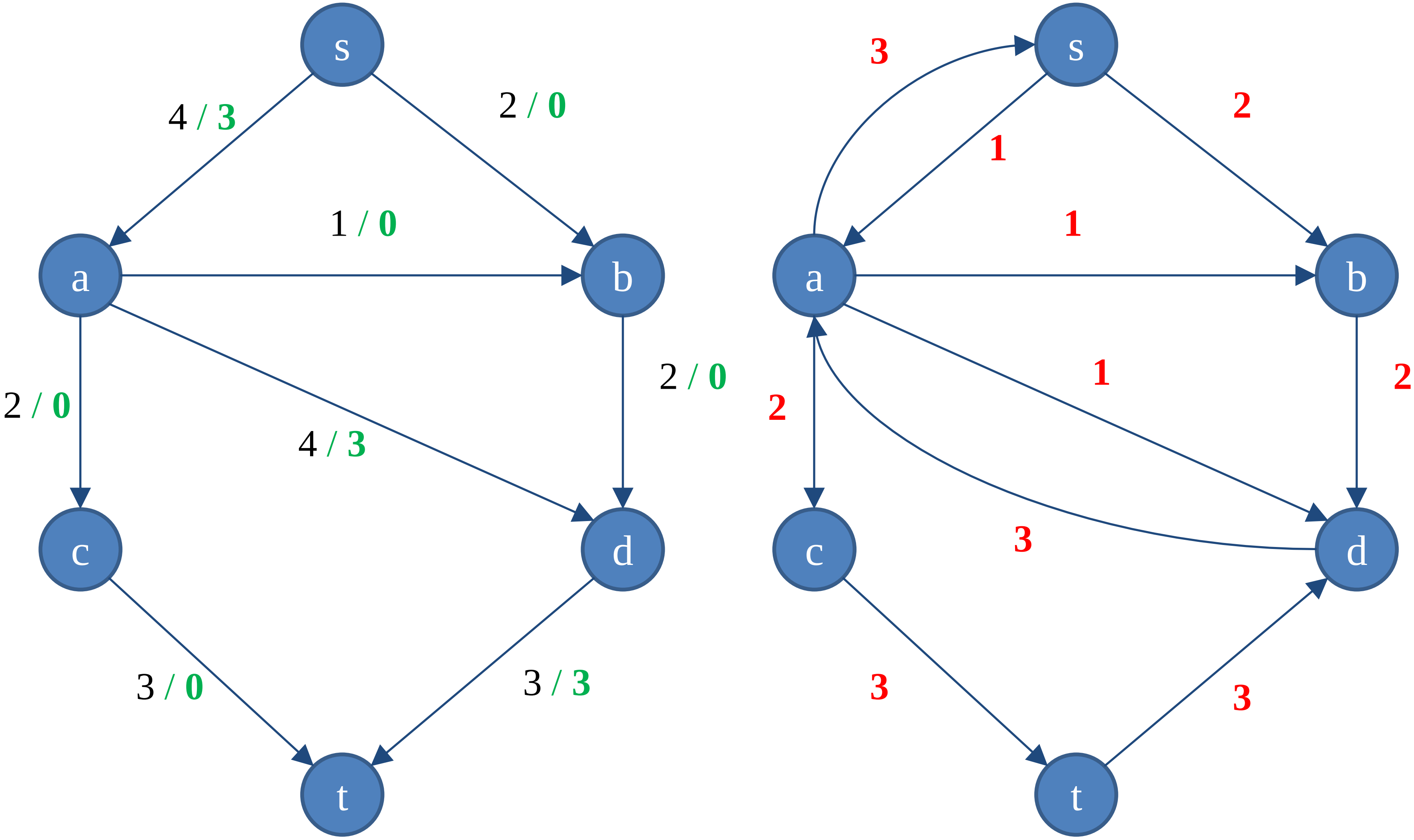
… red residual G_f :

- el conjunto de nodos de G_f es el mismo que el de G
- para cada arista $e = (u, v)$ en que $f(e) < c_e$, incluimos la arista $e = (u, v)$ en G_f , con capacidad $c_e - f(e)$: **arista *forward***
- para acada arista $e = (u, v)$ en que $f(e) > 0$, incluimos la arista $e' = (v, u)$ en G_f , con capacidad $f(e)$: **arista *backward***


s
a
b
c
d
t
4 / 3
2 / 0
1 / 0
2 / 0
4 / 3
2 / 0
3 / 0
3 / 3
3
2
1
1
1
2
2
3
3
3

Es decir, cada arista e en G puede dar origen a una o dos aristas en G_f :

