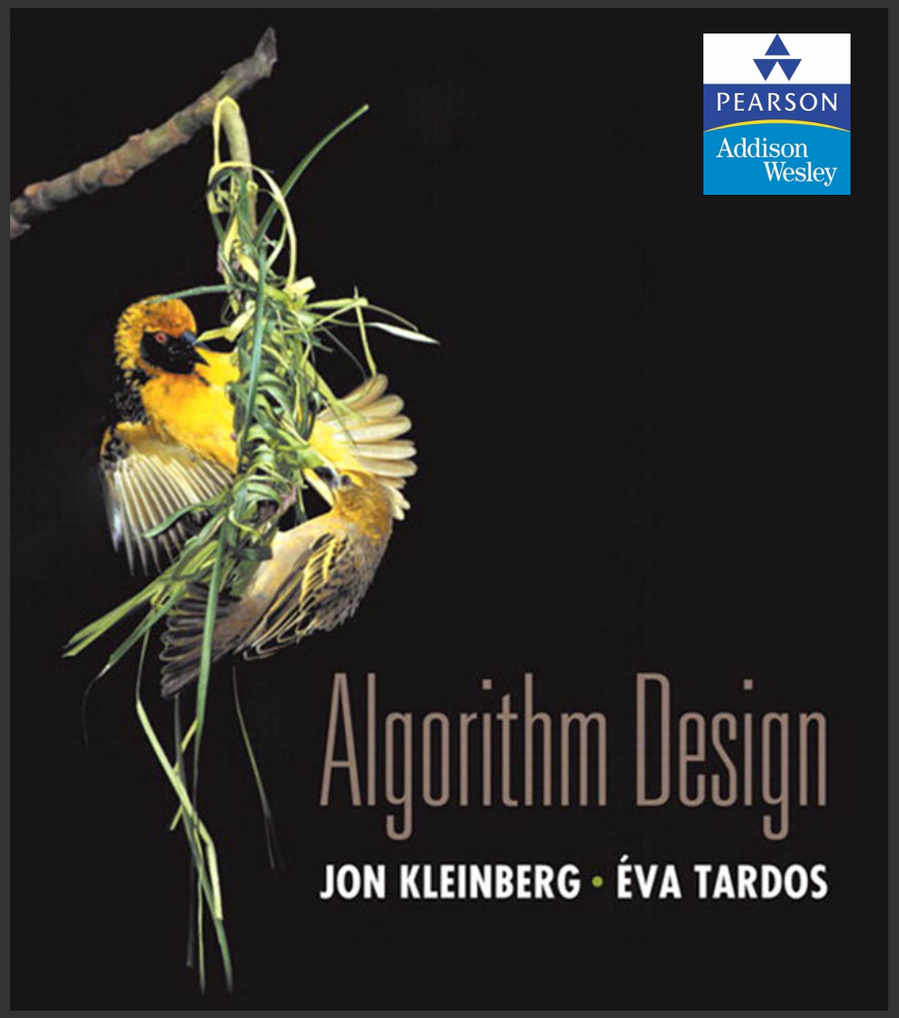


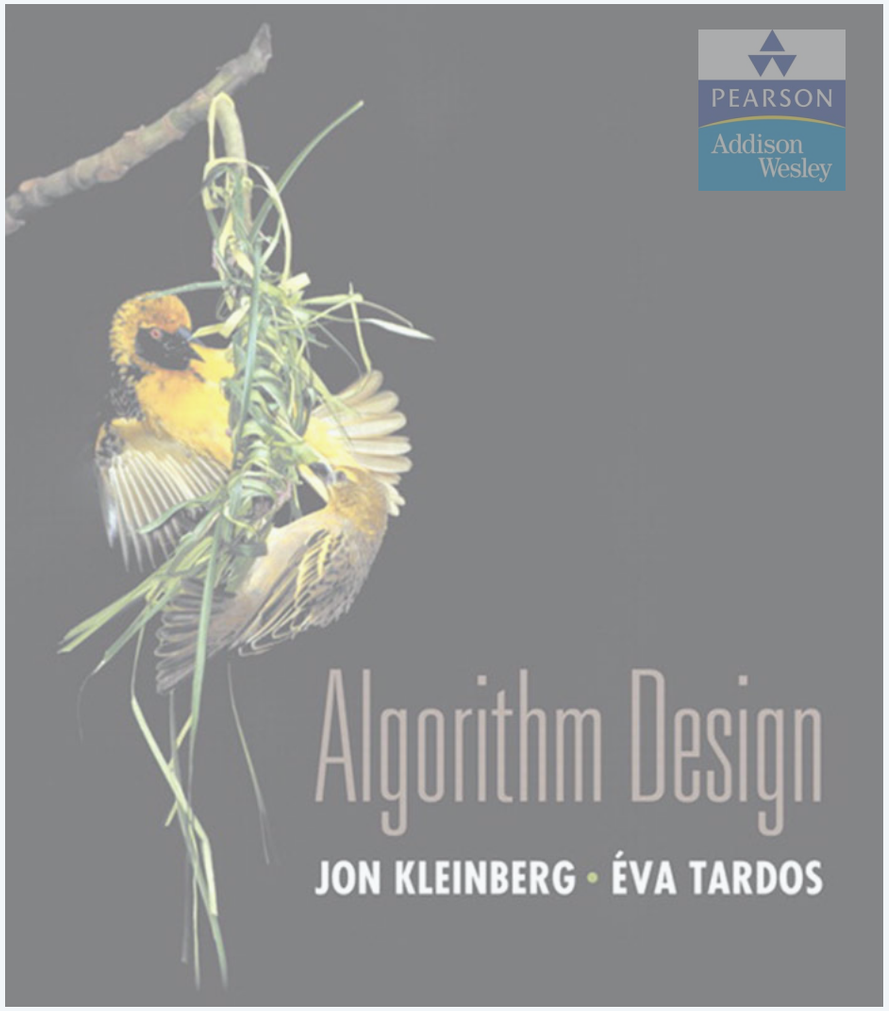
7. Flussi di Rete I

- *massimo flusso e minimo taglio*
- *algoritmo Ford–Fulkerson*
- *teorema massimo flusso - minimo taglio*
- *algoritmo di scaling di capacità*
- *cammini aumentanti minimi*

Traduzione e adattamento di Vincenzo Bonifaci
Original lecture slides by Kevin Wayne

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

Last updated on 03/05/2020 20:37



SECTION 7.1

7. FLUSSI DI RETE I

- *massimo flusso e minimo taglio*
- *Ford–Fulkerson algorithm*
- *max-flow min-cut theorem*
- *capacity-scaling algorithm*
- *shortest augmenting paths*
- *Dinitz' algorithm*
- *simple unit-capacity networks*

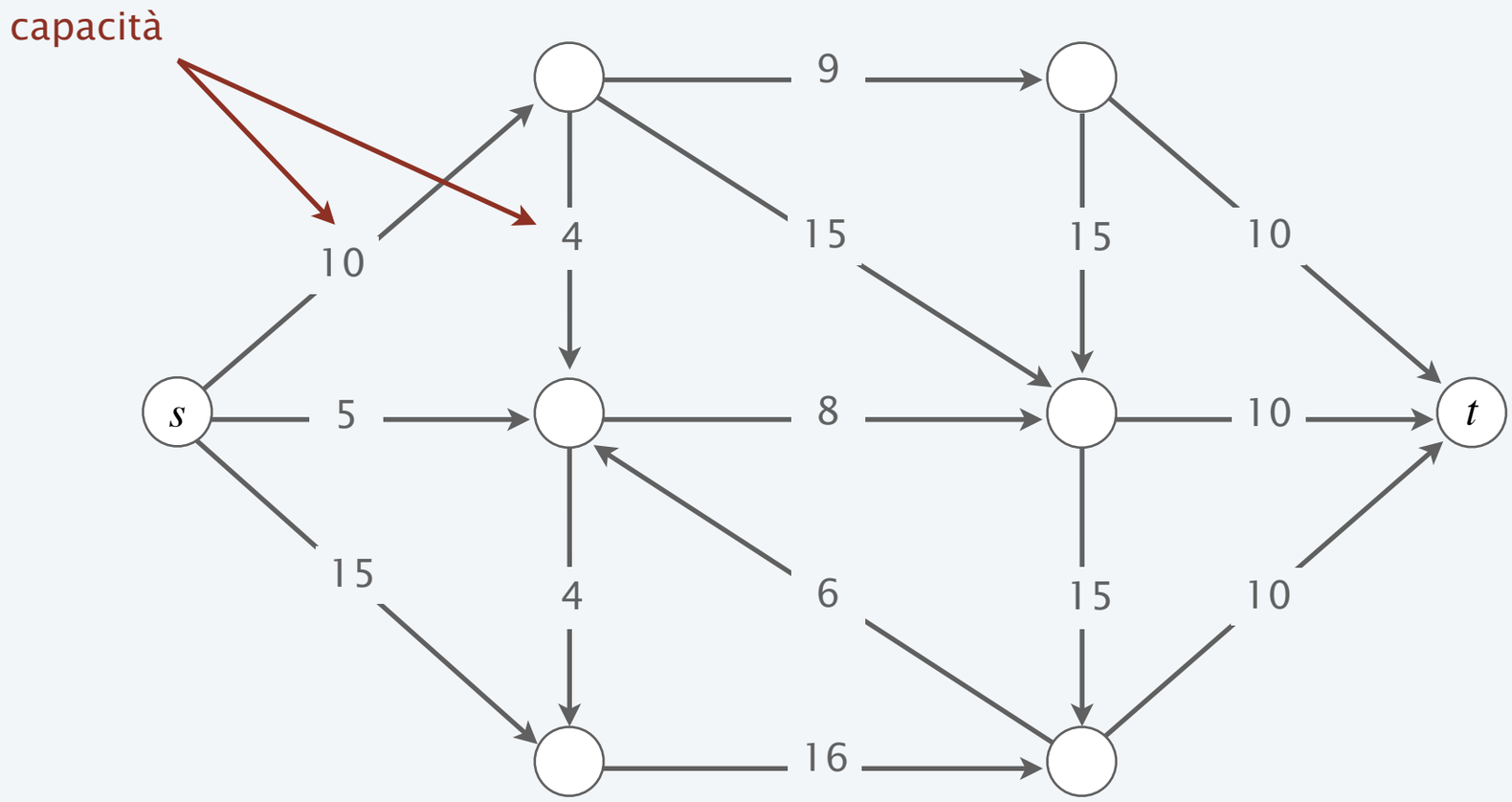
Rete di flusso

Una rete di flusso (o rete capacitata) è una tupla $G = (V, E, s, t, c)$.

- Digrafo (V, E) con sorgente $s \in V$ e pozzo $t \in V$.
- Capacità $c(e) \geq 0$ per ogni $e \in E$.

assume che tutti i nodi siano raggiungibili da s

Intuizione. Materiali che fluiscono attraverso una rete di trasporto; i materiali sono originati alla sorgente e inviati al pozzo.

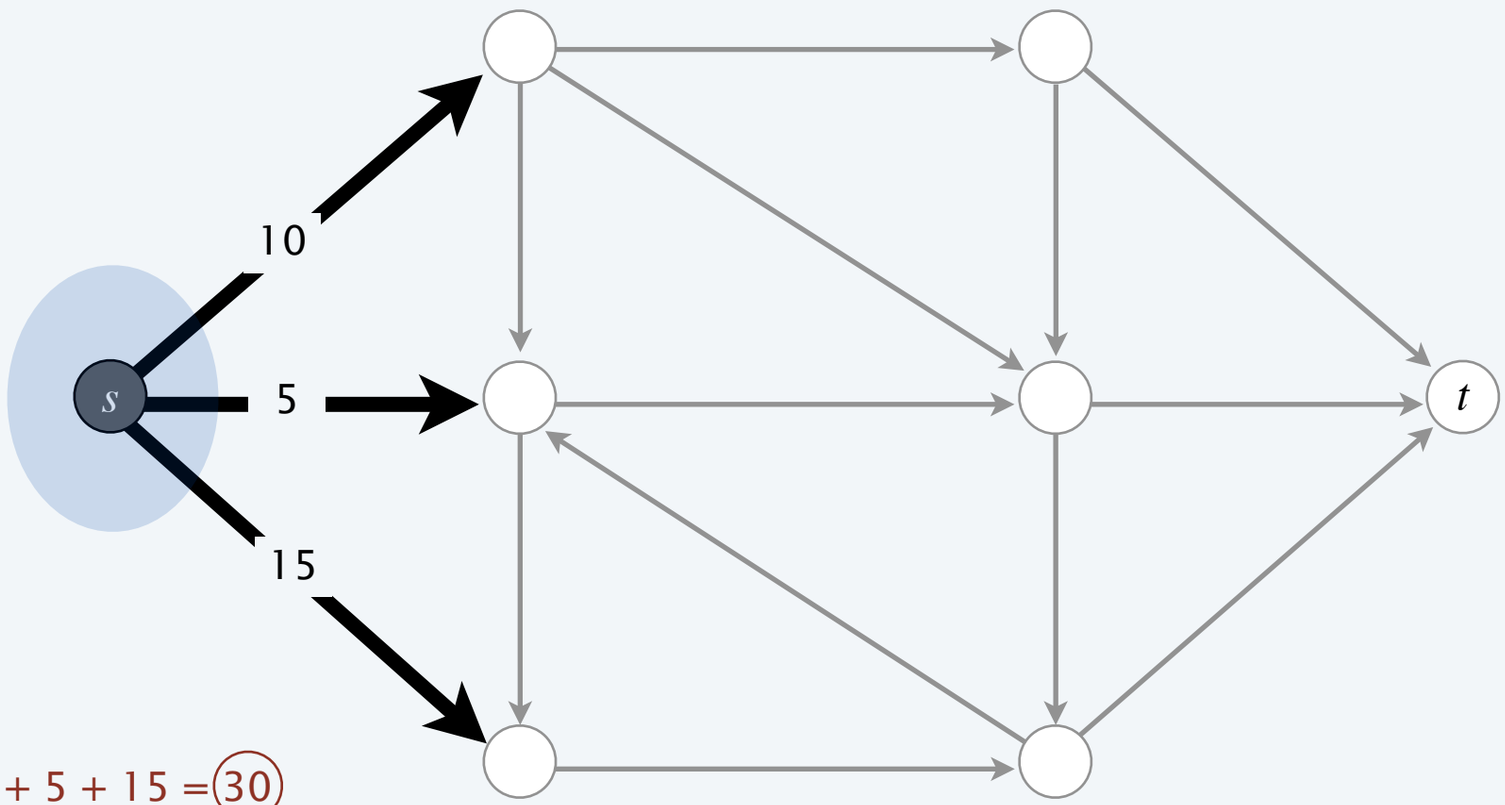


Problema del minimo taglio

Def. Un taglio s-t (o semplicemente un taglio) è una partizione (A, B) dei nodi con $s \in A$ e $t \in B$.

Def. La sua capacità è la somma delle capacità degli archi da A a B.

$$cap(A, B) = \sum_{e \text{ out of } A} c(e)$$



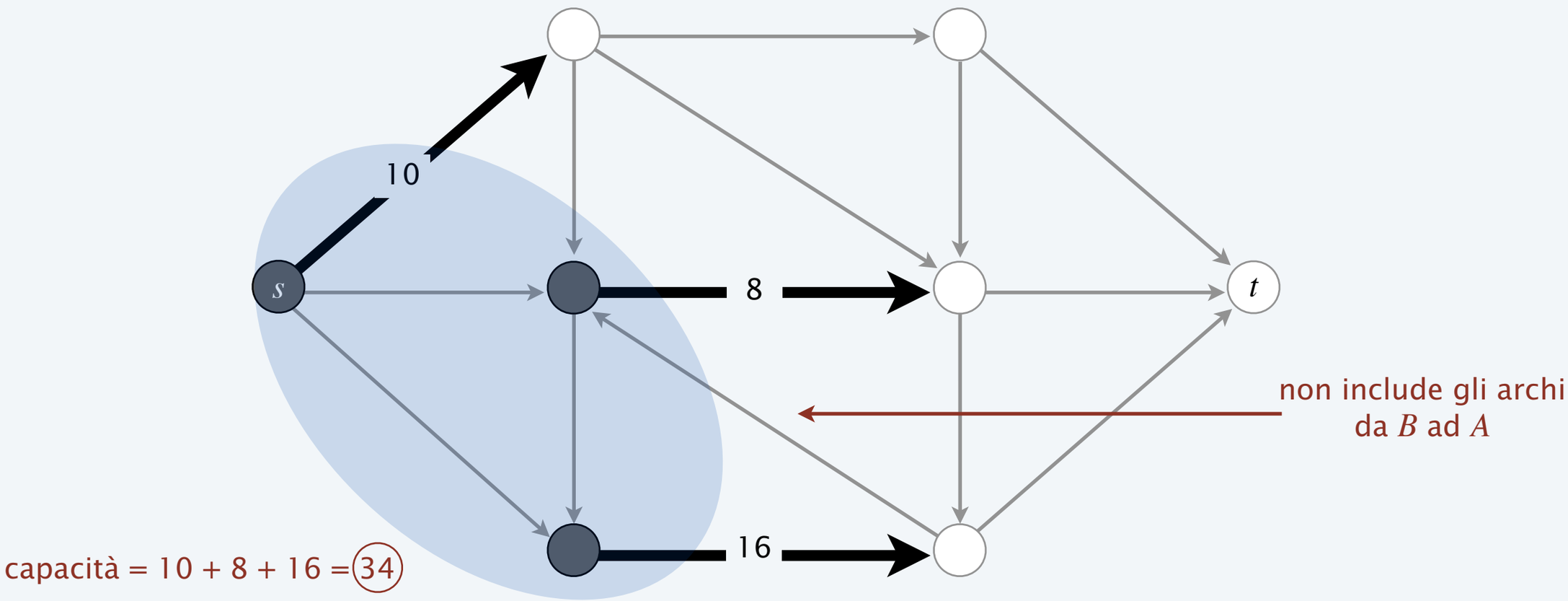
capacità = 10 + 5 + 15 = 30

Problema del minimo taglio

Def. Un taglio s-t (o semplicemente un taglio) è una partizione (A, B) dei nodi con $s \in A$ e $t \in B$.

