

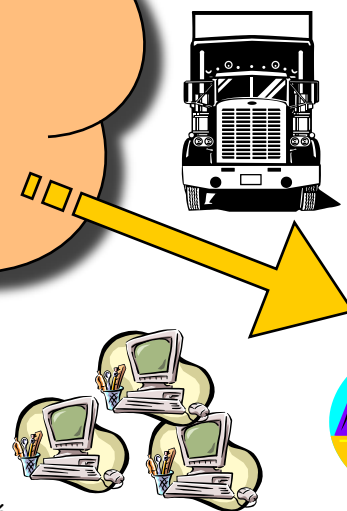
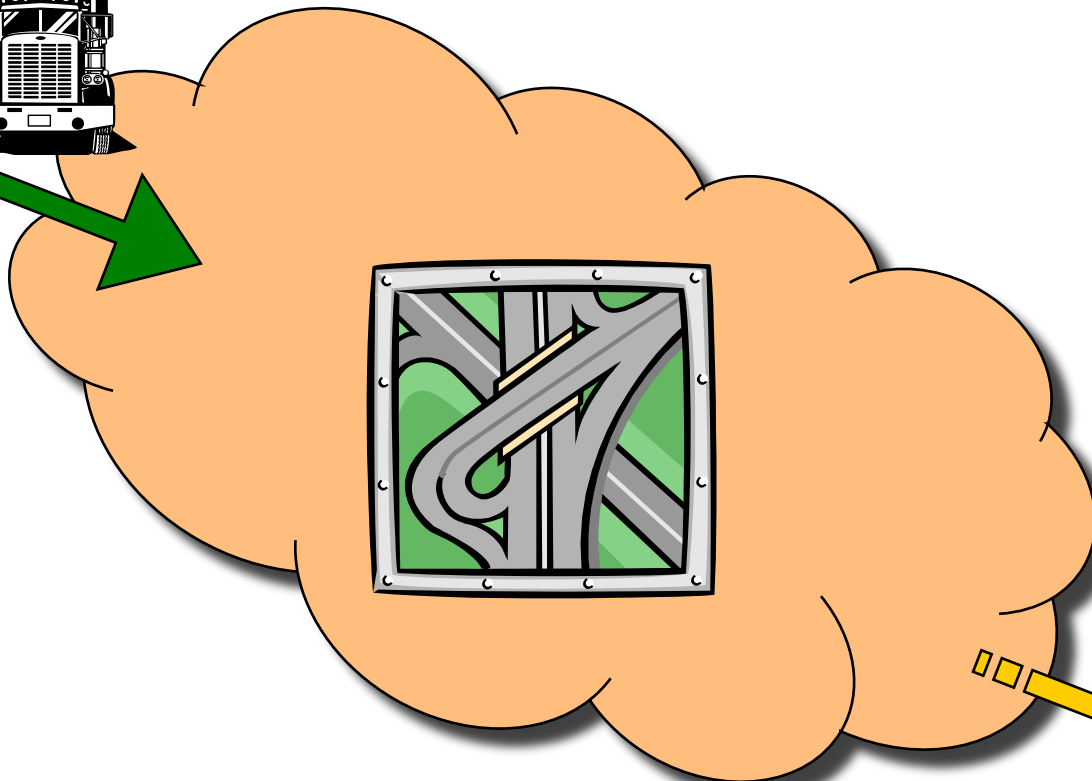
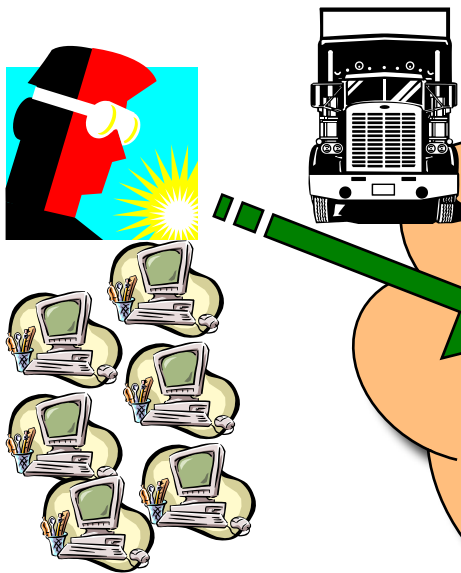
Massimo flusso

Progettazione di Algoritmi a.a. 2015-16

Matricole congrue a 1

Docente: Annalisa De Bonis

Massimizzare il #
di PC prodotti



Descrizione del problema

- Una fabbrica (**sorgente**) di PC deve stabilire il numero di PC da assemblare giornalmente.
- Tutti i PC prodotti verranno venduti in un negozio (**destinazione**).
- La fabbrica ed il negozio sono collegati attraverso una rete di comunicazione.
- Su di ogni tratto della rete è in servizio un furgone che può trasportare un numero fissato di PC (numero che dipende dalla grandezza del furgone).

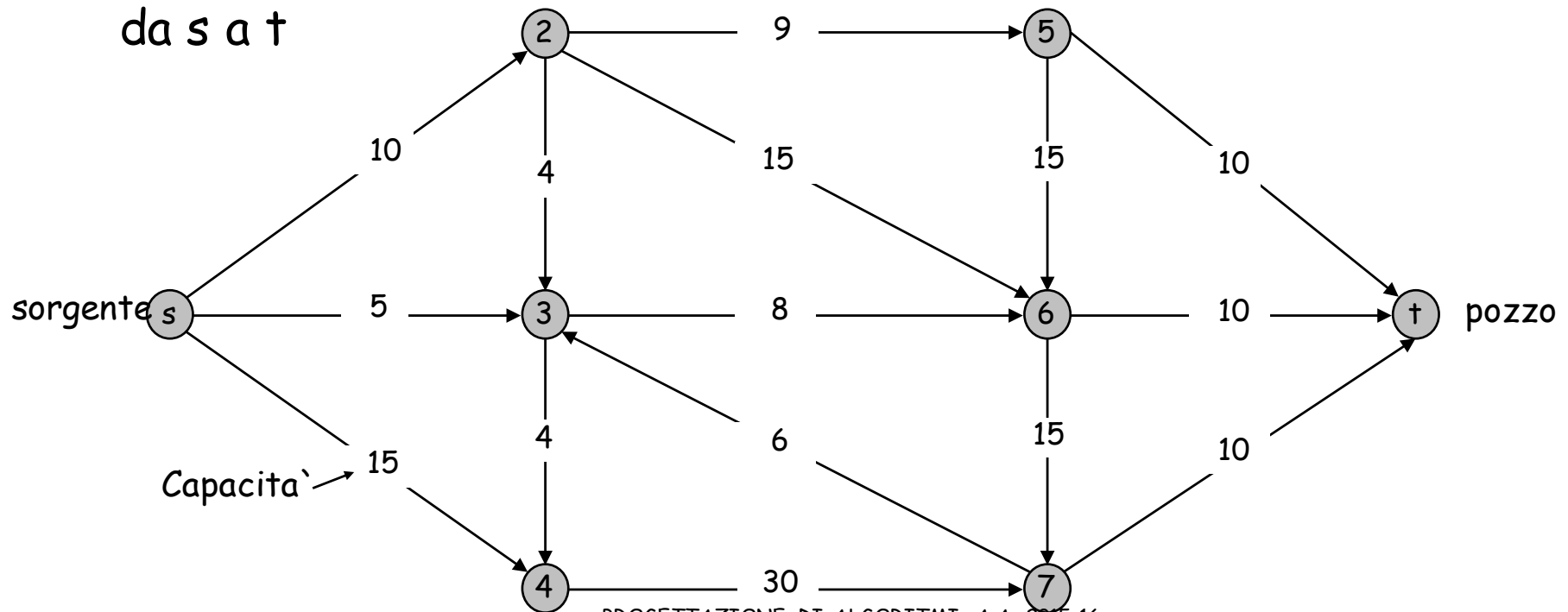
Ulteriori vincoli

- In ogni nodo della rete di comunicazione:
 - Non è possibile produrre PC
 - Non è possibile stoccare PC
- In altre parole ... il numero di PC che entra in un nodo è uguale al numero di PC che esce dal nodo.
- **Obiettivo:** Qual è il maggior numero di PC che può essere trasportato dalla sorgente alla destinazione senza violare i vincoli del problema?

Il problema del minimo taglio

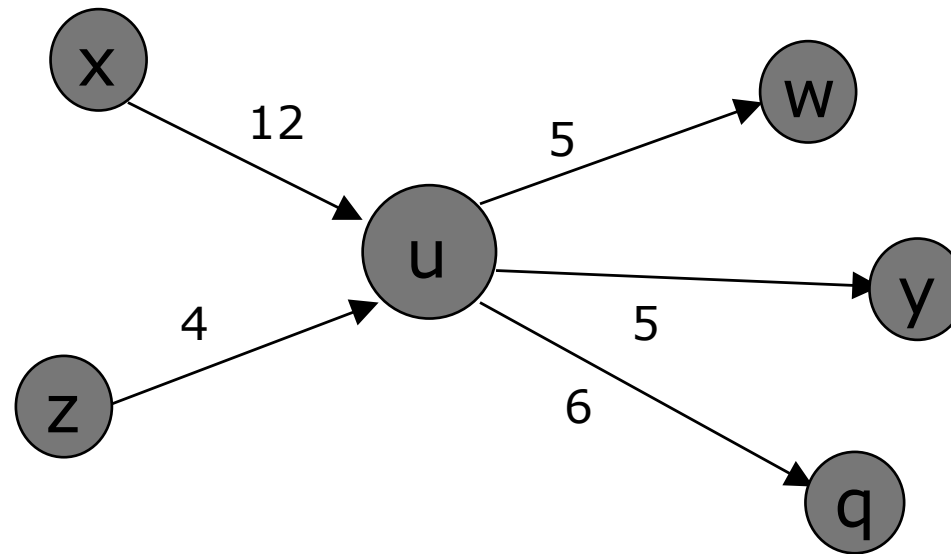
- Rete di flusso

- Gli archi rappresentano condotte attraverso le quali fluisce materiale.
- $G = (V, E)$: grafo direzionato senza archi paralleli (il materiale fluisce in una sola direzione)
- Due nodi speciali sorgente s e pozzo t .
- $c(e) \geq 0$: capacita` dell'arco e
- Assumiamo inoltre che ogni vertice si trovi lungo un percorso da s a t



Conservazione del flusso

$$\begin{aligned}\sum_{v \in V} f(u,v) &= f(u,x) + f(u,z) + f(u,w) + f(u,y) + f(u,q) \\ &= (-4) + (-12) + 5 + 5 + 6 = 0\end{aligned}$$



Valore del flusso

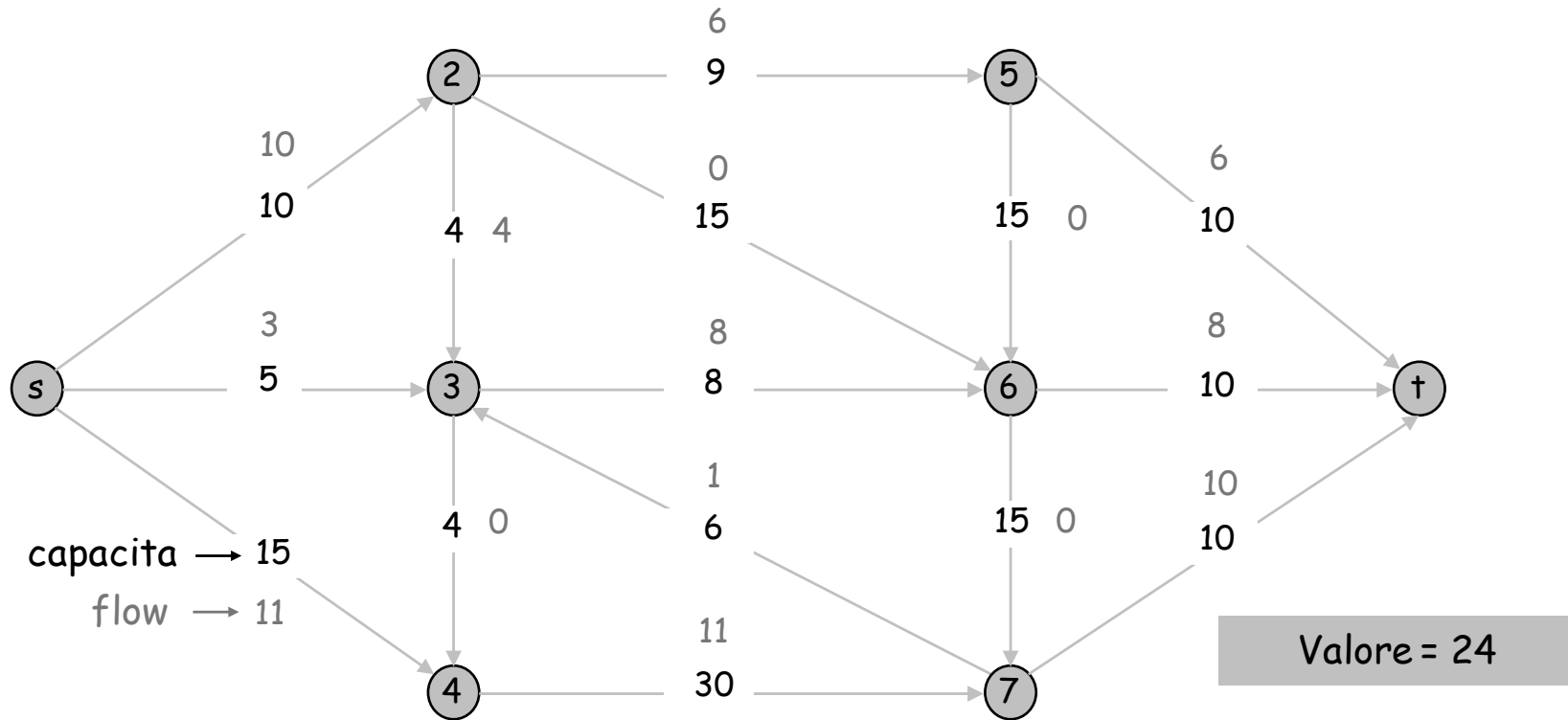
Dato un flusso f di G , il valore del flusso è definito come:

- $v(f) = \sum_{v \in V} f(s, v)$

E' possibile verificare che vale anche

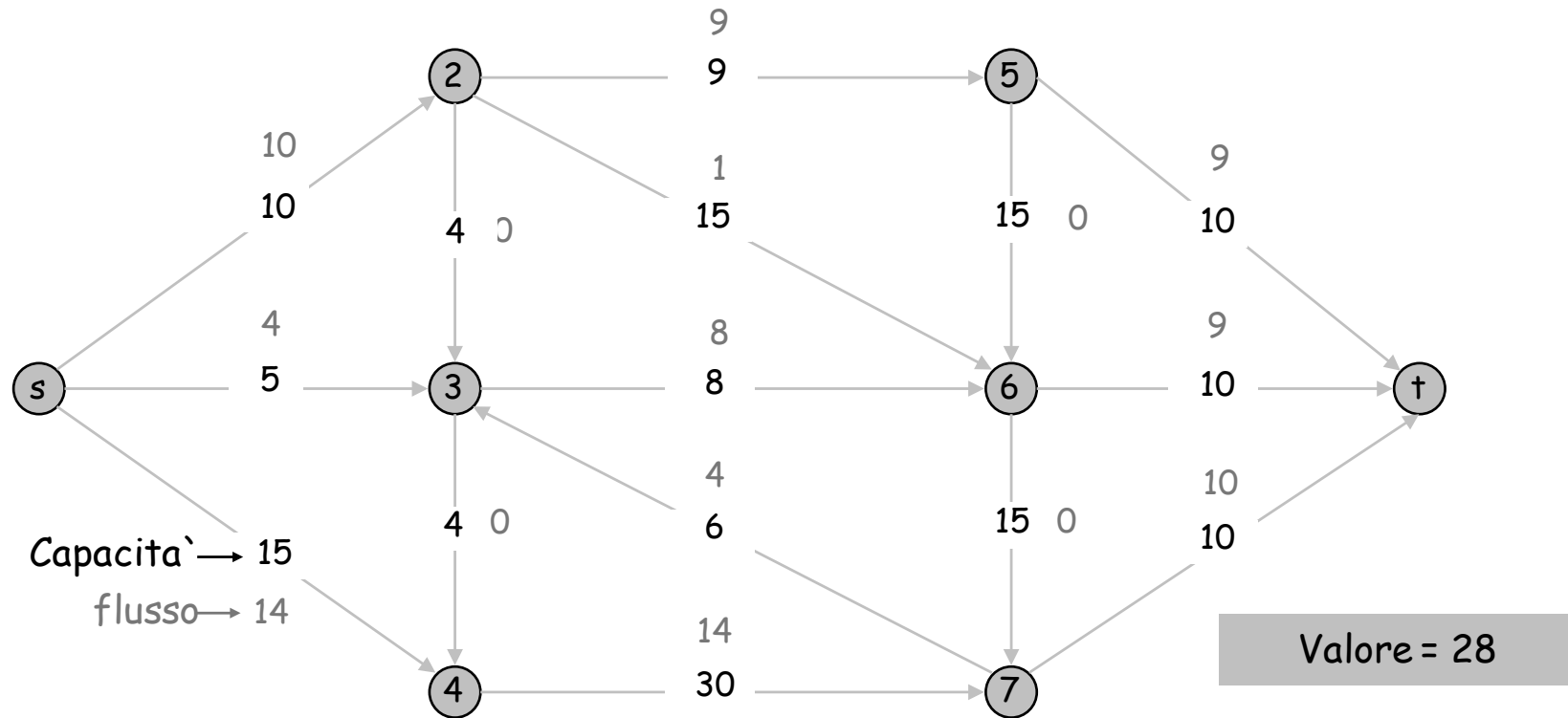
$$v(f) = \sum_{v \in V} f(v, t)$$

Flussi



Il problema del massimo flusso

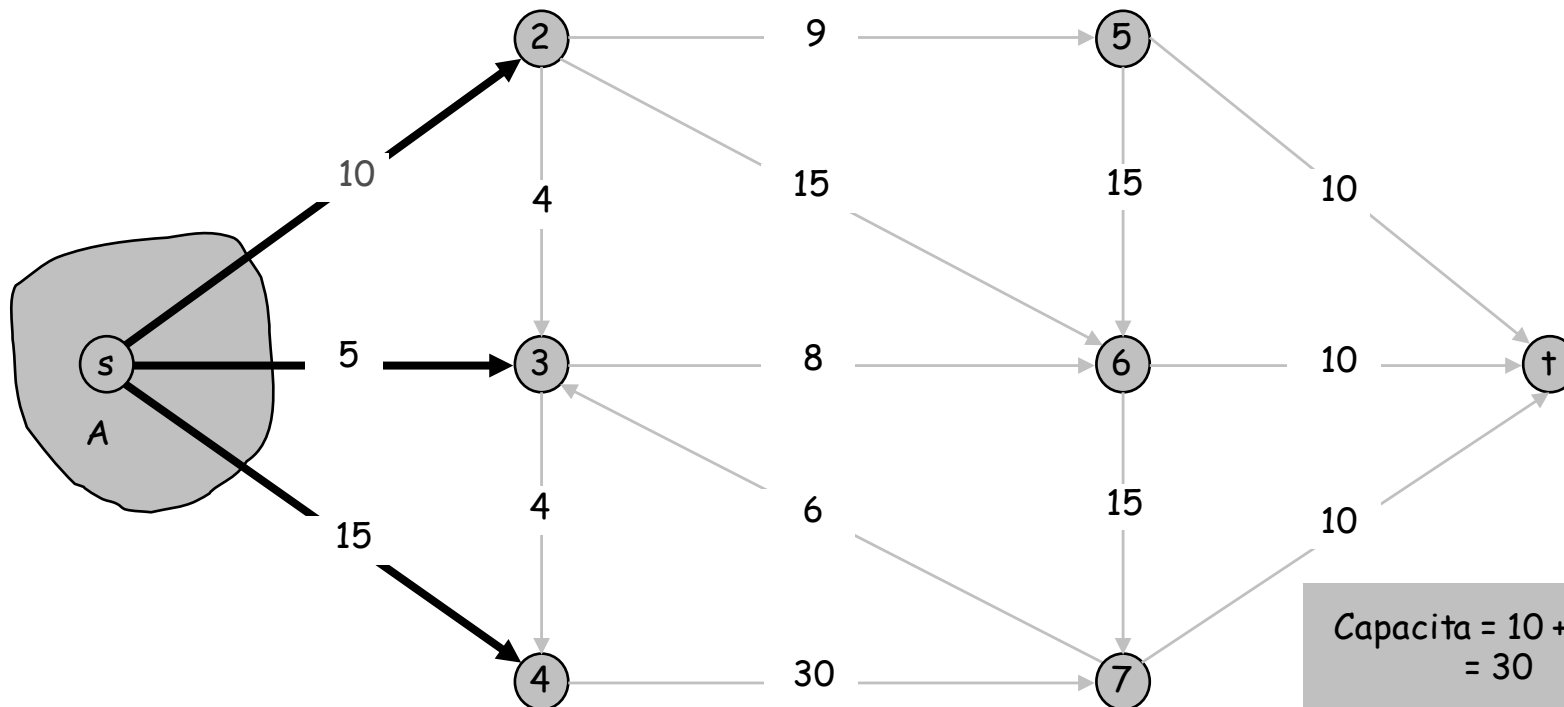
- Il problema del massimo flusso. Trova un flusso da s a t di valore massimo.



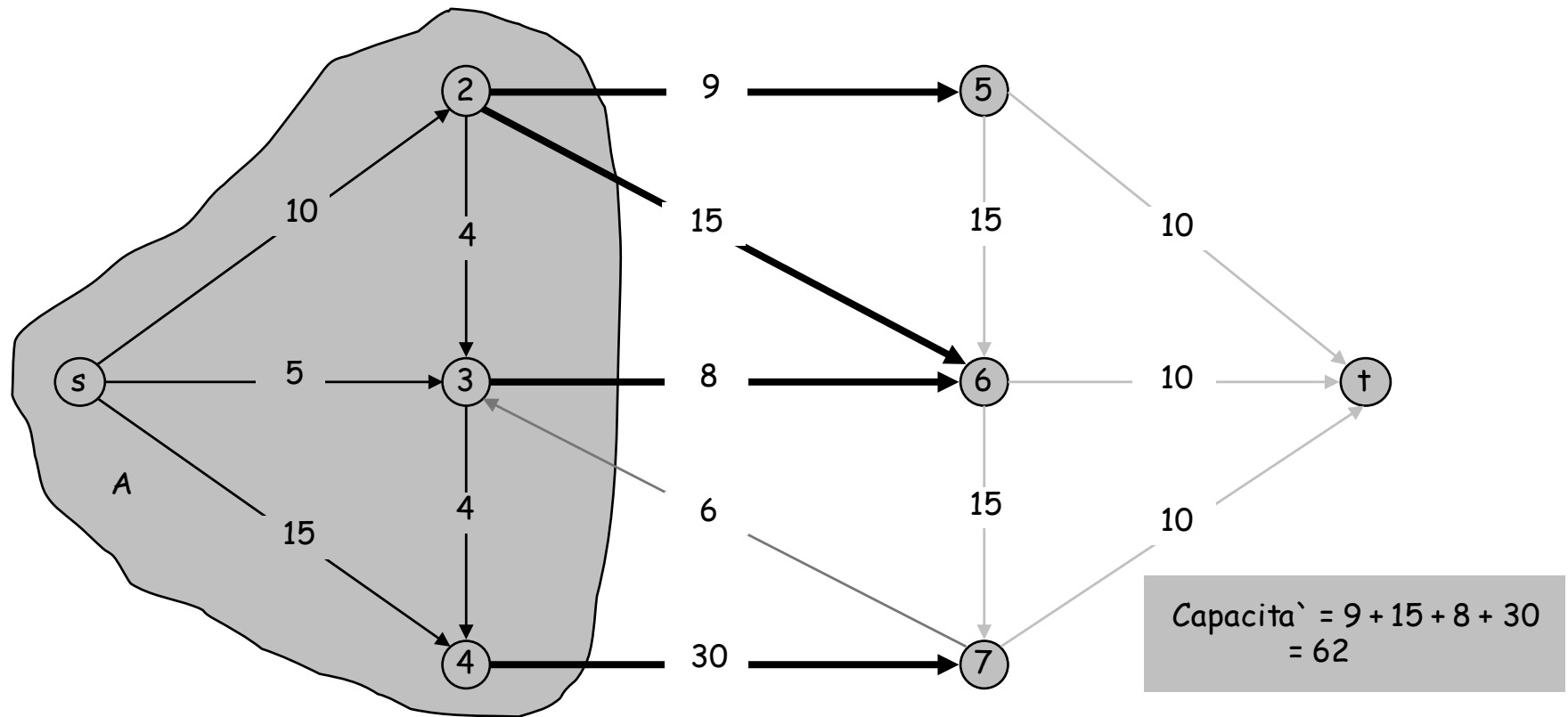
Taglio

• Def. Un **taglio s-t** e' una partizione (A, B) di V con $s \in A$ e $t \in B$.

