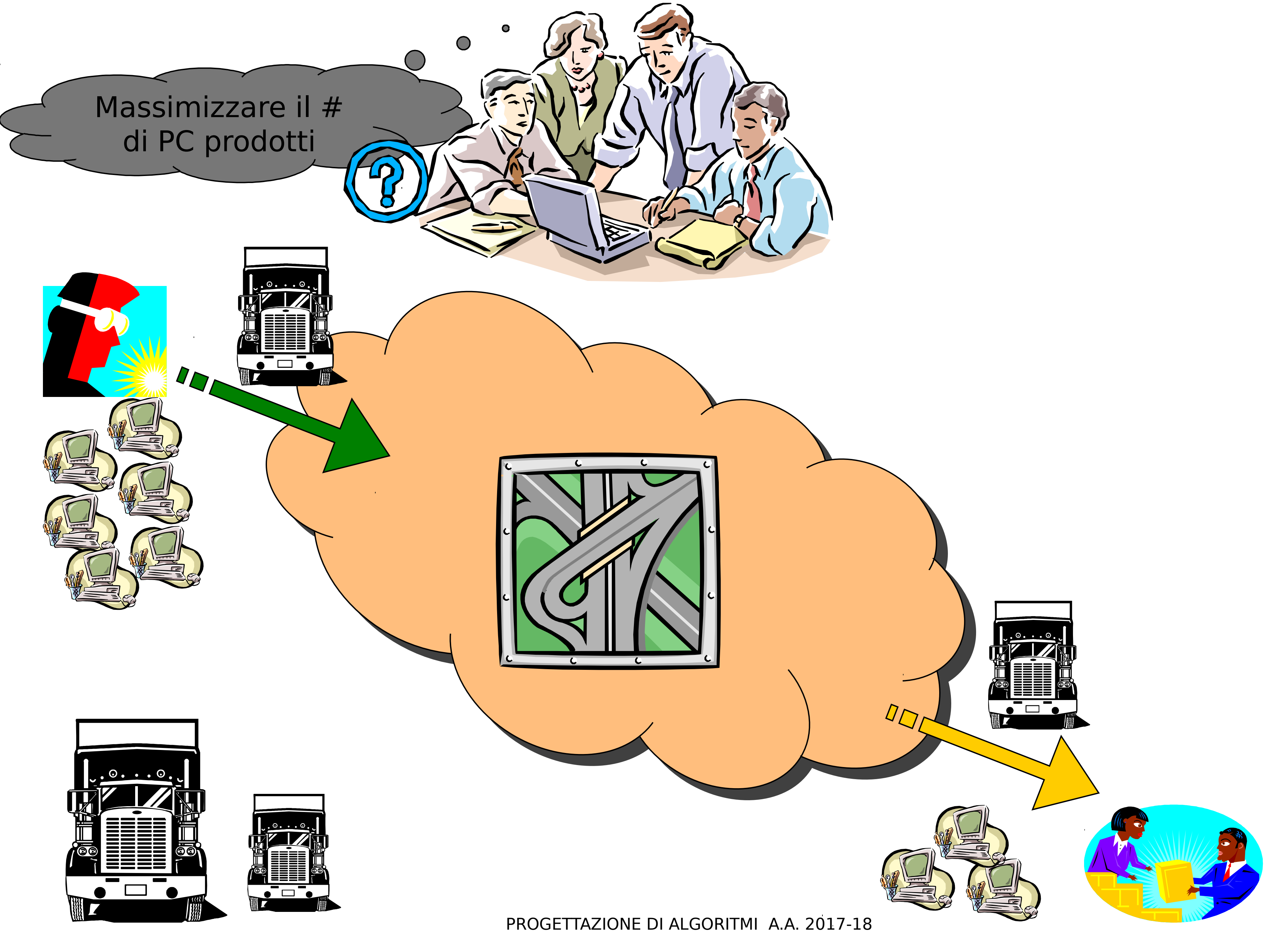
Massimo flusso

Progettazione di Algoritmi a.a. 2017-18

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

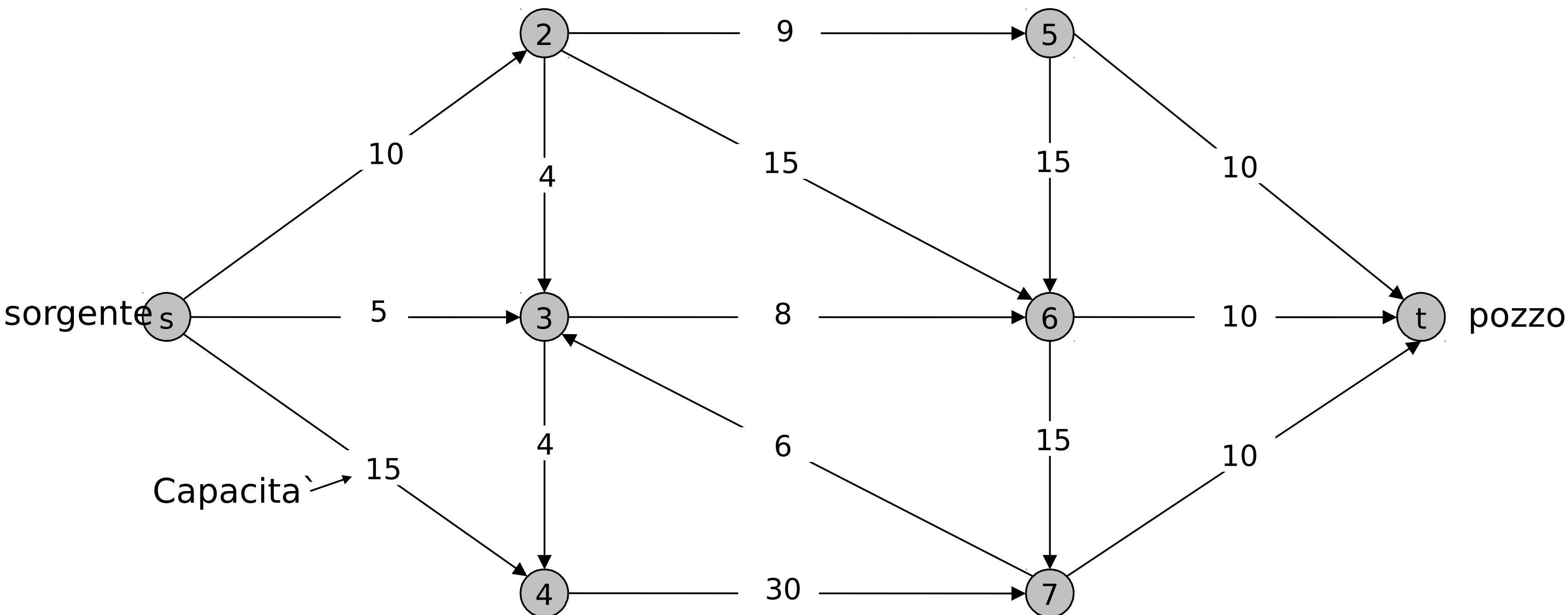
- In ogni nodo della rete di comunicazione:
  - Non è possibile produrre PC
  - Non è possibile stoccare PC

- 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?

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



Funzione flusso

- Una funzione flusso f e` una funzione che assegna ad ogni arco (u,v) un valore reale f(u,v) **maggiore o uguale di 0** e soddisfa le due seguenti proprieta`:

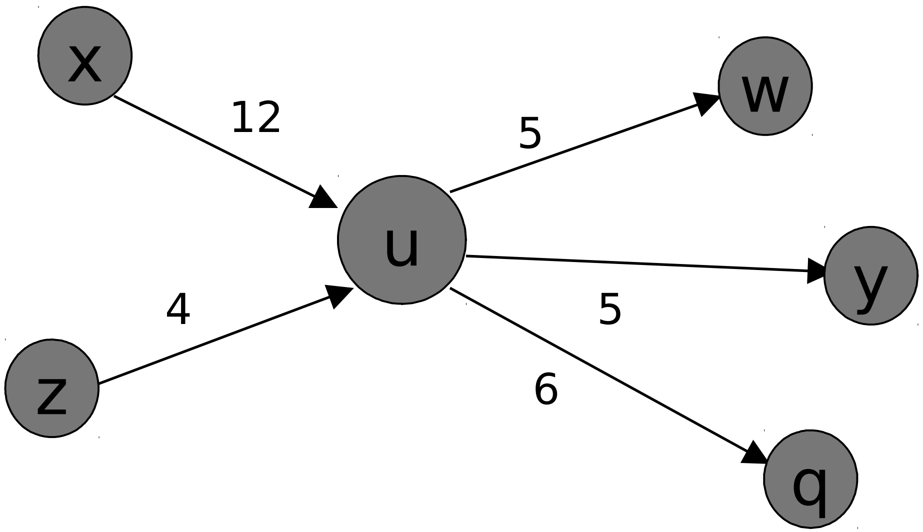
  - Vincolo sulla capacita`: per ogni arco (u,v) si ha f(u,v)≦c(u,v)
    - il flusso su un arco (u,v) deve essere minore o uguale della capacita§ dell'arco (u,v)

  - Conservazione del flusso: per ogni nodo u diverso da s e t si ha

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

    - la quantita` di flusso che entra in un nodo u deve essere uguale alla quantita` di flusso che esce da u.