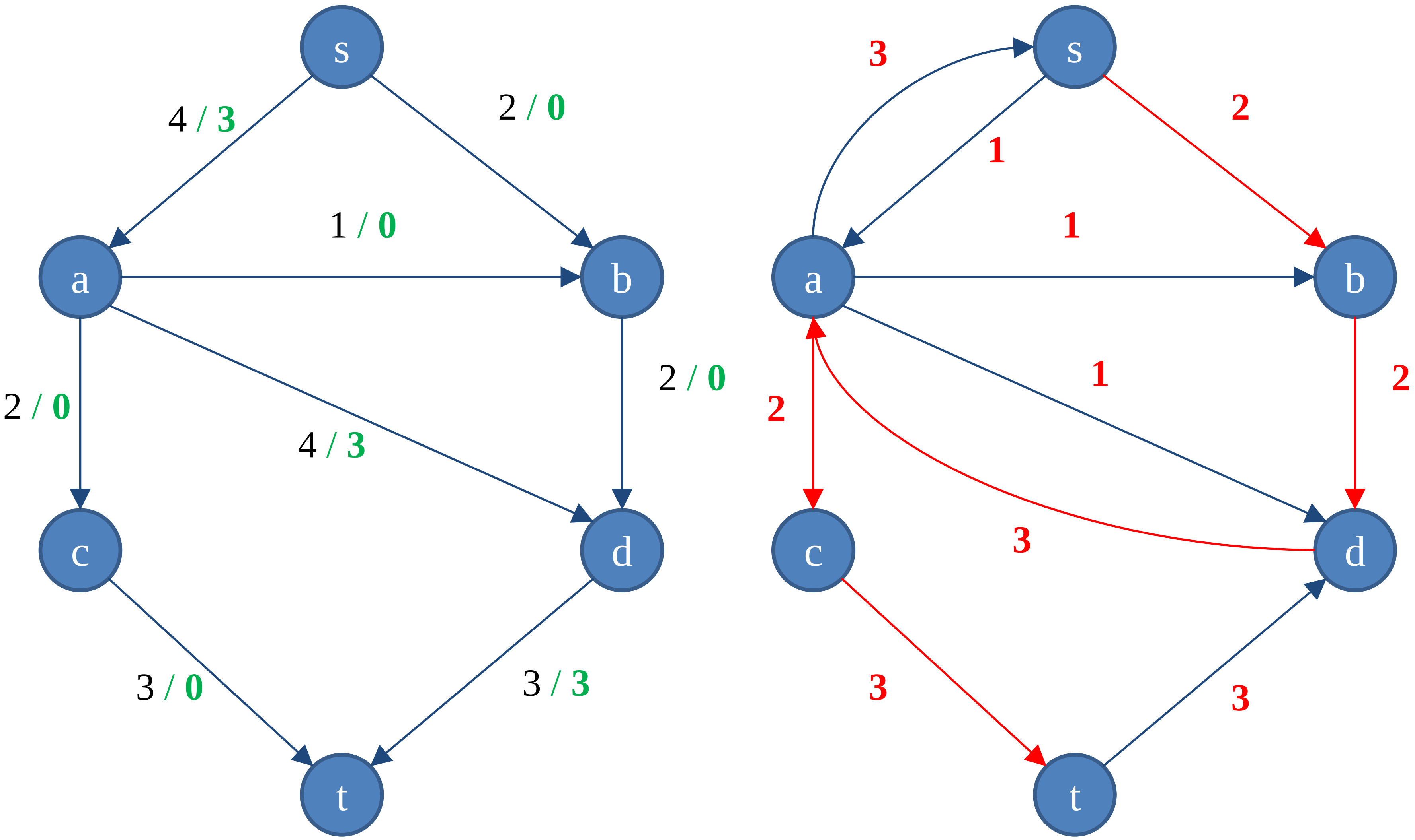
- si $0 < f(e) < c_e$, entonces se traduce tanto en una arista *forward* como en una arista *backward* incluidas en G_f
- G_f tiene a lo más el doble de aristas que G


s
a
b
c
d
t
4 / 3
2 / 0
1 / 0
2 / 0
4 / 3
2 / 0
3 / 0
3 / 3
s
a
b
c
d
t
3
1
2
1
1
2
2
3
3
3

En G_f podemos encontrar una ruta simple p de s a t —llamada una **ruta de aumento**,

… y sea $g_{p,f}$ la menor capacidad de cualquier arista en p (con respecto al flujo f),

… entonces podemos definir un nuevo flujo f' en G …


s
4 / 3
2 / 0
1 / 0
a
b
2 / 0
2 / 0
4 / 3
c
d
3 / 0
3 / 3
t
3
s
1
2
1
a
b
1
2
2
3
c
d
3
3
t