Def. La sua capacità è la somma delle capacità degli archi da A a B.

$$cap(A, B) = \sum_{e \text{ out of } A} c(e)$$



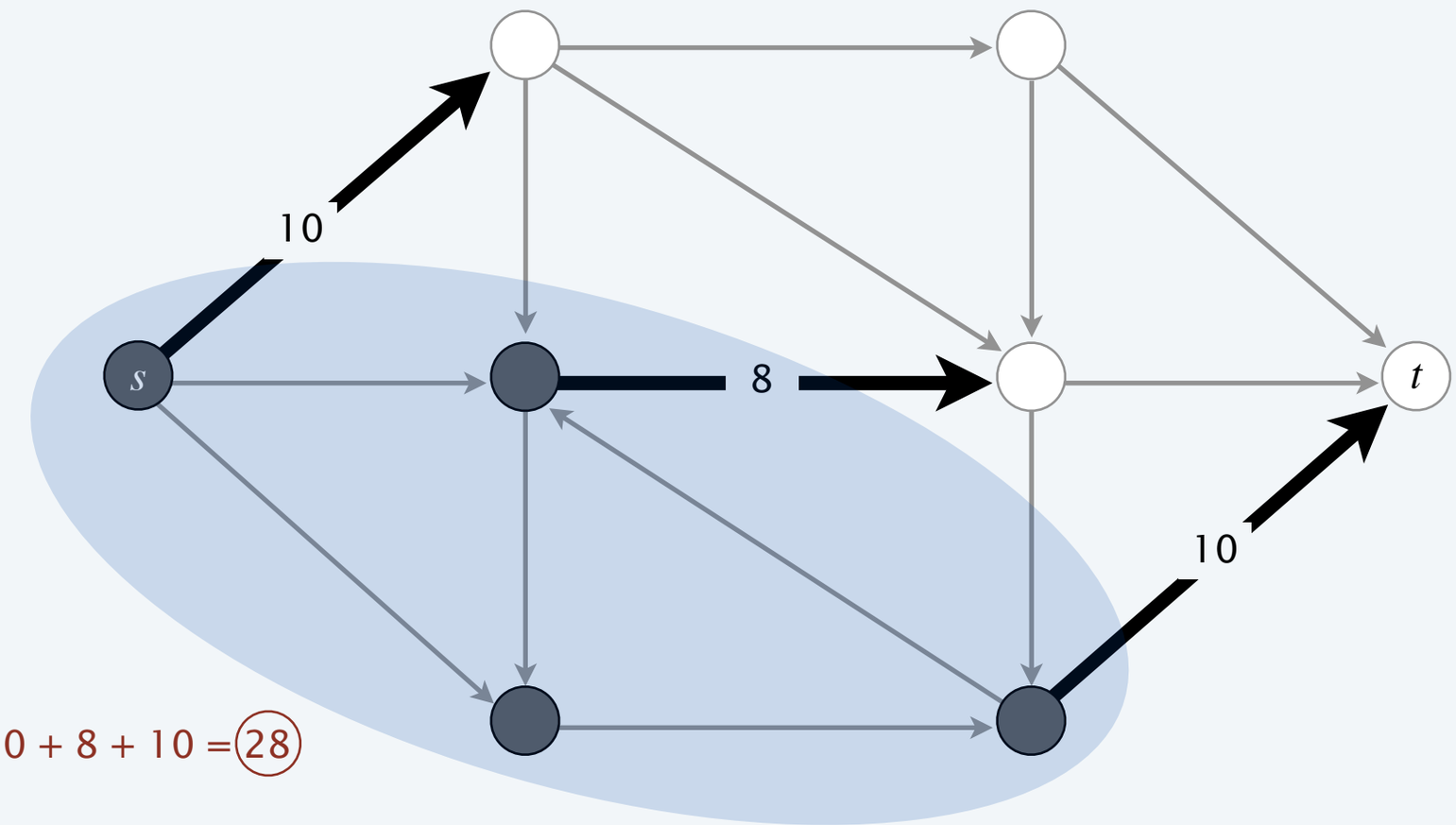
Problema del minimo taglio

Def. Un taglio s-t (o semplicemente un taglio) è una partizione (A, B) dei nodi con $s \in A$ e $t \in B$.

Def. La sua capacità è la somma delle capacità degli archi da A a B.

$$cap(A, B) = \sum_{e \text{ out of } A} c(e)$$

Problema del minimo taglio. Trovare un taglio a capacità minima.



capacità = 10 + 8 + 10 = 28

Flussi di rete: quiz 1

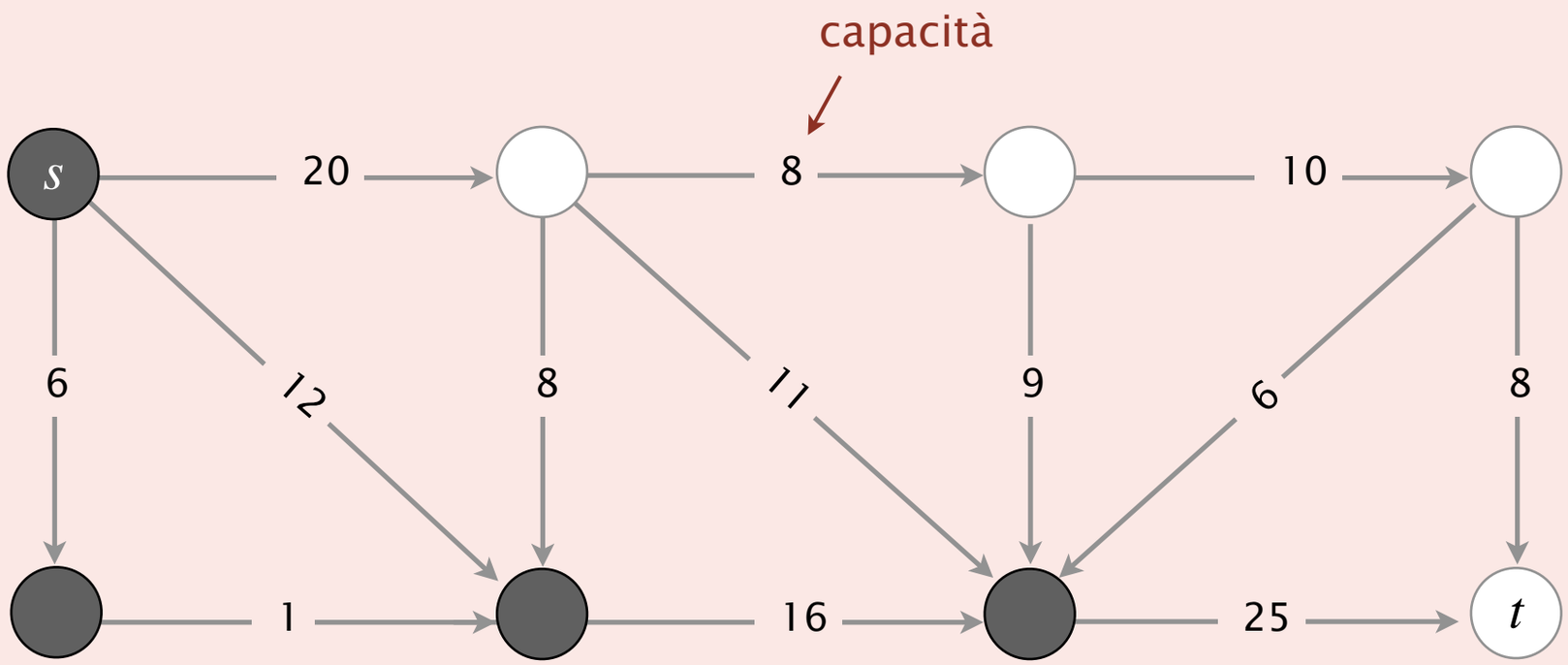
Qual è la capacità del taglio *s*-*t* raffigurato?

A. 11 (20 + 25 – 8 – 11 – 9 – 6)

B. 34 (8 + 11 + 9 + 6)

C. 45 (20 + 25)

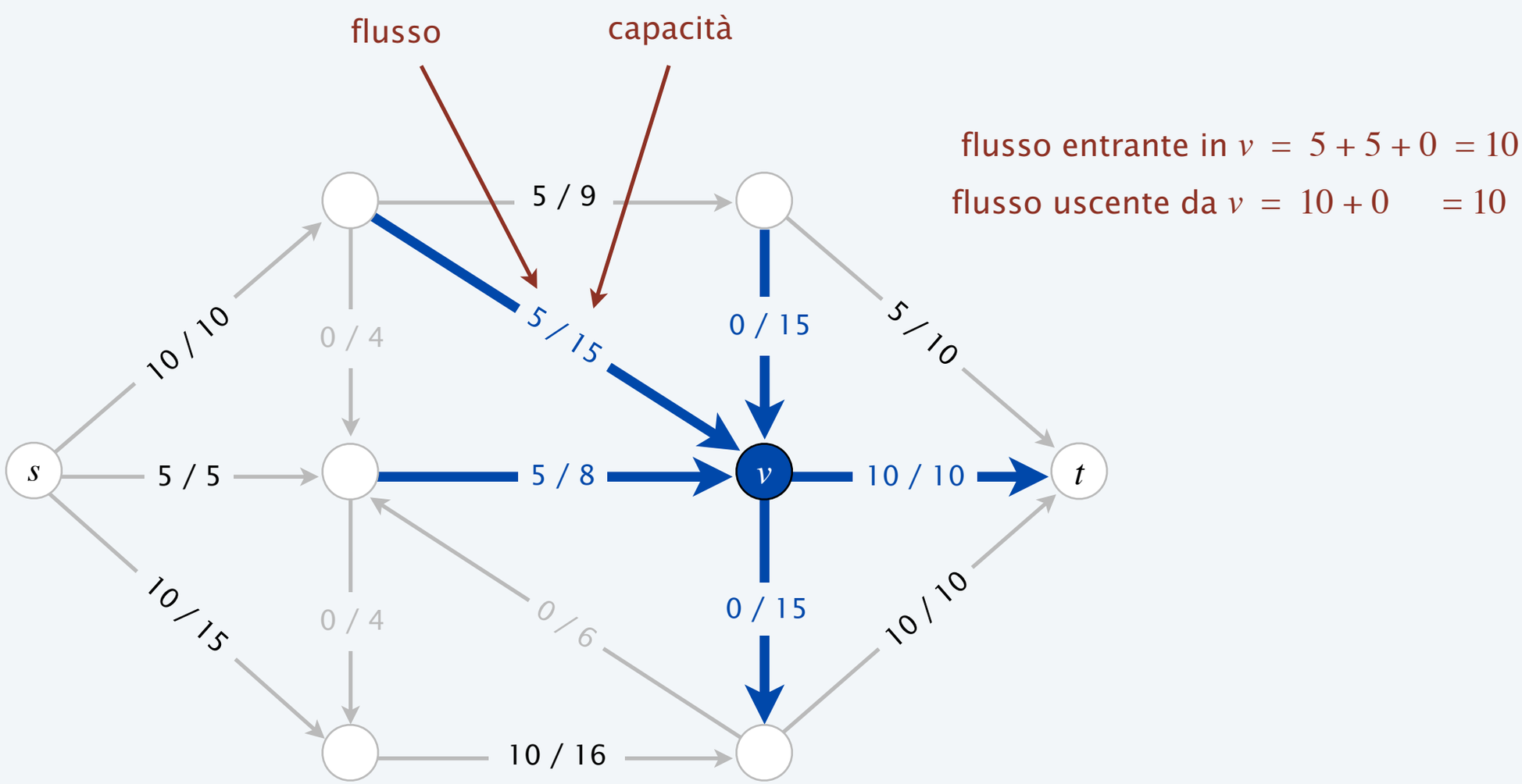
D. 79 (20 + 25 + 8 + 11 + 9 + 6)



Problema del massimo flusso

Def. Un flusso *s-t* (o semplicemente un flusso) f è una funzione che soddisfa:

- Per ogni $e \in E$: $0 \le f(e) \le c(e)$ [capacità]
- Per ogni $v \in V - \{s, t\}$: $\sum_{e \text{ in to } v} f(e) = \sum_{e \text{ out of } v} f(e)$ [conservazione del flusso]

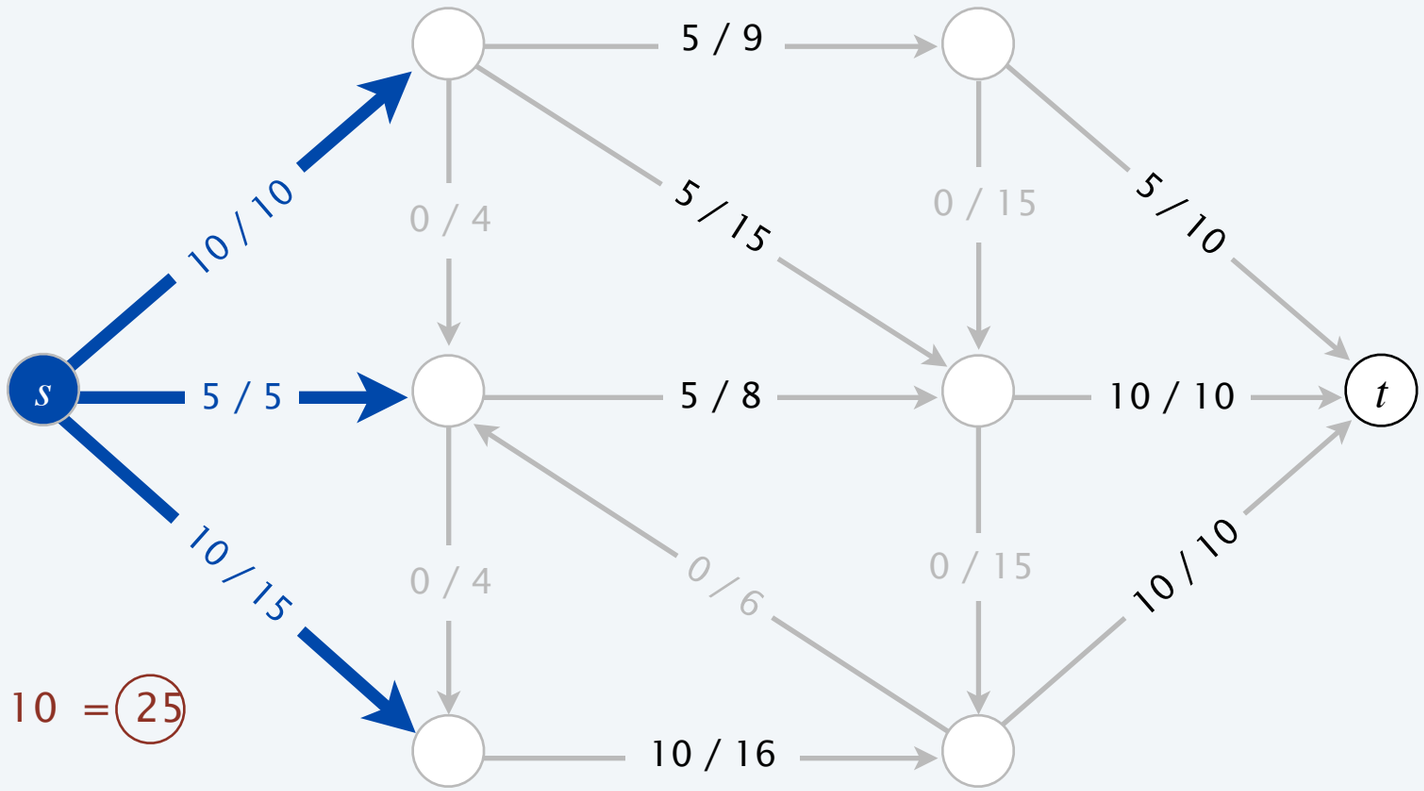


Problema del massimo flusso

Def. Un flusso *s*-*t* (o semplicemente un flusso) f è una funzione che soddisfa:

- Per ogni $e \in E$: $0 \le f(e) \le c(e)$ [capacità]
- Per ogni $v \in V - \{s, t\}$: $\sum_{e \text{ in to } v} f(e) = \sum_{e \text{ out of } v} f(e)$ [conservazione del flusso]

Def. Il valore di un flusso f è: $val(f) = \sum_{e \text{ out of } s} f(e) - \sum_{e \text{ in to } s} f(e)$



valore = 5 + 10 + 10 = 25

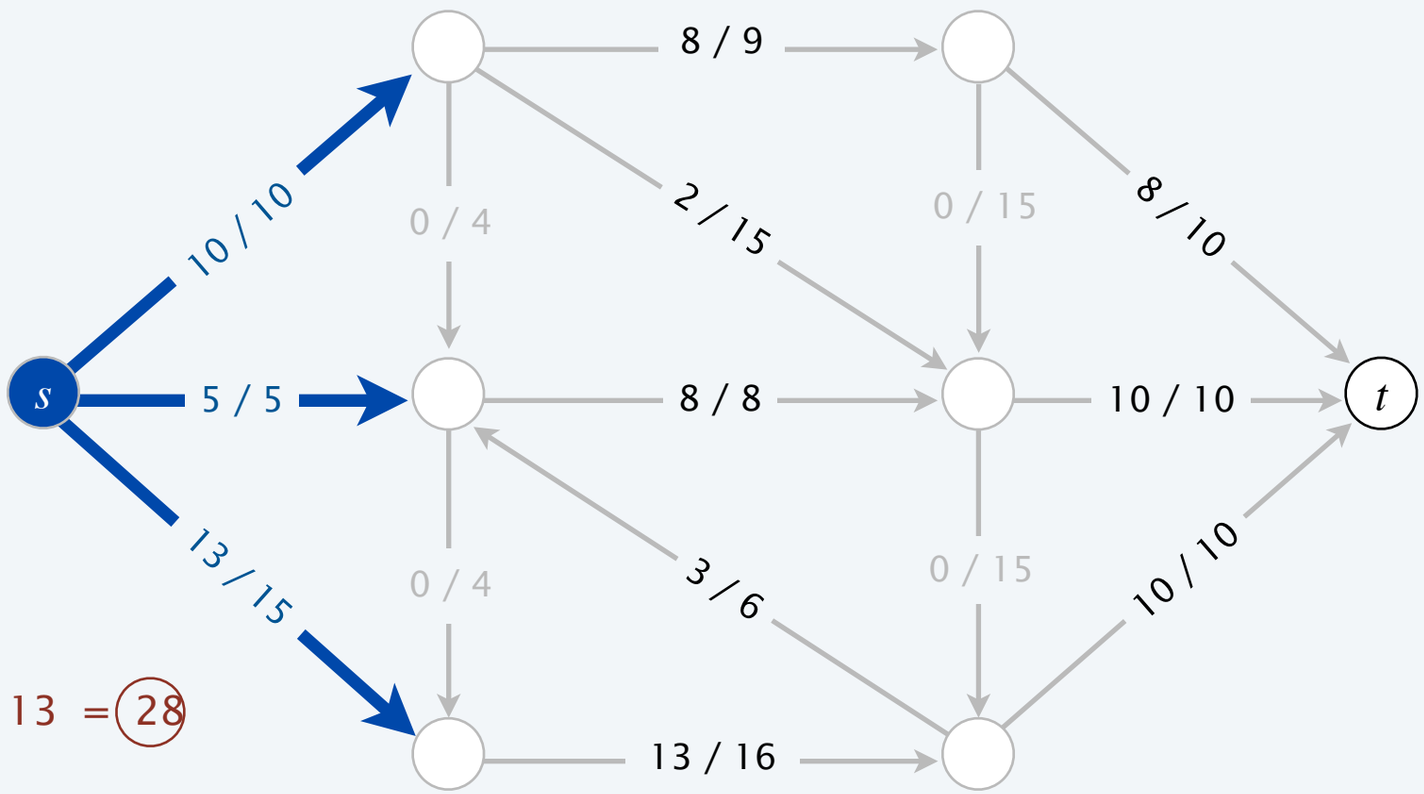
Problema del massimo flusso

Def. Un flusso s-t (o semplicemente un flusso) f è una funzione che soddisfa:

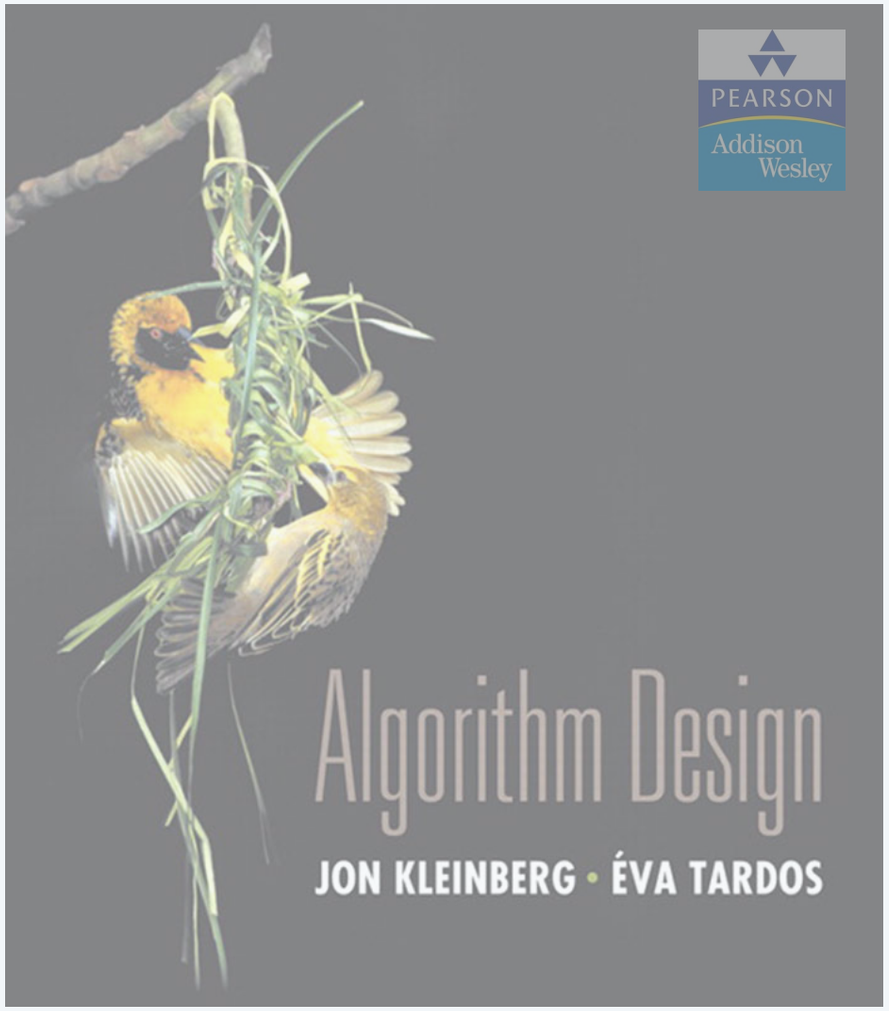
- Per ogni $e \in E$: $0 \le f(e) \le c(e)$ [capacità]
- Per ogni $v \in V - \{s, t\}$: $\sum_{e \text{ in to } v} f(e) = \sum_{e \text{ out of } v} f(e)$ [conservazione del flusso]

Def. Il valore di un flusso f è: $val(f) = \sum_{e \text{ out of } s} f(e) - \sum_{e \text{ in to } s} f(e)$

Problema del massimo flusso. Trovare un flusso di valore massimo.



valore = 10 + 5 + 13 = 28



SECTION 7.1

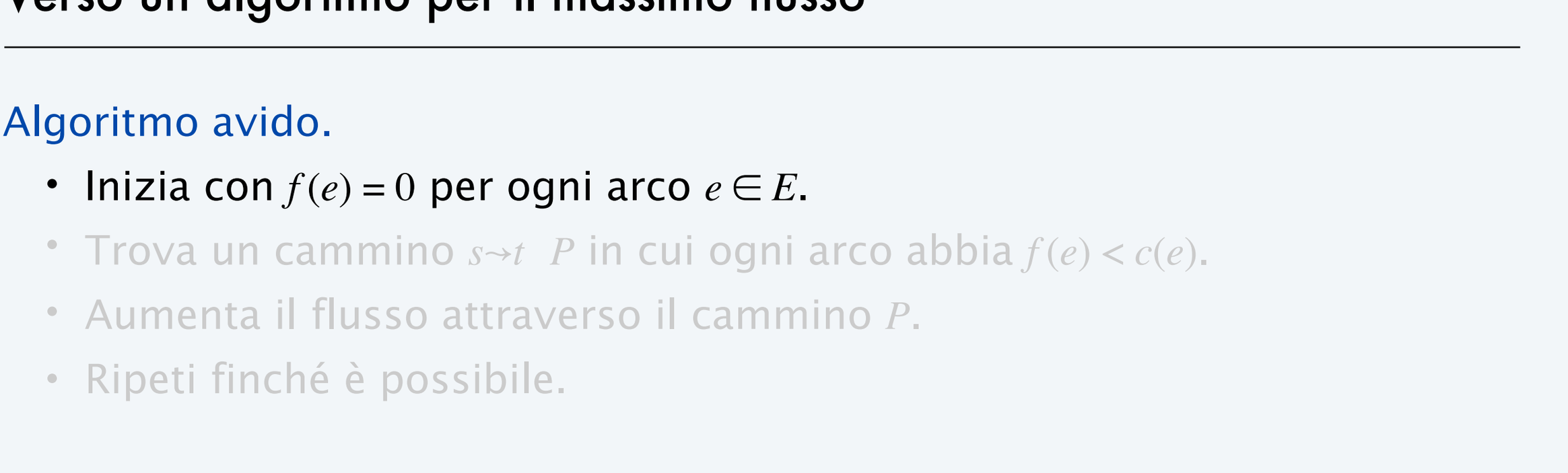
7. FLUSSI DI RETE I

- *max-flow and min-cut problems*
- *algoritmo di Ford–Fulkerson*
- *max-flow min-cut theorem*
- *capacity-scaling algorithm*
- *shortest augmenting paths*
- *Dinitz' algorithm*
- *simple unit-capacity networks*

Verso un algoritmo per il massimo flusso

Algoritmo avido.

- Inizia con $f(e) = 0$ per ogni arco $e \in E$.
- Trova un cammino $s \rightarrow t$ P in cui ogni arco abbia $f(e) < c(e)$.
- Aumenta il flusso attraverso il cammino P.
- Ripeti finché è possibile.

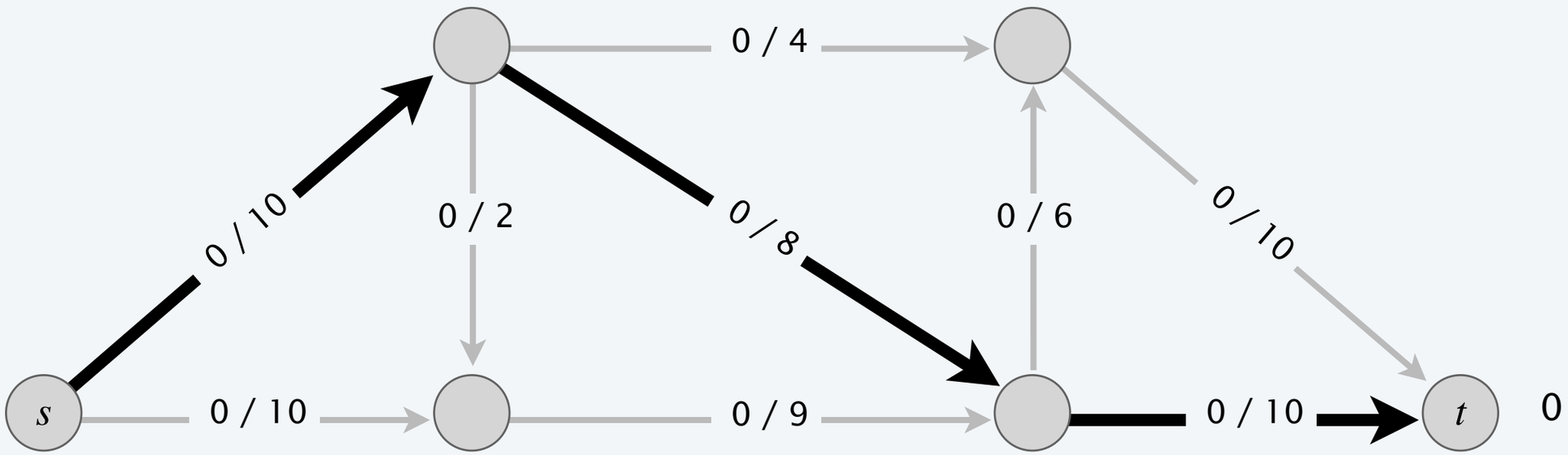


Verso un algoritmo per il massimo flusso

Algoritmo avido.

- Inizia con $f(e) = 0$ per ogni arco $e \in E$.
- Trova un cammino $s \rightarrow t$ P in cui ogni arco abbia $f(e) < c(e)$.
- Aumenta il flusso attraverso il cammino P.
- Ripeti finché è possibile.

rete di flusso G e flusso f

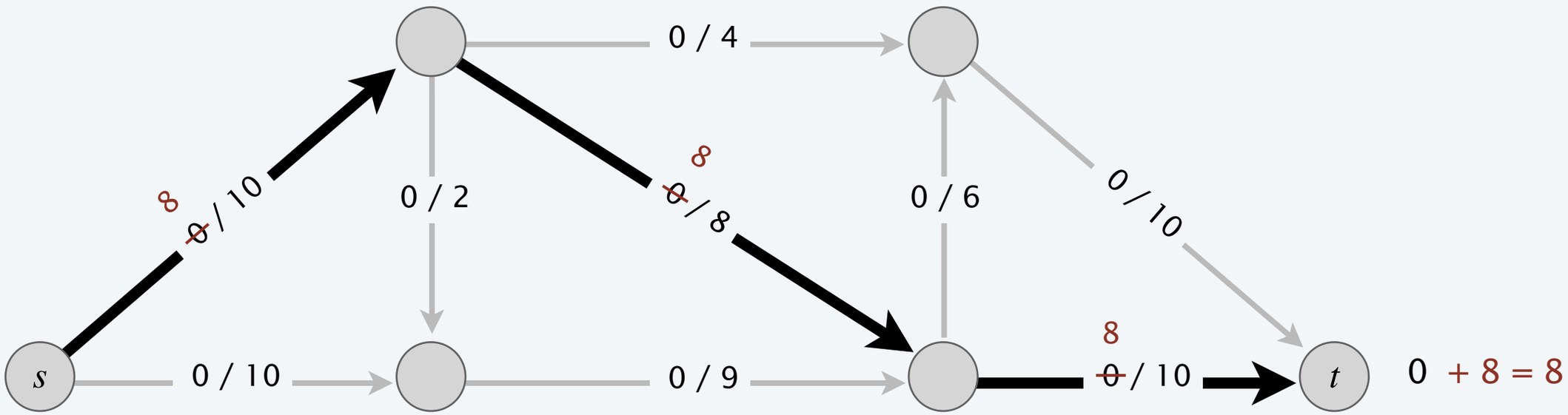


Verso un algoritmo per il massimo flusso

Algoritmo avido.