Conservazione del flusso: esempio



Valore del flusso

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

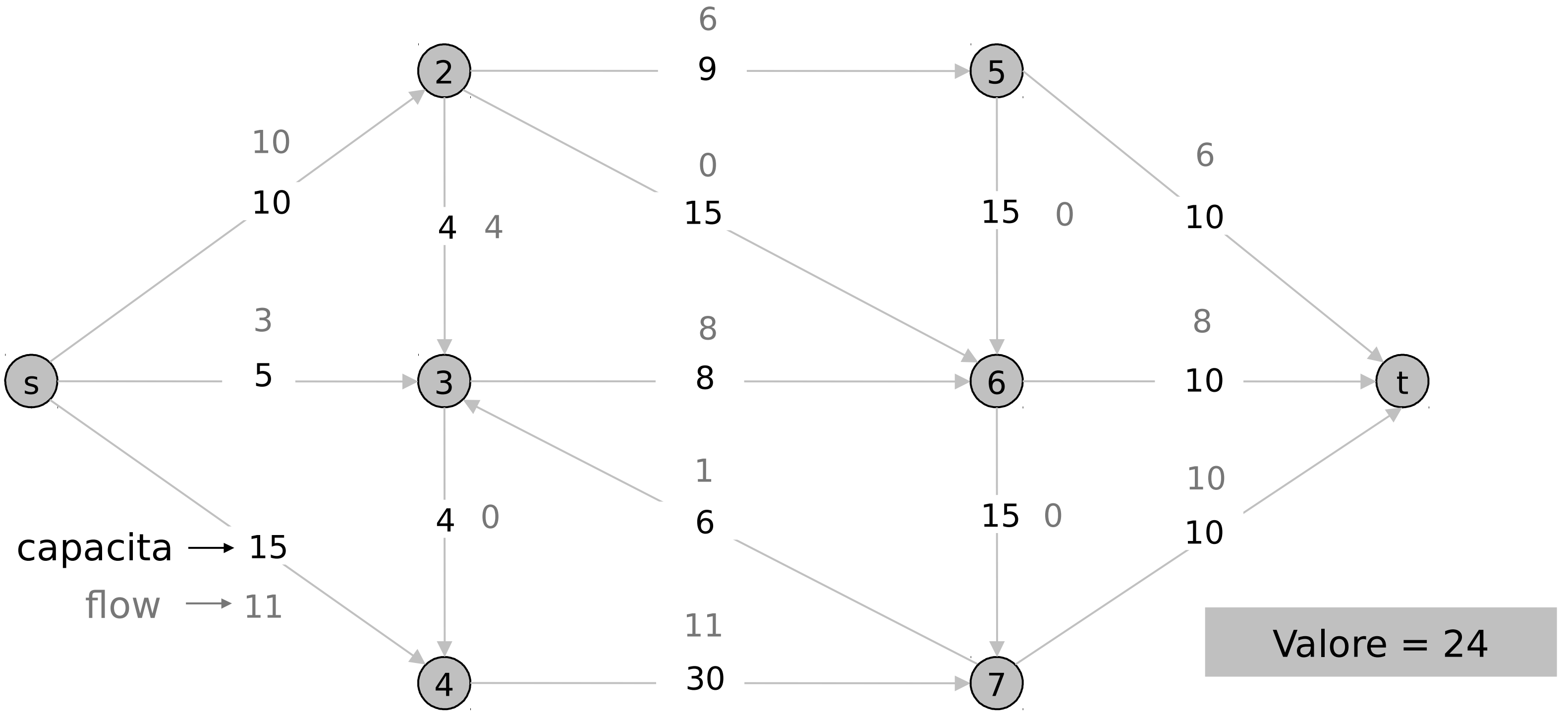
- $v(f) = \sum_{v \in V} f(s,v)$

E' possibile verificare che vale anche

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

Flussi

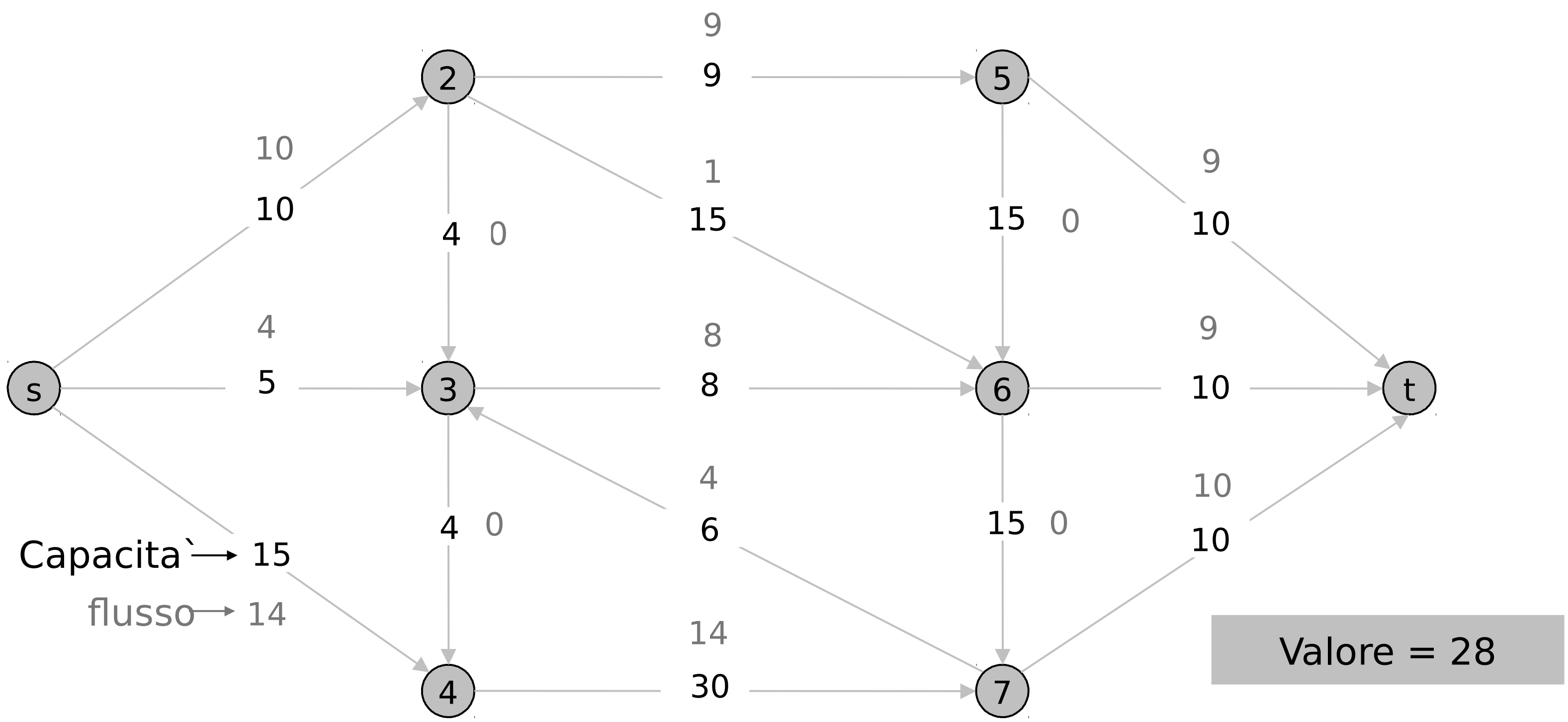
✂



6
9
10
10
0
15
15
0
6
10
4
4
3
5
8
8
8
10
1
6
4
0
15
0
10
10
capacita
15
flow
11
11
30
s
2
3
4
5
6
7
t
Valore = 24

Il problema del massimo flusso

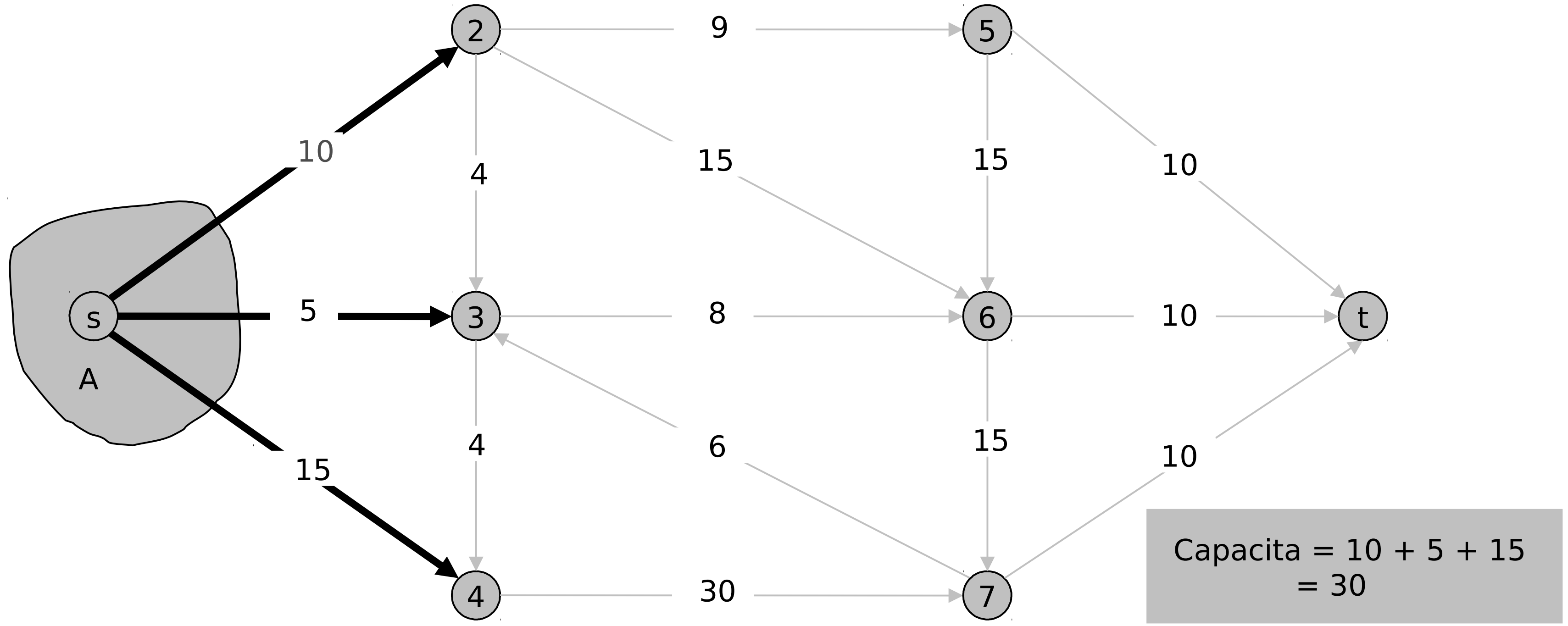
- Il problema del massimo flusso. Trova un flusso da s a t di valore massimo.