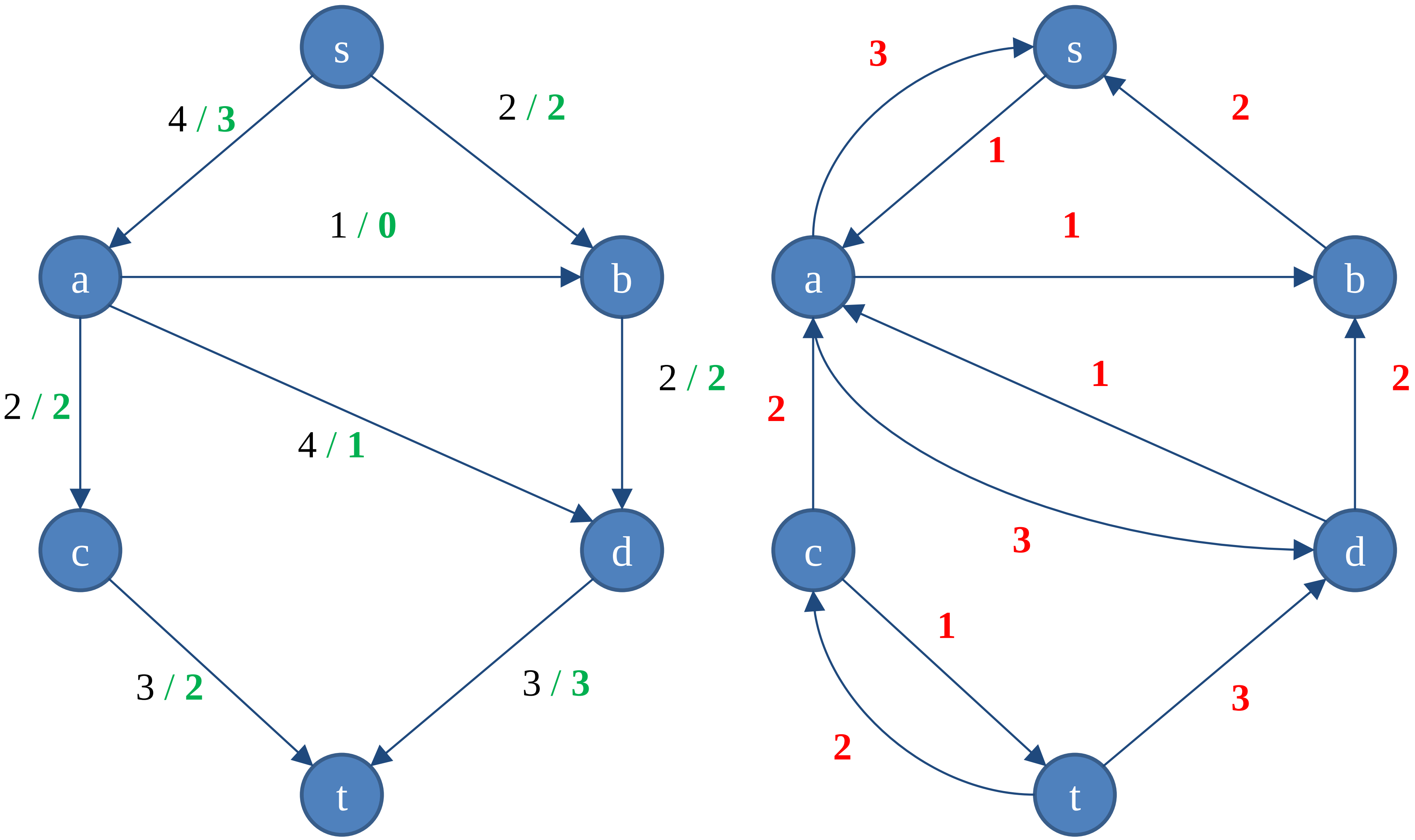
… entonces podemos definir un nuevo flujo f' en G :

para cada arista (u, v) en p :

si $e = (u, v)$ es una arista *forward*:
aumentar $f(e)$ en la red G en la cantidad $g_{p,f}$

si (u, v) es una arista *backward* y sea $e = (v, u)$:
disminuir $f(e)$ en la red G en la cantidad $g_{p,f}$

Se puede verificar que f' es un flujo válido en G —satisface las dos propiedades que aparecen en la diap. # 4


s
a
b
c
d
t
4 / 3
2 / 2
1 / 0
2 / 2
4 / 1
2 / 2
3 / 2
3 / 3
3
2
1
1
1
2
2
3
1
2
3