• Def. La **capacita'** di un taglio (A, B) e': $cap(A, B) = \sum_{e \text{ uscente da } A} c(e)$

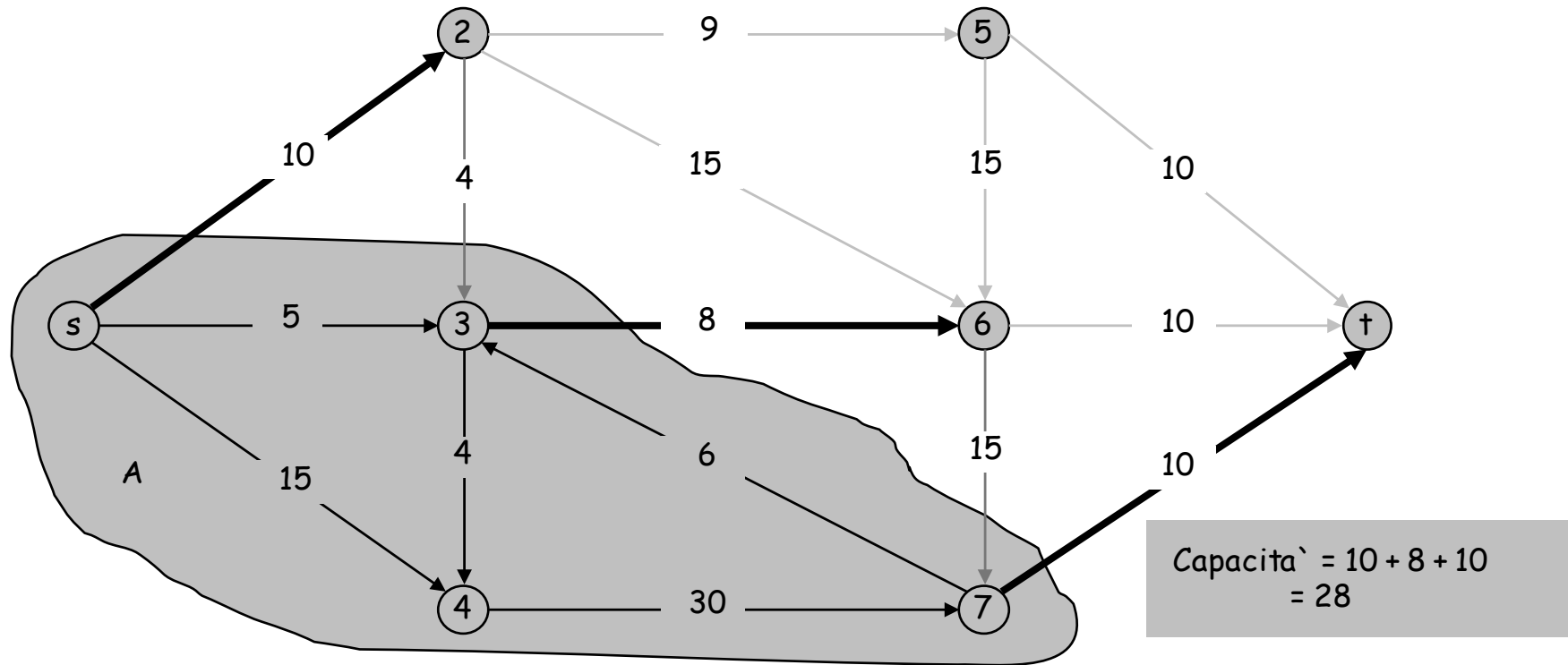


Tagli



Il problema del minimo taglio

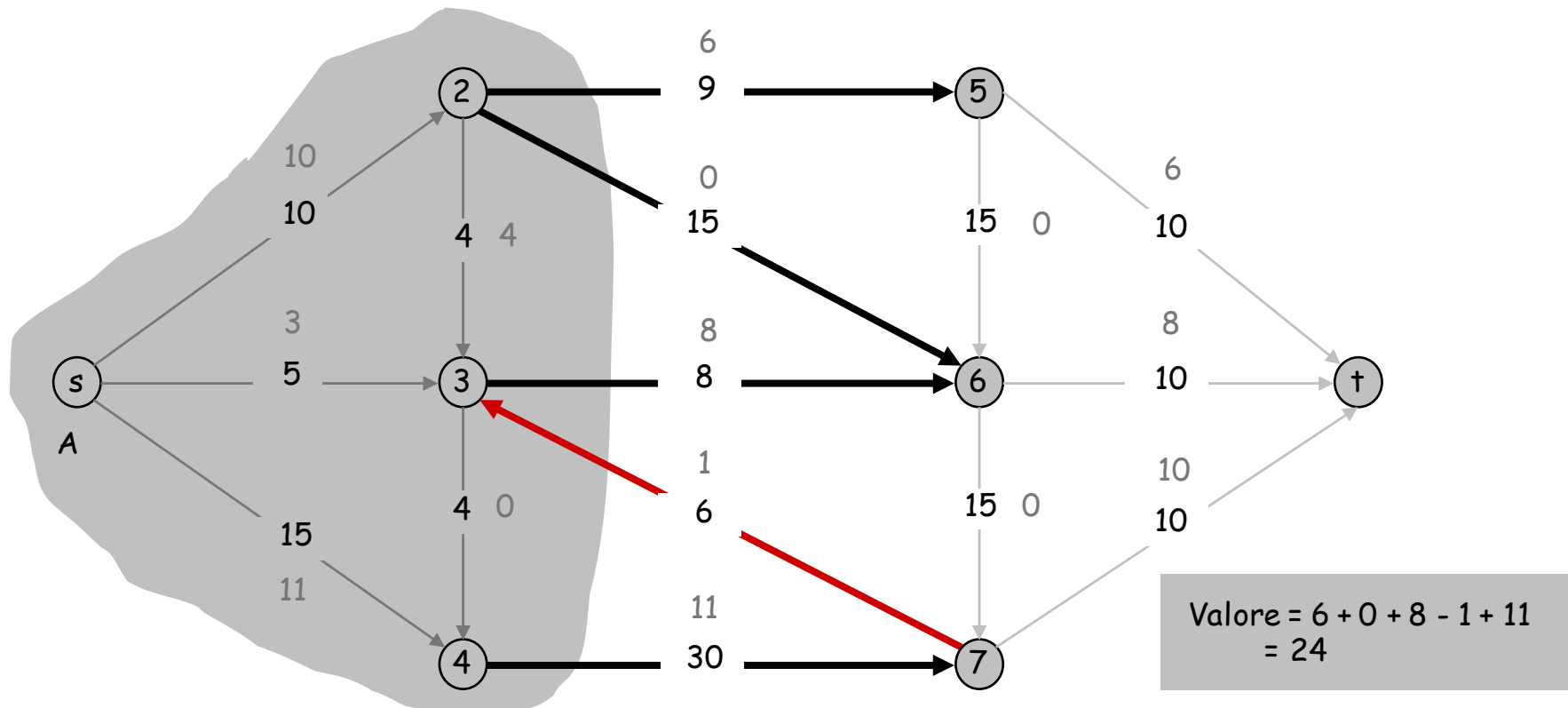
- Problema del minimo taglio s-t. Trova un taglio s-t di capacita` minima



Flussi e tagli

- Lemma del valore del taglio.** Sia f un qualsiasi flusso e sia (A,B) un qualsiasi taglio s - t . Il flusso netto inviato attraverso il taglio e' uguale alla quantita' di flusso uscente da s .

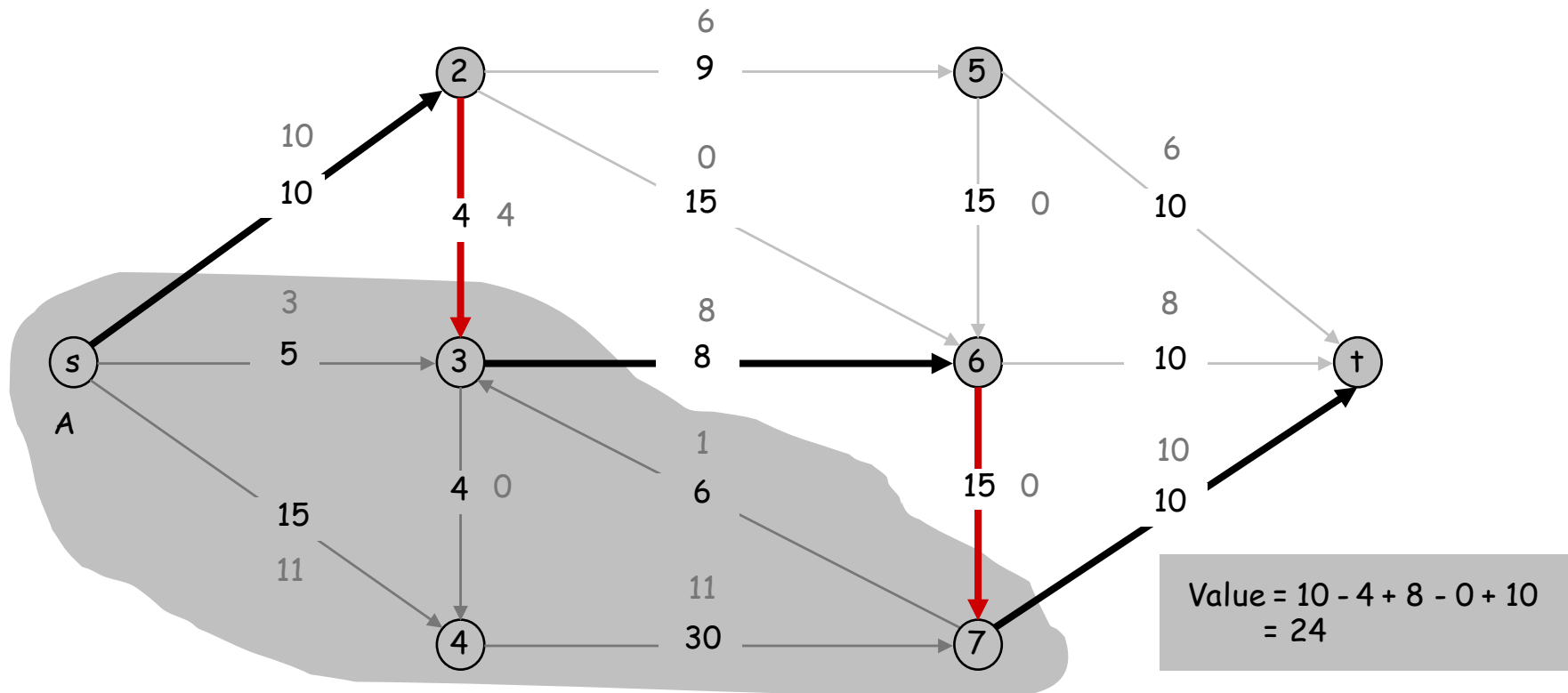
$$\sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e) = v(f)$$



Flussi e Tagli

- **Lemma del valore del taglio.** Sia f un qualsiasi flusso e sia (A,B) un qualsiasi taglio s - t . Il flusso netto inviato attraverso il taglio e' uguale alla quantita' di flusso uscente da s .