- Inizia con $f(e) = 0$ per ogni arco $e \in E$.
- Trova un cammino $s \rightarrow t$ P in cui ogni arco abbia $f(e) < c(e)$.
- Aumenta il flusso attraverso il cammino P.
- Ripeti finché è possibile.

rete di flusso G e flusso f

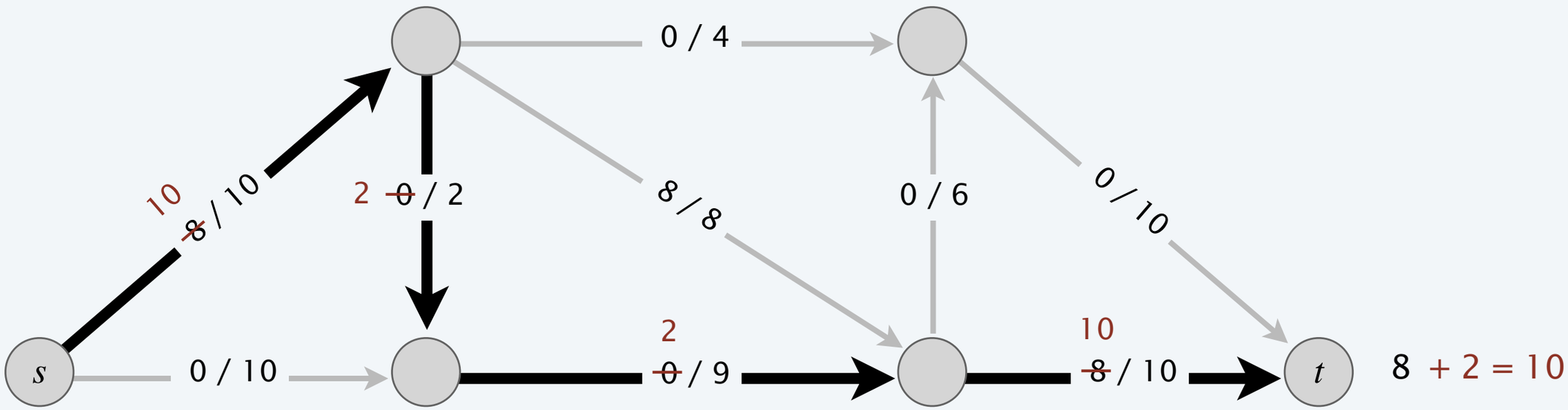


Verso un algoritmo per il massimo flusso

Algoritmo avido.

- Inizia con $f(e) = 0$ per ogni arco $e \in E$.
- Trova un cammino $s \rightarrow t$ P in cui ogni arco abbia $f(e) < c(e)$.
- Aumenta il flusso attraverso il cammino P.
- Ripeti finché è possibile.

rete di flusso G e flusso f

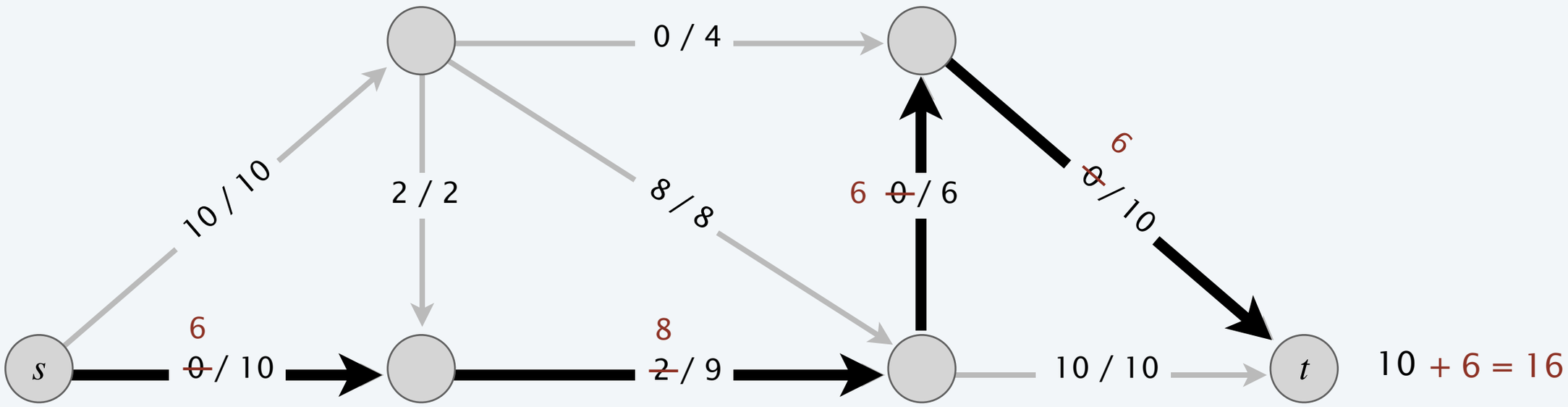


Verso un algoritmo per il massimo flusso

Algoritmo avido.

- Inizia con $f(e) = 0$ per ogni arco $e \in E$.
- Trova un cammino $s \rightarrow t$ P in cui ogni arco abbia $f(e) < c(e)$.
- Aumenta il flusso attraverso il cammino P.
- Ripeti finché è possibile.

rete di flusso G e flusso f



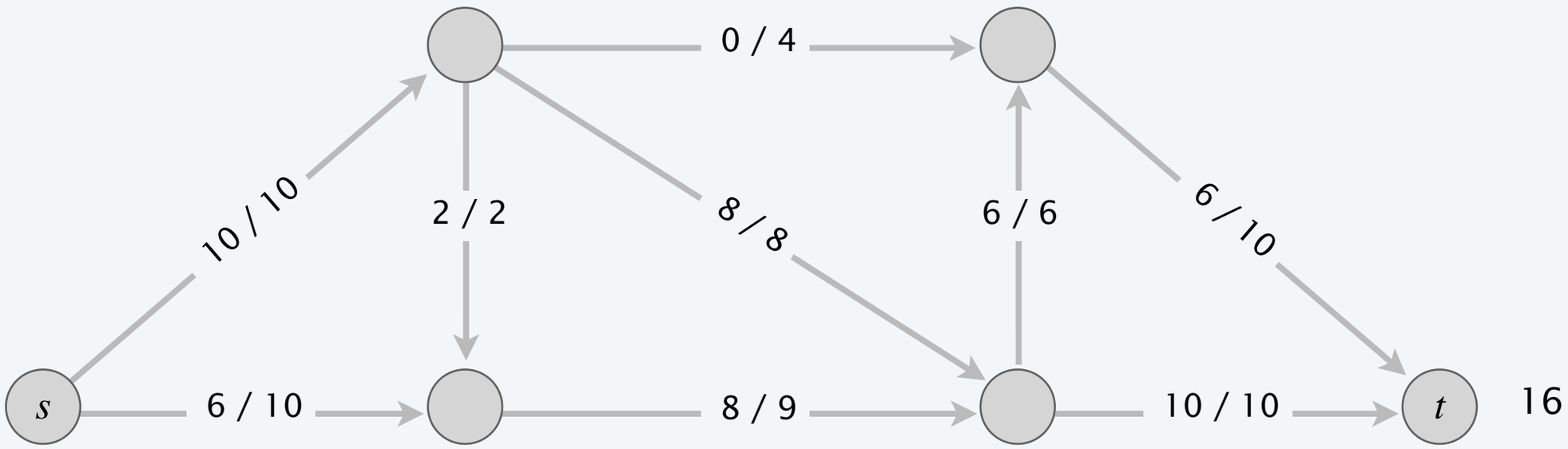
Verso un algoritmo per il massimo flusso

Algoritmo avido.

- Inizia con $f(e) = 0$ per ogni arco $e \in E$.
- Trova un cammino $s \rightarrow t$ P in cui ogni arco abbia $f(e) < c(e)$.
- Aumenta il flusso attraverso il cammino P.
- Ripeti finché è possibile.

valore finale del flusso = 16

rete di flusso G e flusso f



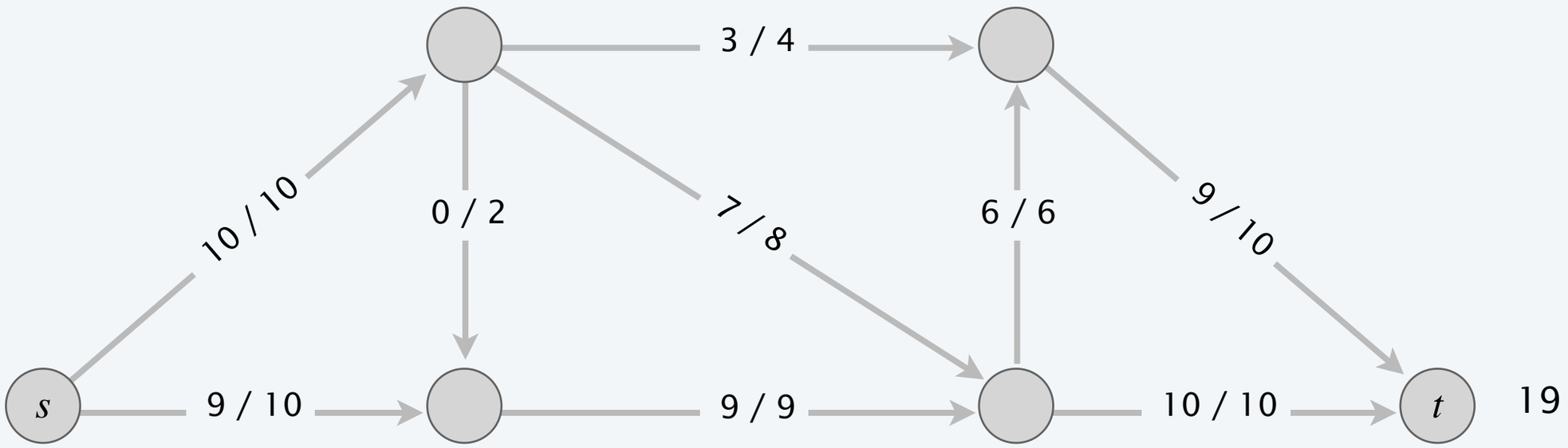
Verso un algoritmo per il massimo flusso

Algoritmo avido.

- Inizia con $f(e) = 0$ per ogni arco $e \in E$.
- Trova un cammino $s \rightarrow t$ P in cui ogni arco abbia $f(e) < c(e)$.
- Aumenta il flusso attraverso il cammino P.
- Ripeti finché è possibile.

ma valore del massimo flusso = 19

rete di flusso G e flusso f



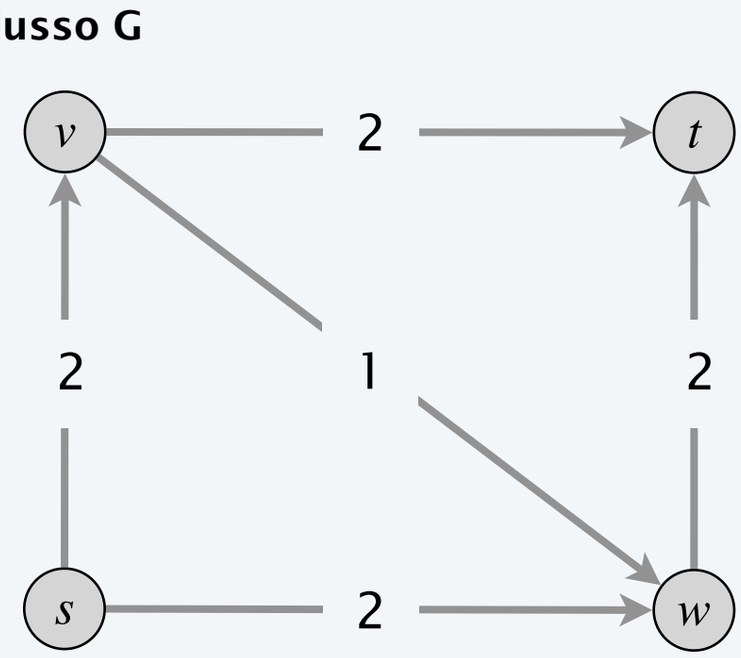
Perché l'algoritmo avido fallisce

D. Perché l'algoritmo avido fallisce?

R. Dopo che l'algoritmo avido aumenta il flusso su un arco, non lo diminuisce più.

Es. Consideriamo la rete di flusso G.

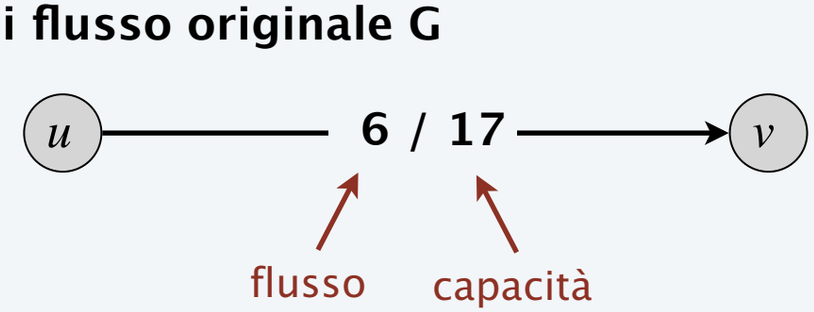
- Il flusso massimo f^* è unico e soddisfa $f^*(v, w) = 0$.
- L'algoritmo avido potrebbe scegliere $s \to v \to w \to t$ come primo cammino.



Rete residua

Arco originale. $e = (u, v) \in E$.

- Flusso $f(e)$.
- Capacità $c(e)$.

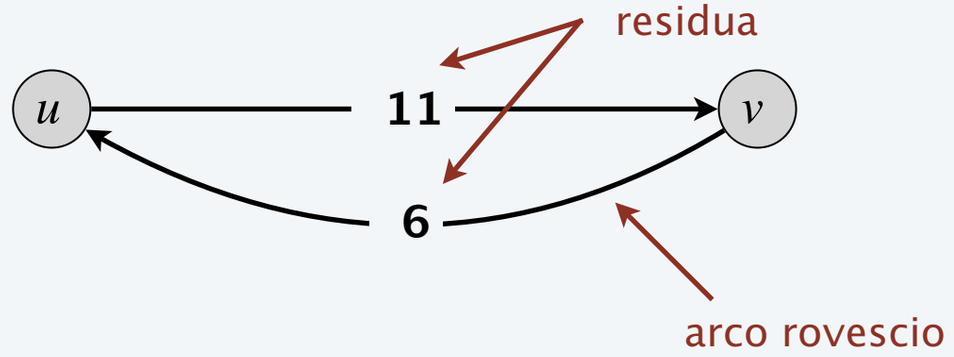


Arco rovescio. $e^{\text{reverse}} = (v, u)$.

- "Annulla" il flusso inviato.

Capacità residua.

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



Rete residua. $G_f = (V, E_f, s, t, c_f)$.

- $E_f = \{e : f(e) < c(e)\} \cup \{e^{\text{reverse}} : f(e) > 0\}$.
- Proprietà chiave: f' è un flusso in G_f sse $f + f'$ è un flusso in G.

archi con capacità residua positiva

qui il flusso su un arco rovescio nega il flusso sul corrispondente arco opposto

Cammino aumentante

Def. Un cammino aumentante è un cammino $s \rightsquigarrow t$ semplice nella rete residua G_f.

Def. La capacità di strozzatura di un cammino aumentante P è la minima capacità residua di un arco di P.

Proprietà chiave. Sia f un flusso e sia P un cammino aumentante in G_f. Allora, dopo la chiamata $f' \leftarrow \text{AUGMENT}(f, c, P)$, f' è un flusso e $val(f') = val(f) + strozzatura(G_f, P)$.

AUGMENT(f, c, P)

$\delta \leftarrow$ capacità di strozzatura del cammino P.

FOREACH arco $e \in P$:

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

  ELSE $f(e^{\text{reverse}}) \leftarrow f(e^{\text{reverse}}) - \delta$.

RETURN f.

Flusso di rete: quiz 2

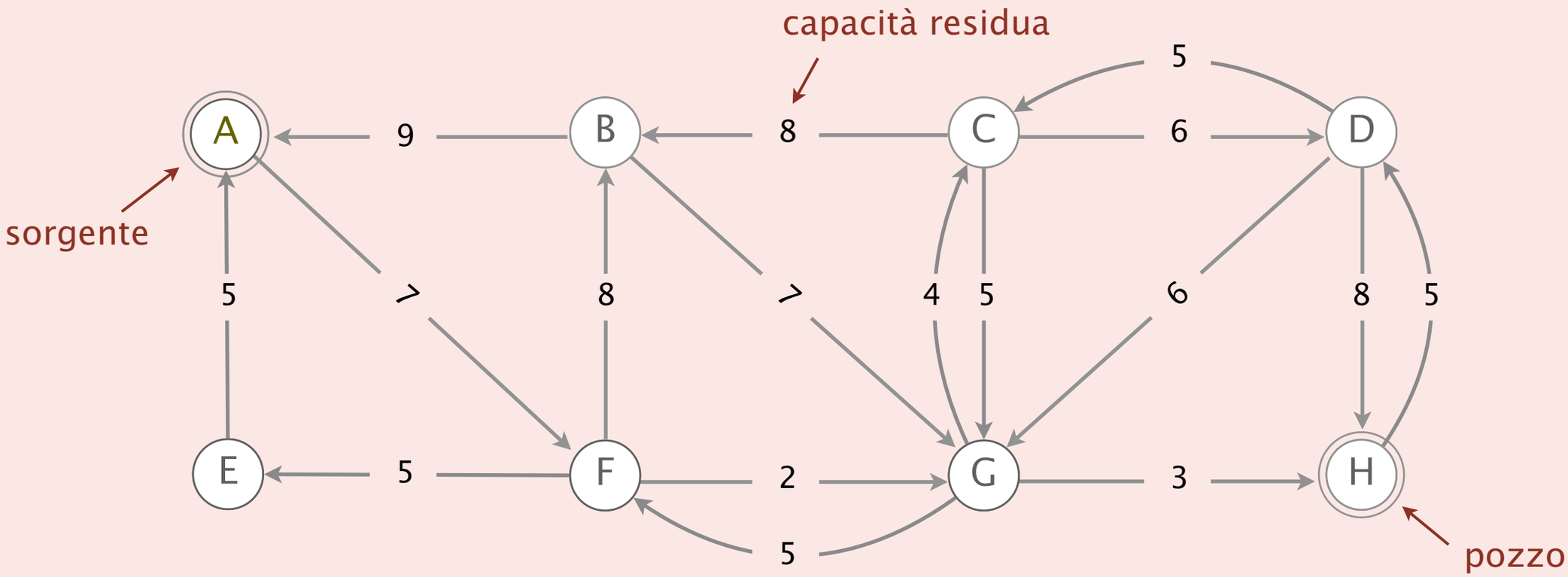
Quale di questi cammini ha la più alta capacità di strozzatura?