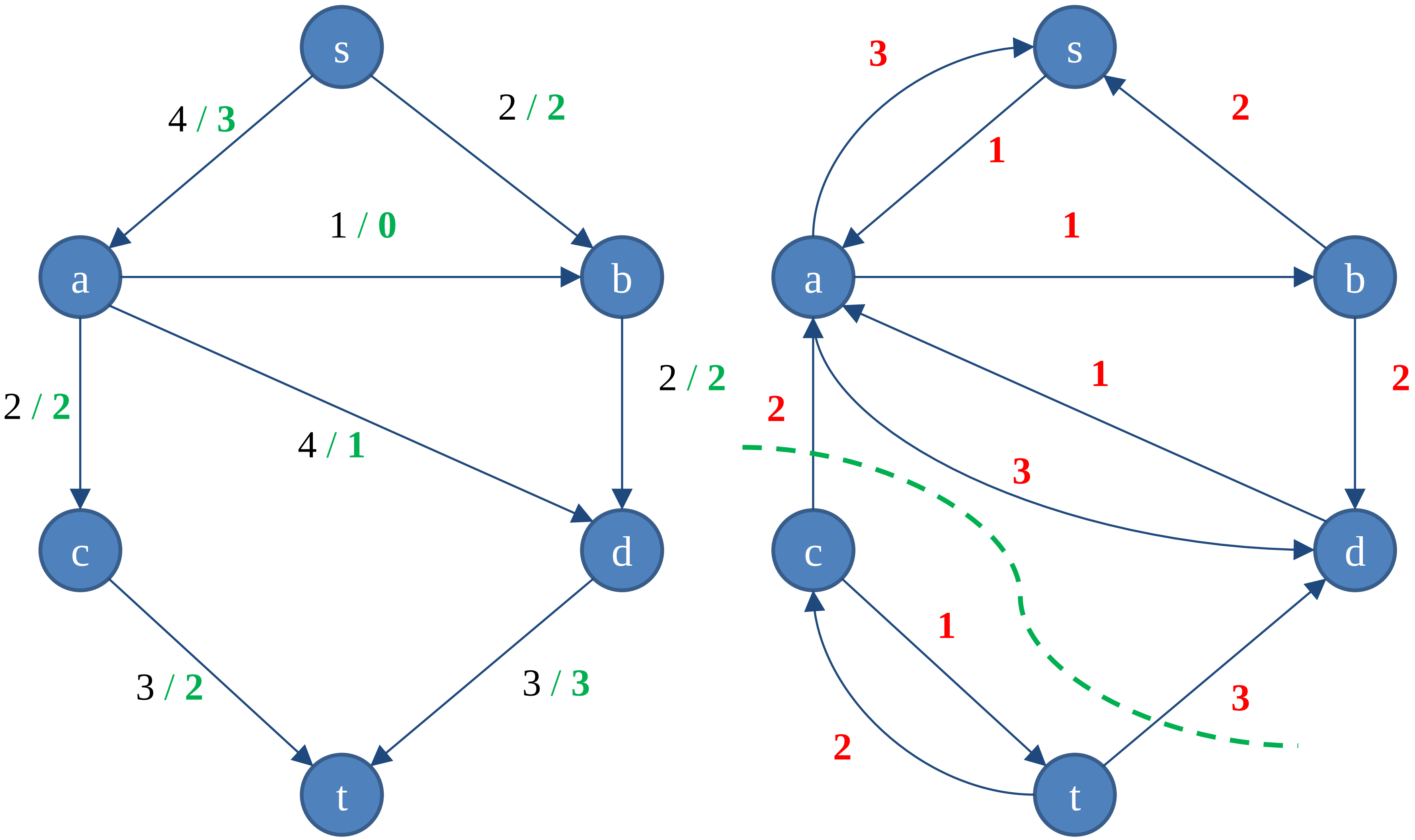
La aplicación repetida de los pasos de la diap. anterior constituye el **algoritmo de Ford-Fulkerson** para encontrar flujos máximos

… la repetición se detiene cuando no hay rutas simples de s a t en G_f

Se puede demostrar que si f es un flujo máximo en G,

… entonces la red residual G_f no tiene rutas de aumento de s a t

… y vice versa


s
a
b
c
d
t
4 / 3
2 / 2
1 / 0
2 / 2
4 / 1
2 / 2
3 / 2
3 / 3
3
2
1
1
1
2
2
3
1
3
2

El algoritmo de Ford-Fulkerson [1956]

max-flow:

inicialmente $f(e) = 0$ para toda e en G

`while` hay una ruta s-t en la red residual G_f :

sea P una ruta simple s-t en G_f

$f' =$ `augment`(f, P)

$f \leftarrow f'$

$G_f \leftarrow G_{f'}$

`return` f

`augment`(f, P):

—*diapositivas 30 y 32*

El algoritmo termina

En cualquier etapa intermedia del algoritmo, los valores de flujo y las capacidades residuales en G_f son enteros

Por otra parte, $v(f') > v(f)$

Si todas las aristas pudieran saturarse completamente con flujo, el valor del flujo sería la suma de las capacidades de estas aristas

… llamemos C a esta suma

… $v(f) \le C$ —sí, C puede ser una tremenda sobreestimación del flujo máximo

(el algoritmo podría no terminar solo si las capacidades de las aristas son números irracionales)

Como el flujo empieza en 0, aumenta en cada iteración a lo menos en 1, y no puede superar C

… el ciclo **`while`** se va a ejecutar a lo más C veces

… y el algoritmo es $O(|E|C)$

El teorema del **flujo máximo corte mínimo** (*max-flow min-cut*) asegura que las siguientes tres condiciones son equivalentes:

1. f es un flujo máximo en G

2. La red residual G_f no tiene rutas de aumento

3. $|f|$ = capacidad del corte (S, T) para algún corte (S, T) de G

El flujo *f* al final del algoritmo es máximo

Partimos de que la red residual G_f no tiene rutas direccionales de s a t (de aumento)