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



Flussi e tagli

- **Lemma del valore del taglio.** Sia f un qualsiasi flusso e sia (A, B) un qualsiasi taglio s - t . Il flusso netto inviato attraverso il taglio e' uguale alla quantita' di flusso uscente da s .

$$\sum_{e \text{ out of } A} f(e) - \sum_{e \text{ into } A} f(e) = v(f).$$

- **Dim.**

$$v(f) = \sum_{e \text{ out of } s} f(e)$$

Dalla definizione del valore del flusso

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

Per la conservazione del flusso la quantita' sommata e' 0

$$= \sum_{e \text{ out of } s} f(e) - \sum_{e \text{ into } s} f(e) + \sum_{\substack{v \in A \\ v \neq s}} \left(\sum_{e \text{ out of } v} f(e) - \sum_{e \text{ into } v} f(e) \right)$$

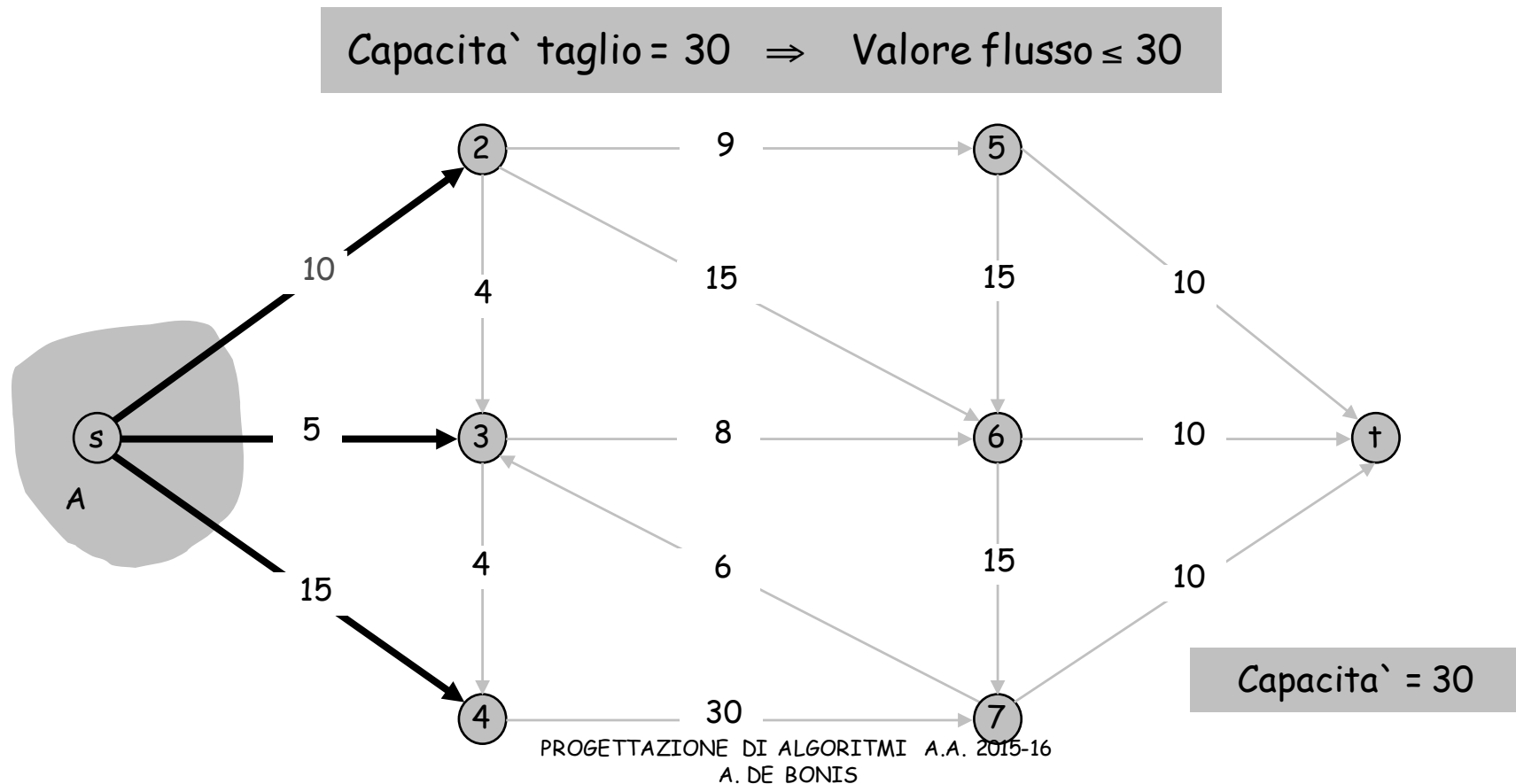
La quantita' sottratta e' 0 perche' nella sorgente non entra niente

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

$$= \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ into } A} f(e).$$

Flussi e tagli

- **Dualita` debole.** Sia f un qualsiasi flusso, e sia (A,B) un qualsiasi taglio s-t. Il valore del flusso e` al piu` la capacita` del taglio.



Flussi e tagli

- **Dualita' debole.** Sia f un qualsiasi flusso, e sia (A,B) un qualsiasi taglio s-t. Il valore del flusso e' al piu' la capacita' del taglio.
- In altre parole, per ogni taglio s-t (A,B) , si ha $v(f) \leq \text{cap}(A, B)$.

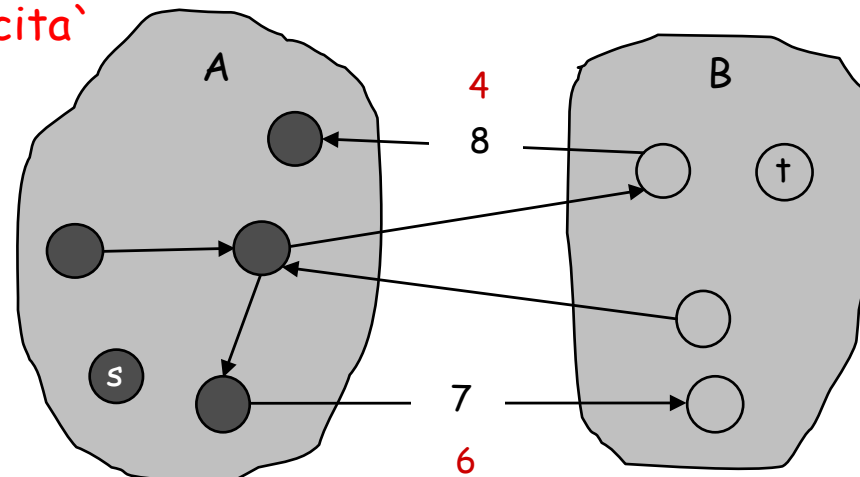
$$v(f) \stackrel{\text{Dim.}}{=} \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ into } A} f(e)$$

Dal lemma del valore del taglio

$$\leq \sum_{e \text{ out of } A} f(e)$$

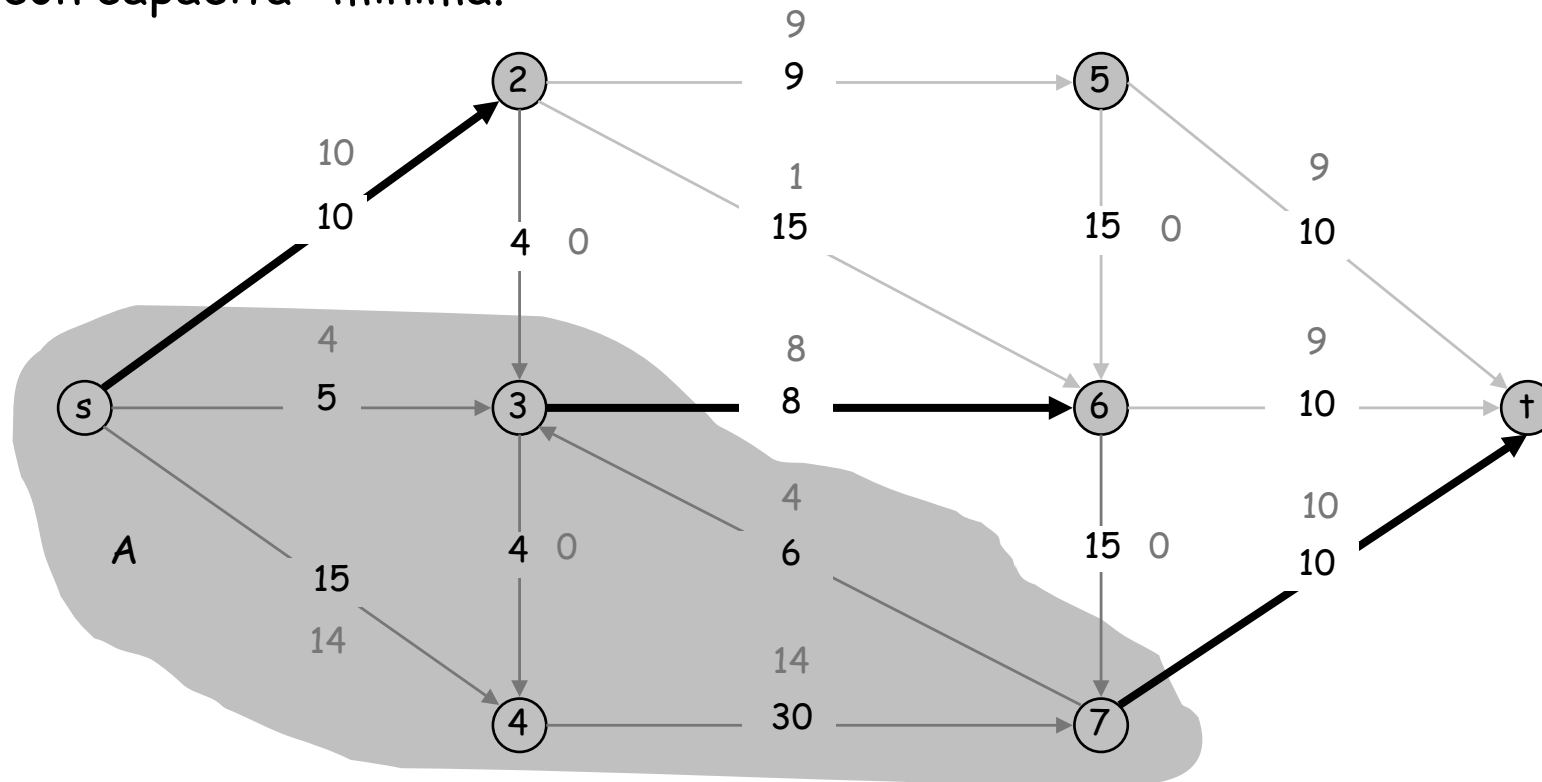
$$\leq \sum_{e \text{ out of } A} c(e) \quad \text{Dal vincolo della capacita'}$$

$$= \text{cap}(A, B)$$



Massimo flusso e Minimo taglio

- **Corollario.** Sia f un qualsiasi flusso e sia (A, B) un qualsiasi taglio s - t . Se $v(f) = \text{cap}(A, B)$ allora f e' il massimo flusso e (A, B) e' un minimo taglio.
- **Dim.** Dal teorema sulla dualita' debole, si ha che per ogni taglio s - t (X, Y) risulta $v(f) \leq \text{cap}(X, Y)$. Se per un certo taglio s - t (A, B) si ha $v(f) = \text{cap}(A, B)$ allora $\text{cap}(A, B) = v(f) \leq \text{cap}(X, Y)$ per cui (A, B) e' un taglio con capacita' minima.