A. $A \to F \to G \to H$

B. $A \to B \to C \to D \to H$

C. $A \to F \to B \to G \to H$

D. $A \to F \to B \to G \to C \to D \to H$



Algoritmo di Ford-Fulkerson

Algoritmo dei cammini aumentanti di Ford–Fulkerson.

- Inizia con $f(e) = 0$ per ogni arco $e \in E$.
- Trova un cammino $s \rightsquigarrow t$ P nella rete residua G_f.
- Aumenta il flusso attraverso P.
- Ripeti finché è possibile.

FORD–FULKERSON(G)

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

$G_f \leftarrow$ rete residua di G rispetto al flusso f.

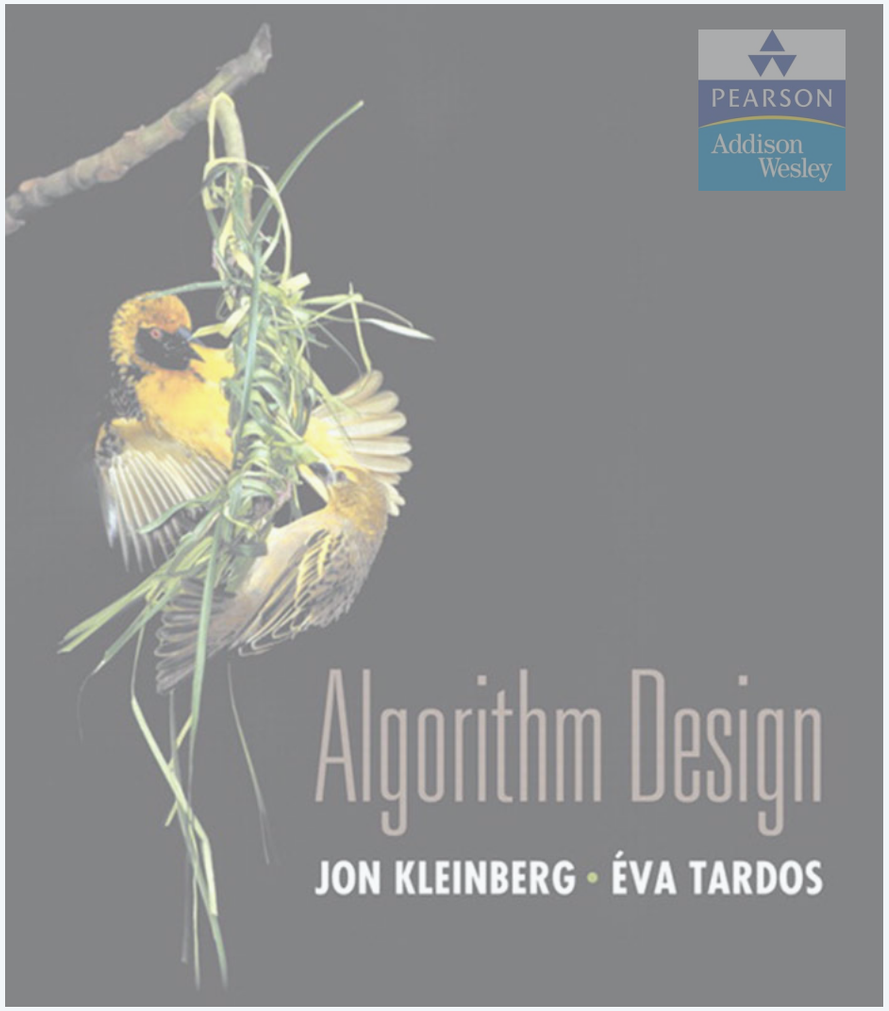
WHILE (esiste un cammino s$\rightsquigarrow$t P in G_f)

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

  Aggiorna G_f.

RETURN f.

cammino aumentante



SECTION 7.2

7. FLUSSI DI RETE I

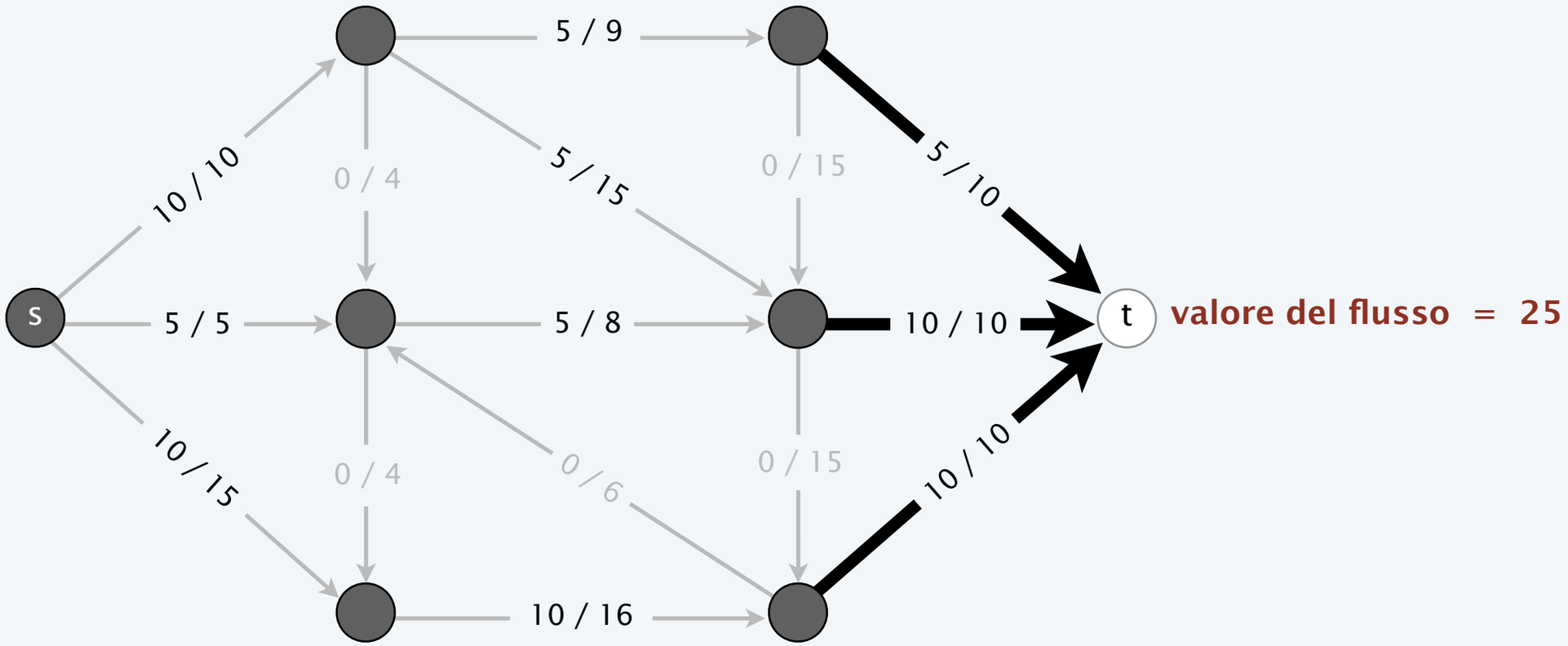
- *max-flow and min-cut problems*
- *Ford–Fulkerson algorithm*
- *teorema massimo flusso - minimo taglio*
- *capacity-scaling algorithm*
- *shortest augmenting paths*
- *Dinitz' algorithm*
- *simple unit-capacity networks*

Relazione tra flussi e tagli

Lemma valore del flusso. Sia f un flusso e sia (A, B) un taglio. Allora, il valore del flusso f uguaglia il flusso netto attraverso il taglio (A, B).

$$val(f) = \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e)$$

flusso netto = 5 + 10 + 10 = 25

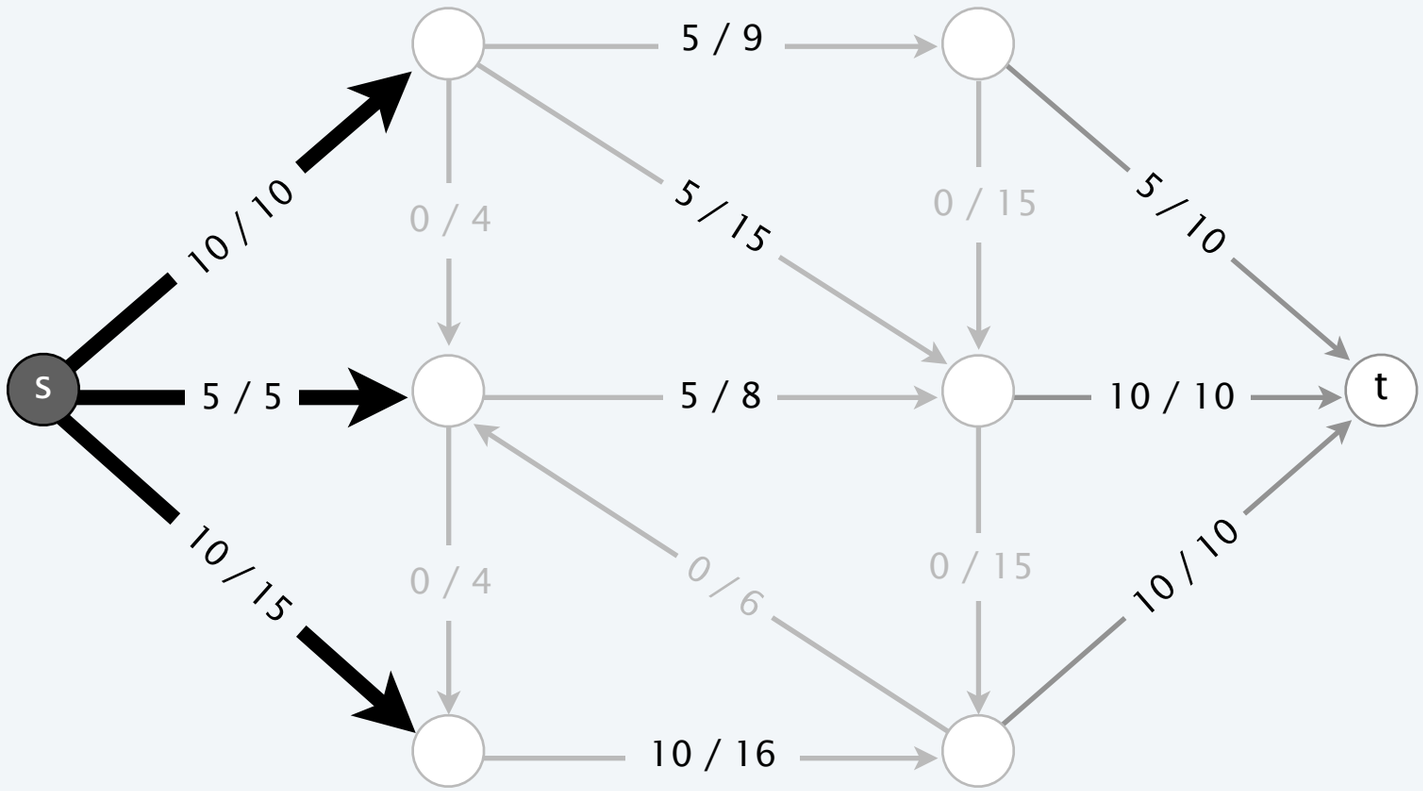


Relazione tra flussi e tagli

Lemma valore del flusso. Sia f un flusso e sia (A, B) un taglio. Allora, il valore del flusso f uguaglia il flusso netto attraverso il taglio (A, B).

$$val(f) = \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e)$$

flusso netto = 10 + 5 + 10 = 25



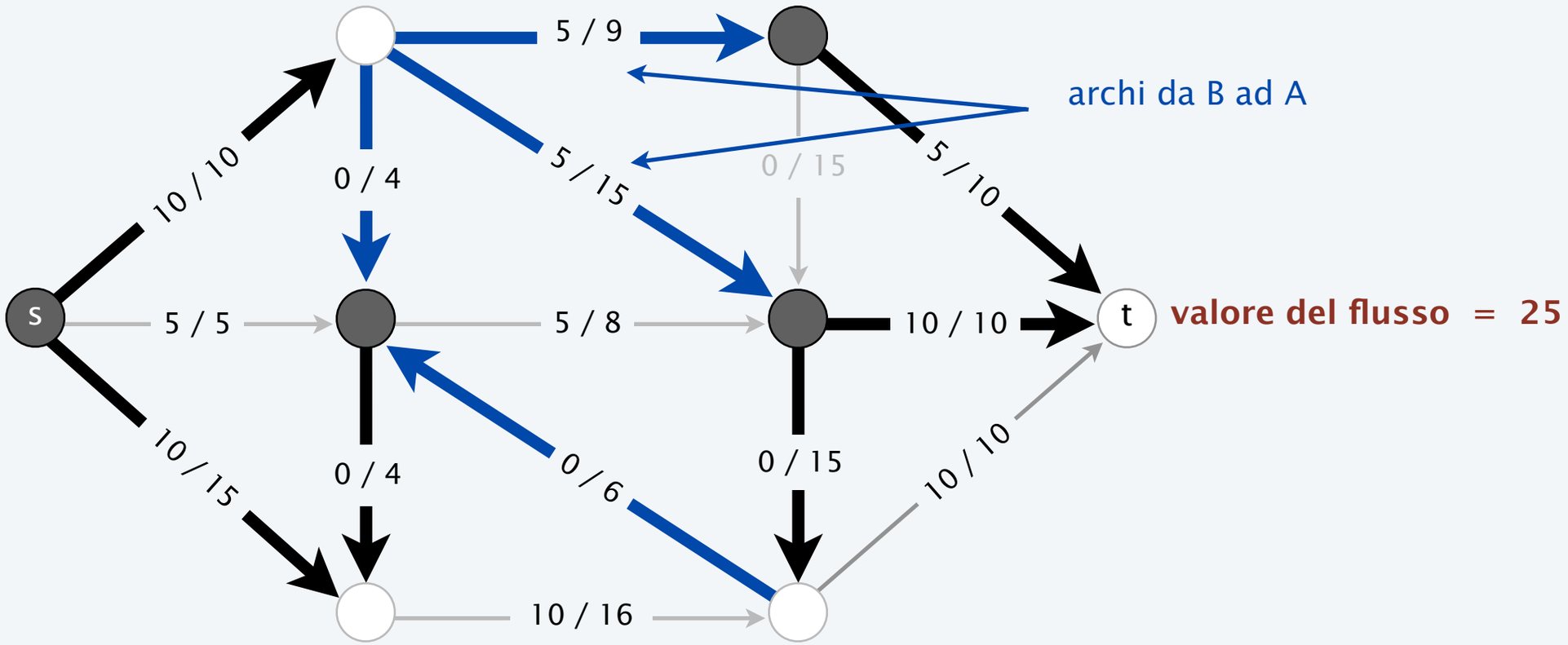
valore del flusso = 25

Relazione tra flussi e tagli

Lemma valore del flusso. Sia f un flusso e sia (A, B) un taglio. Allora, il valore del flusso f uguaglia il flusso netto attraverso il taglio (A, B).