Sea S el conjunto de los nodos alcanzables desde s en G_f

… y $T = V - S$ —la partición (S, T) es un corte

Consideremos los nodos $x \in S$ y $y \in T$:

- si $(x, y) \in E$, entonces $f(x, y) = c(x, y)$
- si $(y, x) \in E$, entonces $f(y, x) = 0$

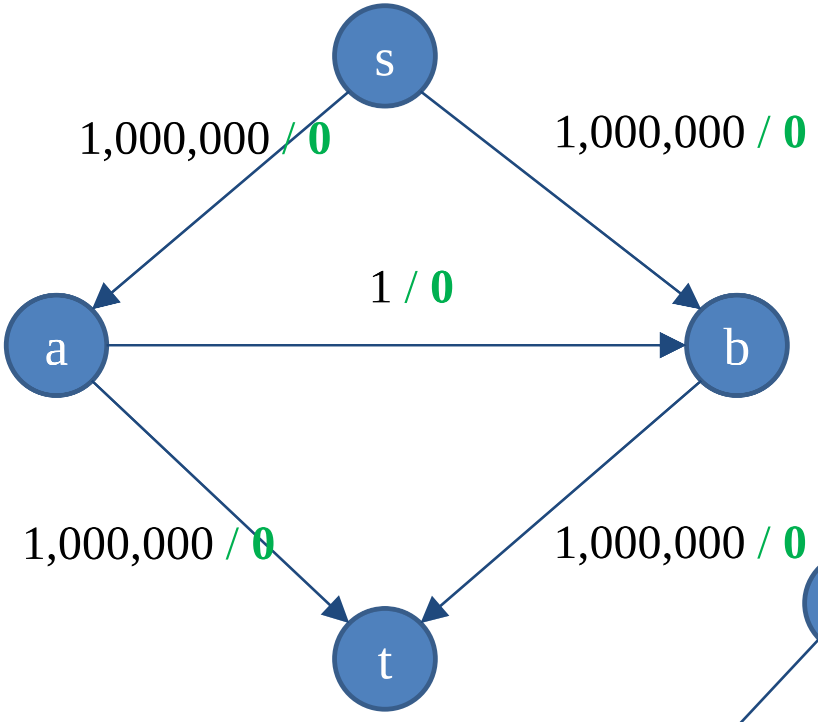
  … de lo contrario $(x, y) \in G_f$

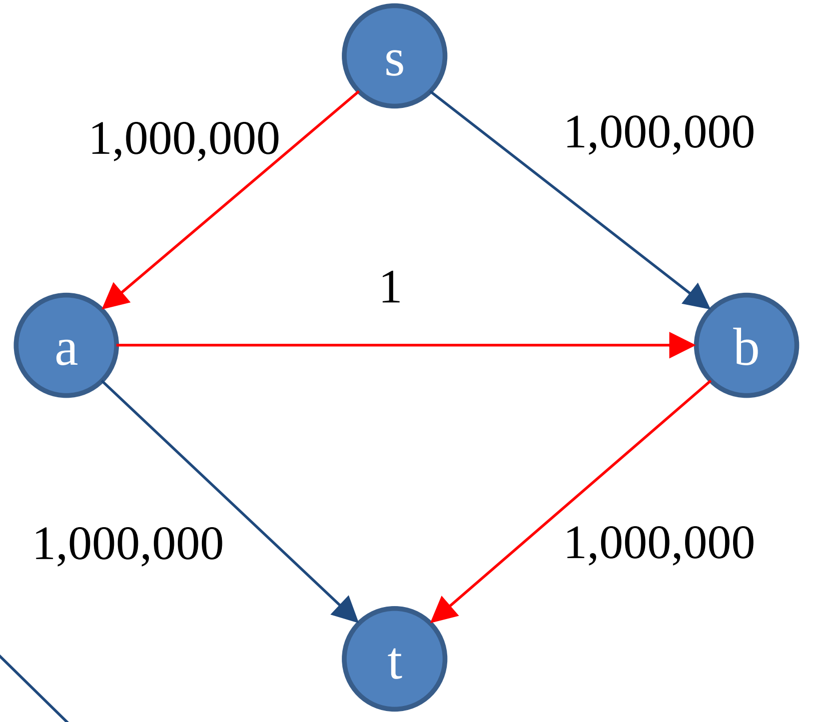
Luego, el flujo de S a T es igual a la capacidad del corte (S, T)