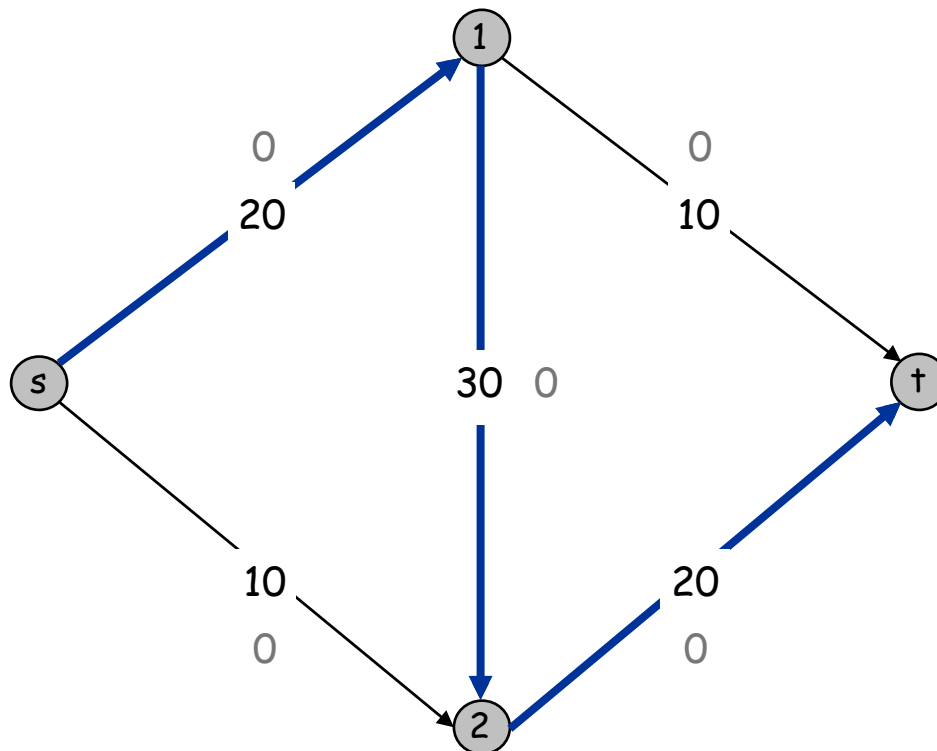


Valore del flusso = 28

Capacita' del taglio = 28 \Rightarrow Valore del flusso \leq 28

Algoritmo per il max flusso: approccio sbagliato

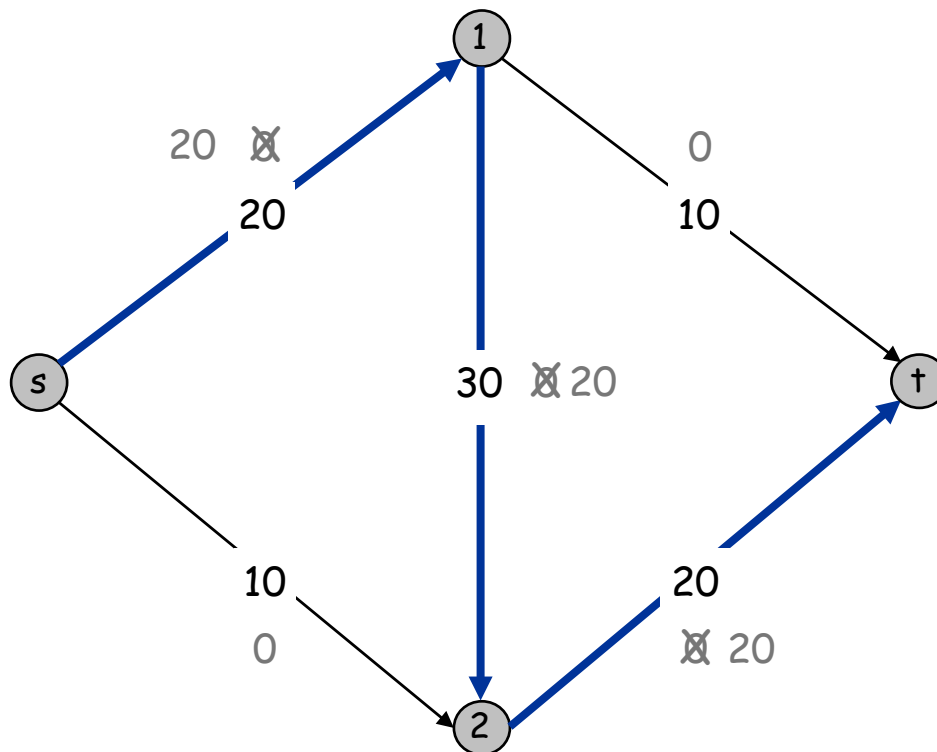
- *Algoritmo greedy.*
 - Cominciamo con $f(e) = 0$ per ogni arco $e \in E$.
 - Troviamo un percorso P da s a t dove per ogni arco sul percorso si ha $f(e) < c(e)$.
 - Aumentiamo il flusso lungo il percorso P in modo da rispettare il vincolo della capacita` e quello sulla conservazione del flusso.
 - Ripetiamo fino a che non e` piu` possibile trovare un percorso lungo il quale e` possibile aumentare il flusso



Valore del flusso= 0

Algoritmo per il max flusso: approccio sbagliato

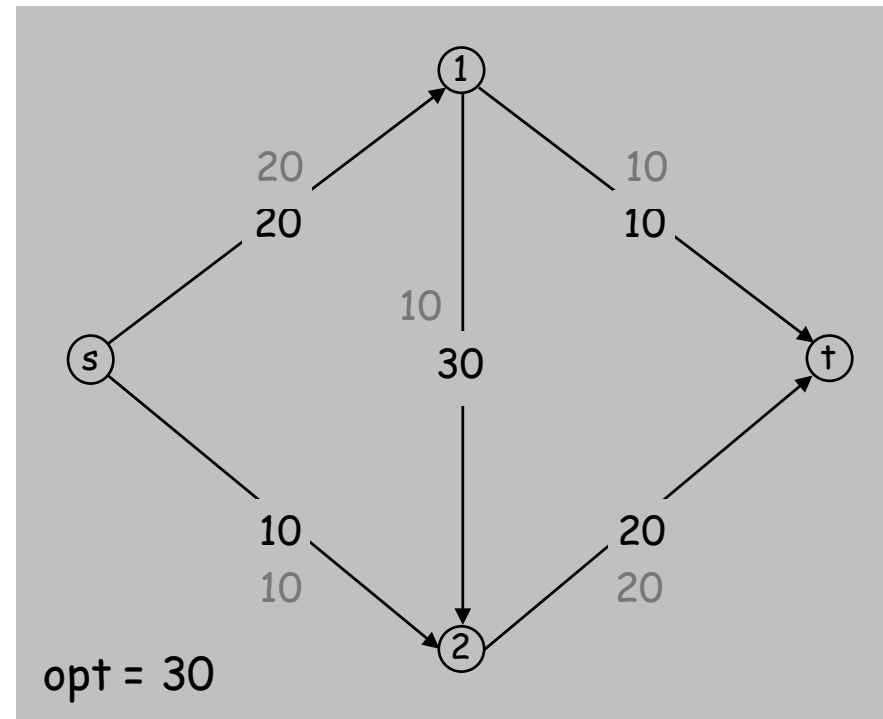
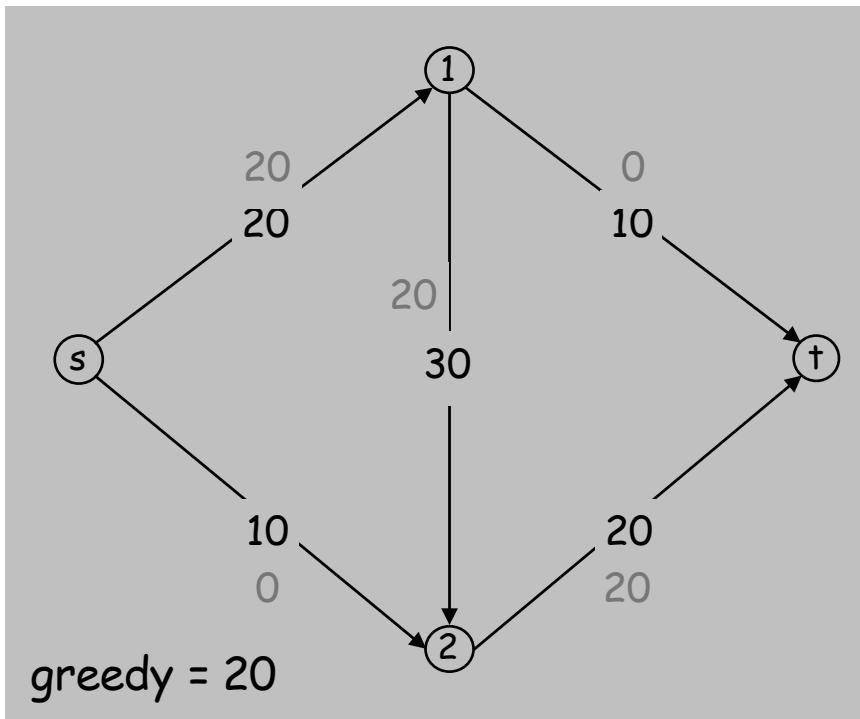
- *Algoritmo greedy.*
 - Cominciamo con $f(e) = 0$ per ogni arco $e \in E$.
 - Troviamo un percorso P da s a t dove per ogni arco sul percorso si ha $f(e) < c(e)$.
 - Aumentiamo il flusso lungo il percorso P in modo da rispettare il vincolo della capacità e quello sulla conservazione del flusso.
 - Ripetiamo fino a che non è più possibile trovare un percorso lungo il quale è possibile aumentare il flusso



Valore del flusso= 20

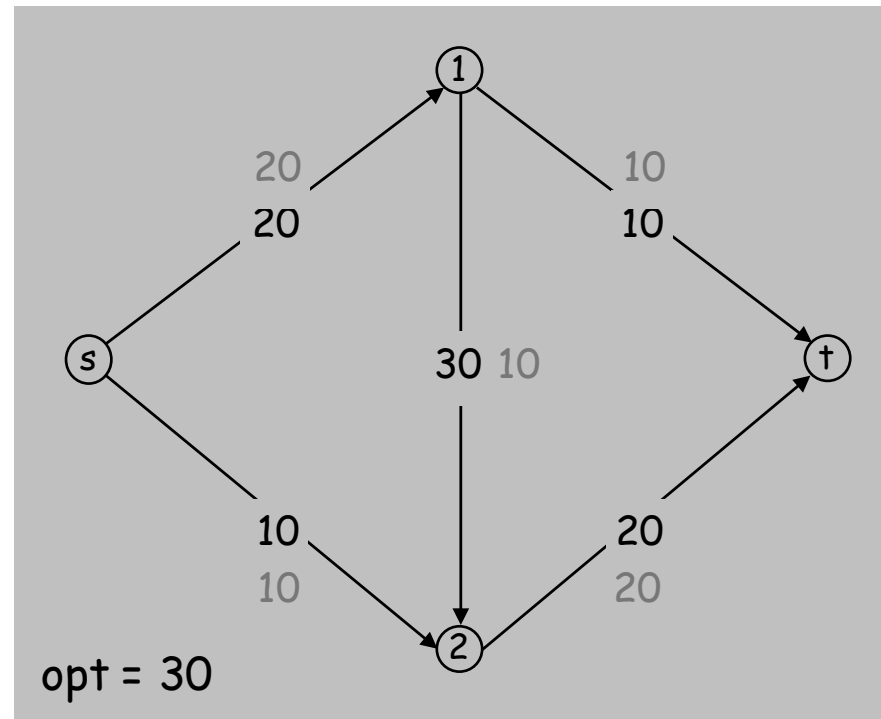
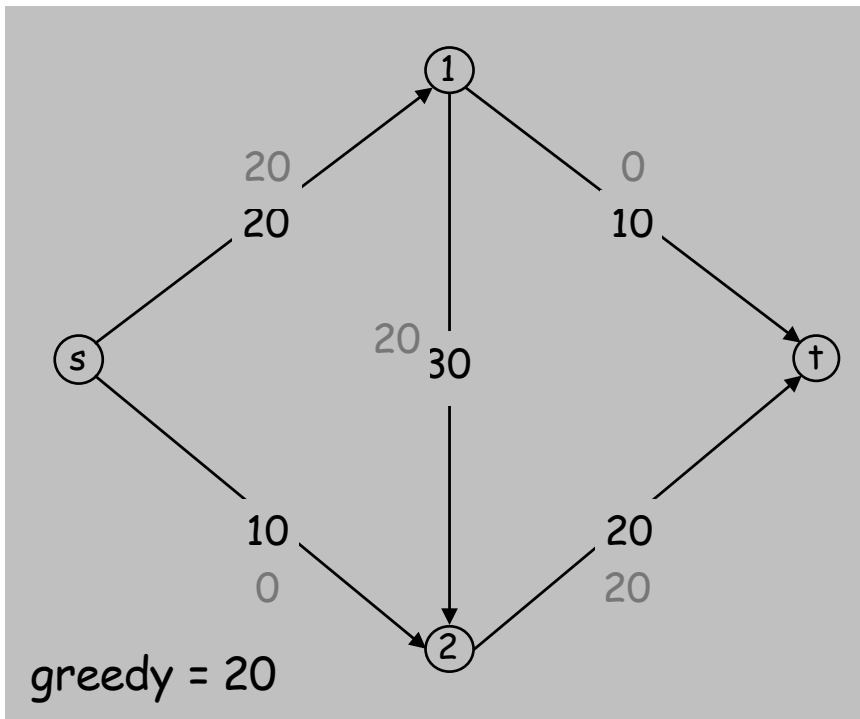
Algoritmo per il max flusso: approccio sbagliato

- *Algoritmo greedy.*
 - Cominciamo con $f(e) = 0$ per ogni arco $e \in E$.
 - Troviamo un percorso P da s a t dove per ogni arco sul percorso si ha $f(e) < c(e)$.
 - Aumentiamo il flusso lungo il percorso P in modo da rispettare il vincolo della capacità e quello sulla conservazione del flusso.
 - Ripetiamo fino a che non è più possibile trovare un percorso lungo il quale è possibile aumentare il flusso



Algoritmo per il max flusso: approccio sbagliato

- Il problema con l'algoritmo di prima è che si blocca quando non riesce a trovare un percorso lungo il quale spingere altro flusso.
- Per sbloccare la situazione, possiamo provare ad annullare in parte o del tutto il flusso inviato su alcuni degli archi.
- Come facciamo ad annullare in parte o del tutto il flusso lungo un certo arco $e=(u,v)$? **Risposta:** Immaginiamo di far tornare indietro il flusso.
- Quanto flusso possiamo far tornare indietro? **Risposta:** Al più una quantità pari al flusso $f(e)$ spinto lungo e .