$$val(f) = \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e)$$

flusso netto = (10 + 10 + 5 + 10 + 0 + 0) – (5 + 5 + 0 + 0) = 25



Flussi di rete: quiz 3

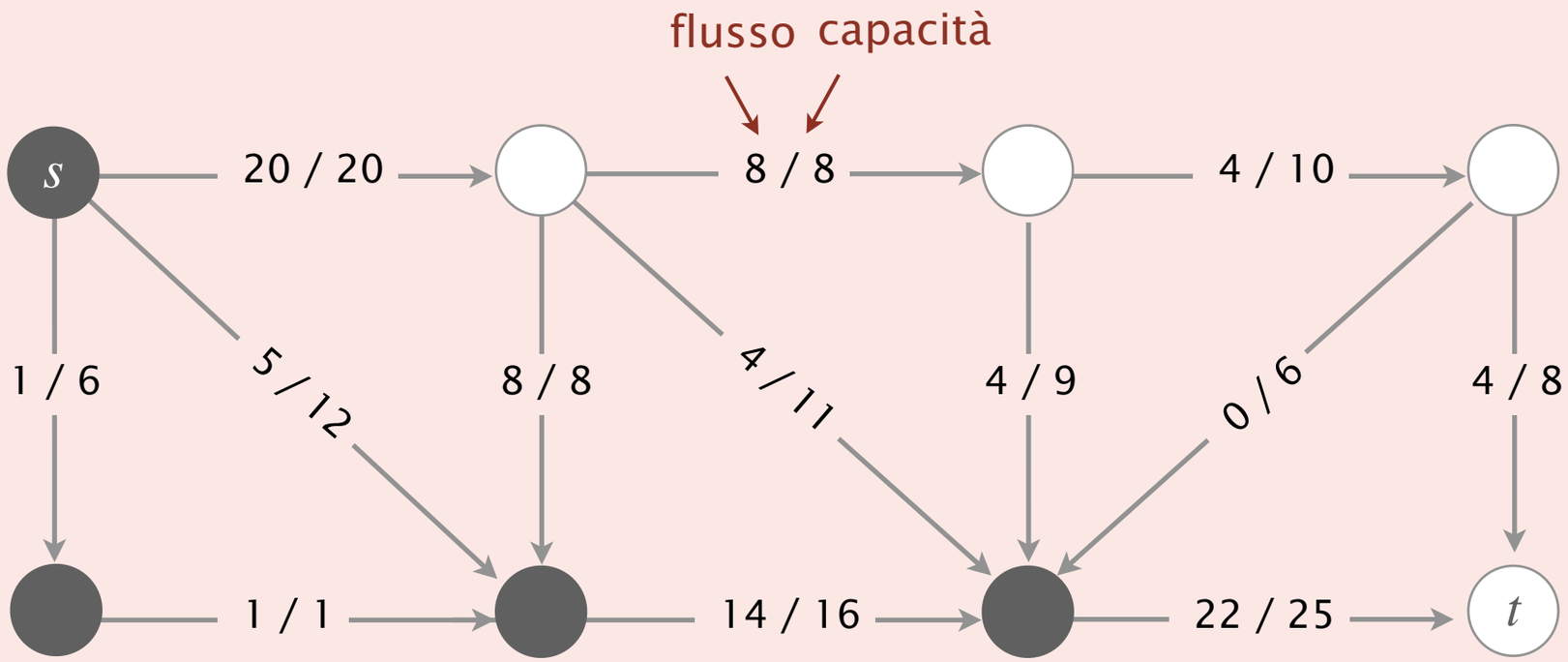
Quanto vale il flusso netto attraverso il taglio in figura?

A. 11 (20 + 25 – 8 – 11 – 9 – 6)

B. 26 (20 + 22 – 8 – 4 – 4)

C. 42 (20 + 22)

D. 45 (20 + 25)



Relazione tra flussi e tagli

Lemma valore del flusso. Sia f un flusso e sia (A, B) un taglio. Allora, il valore del flusso f uguaglia il flusso netto attraverso il taglio (A, B).

$$val(f) = \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e)$$

Dim.

$$val(f) = \sum_{e \text{ out of } s} f(e) - \sum_{e \text{ in to } s} f(e)$$

per la conservazione del flusso, tutti i termini eccetto che $v = s$ sono 0 →

$$= \sum_{v \in A} \left(\sum_{e \text{ out of } v} f(e) - \sum_{e \text{ in to } v} f(e) \right)$$

$$= \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e) \quad \blacksquare$$

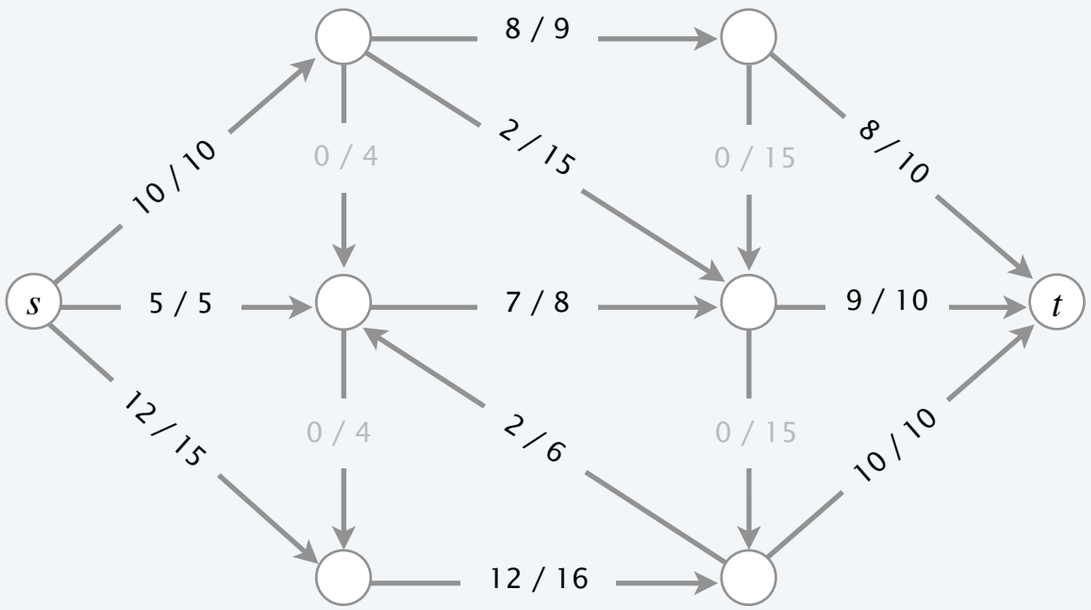
Relazione tra flussi e tagli

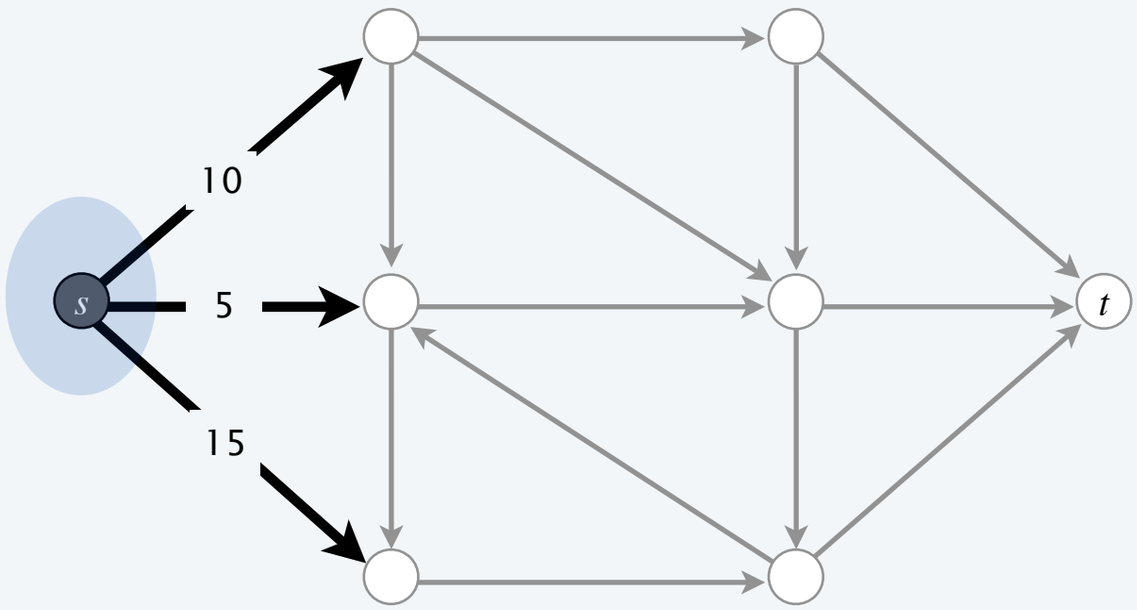
Dualità debole. Sia f un flusso e (A, B) un taglio. Allora $val(f) \leq cap(A, B)$.

Dim.

$$\begin{aligned} val(f) &= \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e) \\ &\leq \sum_{e \text{ out of } A} f(e) \\ &\leq \sum_{e \text{ out of } A} c(e) \\ &= cap(A, B) \quad \blacksquare \end{aligned}$$

lemma valore del flusso





valore del flusso = 27 ≤ **capacità del taglio = 30**

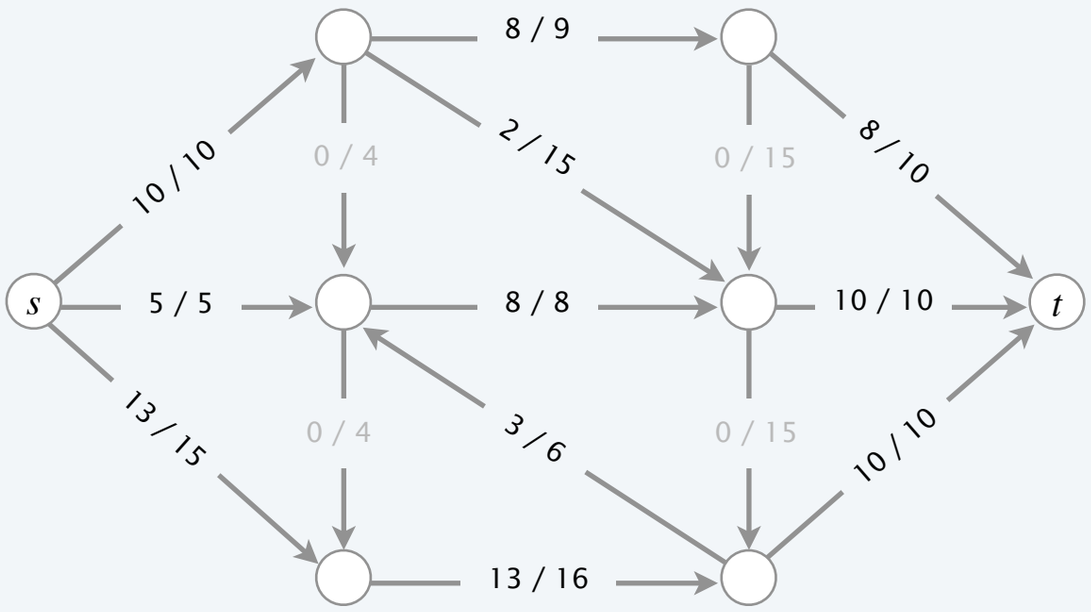
Certificato di ottimalità

Corollario. Sia f un flusso e sia (A, B) un taglio.

Se $val(f) = cap(A, B)$, allora f è un flusso massimo e (A, B) è un taglio minimo.

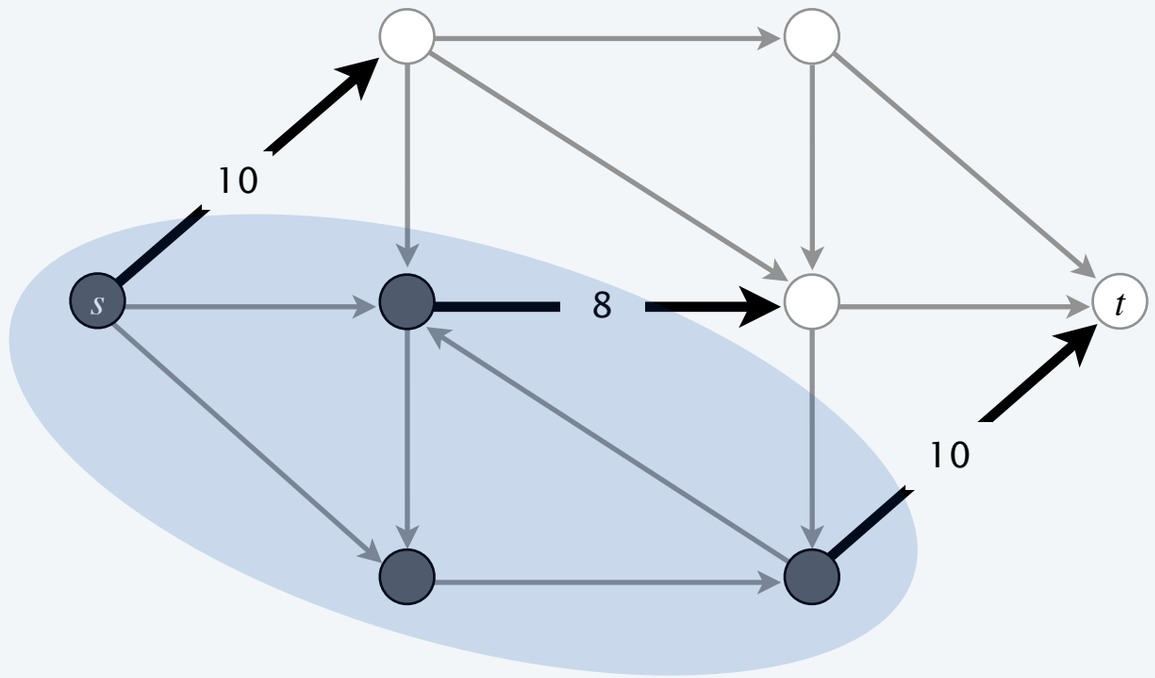
Dim.

- Per ogni flusso f': $val(f') \leq cap(A, B) = val(f)$. (dualità debole)
- Per ogni taglio (A', B'): $cap(A', B') \geq val(f) = cap(A, B)$. ▪ (dualità debole)



valore del flusso = 28

=



capacità del taglio = 28

Teorema del massimo flusso – minimo taglio

Teorema massimo flusso – minimo taglio. Valore del flusso massimo = capacità del taglio minimo.

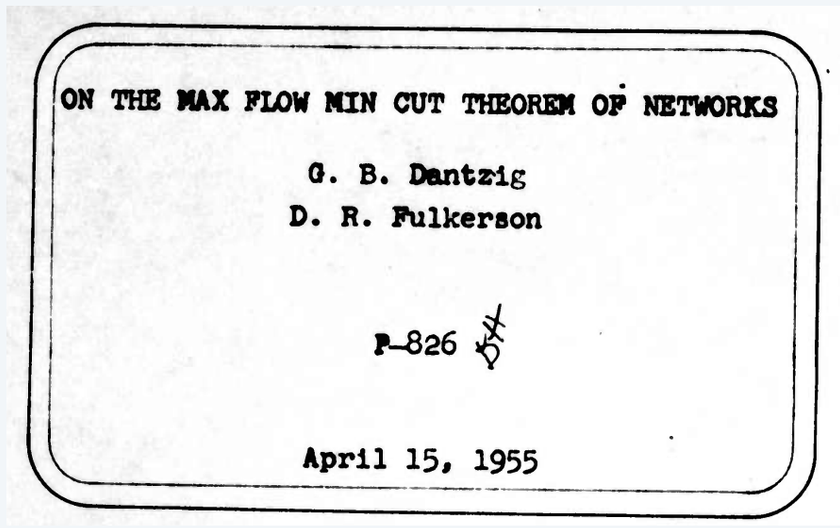
↑ dualità forte

MAXIMAL FLOW THROUGH A NETWORK

L. R. FORD, JR. AND D. R. FULKERSON

Introduction. The problem discussed in this paper was formulated by T. Harris as follows:

"Consider a rail network connecting two cities by way of a number of intermediate cities, where each link of the network has a number assigned to it representing its capacity. Assuming a steady state condition, find a maximal flow from one given city to the other."



ON THE MAX FLOW MIN CUT THEOREM OF NETWORKS

G. B. Dantzig
D. R. Fulkerson

P-826

April 15, 1955

A Note on the Maximum Flow Through a Network*

P. ELIAS†, A. FEINSTEIN‡, AND C. E. SHANNON§

Summary—This note discusses the problem of maximizing the rate of flow from one terminal to another, through a network which consists of a number of branches, each of which has a limited capacity. The main result is a theorem: The maximum possible flow from left to right through a network is equal to the minimum value among all simple cut-sets. This theorem is applied to solve a more general problem, in which a number of input nodes and a number of output nodes are used.

from one terminal to the other in the original network passes through at least one branch in the cut-set. In the network above, some examples of cut-sets are (d, e, f), and (b, c, e, g, h), (d, g, h, i). By a *simple cut-set* we will mean a cut-set such that if any branch is omitted it is no longer a cut-set. Thus (d, e, f) and (b, c, e, g, h) are simple cut-sets while (d, g, h, i) is not. When a simple cut-set is

Teorema del massimo flusso – minimo taglio

Teorema massimo flusso – minimo taglio. Valore del flusso massimo = capacità del taglio minimo.