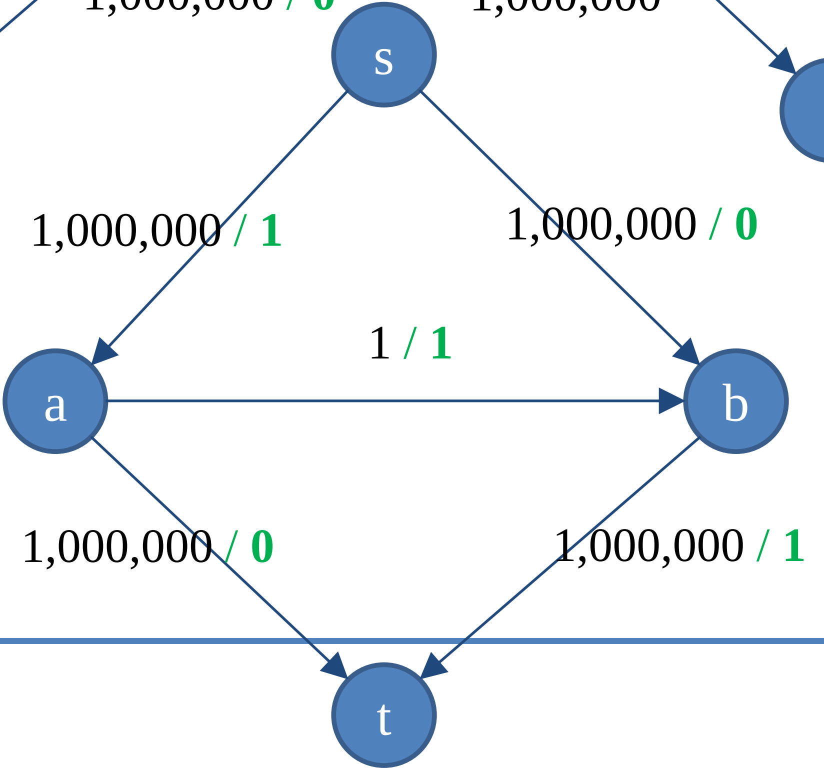
Como $v(f) \leq$ capacidad de cualquier corte, entonces f debe ser máximo

De las diapositivas # 38 y 39

… el algoritmo de Ford-Fulkerson es O(E $|f^*|$)


s
a
b
t
1,000,000 / 0
1,000,000 / 0
1 / 0
1,000,000 / 0
1,000,000 / 0


s
a
b
t
1,000,000
1,000,000
1
1,000,000
1,000,000


s
a
b
t
1,000,000 / 1
1,000,000 / 0
1 / 1
1,000,000 / 0
1,000,000 / 1

Podemos mejorar la cota O(E $|f^*|$) si buscamos rutas s-t en G_f usando BFS

→ la ruta de aumento es una ruta *más corta* de s a t en G_f

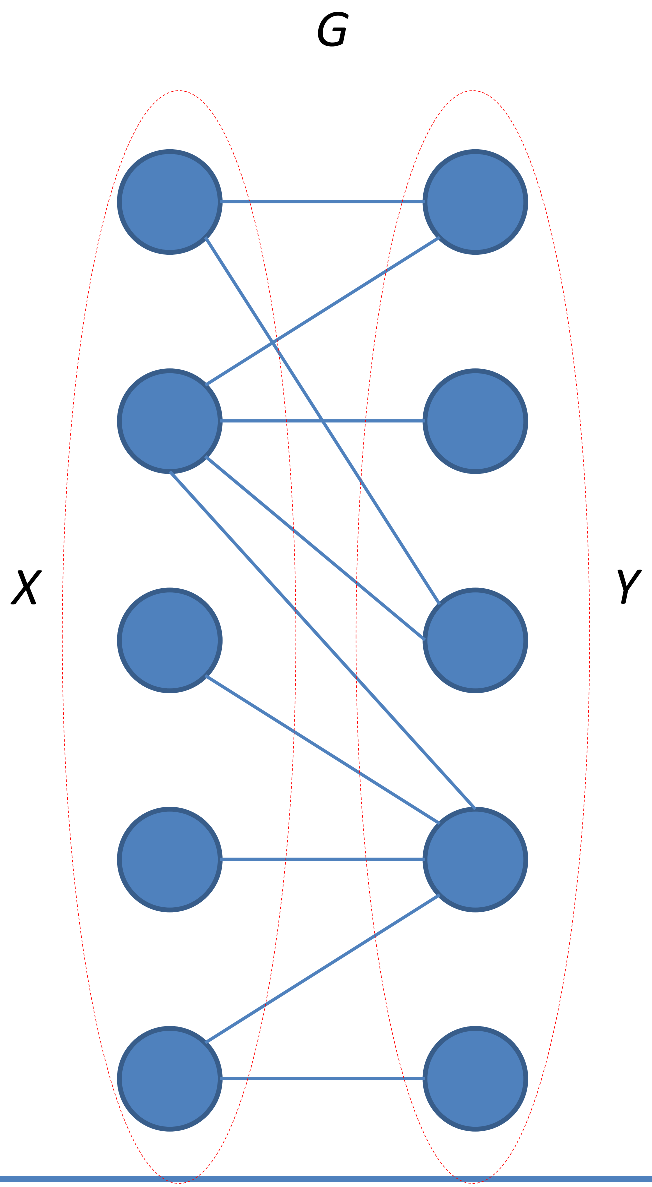
… en que cada arista tiene distancia (costo) unitaria —**algoritmo de Edmonds-Karp** es O(VE^2)

… es decir, el número de aumentos (o iteraciones) realizados por el algoritmo de Edmonds-Karp es O(VE)