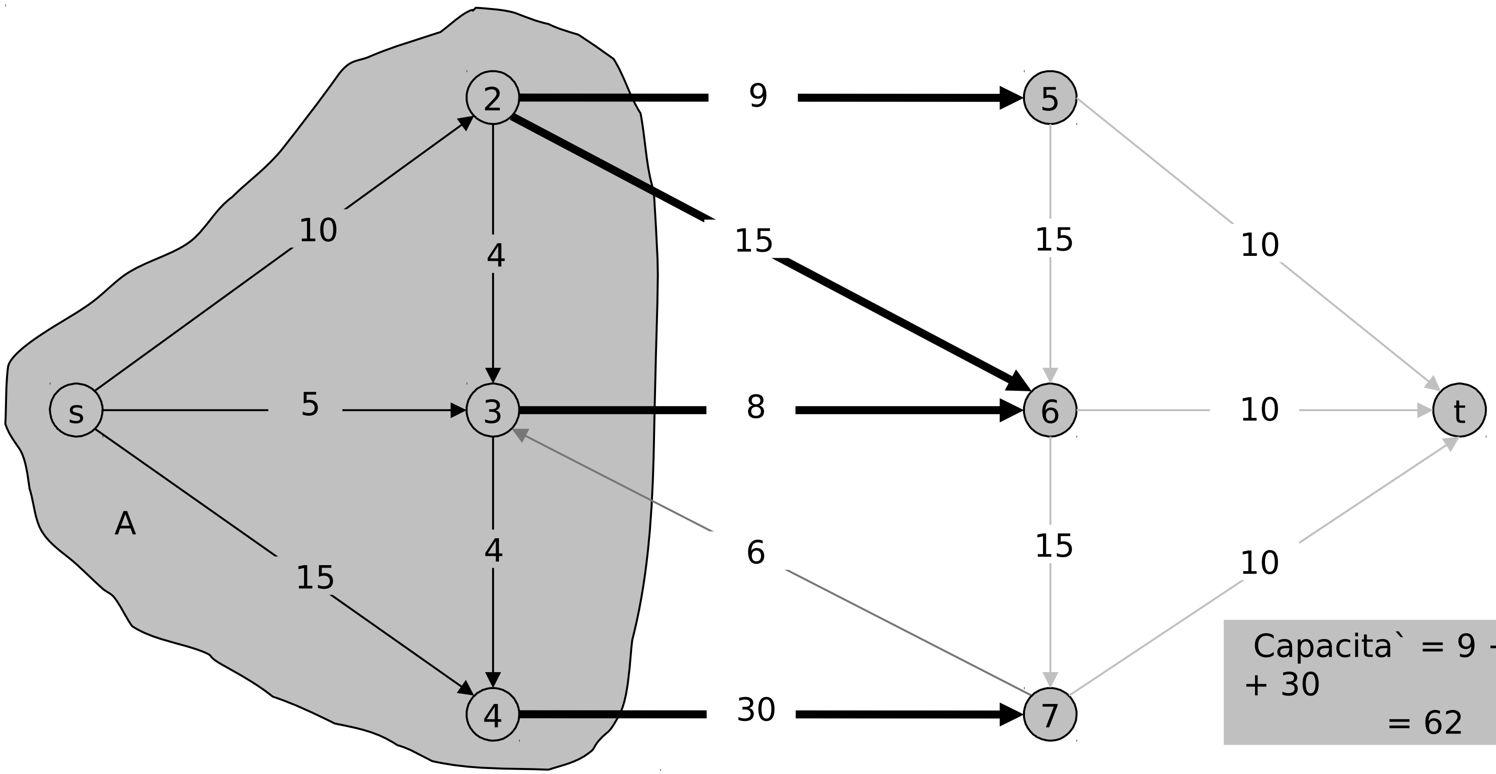
Taglio

- **Def.** Un taglio s-t (o semplicemente taglio) e` una partizione (A, B) di V con s $\in$ A e t $\in$ B.

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



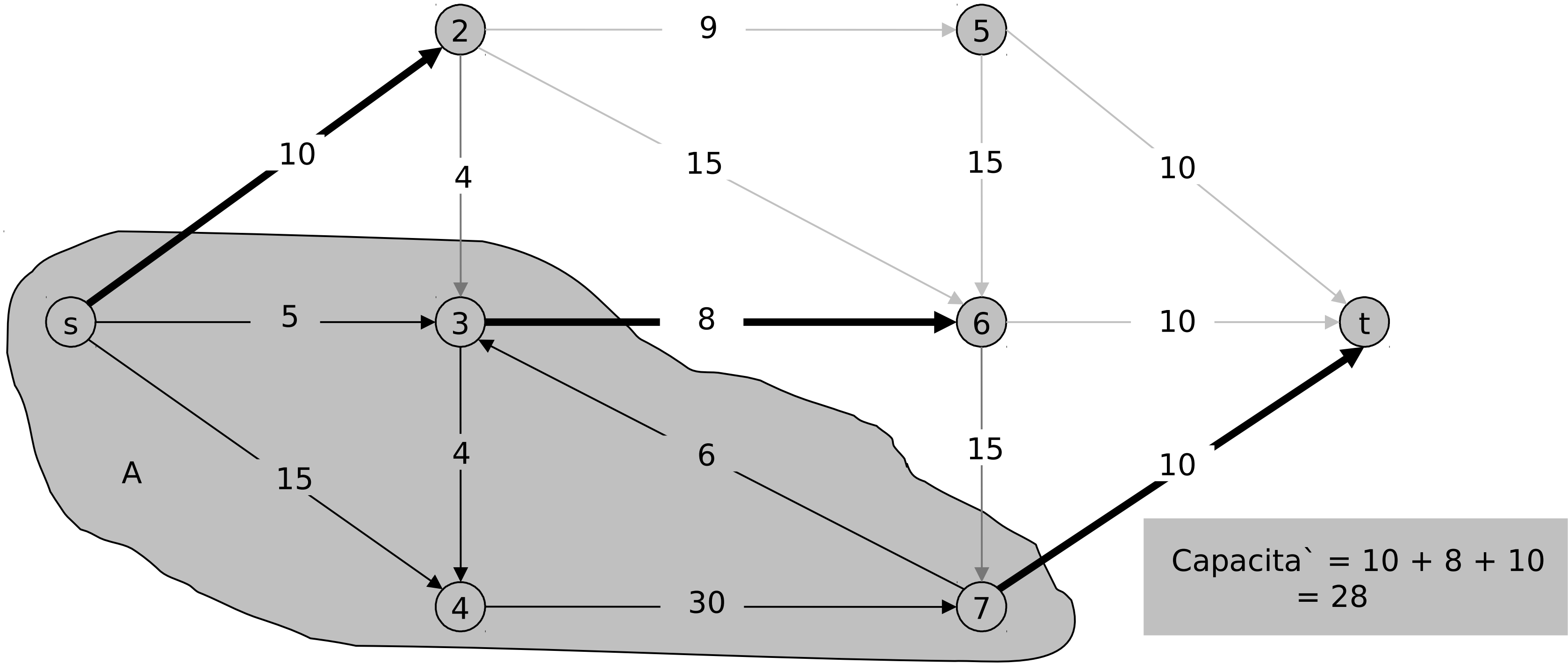
Tagli



Capacita` = 9 + 15 + 8 + 30
= 62

Il problema del minimo taglio

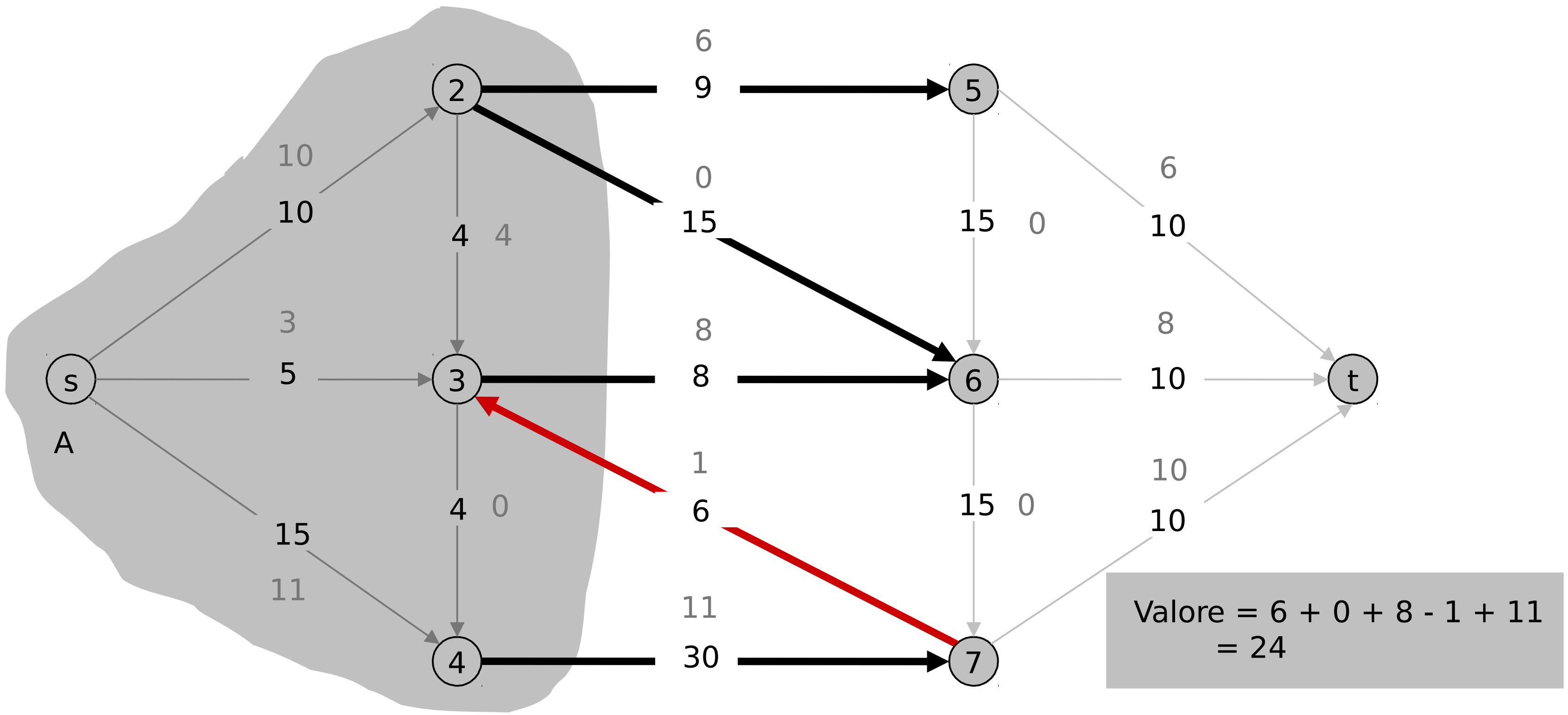
- Problema del minimo taglio s-t. Trova un taglio s-t di capacita` minima



Flussi e tagli

- Def. Sia (A,B) un qualsiasi taglio s-t. Il flusso netto inviato attraverso il taglio e` la differenza