Grafo residuo

Arco originale.: $e = (u, v) \in E$ con capacita' $c(e)$.

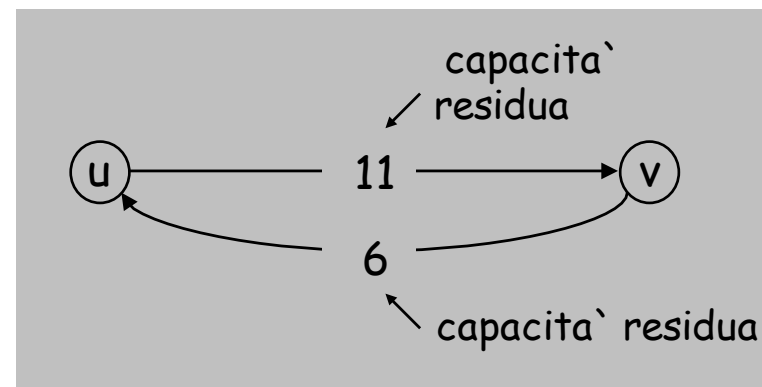
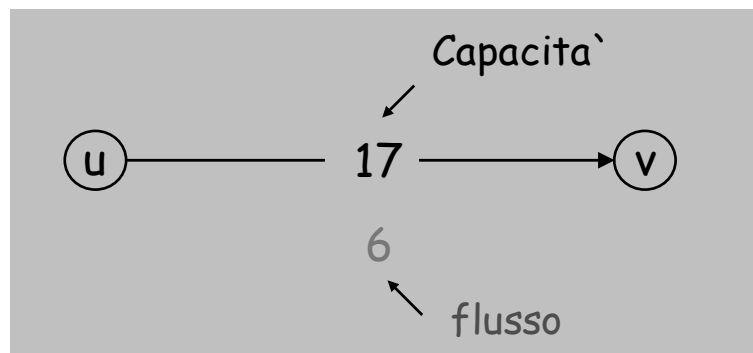
Sia $f(e)$ il flusso lungo e

Def. Insieme di archi residui E_f rispetto ad f

- Per ogni arco $e = (u, v) \in E$,
 - se $f(e) < c(e)$ allora $e \in E_f$ e ha capacita' $c_f(e) = c(e) - f(e)$.
 - Se $f(e) > 0$ allora $e^B = (v, u) \in E_f$ e ha capacita' $c_f(e^B) = f(e)$
- Quindi $E_f = \{e: e \in E, f(e) < c(e)\} \cup \{e^B: e \in E, f(e) > 0\}$

Le capacita' c_f degli archi nel grafo residuo vengono dette capacita' residue

Def. Grafo residuo: $G_f = (V, E_f)$.



Grafo residuo rispetto ad f

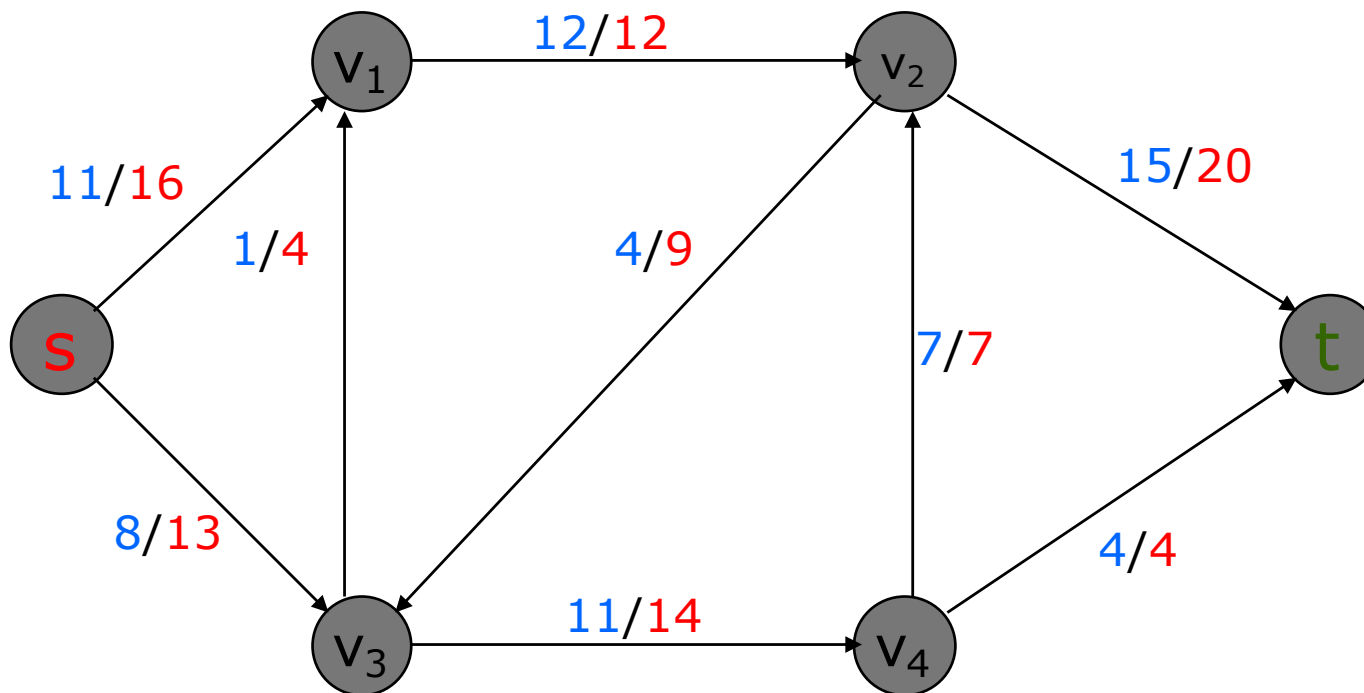
- Per definizione, un arco (u,v) appartiene ad E_f se
 - $(u,v) \in E$, $c(u,v) > 0$ e $f(u,v) < c(u,v)$. In questo caso $c_f(u,v) = c(u,v) - f(u,v)$

oppure se

- $(v,u) \in E$ e $f(v,u) > 0$. In questo caso $c_f(u,v) = f(v,u)$

- NB: $|E_f| \leq 2|E|$

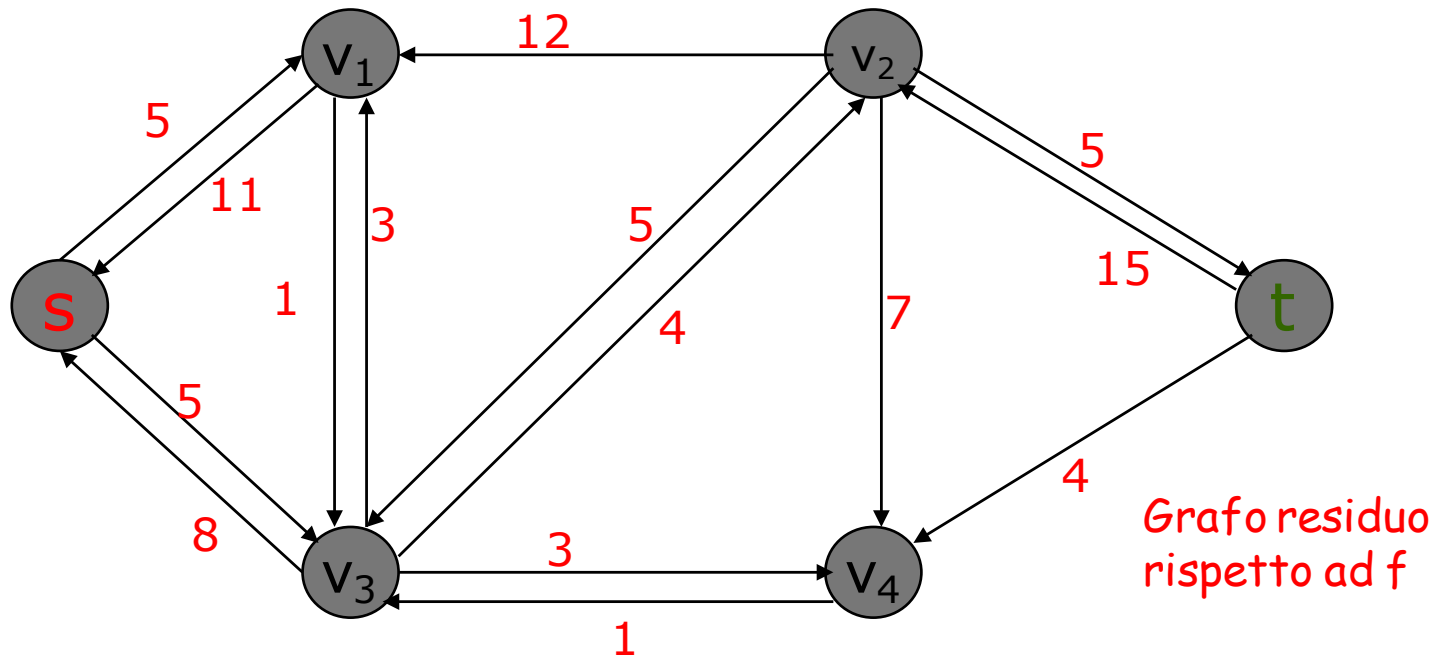
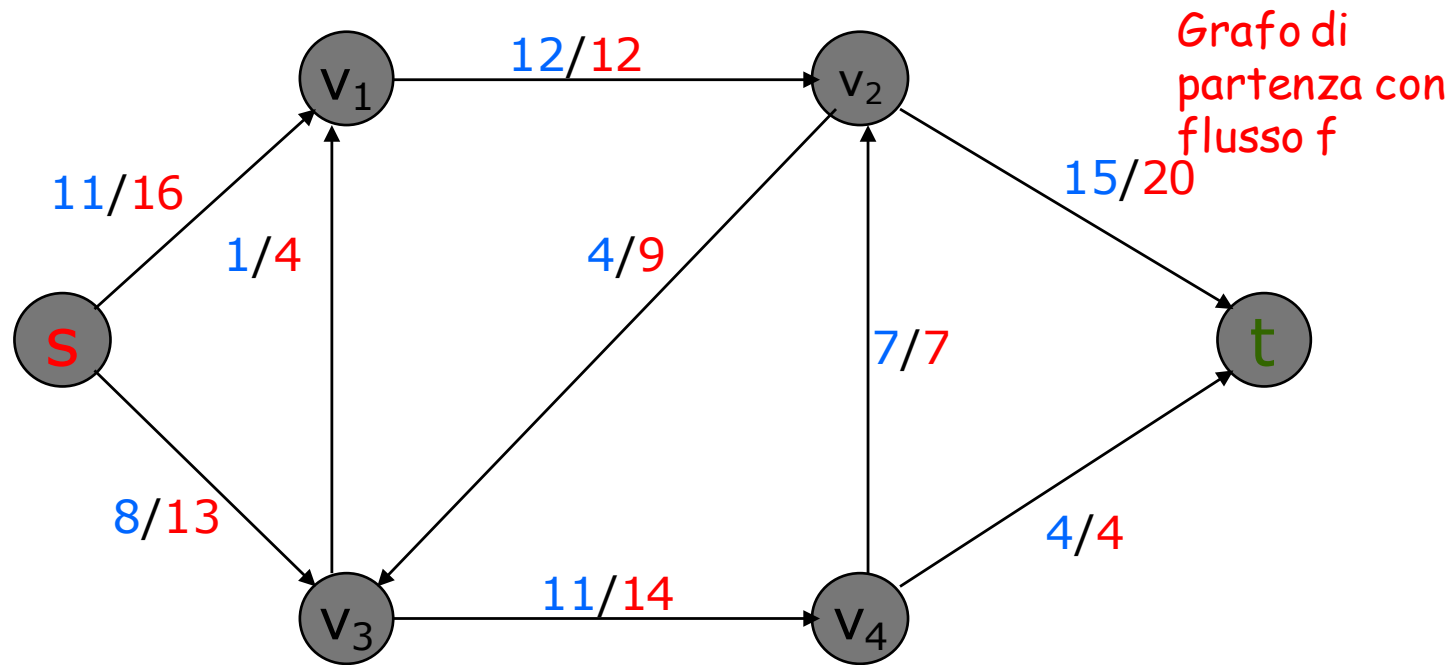
Esempio di rete di flusso