Emparejamiento bipartito

Un grafo **bipartito** $G = (V, E)$ es un grafo no direccional cuyo conjunto de nodos puede ser particionado como $V = X \cup Y$

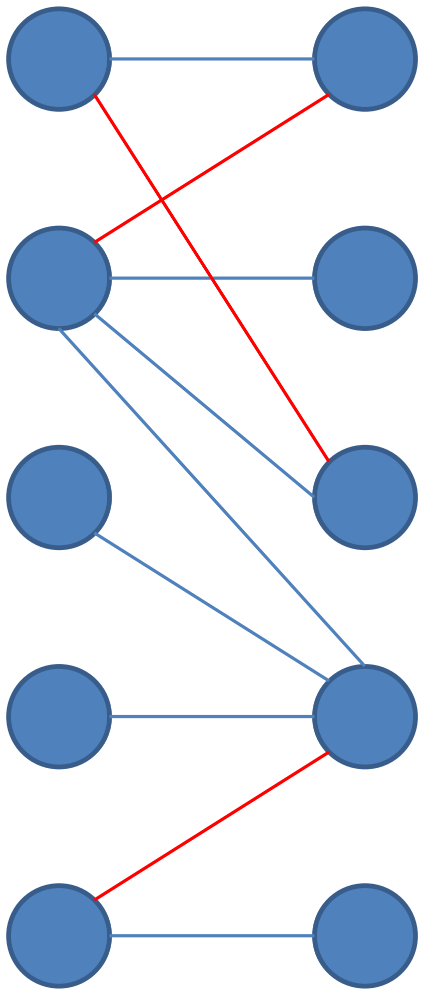
… tal que toda arista $e \in E$ tiene un extremo en X y el otro en Y


G
X
Y

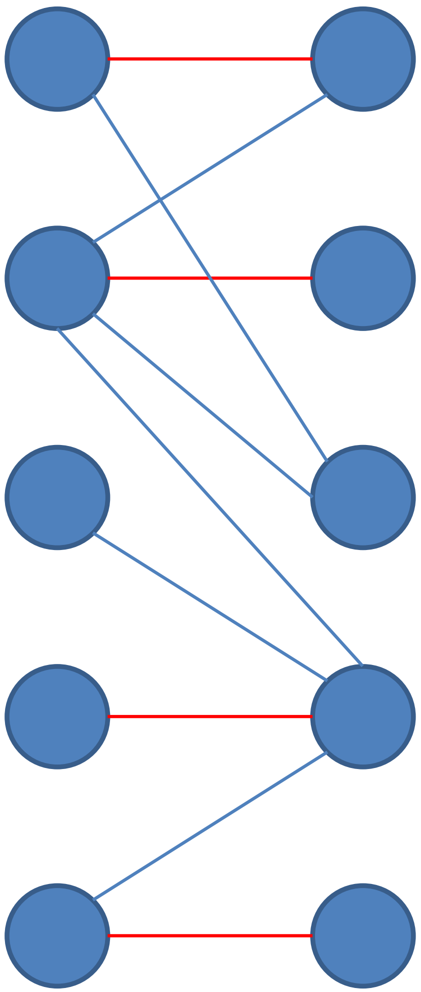
Un **emparejamiento** *M* en *G* es un subconjunto de las aristas $M \subsetneq E$ tal que cada nodo aparece en a lo más una arista en *M*

El problema del emparejamiento bipartito consiste en **encontrar un emparejamiento en *G* del mayor tamaño posible**