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

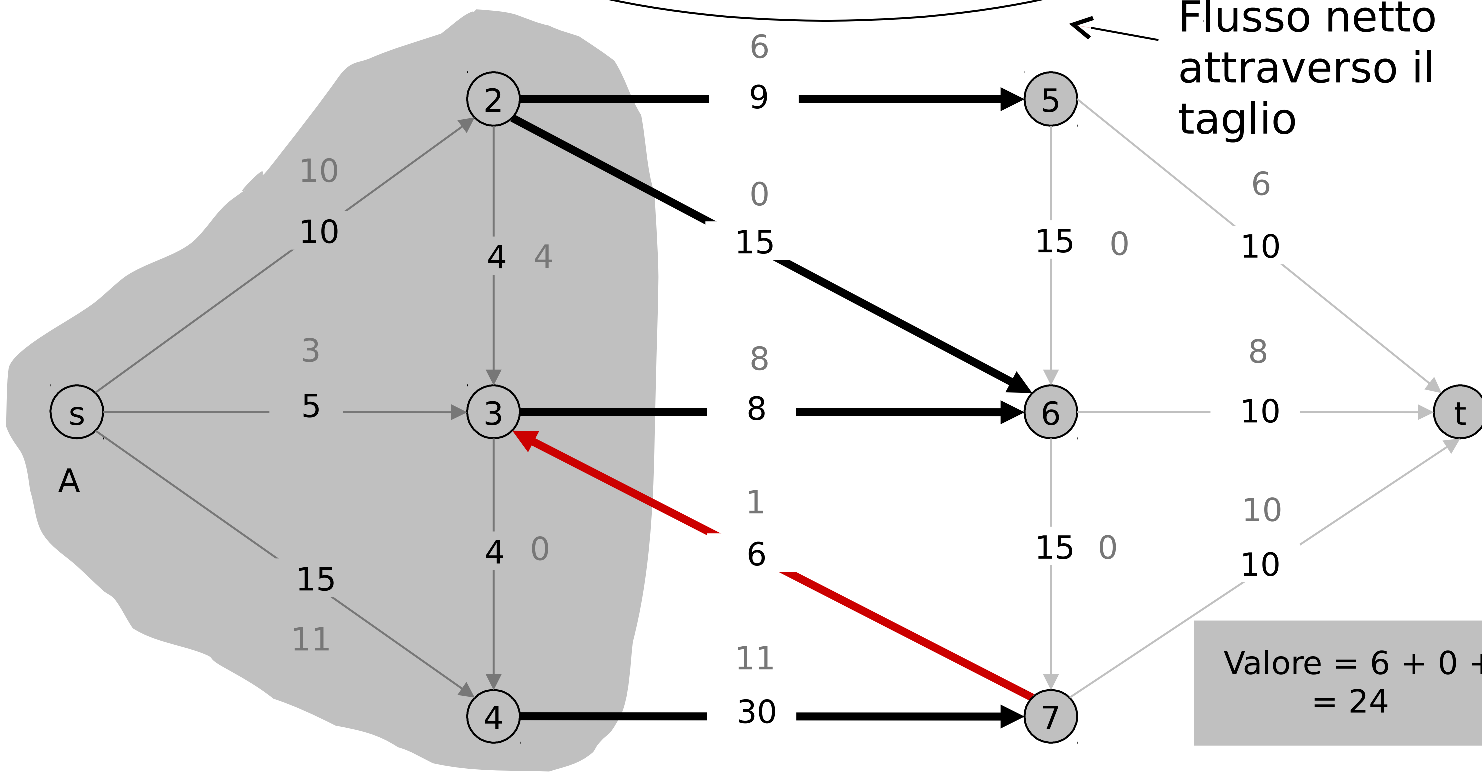


Flussi e tagli

- **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 a:

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

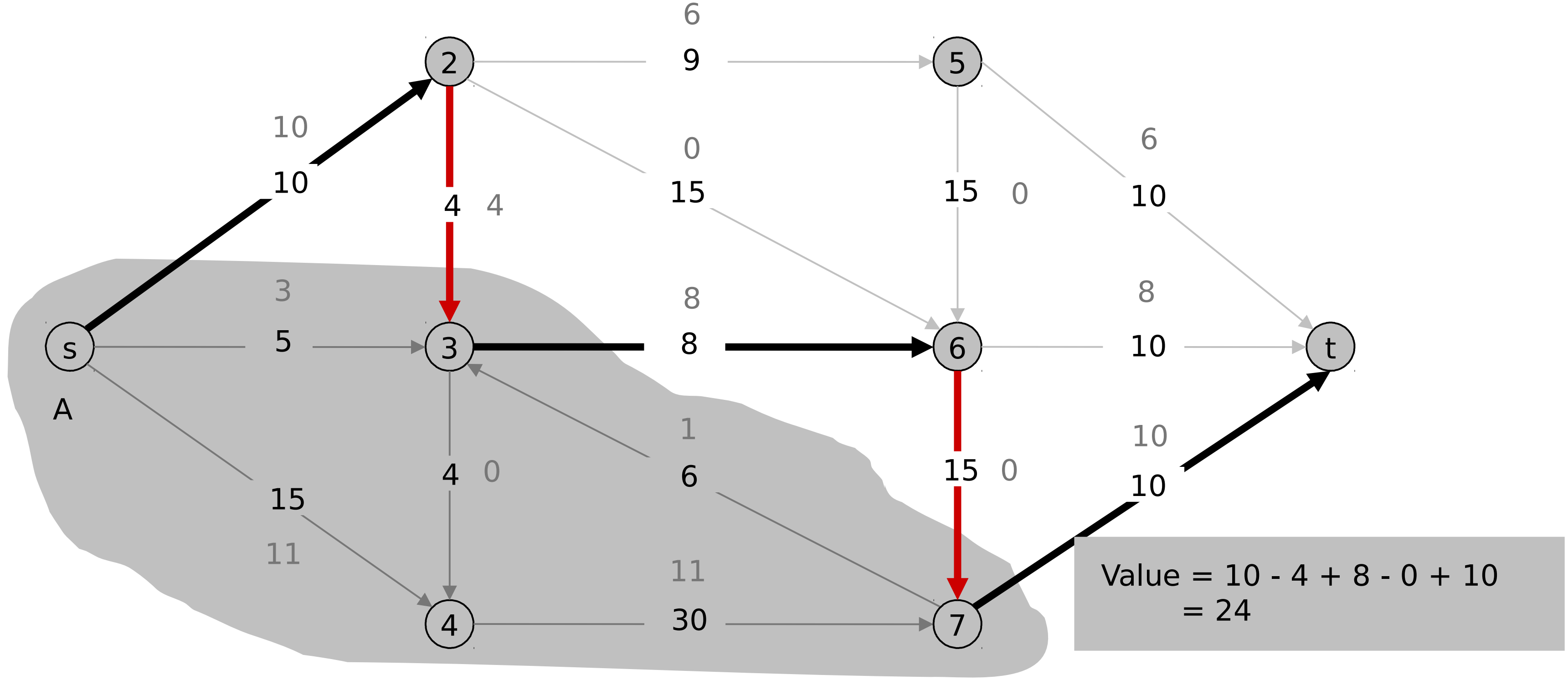
Flusso netto attraverso il taglio



Valore = 6 + 0 + 8 - 1 + 11
= 24

- 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.

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



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 al valore del flusso.

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

- Dim.

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

Dalla definizione del valore del flusso

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

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

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

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

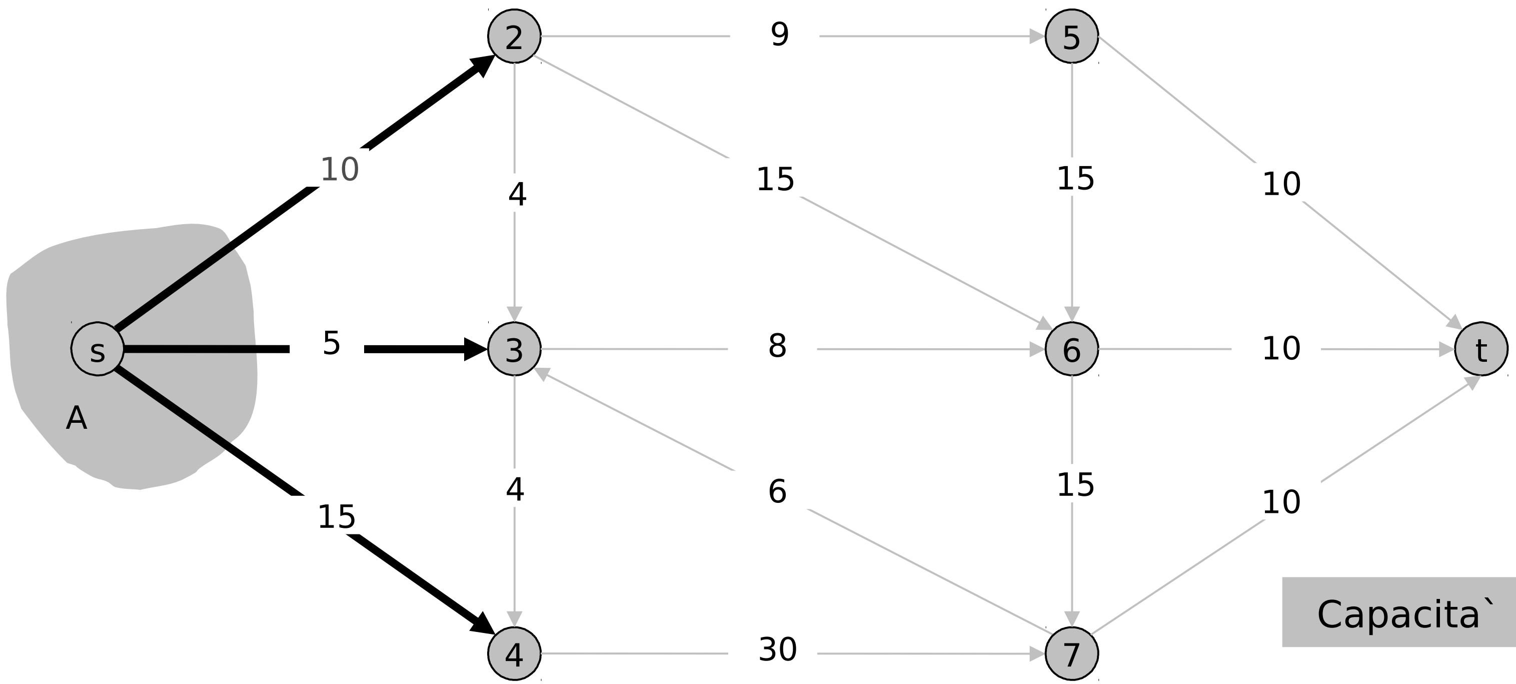
$$= \sum_{v \in A} \left(\sum_{e \text{ out of } v} f(e) - \sum_{e \text{ into v}} f(e) \right)$$

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

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.

Capacita` taglio = 30 $\Rightarrow$ Valore flusso $\leq$ 30



Capacita` = 30

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 (A,B).
- In altre parole, per ogni taglio s-t (A,B), si ha v(f) ≤ cap(A, B).

- Dim.

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

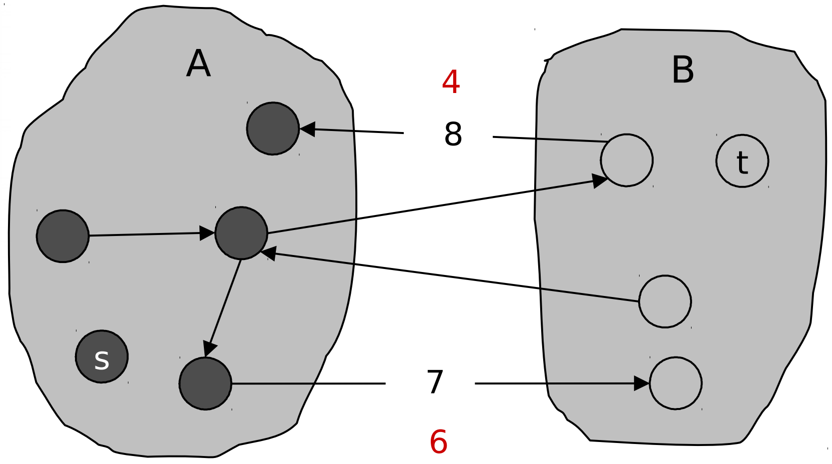
Dal lemma del valore del taglio

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

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

Dal vincolo della capacita`

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



cap(A,B)=v(f)

Massimo flusso e Minimo taglio

Corollario. Sia f un flusso di G e sia (A, B) un taglio s-t di G. Se si ha che v(f) = cap(A, B) allora f e\` un massimo flusso e (A, B) e\` un minimo taglio.