$$c_f(s, V_1) = c(s, V_1) - f(s, V_1) = 16 - 11 = 5$$

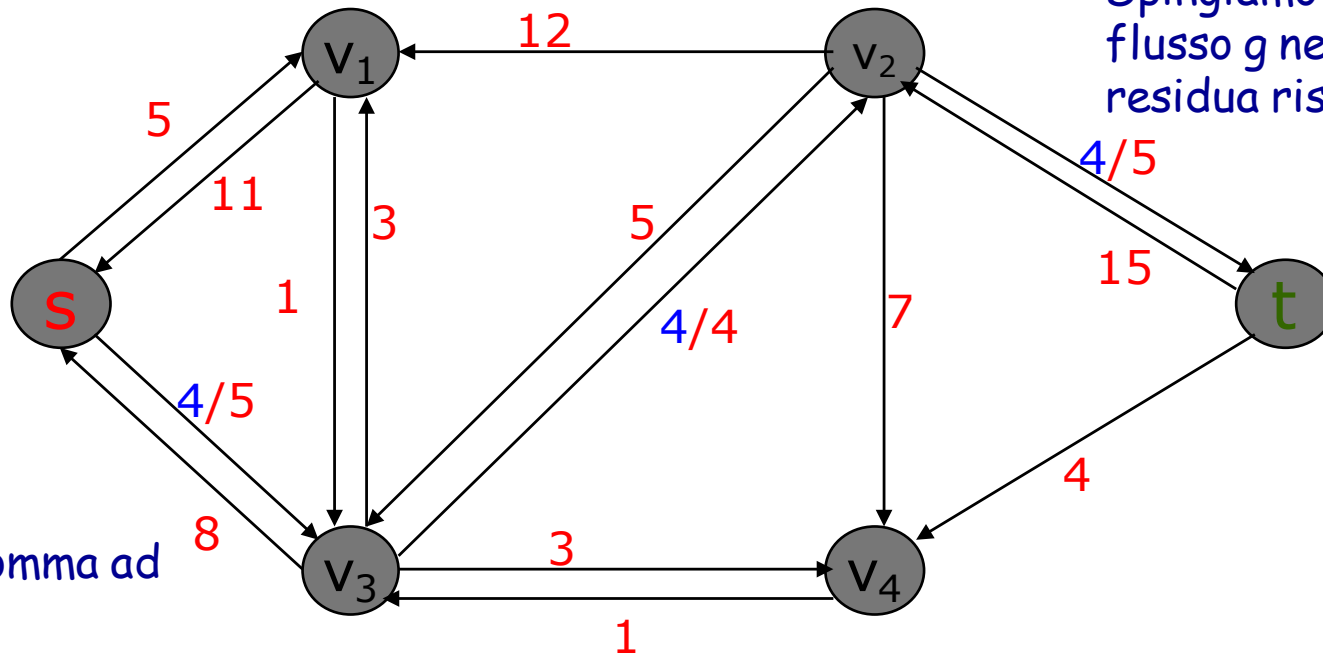
$$c_f(V_1, s) = c(V_1, s) - f(V_1, s) = 0 - (-11) = 11$$

Esempio di rete residua

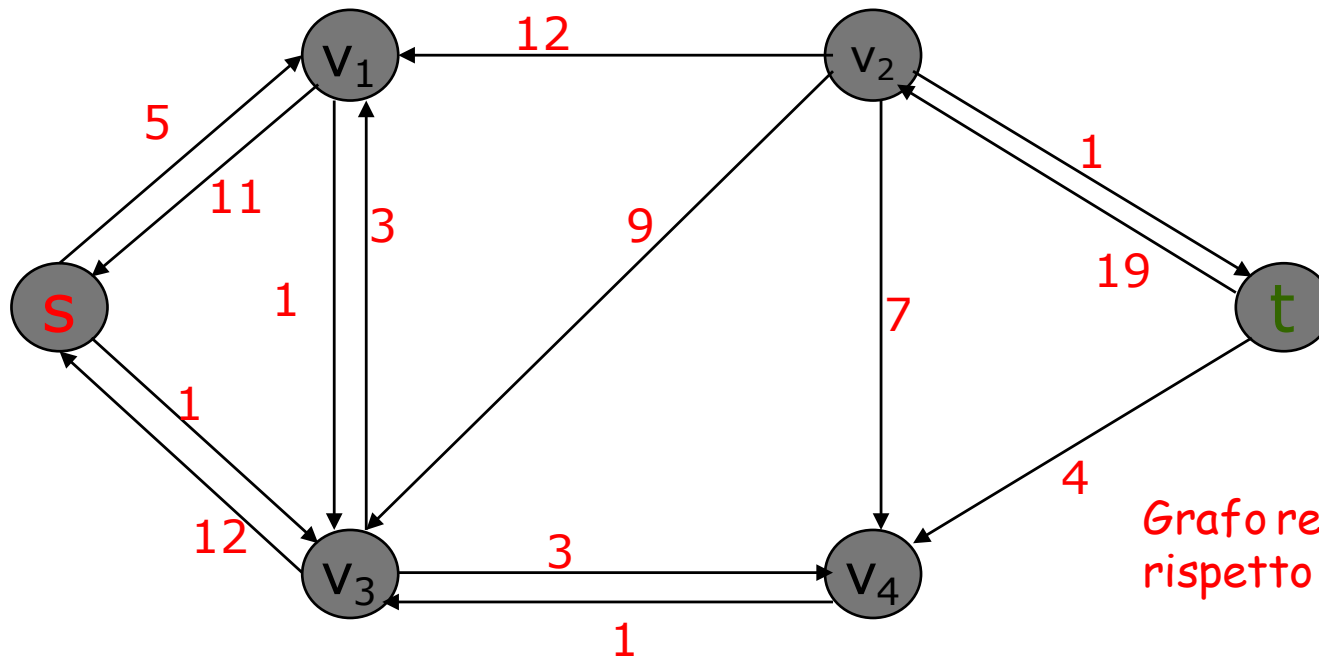


Esempio di rete residua

Spingiamo un nuovo flusso g nella rete residua rispetto ad f



Il flusso g si somma ad f per cui...



Grafo residuo rispetto ad $f+g$

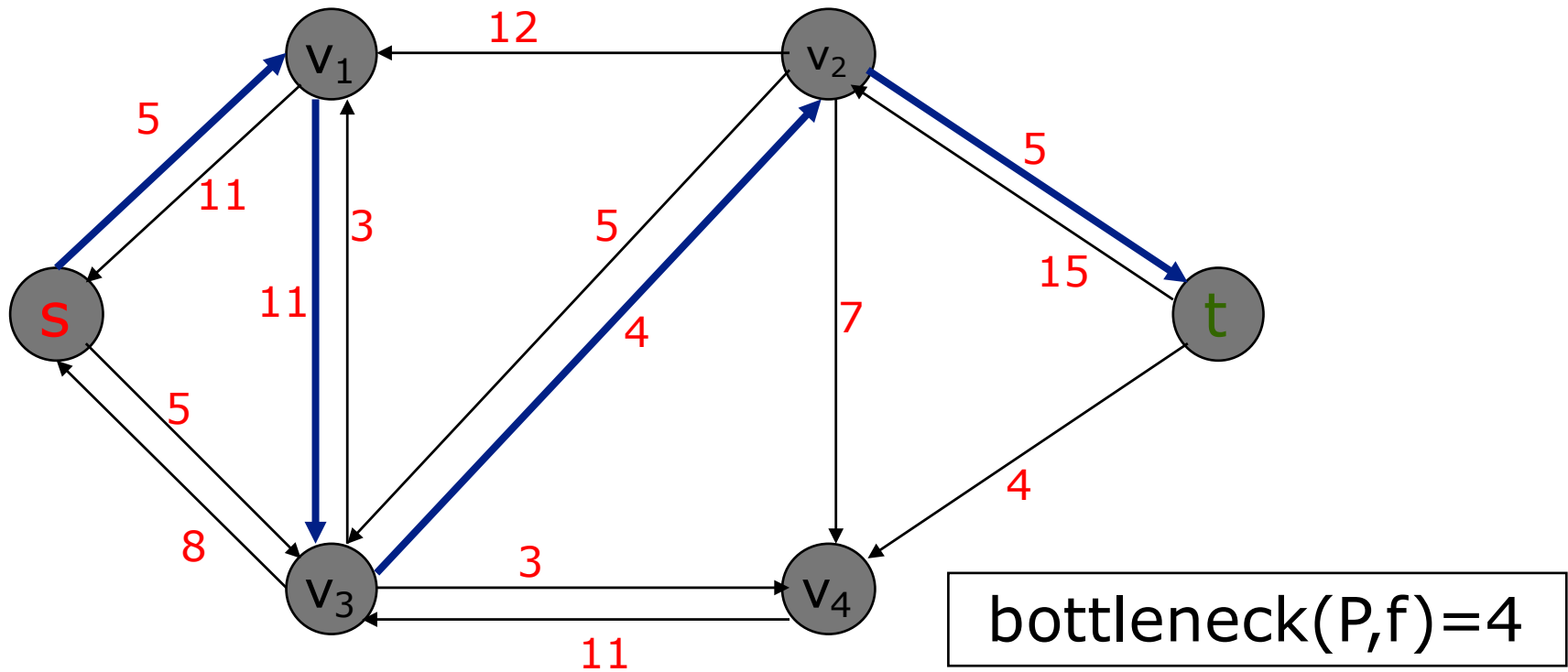
Augmentig path (cammino aumentante)

- Data una rete di flusso $G=(V,E)$ ed un flusso f , un *augmenting path* (cammino aumentante) P è un cammino semplice da s a t nella rete residua G_f
- Dato un *augmenting path* P definiamo la *capacità residua* di P (collo di bottiglia di P rispetto al flusso f) come

$$\text{bottleneck}(P,f) = \min\{c_f(u,v) : (u,v) \text{ è un arco in } P\}$$

- La capacità residua rappresenta la massima quantità di flusso che si può trasportare lungo gli archi di p