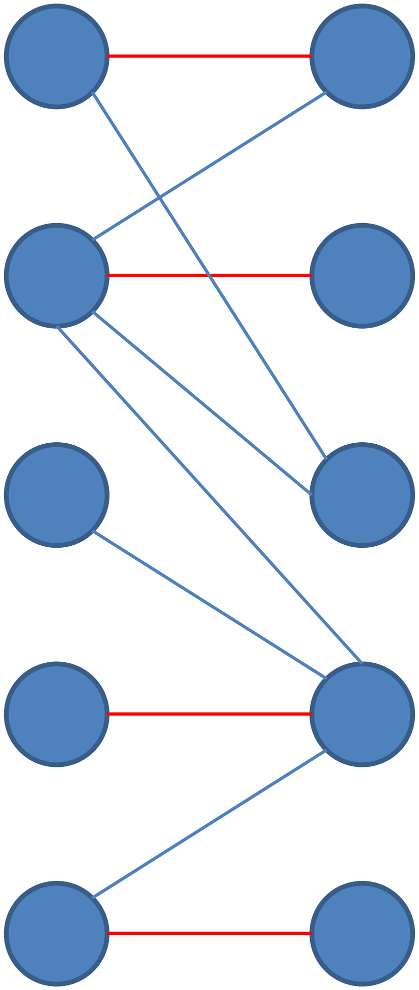
G



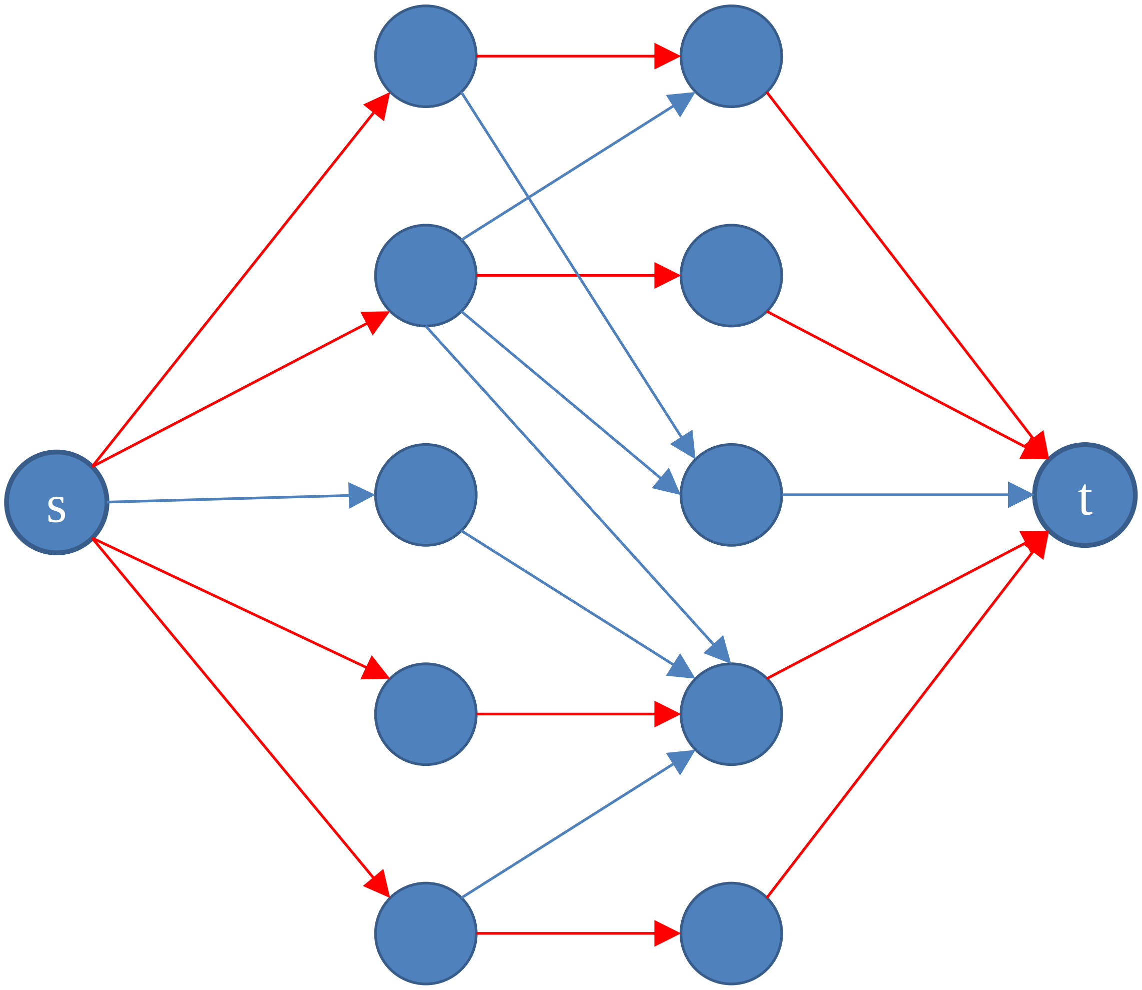
G



G



G', correspondiente a *G*, con todas las capacidades igual a 1



Los flujos en *G'* representan emparejamientos en *G*

Primero, supongamos que hay un emparejamiento en G consistente en k aristas (x_{i1}, y_{i1}), …, (x_{ik}, y_{ik})

Consideremos el flujo f que envía una unidad por cada ruta de la forma s, x_{ij}, y_{ij}, t

Las condiciones de capacidad y conservación se cumplen

… por lo que f es un flujo de valor k

Segundo, supongamos que hay un flujo *f’* en *G’* de valor *k*

Podemos demostrar que si todas las capacidades en la red son enteros,

… entonces hay un flujo de valor (total) *k* de puros números enteros

… y como todas las capacidades son 1, entonces *f*(*e*) es 0 o 1 para cada arista *e*

Consideremos el conjunto *M'* de aristas (*x*, *y*) en que el valor del flujo es 1:

- *M'* contiene (¿obviamente?) *k* aristas
- cada nodo en *X* es la cola de a lo más una arista en *M'* (de lo contrario, más de una unidad de flujo debería entrar al nodo)

Cada nodo en *Y* es la cabeza de a lo más una arista en *M'*

Así, el tamaño de un emparejamiento máximo en *G* es igual al valor del flujo máximo en *G’*

… y las aristas del emparejamiento son las aristas que llevan flujo de *X* a *Y* en *G’*