Dim.

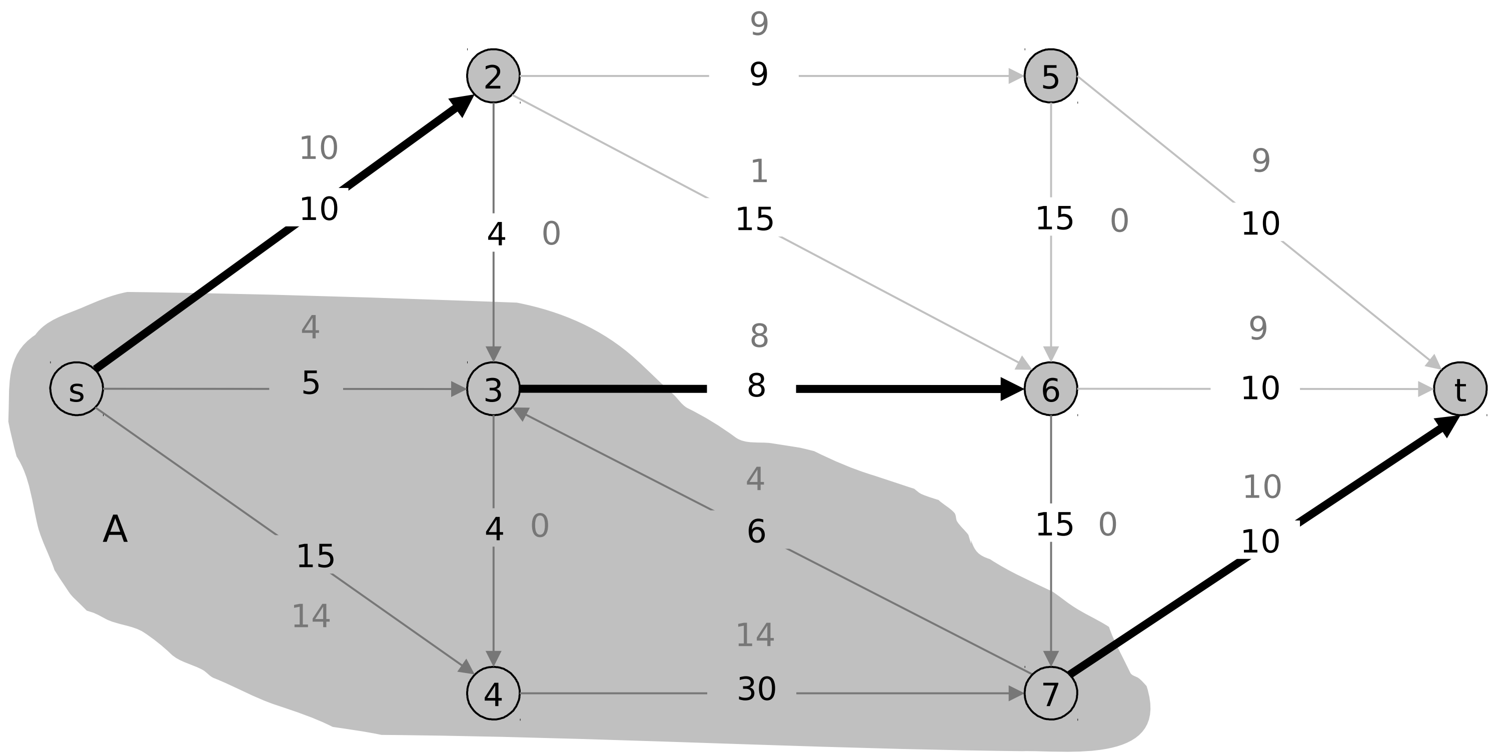
- Sia (X,Y) un qualsiasi taglio s-t e sia f' una qualsiasi funzione flusso e siano (A,B) ed f rispettivamente il taglio s-t e la funzione flusso dell'ipotesi per cui si ha che cap(A,B)=v(f).

- Risulta allora v(f') ≤ cap(A,B)=v(f) ≤ cap(X,Y),
  - la prima e l'ultima disuguaglianza sono conseguenza della dualita\` debole

- Dal punto precedente si ha che

1. per ogni flusso f', v(f) e\` maggiore o uguale di v(f') per cui f e\` un flusso massimo
2. Per ogni taglio s-t (X,Y), cap(A,B) e\` minore o uguale di cap(X,Y) per cui (A,B) e\` un taglio con capacita\` minima.

Massimo flusso e Minimo taglio

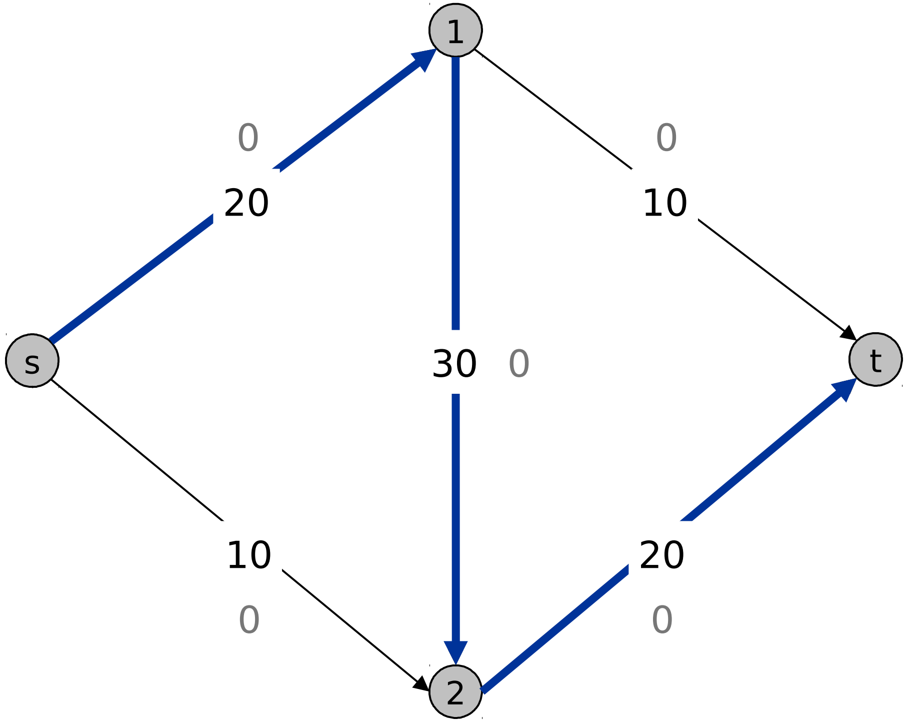


Valore del flusso= 28

Capacita` del taglio = 28 $\Rightarrow$ Valore del flusso $\leq$ 28

Algoritmo per il max flusso: approccio sbagliato

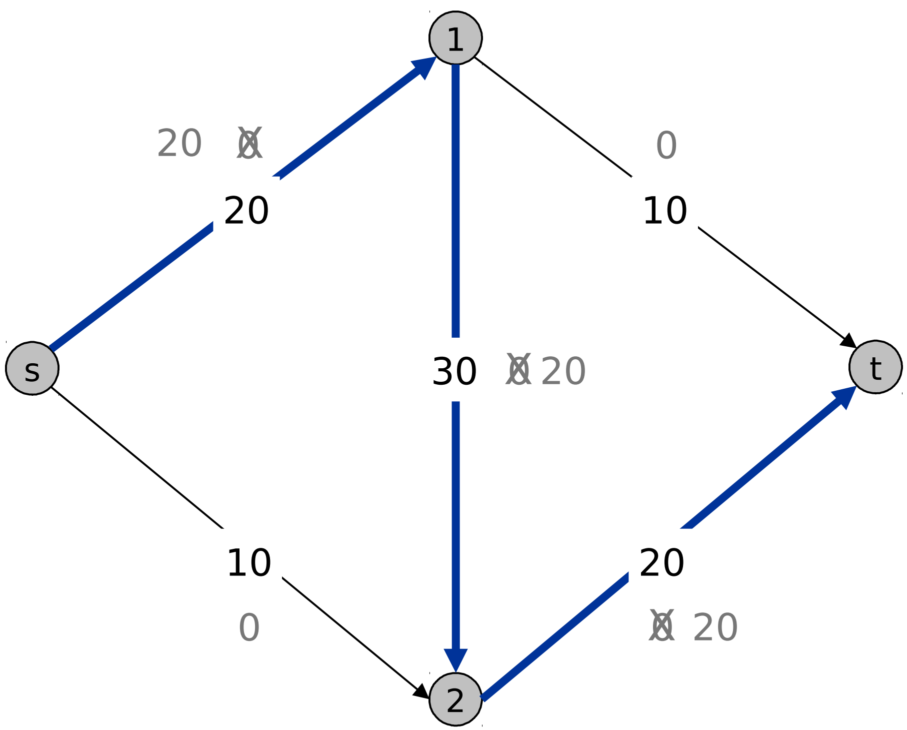
- Algoritmo greedy.
  - Cominciamo con f(e) = 0 per ogni arco e $\in$ E.
  - Troviamo un percorso P da s a t dove per ogni arco sul percorso si ha f(e) < c(e).
  - Aumentiamo il flusso lungo il percorso P in modo da rispettare il vincolo della capacita` e quello sulla conservazione del flusso.
  - Ripetiamo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

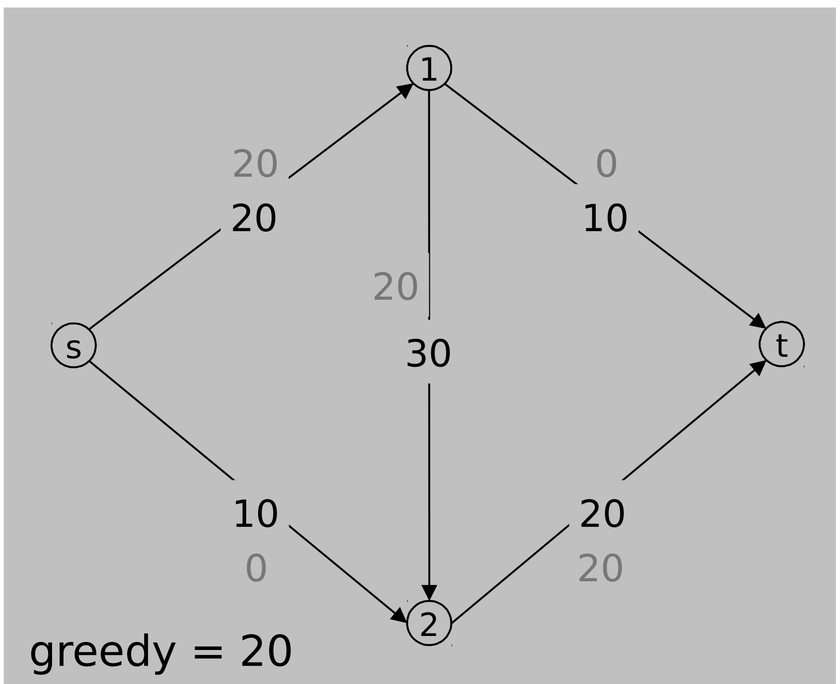
- Algoritmo greedy.
  - Cominciamo con $f(e) = 0$ per ogni arco $e \in E$.
  - Troviamo un percorso P da s a t dove per ogni arco sul percorso si ha $f(e) < c(e)$.
  - Aumentiamo il flusso lungo il percorso P in modo da rispettare il vincolo della capacita` e quello sulla conservazione del flusso.
 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