Esempio di augmentig path



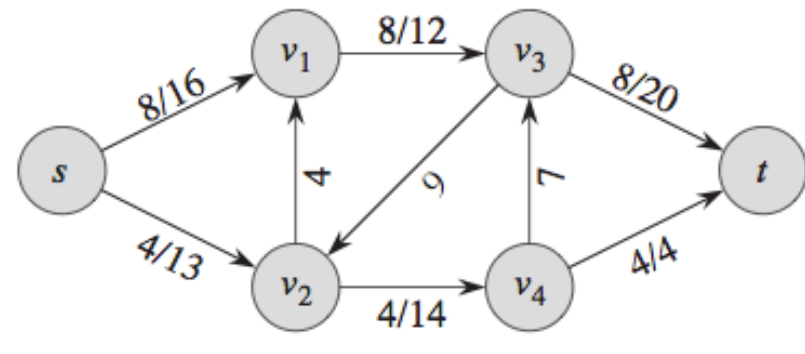
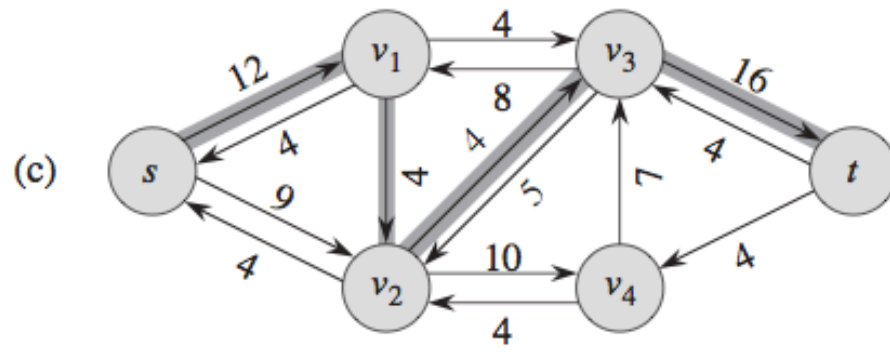
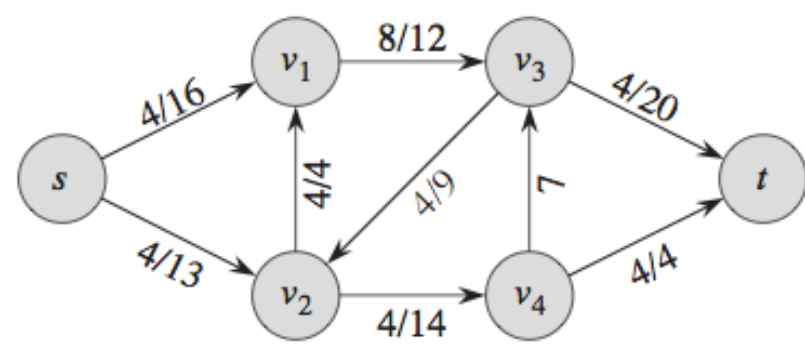
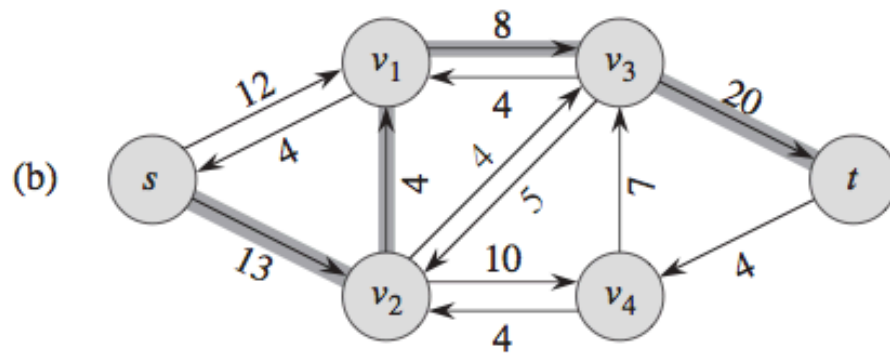
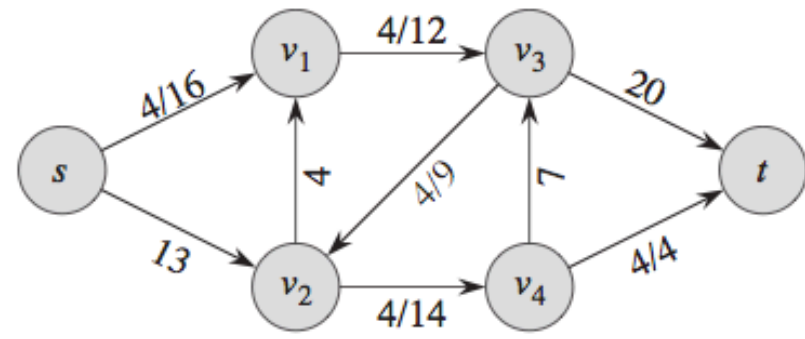
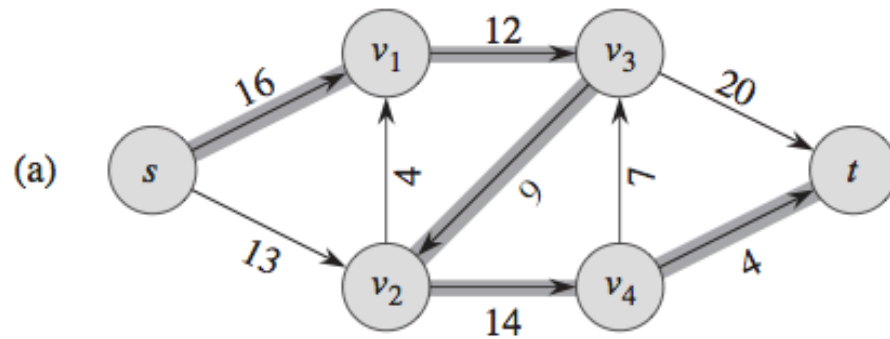
$$P = (s, v_1)(v_1, v_3)(v_3, v_2)(v_2, t)$$

Algorithm di Ford-Fulkerson

```
Augment(f, c, P) {  
  b ← bottleneck(P, f)  
  foreach e ∈ P {  
    if (e ∈ E) f(e) ← f(e) + b  
    else      f(eB) ← f(eB) - b  
  }  
  return f  
}
```

```
Ford-Fulkerson(G, s, t, c) {  
  foreach e ∈ E f(e) ← 0  
  Gf ← residual graph  
  
  while (there exists augmenting path P) {  
    f ← Augment(f, c, P)  
    update Gf  
  }  
  return f  
}
```

Esempio



Algoritmo di Ford Fulkerson

- **Teorema.** Sia $G=(V,E)$ una rete di flusso e sia f un flusso in G . Sia P un *augmentig path* nella rete residua G_f . Indichiamo con f' il flusso restituito da $\text{Augment}(f, c, P)$. Allora f' è un flusso in G con valore $\text{val}(f') = \text{val}(f) + \text{bottleneck}(P, f) > \text{val}(f)$

Teorema del max flusso e minimo taglio

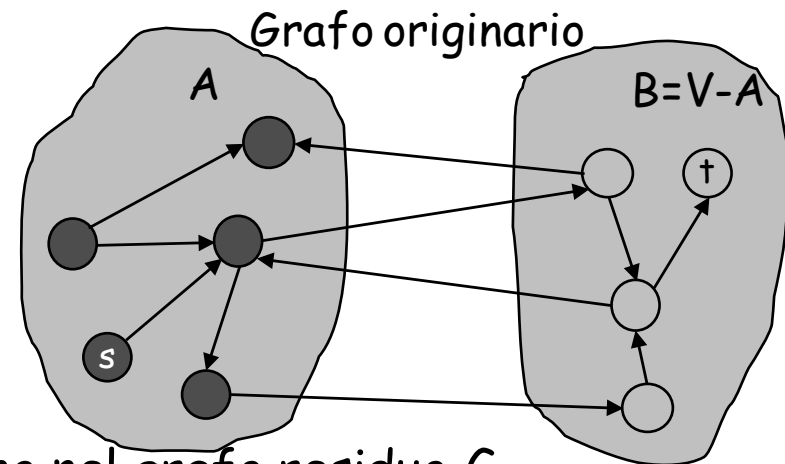
- Teorema: Sia dato un flusso f per la rete di flusso $G=(V,E)$. Le seguenti affermazioni sono equivalenti
 - (i) Esiste un taglio (A, B) tale che $v(f) = \text{cap}(A, B)$.
 - (ii) Il flusso f e` un max flusso
 - (iii) Non esiste un cammino aumentante in G_f .
- (i) \Rightarrow (ii) Questo e` il corollario che abbiamo gia` dimostrato
- (ii) \Rightarrow (iii)
 - Se esistesse un percorso aumentante in G_f allora potremmo aumentare il flusso spingendo altro flusso lungo il cammino aumentante e di conseguenza f non sarebbe massimo. Infatti il teorema precedente implica che il nuovo flusso avrebbe valore $= f + \text{bottleneck}(P, f) > f$

Continua nella prossima slide

Teorema del max flusso e minimo taglio

- (iii) \Rightarrow (i)
 - Supponiamo che G_f non contenga cammini aumentanti.
 - Sia A l'insieme di vertici raggiungibili da s in G_f
 - Per definizione di A , $s \in A$.
 - Siccome per ipotesi G_f non contiene cammini aumentanti allora t non è raggiungibile da s e di conseguenza $t \notin A$. Quindi (A, B) con $B = V - A$ è un taglio s - t .
 - Possiamo allora applicare il lemma del valore del taglio al flusso f e al taglio (A, B)

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



Gli archi di G che vanno da A a B non sono nel grafo residuo G_f

Gli archi di G che vanno da B ad A hanno flussi uguali a 0: supponiamo che un arco (v, u) con u in A e v in B abbia flusso $f(u, v) > 0$. Questo vuol dire che $c_f(u, v) > 0$ e quindi (u, v) è in G_f e di conseguenza v è in anch'esso in A . Contraddizione!

Conseguenze del teorema

- Un flusso f è massimo se e solo se non ci sono cammini aumentanti
- Il valore del massimo flusso è uguale alla capacità del minimo taglio

Tempo di esecuzione

- Il *tempo di esecuzione* dell'algoritmo di Ford-Fulkerson dipende da come si determina il cammino aumentante
- Quando la capacità assume valori reali (irrazionali), se il cammino aumentante non è scelto con cura Ford-Fulkerson potrebbe non convergere mai
- Se l'AP è scelta usando la BFS, l'algoritmo ha un *running-time* polinomiale

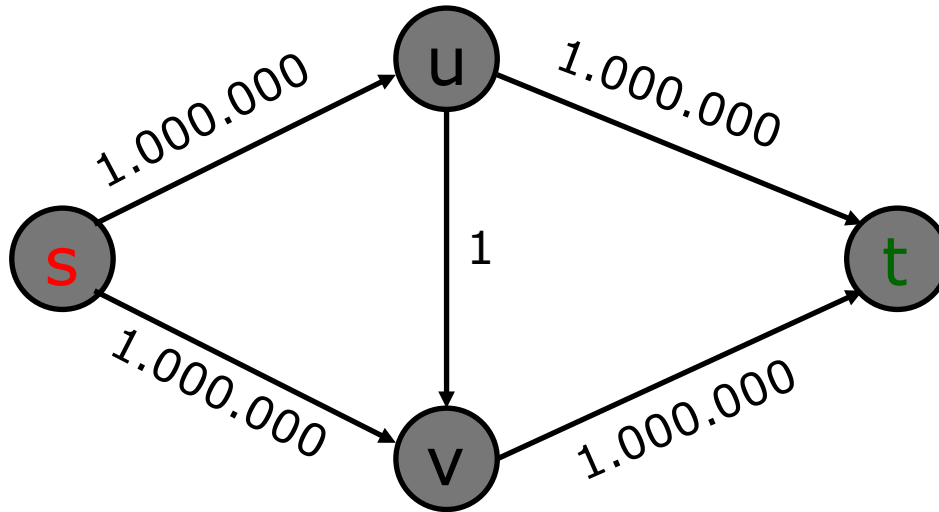
Tempo di esecuzione nel caso di capacità intere

- **Assunzione.** Tutte le capacità sono interi tra uno e C .
- \rightarrow somma capacità archi uscenti da $s \leq nC \rightarrow$ flusso massimo $v(f^*) \leq nC$
- **Invariante.** Ciascun valore del flusso $f(e)$ e ciascuna capacità residua $c_f(e)$ rimane un intero durante tutto l'algoritmo
- **Teorema.** L'algoritmo termina in $v(f^*)$ iterazioni.
- **Dim.** Ogni iterazione aumenta il flusso di uno
- **Teorema.** L'algoritmo ha tempo di esecuzione $O(m \times v(f^*)) = O(m \times (\text{somma cap. archi uscenti da } s)) = O(mnC)$
- **Dim.** Ogni iterazione impiega $O(n+m) = O(m)$ per trovare il cammino aumentante con BFS o DFS. Inoltre $O(n) = O(m)$ per Augment e $O(m)$ per costruire il nuovo grafo residuo.
- **Corollario.** Se $C=1$, Ford-Fulkerson ha tempo di esecuzione $O(mn)$.

Tempo di esecuzione nel caso di capacità intere

- **Teorema.** Se tutte le capacità sono intere allora il valore del flusso max $v(f^*)$ è intero ed esiste una funzione flusso f con valore $v(f^*)$ tale che $f(e)$ è un intero per ogni arco e
- **Dim.** Basta considerare l'algoritmo di Ford-Fulkerson: l'algoritmo termina quando non ci sono più cammini aumentanti e , per il teorema del massimo flusso e minimo taglio, il flusso restituito è max. L'invariante implica che il valore del flusso restituito è max e assegna ad ogni arco e un valore $f(e)$ intero.

Scelta del cammino aumentante



$$f^* = 2.000.000$$

$$AP_1 = (s, u)(u, v)(v, t)$$

Percorso aumentante nelle iterazioni di ordine dispari

$$AP_2 = (s, v)(v, u)(u, t)$$

Percorso aumentante nelle iterazioni di ordine pari

Numero iterazioni = 2.000.000

Scegliere buoni cammini aumentanti

- Alcune scelte dei cammini aumentanti possono portare ad un numero elevato di iterazioni
- Scelte piu` attente possono portare ad algoritmi polinomiali
- Come gia` osservato, se le capacita` sono numeri irrazionali, l'algoritmo potrebbe non terminare.
 - NB: capacita` reali razionali possono essere trasformate in interi per cui l'algoritmo converge.

[Edmonds-Karp 1972, Dinitz 1970]

- Viene scelto il cammino aumentante piu` corto (BFS!).
- $O(nm)$ iterazioni.
- Si puo` implementare in modo il tempo di esecuzione sia $O(nm^2)$