Teorema del cammino aumentante. Un flusso f è un flusso massimo sse non esistono cammini aumentanti.

Dim. Le seguenti tre condizioni sono equivalenti per qualunque flusso f:

i. Esiste un taglio (A, B) tale che $cap(A, B) = val(f)$.

ii. f è un flusso massimo.

iii. Non esiste un cammino aumentante rispetto ad f. ← se Ford–Fulkerson termina, allora f è un flusso massimo

[i $\Rightarrow$ ii]

- È il corollario già visto della dualità debole. ▪

Teorema del massimo flusso – minimo taglio

Teorema massimo flusso – minimo taglio. Valore del flusso massimo = capacità del taglio minimo.
Teorema del cammino aumentante. Un flusso f è un flusso massimo sse non esistono cammini aumentanti.

Dim. Le seguenti tre condizioni sono equivalente per qualunque flusso f:

i. Esiste un taglio (A, B) tale che $cap(A, B) = val(f)$.
ii. f è un flusso massimo.
iii. Non esiste un cammino aumentante rispetto ad f.

[ii $\Rightarrow$ iii] Dimostriamo il contrapposto: $\neg$ iii $\Rightarrow \neg$ ii.

- Supponiamo che esista un cammino aumentante rispetto ad f.
- Possiamo migliorare f inviando flusso attraverso tale cammino.
- Quindi, f non è un flusso massimo. ▪

Teorema del massimo flusso – minimo taglio

[iii ⇒ i]

- Sia f un flusso senza cammini aumentanti.
- Sia A = insieme dei nodi raggiungibili da s nella rete residua G_f.
- Per definizione di A: $s \in A$.
- Per definizione del flusso f: $t \notin A$.

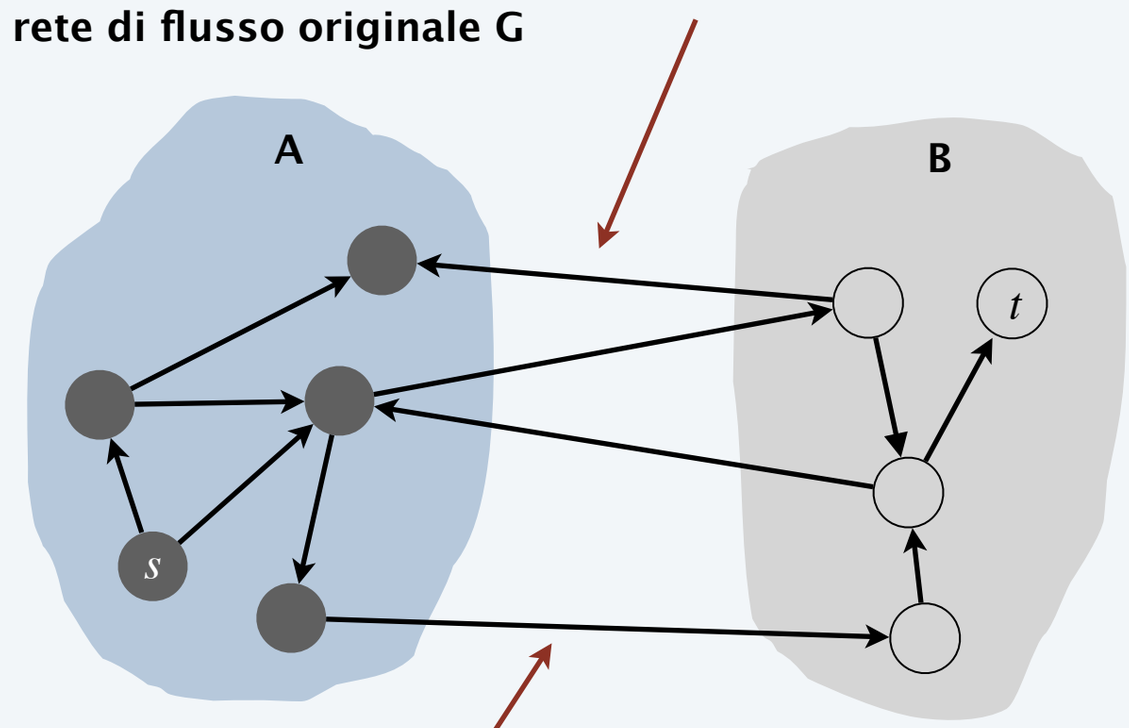
$$val(f) = \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e)$$

lemma valore del flusso

$$= \sum_{e \text{ out of } A} c(e) - 0$$

$$= cap(A, B) \quad \blacksquare$$

un arco $e = (v, w)$ con $v \in B, w \in A$ deve avere $f(e) = 0$



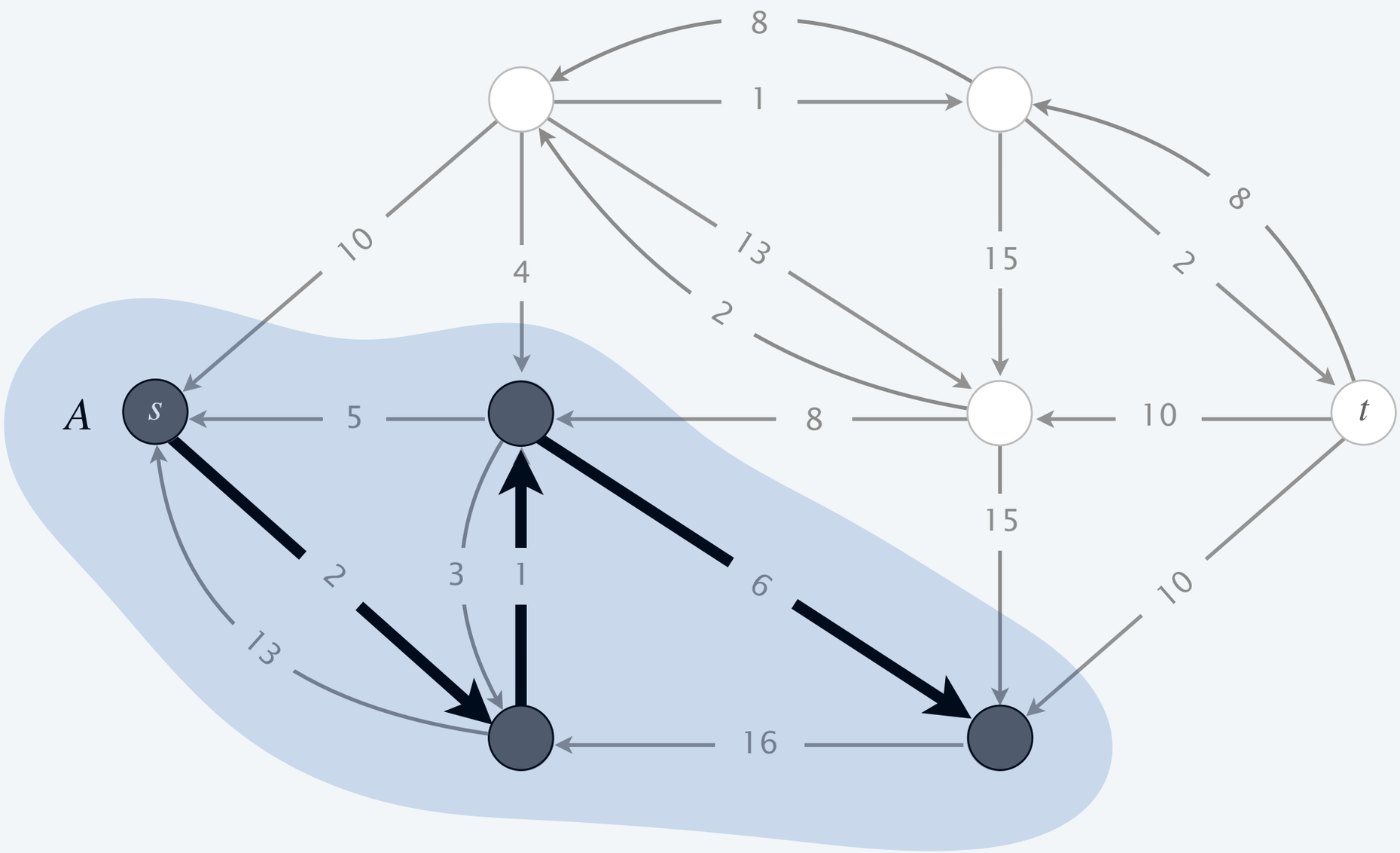
un arco $e = (v, w)$ con $v \in A, w \in B$ deve avere $f(e) = c(e)$

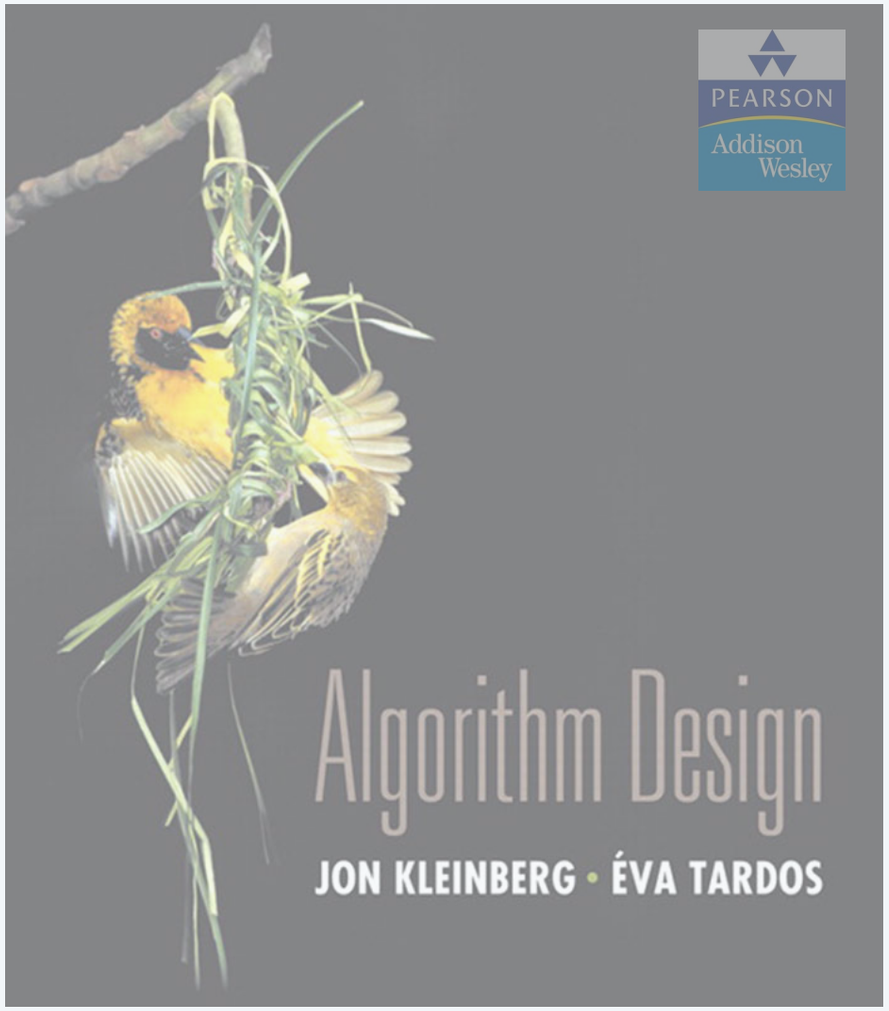
Calcolare un taglio minimo a partire da un flusso massimo

Teorema. Dato un flusso massimo f, si può calcolare un taglio minimo (A, B) in tempo $O(m)$.

Dim. Sia A = insieme di nodi raggiungibili da s nella rete residua G_f. ▪

ragionamento della slide precedente implica che capacità di (A, B) = valore del flusso f





SECTION 7.3

7. NETWORK FLOW I

- *max-flow and min-cut problems*
- *Ford–Fulkerson algorithm*
- *max-flow min-cut theorem*
- *algoritmo di scaling di capacità*
- *shortest augmenting paths*
- *Dinitz' algorithm*
- *simple unit-capacity networks*

Analisi di Ford–Fulkerson (quando le capacità sono intere)

Assunzione. Ogni capacità di arco $c(e)$ è un intero tra 1 e C.

Invariante di interezza. Durante Ford–Fulkerson, ogni flusso di arco $f(e)$ ed ogni capacità residua $c_f(e)$ sono numeri interi.

Dim. Per induzione sul numero di cammini aumentanti. ▪

si consideri il taglio $A = \{ s \}$ (assumiamo assenza di archi paralleli)

Teorema. Ford–Fulkerson termina dopo al più $val(f^*) \leq n C$ cammini aumentanti, dove f^* è un flusso massimo.

Dim. Ogni aumento incrementa il valore del flusso di almeno 1. ▪

Corollario. Il tempo di esecuzione di Ford–Fulkerson è $O(m n C)$.

Dim. Con una BFS o DFS troviamo un cammino aumentante in tempo $O(m)$. ▪

$f(e)$ è intero per ogni e

Teorema di interezza. Esiste un flusso massimo intero f^*.

Dim. Poiché Ford–Fulkerson termina, il teorema segue dall'invariante di interezza (e dal teorema del cammino aumentante). ▪

Ford–Fulkerson: esempio esponenziale

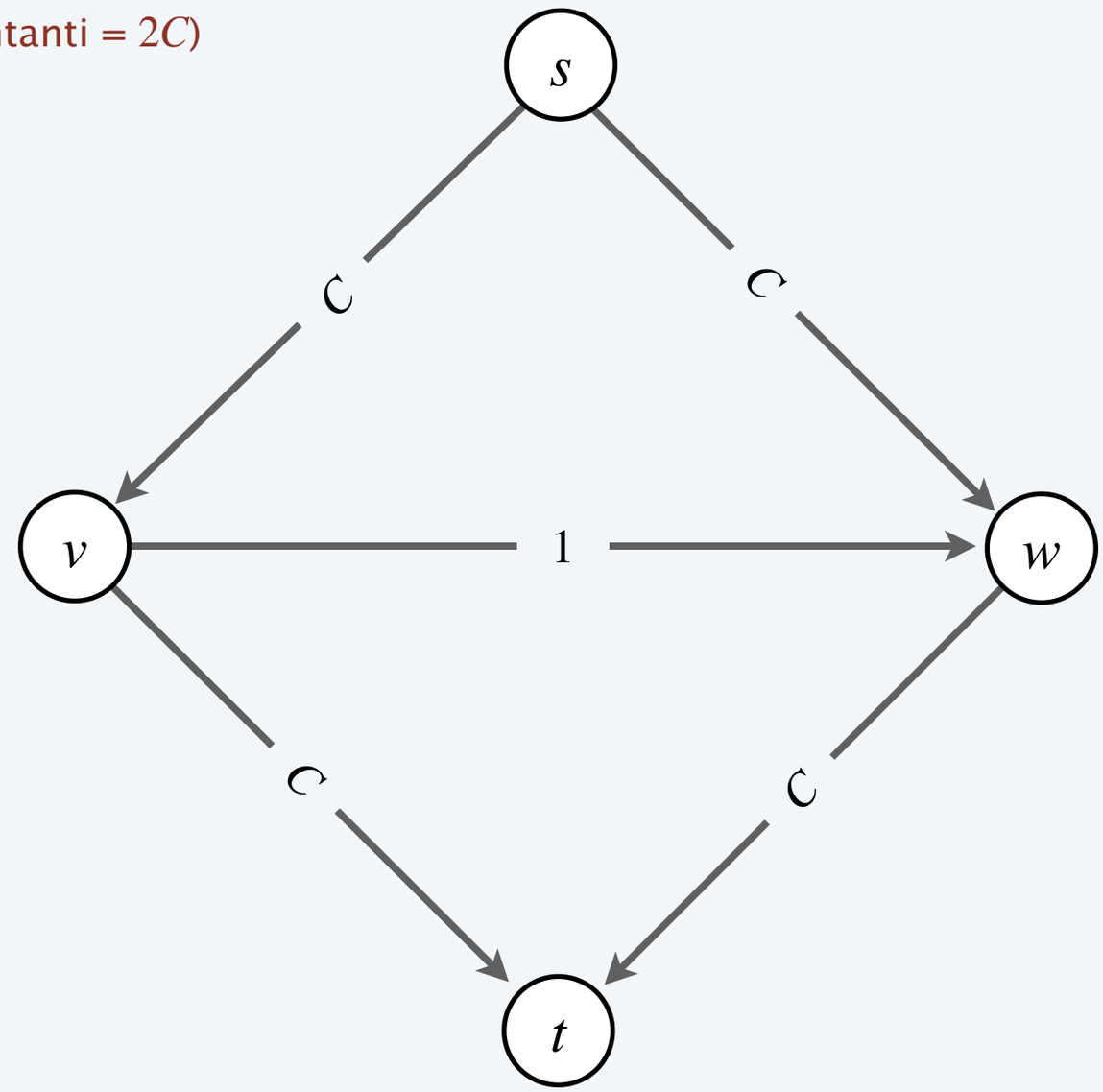
D. L'algoritmo generico Ford–Fulkerson è polinomiale nella taglia dell'input?

m, n, e $\log C$

R. No. Se la capacità massima è C, l'algoritmo può richiedere $\geq C$ iterazioni.

- $s \rightarrow v \rightarrow w \rightarrow t$
- $s \rightarrow w \rightarrow v \rightarrow t$
- $s \rightarrow v \rightarrow w \rightarrow t$
- $s \rightarrow w \rightarrow v \rightarrow t$
- …
- $s \rightarrow v \rightarrow w \rightarrow t$
- $s \rightarrow w \rightarrow v \rightarrow t$

ogni cammino aumentante invia solo 1 unità di flusso (# cammini aumentanti = $2C$)



Flussi di rete: quiz 4

L'algoritmo Ford-Fulkerson di sicuro termina se le capacità degli archi sono…

A. Numeri razionali.

B. Numeri reali.

C. Sia A che B.

D. Né A né B.

Scegliere buoni cammini aumentanti

La selezione dei cammini aumentanti richiede accortezza.