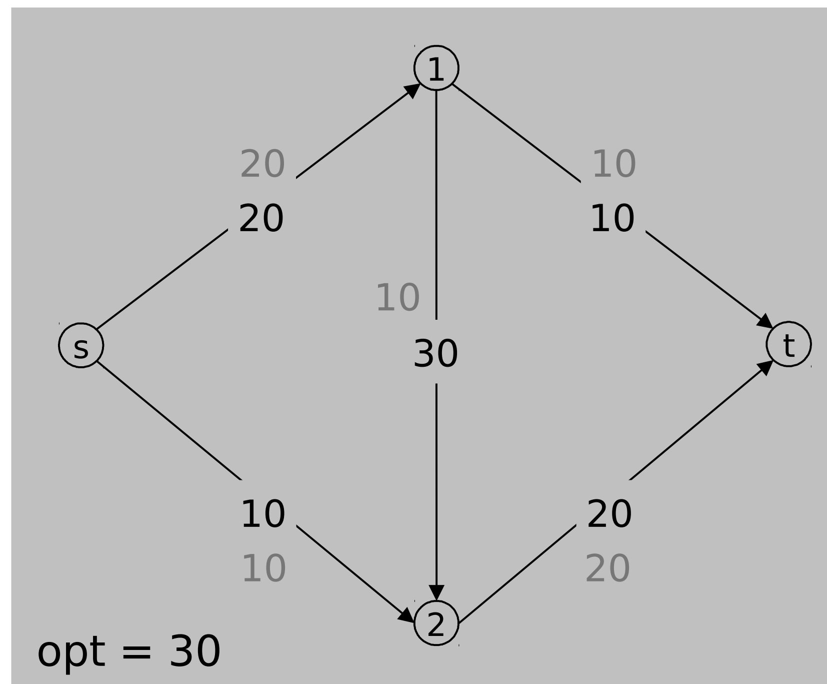


Valore del flusso= 20

Algoritmo per il max flusso: approccio sbagliato

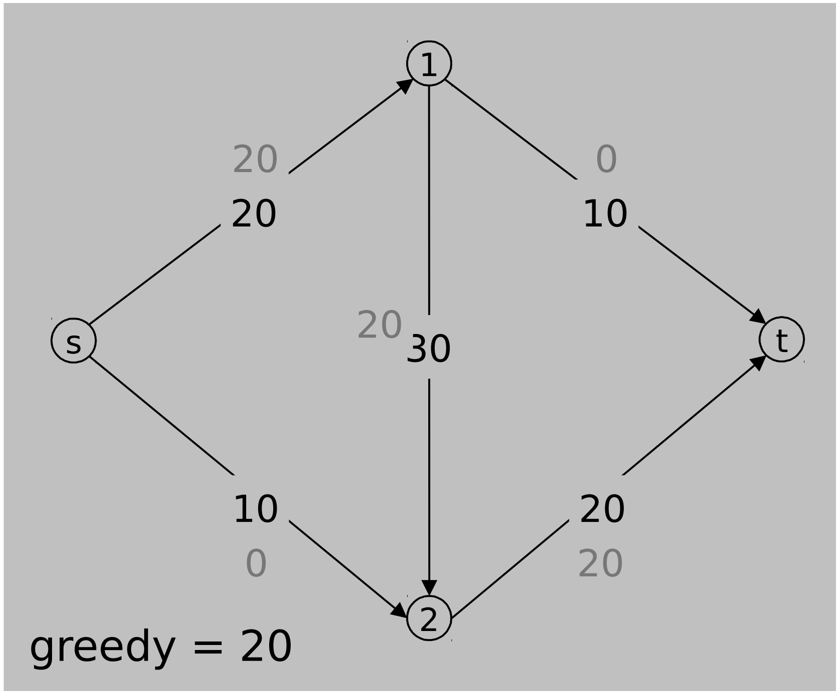
- Algoritmo greedy.
  - Cominciamo con $f(e) = 0$ per ogni arco $e \in E$.
  - Troviamo un percorso P da s a t dove per ogni arco sul percorso si ha $f(e) < c(e)$.
  - Aumentiamo il flusso lungo il percorso P in modo da rispettare il vincolo della capacita` e quello sulla conservazione del flusso.
 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

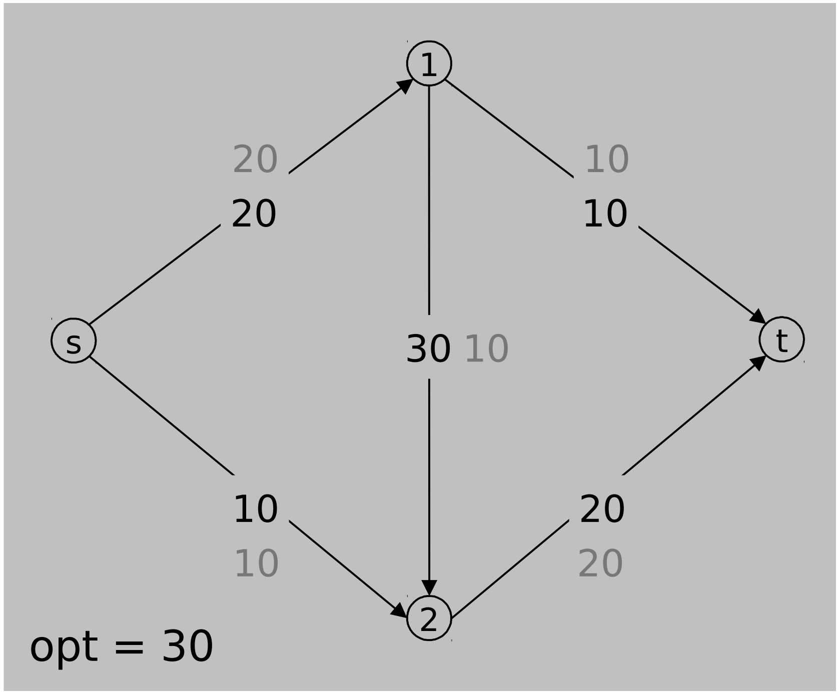




Algoritmo per il max flusso: approccio sbagliato

- Il problema con l'algoritmo di prima e` 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 piu` una quantita` pari al flusso f(e) spinto lungo e.



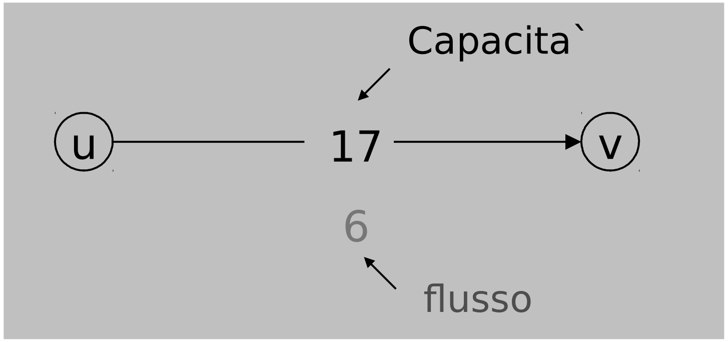


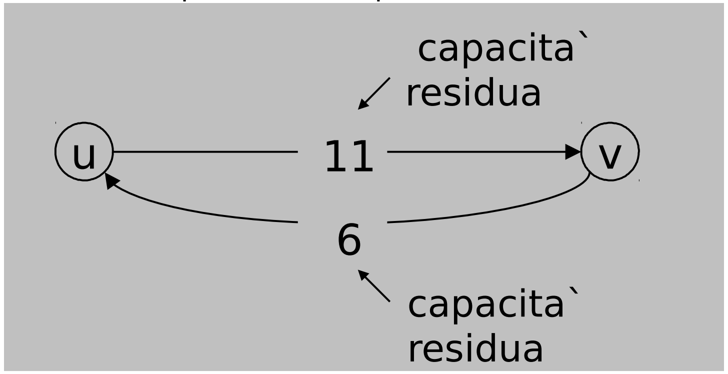
Grafo residuo

- Sia data una rete di flusso $G=(V,E)$
  - Per ogni arco (originale) $e = (u, v) \in E$, indichiamo con $c(e)$ la sua capacita`.
- Sia data una funzione flusso f.
- **Notazione: Per ogni arco $e=(u,v)$, indichiamo con e^R l'arco (v,u)**

Def. Insieme di archi residui E_f rispetto ad f:

- per ogni arco (originale) $e = (u, v) \in E$,
  - 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 $e^R=(v,u) \in E_f$ e ha capacita` $c_f(e^R)=f(e)$.
- Quindi $E_f = \{e: e\in E, f(e) < c(e)\} \cup \{e^R : e\in E, f(e) > 0\}$
- Le capacita` $c_f$ degli archi residui vengono dette capacita` residue.
- Def. Grafo residuo di G rispetto ad f: $G_f = (V, E_f)$.





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).
    - e in questo caso si ha $c_f(u,v)=c(u,v)-f(u,v)$

oppure se

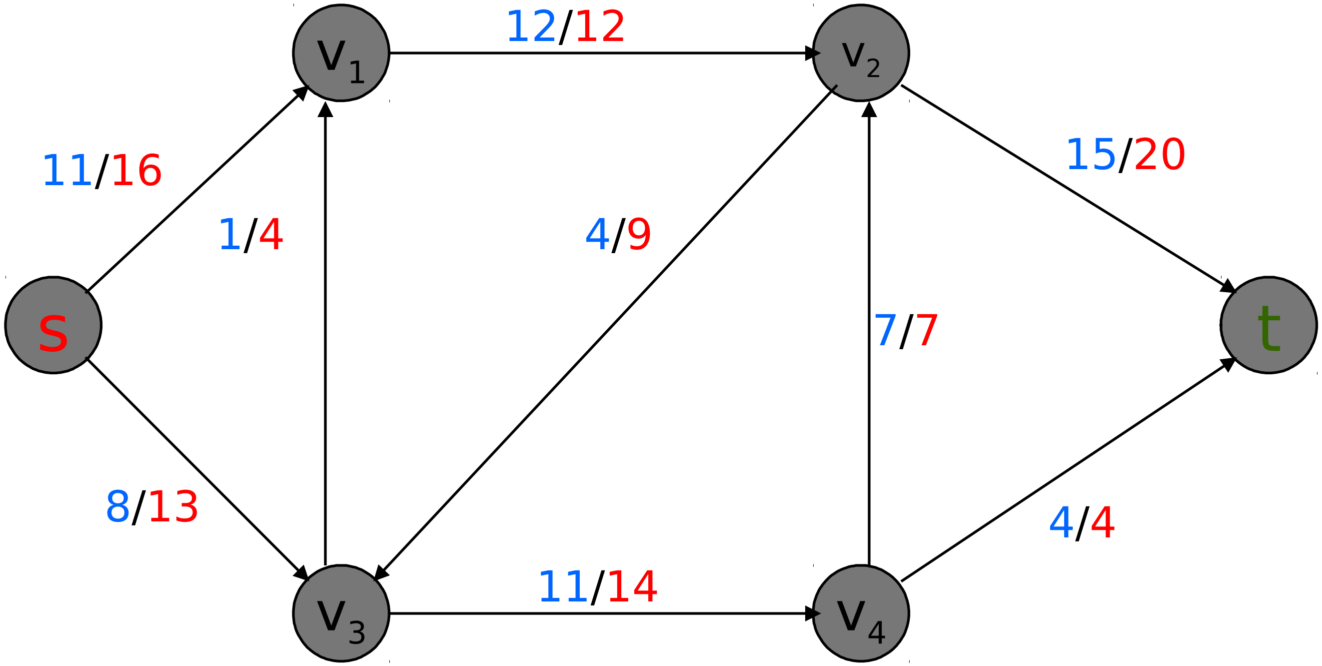
  - (v,u) $\in$ E e f(v,u)>0.
    - e in questo caso si ha $c_f(u,v)=f(v,u)$
    - L'arco (u,v) incluso in questo modo di E_f viene detto arco backward

- Ne consegue che $|E_f| \leq 2|E|$

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
- bottleneck(P,f) = min{c_f(u,v) : (u,v) è un arco in P}
- La capacità residua rappresenta la massima quantità di flusso che si può trasportare lungo P
- Se esiste un cammino aumentante P in G_f e\` quindi possibile aumentare il valore del flusso di una quantita\` pari a bottleneck(P,f)
- Si computa una nuova funzione di flusso f’ che differisce da f solo per i valori assegnati agli archi e (del grafo originario G) tali che e o e^R si trovano lungo P.
  - Se e= (u,v) e\` un arco del grafo originario G e (u,v) si trova lungo P allora f’(u,v)=f(u,v)+ bottleneck(P,f)
  - Se e=(u,v) e\` un arco del grafo originario G e (v,u) si trova lungo P allora f’(u,v)=f(u,v)- bottleneck(P,f)

Esempio di rete di flusso

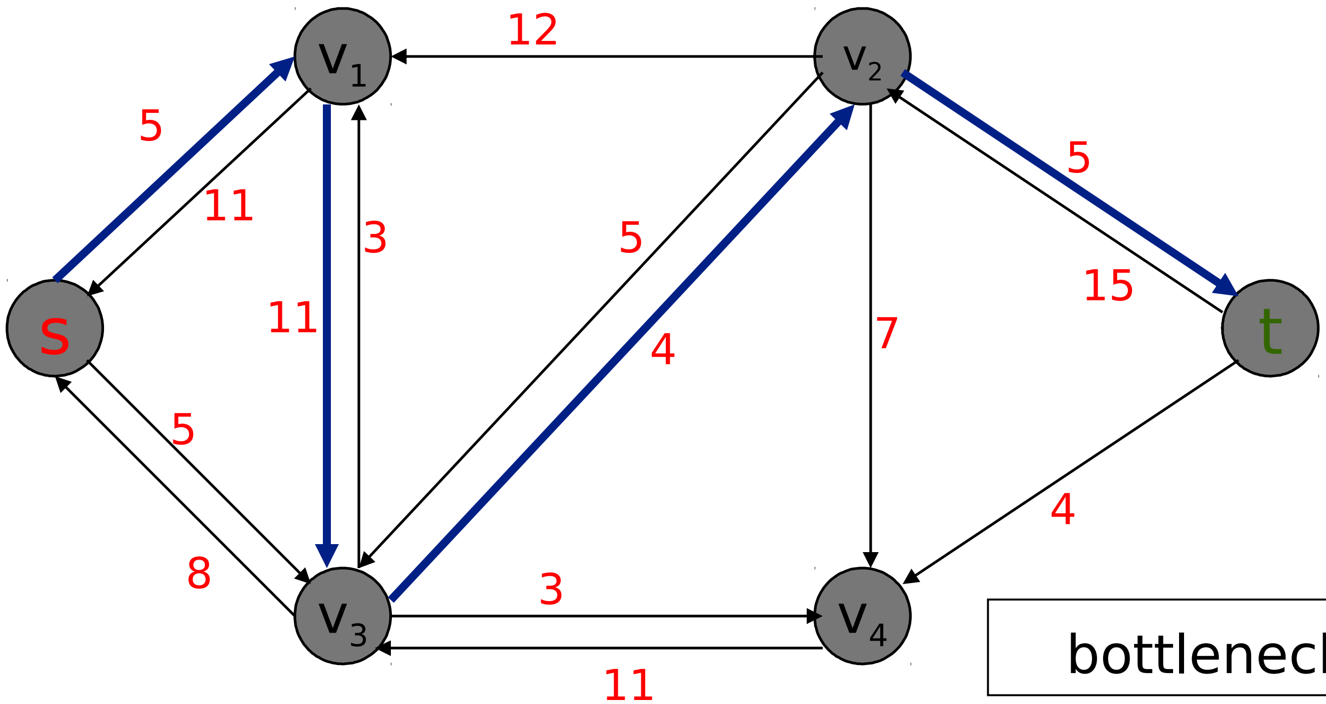


$$c_f(s,v_1)=c(s,v_1)-f(s,v_1)=16-11=5$$

$$c_f(v_1,s)=c(v_1,s)-f(v_1,s)=0-(-11)=11$$

Esempio di augmentig path

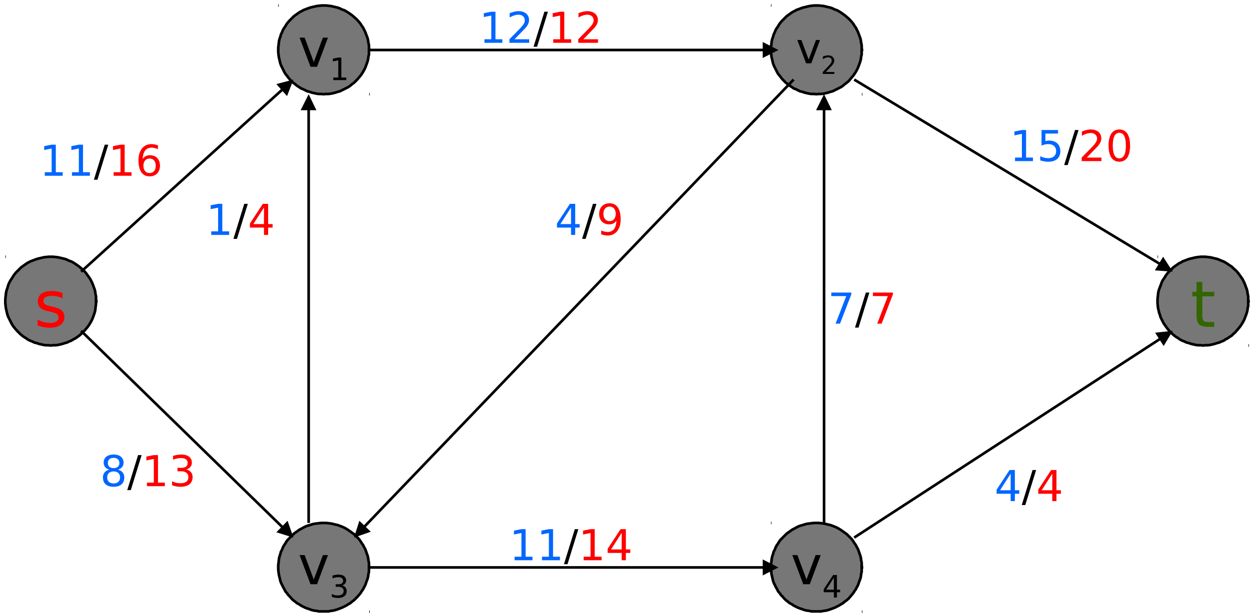
Rete residua della rete disegnata nella slide precedente (rispetto alla funzione flusso li` data)



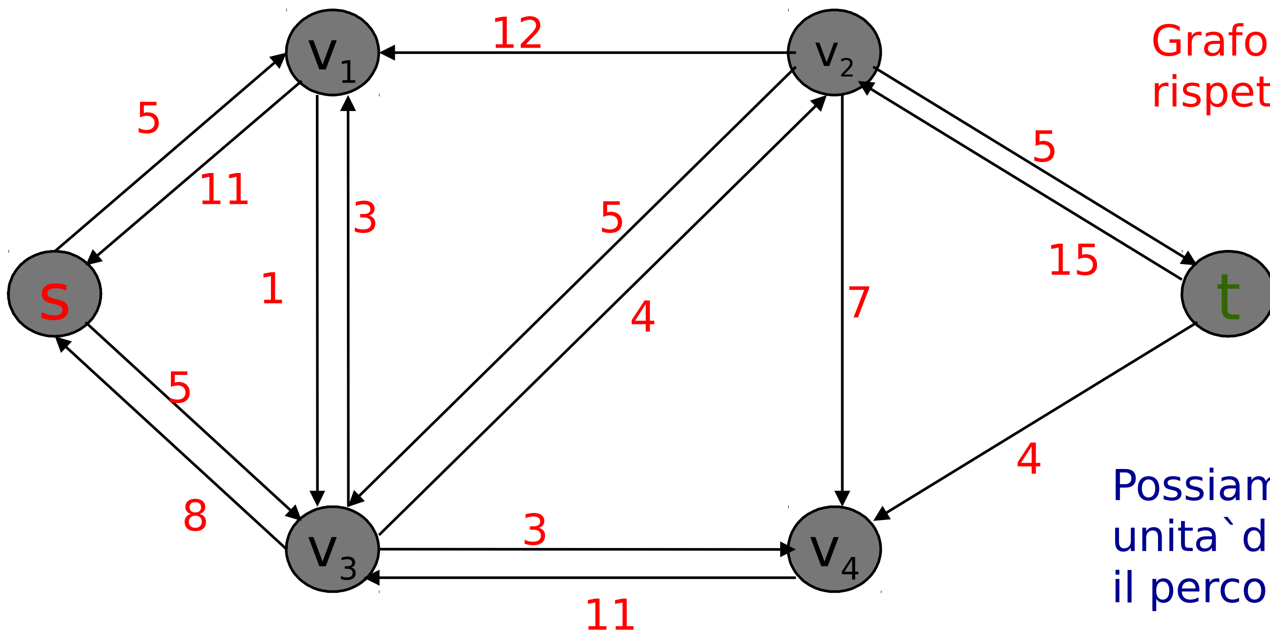
bottleneck(P,f)=4

P=(s,v$_1$)(v$_1$,v$_3$)(v$_3$,v$_2$)(v$_2$,t)