- Alcune scelte portano ad algoritmi esponenziali.
- Scelte intelligenti portano ad algoritmi polinomiali.

Patologia. Se le capacità possono essere irrazionali, non vi è garanzia che Ford–Fulkerson termini (o che converga al flusso massimo)!

Obiettivo. Scegliere i cammini aumentanti in modo che:

- I cammini aumentanti possano essere trovati in modo efficiente.
- Le iterazioni dell'algoritmo siano poche.

Scegliere buoni cammini aumentanti

Scegliere cammini aumentanti caratterizzati da:

- Massima capacità di strozzatura ("spessi").
- Capacità di strozzatura abbastanza grande. ← ora
- Pochi archi.

Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems

JACK EDMONDS

University of Waterloo, Waterloo, Ontario, Canada

AND

RICHARD M. KARP

University of California, Berkeley, California

ABSTRACT. This paper presents new algorithms for the maximum flow problem, the Hitchcock transportation problem, and the general minimum-cost flow problem. Upper bounds on the numbers of steps in these algorithms are derived, and are shown to compare favorably with upper bounds on the numbers of steps required by earlier algorithms.

Edmonds-Karp 1972 (USA)

Dokl. Akad. Nauk SSSR
Tom 194 (1970), No. 4

Soviet Math. Dokl.
Vol. 11 (1970), No. 5

ALGORITHM FOR SOLUTION OF A PROBLEM OF MAXIMUM FLOW IN A NETWORK WITH POWER ESTIMATION

UDC 518.5

E. A. DINIC

Different variants of the formulation of the problem of maximal stationary flow in a network and its many applications are given in [1]. There also is given an algorithm solving the problem in the case where the initial data are integers (or, what is equivalent, commensurable). In the general case this algorithm requires preliminary rounding off of the initial data, i.e. only an approximate solution of the problem is possible. In this connection the rapidity of convergence of the algorithm is inversely proportional to the relative precision.

Dinitz 1970 (Unione Sovietica)

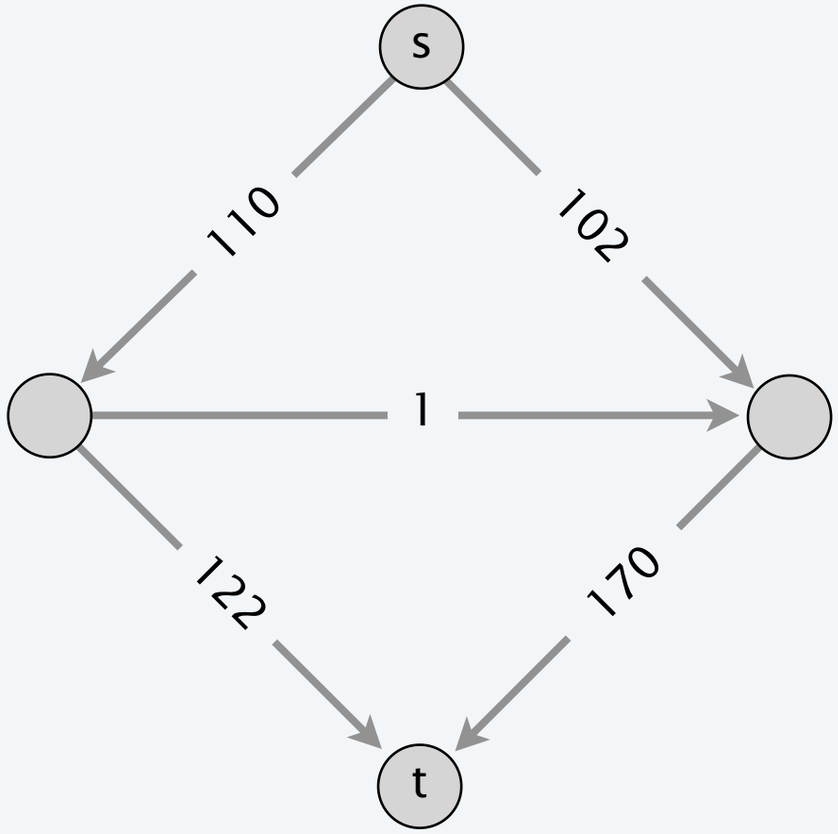
inventato in risposta ad un esercizio
posto a lezione da Adel'son-Vel'skiĭ

Algoritmo di scaling di capacità

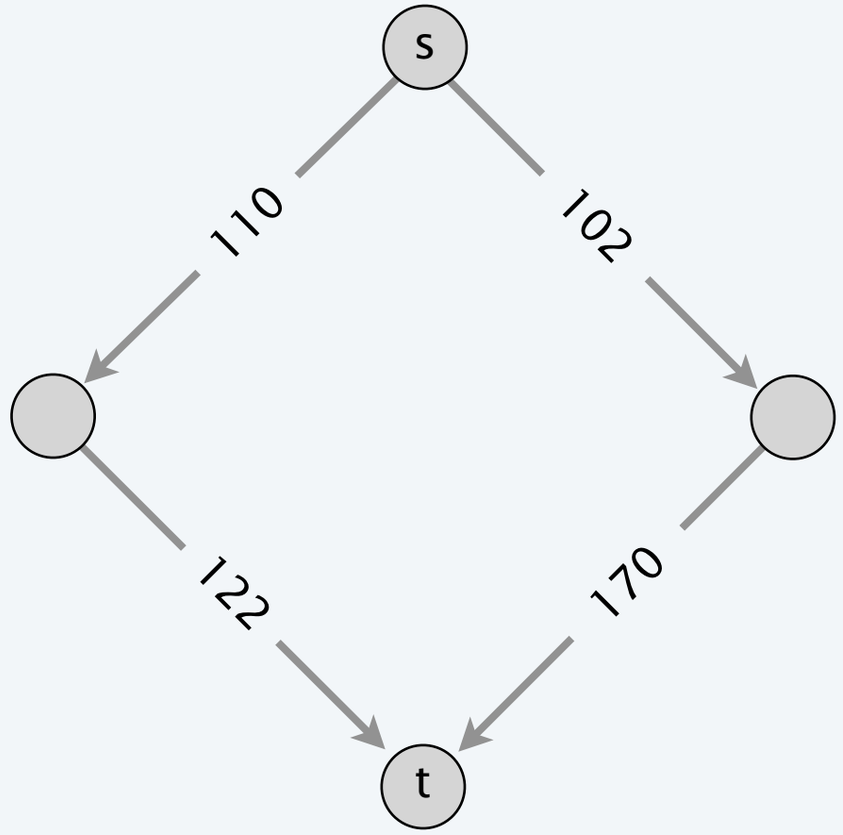
Panoramica. Scegli cammini aumentanti con capacità di strozzatura "grande".

- Mantieni un parametro di scaling Δ.
- Sia $G_f(\Delta)$ la parte della rete residua che contiene solo gli archi di capacità $\geq \Delta$.
- Ogni cammino aumentante in $G_f(\Delta)$ ha capacità di strozzatura $\geq \Delta$.

ma non necessariamente massima



G_f



$G_f(\Delta)$, $\Delta = 100$

Algoritmo di scaling di capacità

CAPACITY-SCALING(G)

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

$\Delta \leftarrow$ massima potenza di 2 $\leq C$.

WHILE ($\Delta \geq 1$)

  $G_f(\Delta) \leftarrow$ rete Δ-residua di G rispetto al flusso f .

  WHILE (esiste un cammino $s \rightsquigarrow t$ P in $G_f(\Delta)$)

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

    Aggiorna $G_f(\Delta)$.

  $\Delta \leftarrow \Delta / 2$.

fase di Δ-scaling

RETURN f.

Algoritmo di scaling di capacità: dimostrazione di correttezza

Assunzione. Tutte le capacità degli archi sono interi tra 1 e C.

Invariante. Il parametro di scaling Δ è una potenza di 2.

Dim. Inizialmente è una potenza di 2; ogni fase divide Δ per 2. ▪

Invariante di interezza. Per tutto l'algoritmo, ogni flusso di arco $f(e)$ e capacità residua $c_f(e)$ sono interi.

Dim. Identica al caso di Ford–Fulkerson generico. ▪

Teorema. Se l'algoritmo di scaling di capacità termina, f è un flusso massimo.

Dim.

- Per l'invariante di interezza, quando $\Delta = 1 \Rightarrow G_f(\Delta) = G_f$.
- Al termine della fase $\Delta = 1$, non ci sono cammini aumentanti.
- Il risultato segue dal teorema del cammino aumentante. ▪

Algoritmo di scaling di capacità: analisi del tempo di esecuzione

Lemma 1. Ci sono $1 + \lfloor \log_2 C \rfloor$ fasi di scaling.

Dim. Inizialmente $C/2 < \Delta \leq C$; Δ diminuisce di un fattore 2 ad ogni fase. ▪

Lemma 2. Sia f il flusso al termine di una fase di Δ-scaling.

Allora il valore del flusso massimo è $\leq val(f) + m\,\Delta$.

Dim. Prossima slide.

Lemma 3. Ci sono $\leq 2m$ aumenti del flusso in ogni fase di scaling.

Dim.

o equivalentemente, alla fine di una fase di 2Δ-scaling

- Sia f il flusso all'inizio della fase di Δ-scaling.
- Lemma 2 $\Rightarrow$ valore del massimo flusso $\leq val(f) + m\,(2\,\Delta)$.
- Ogni aumento in una Δ-fase incrementa $val(f)$ di almeno Δ. ▪

Teorema. L'algoritmo di scaling di capacità prende tempo $O(m^2 \log C)$.

Dim.

- Lemma 1 + Lemma 3 $\Rightarrow$ $O(m \log C)$ aumenti di flusso.
- Trovare un cammino aumentante richiede tempo $O(m)$. ▪

Algoritmo di scaling di capacità: analisi del tempo di esecuzione

Lemma 2. Sia f il flusso al termine di una fase di Δ-scaling.

Allora il valore del massimo flusso è $\leq val(f) + m\,\Delta$.

Dim.

- Mostriamo che esiste un taglio (A, B) tale che $cap(A, B) \leq val(f) + m\,\Delta$.
- Sia A l'insieme dei nodi raggiungibili da s in $G_f(\Delta)$.
- Per definizione di A: $s \in A$.
- Per definizione di f: $t \notin A$.

$$val(f) = \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e)$$

lemma valore del flusso

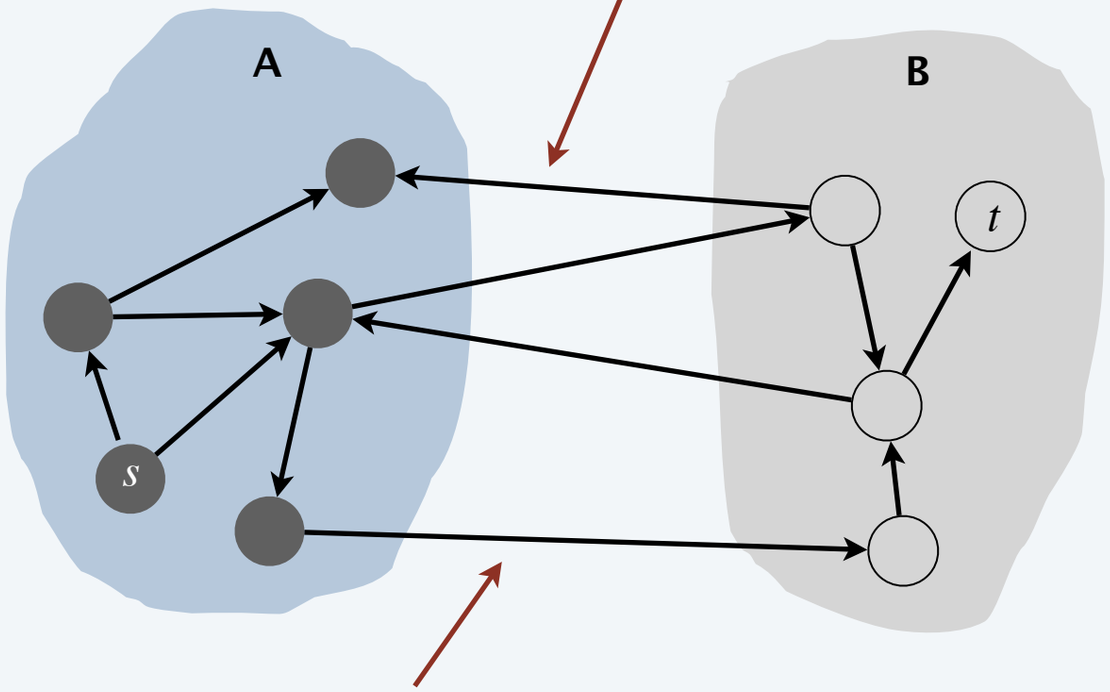
$$\geq \sum_{e \text{ out of } A} (c(e) - \Delta) - \sum_{e \text{ in to } A} \Delta$$

$$\geq \sum_{e \text{ out of } A} c(e) - \sum_{e \text{ out of } A} \Delta - \sum_{e \text{ in to } A} \Delta$$

$$\geq cap(A, B) - m\Delta \quad \blacksquare$$

un arco $e = (v, w)$ con $v \in B, w \in A$
deve avere $f(e) < \Delta$

rete di flusso originale



un arco $e = (v, w)$ con $v \in A, w \in B$
deve avere $f(e) > c(e) - \Delta$

Panoramica degli algoritmi basati su cammini aumentanti

anno	metodo	# aumenti	tempo di esecuzione	
1955	**cammino aumentante**	$n\,C$	$O(m\,n\,C)$	
1972	**cammino più "spesso"**	$m \log (mC)$	$O(m^2 \log n \log (mC))$	cammini "spessi"
1972	**scaling di capacità**	$m \log C$	$O(m^2 \log C)$	
1985	**scaling di capacità migliorato**	$m \log C$	$O(m\,n \log C)$	
1970	**cammino aumentante minimo**	$m\,n$	$O(m^2\,n)$	cammini minimi
1970	**grafo a livelli**	$m\,n$	$O(m\,n^2)$	
1983	**alberi dinamici**	$m\,n$	$O(m\,n \log n)$	

augmenting-path algorithms with m edges, n nodes, and integer capacities between 1 and C

Algoritmi per il massimo flusso: risultati teorici

anno	metodo	caso peggiore	scoperto da
1951	**simplex**	$O(m\, n^2\, C)$	Dantzig
1955	**augmenting paths**	$O(m\, n\, C)$	Ford–Fulkerson
1970	**shortest augmenting paths**	$O(m\, n^2)$	Edmonds–Karp, Dinitz
1974	**blocking flows**	$O(n^3)$	Karzanov
1983	**dynamic trees**	$O(m\, n \log n)$	Sleator–Tarjan
1985	**improved capacity scaling**	$O(m\, n \log C)$	Gabow
1988	**push–relabel**	$O(m\, n \log (n^2 / m))$	Goldberg–Tarjan
1998	**binary blocking flows**	$O(m^{3/2} \log (n^2 / m) \log C)$	Goldberg–Rao
2013	**compact networks**	$O(m\, n)$	Orlin
2014	**interior-point methods**	$\tilde{O}(m\, n^{1/2} \log C)$	Lee–Sidford
2016	**electrical flows**	$\tilde{O}(m^{10/7}\, C^{1/7})$	Mądry
20xx			

algoritmi per il massimo flusso con m archi, n nodi, e capacità intere tra 1 e C