Esempio di come si usa la rete residua per aumentare il valore del flusso



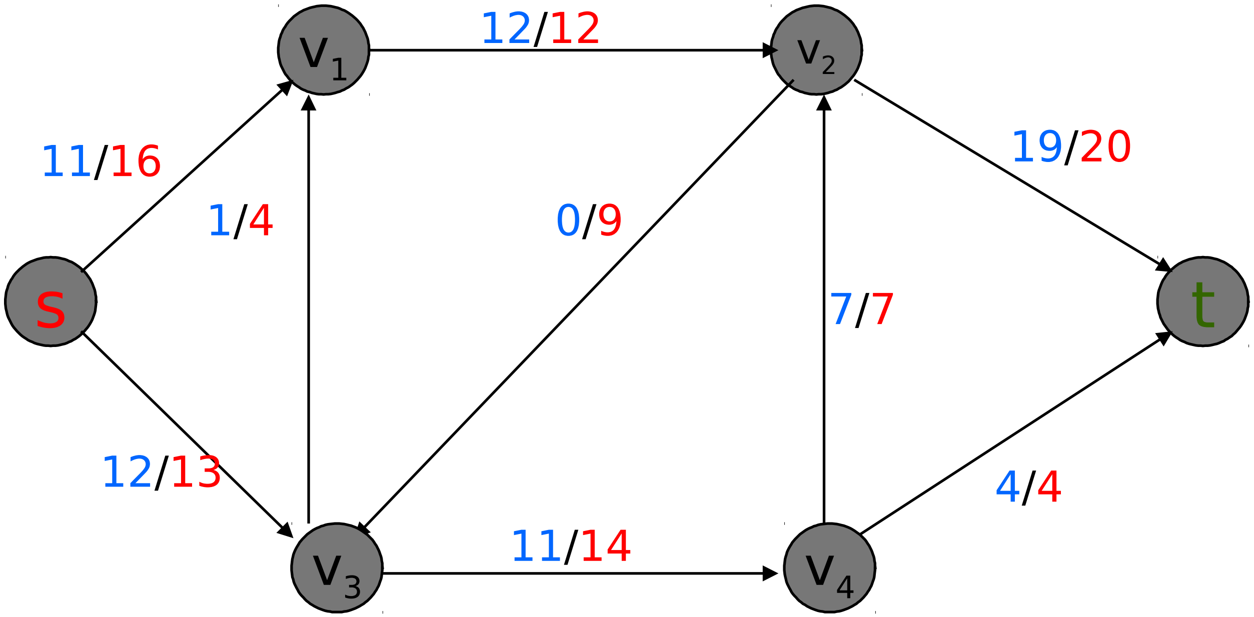
Grafo di partenza con flusso f



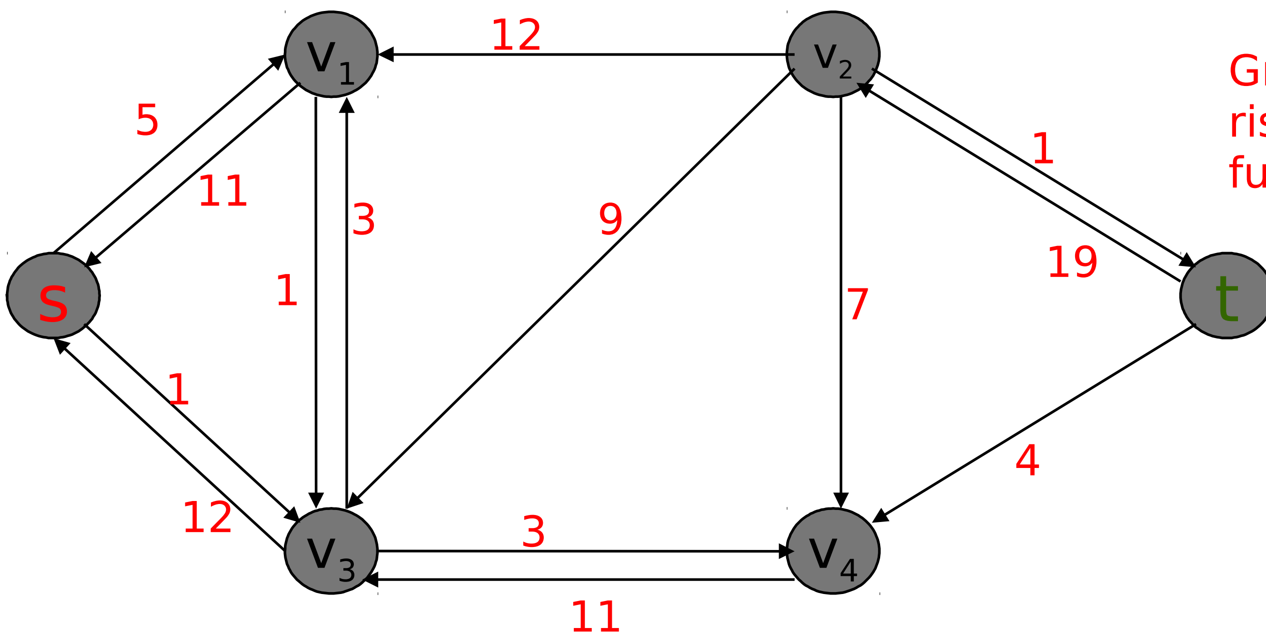
Grafo residuo rispetto ad f

Possiamo spingere 4 unita`di flusso lungo il percorso s, v_3 , v_2 , t

Esempio di come si usa la rete residua per aumentare il valore del flusso



Grafo originale con i valori del flusso aggiornati. Chiamiamo f’ la nuova funzione flusso.



Grafo residuo rispetto alla nuova funzione flusso f’

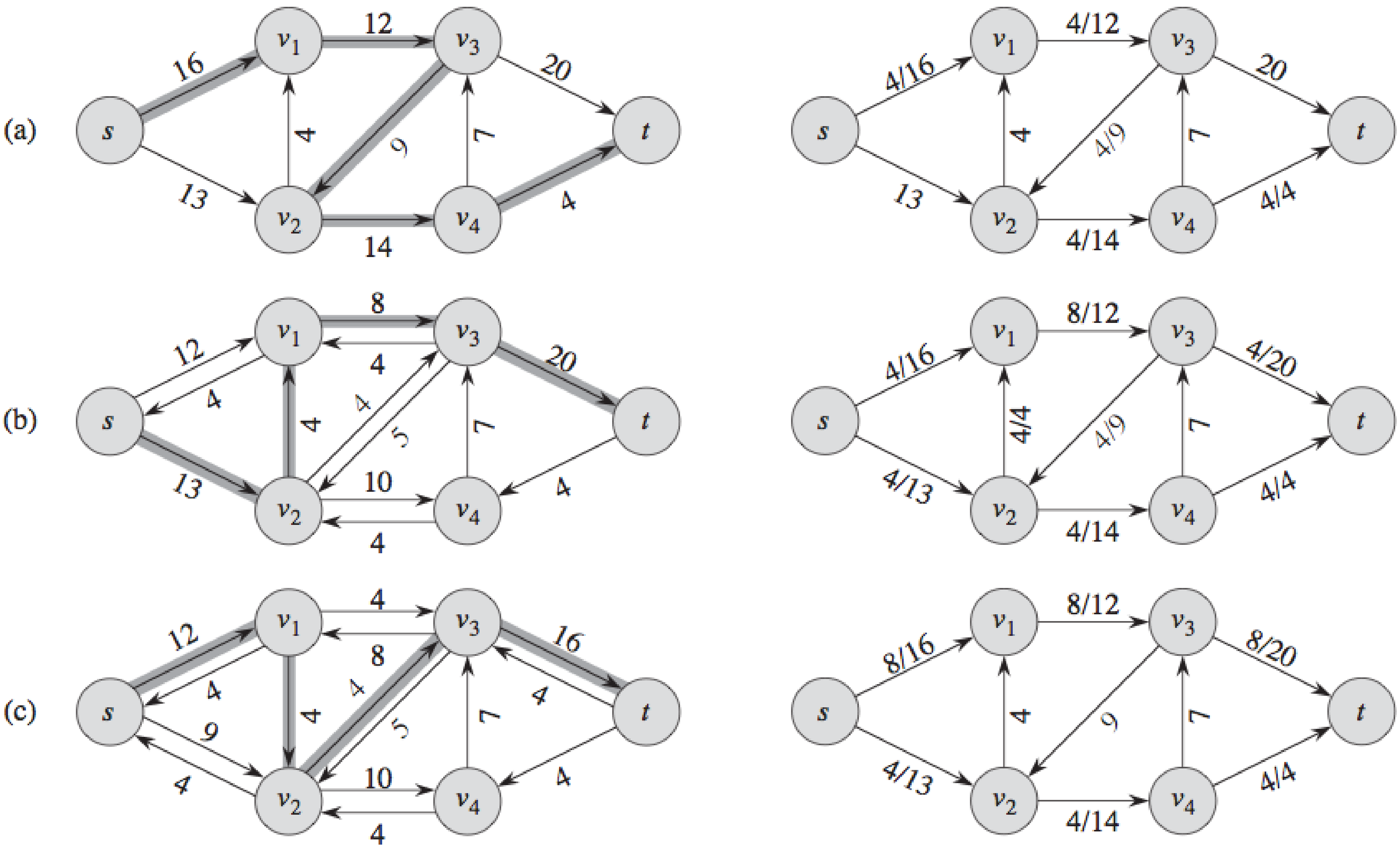
Algorithm di Ford-Fulkerson

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

```
Ford-Fulkerson(G, s, t, c) {
   foreach e ∈ E  f(e) ← 0
   G_f ← residual graph with respect to f

   while (there exists augmenting path P in G_f) {
      f ← Augment(f, c_f , P)
      update G_f
   }
   return f
}
```

Esempio



(a)
s
v1
v2
v3
v4
t
16
12
20
4
9
7
13
14
4
4/16
4/12
20
4
4/9
7
13
4/14
4/4
(b)
12
8
4
20
4
4
4
5
7
13
10
4
4
4/16
8/12
4/20
4/4
4/9
7
4/13
4/14
4/4
(c)
12
4
8
16
4
9
4
4
5
7
4
4
10
4
4
8/16
8/12
8/20
4
9
7
4/13
4/14
4/4

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 **`Augment(f, c, P)`**. Allora f' è un flusso in G con valore $val(f')=val(f)+bottleneck(P,f)>val(f)$

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) = 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)

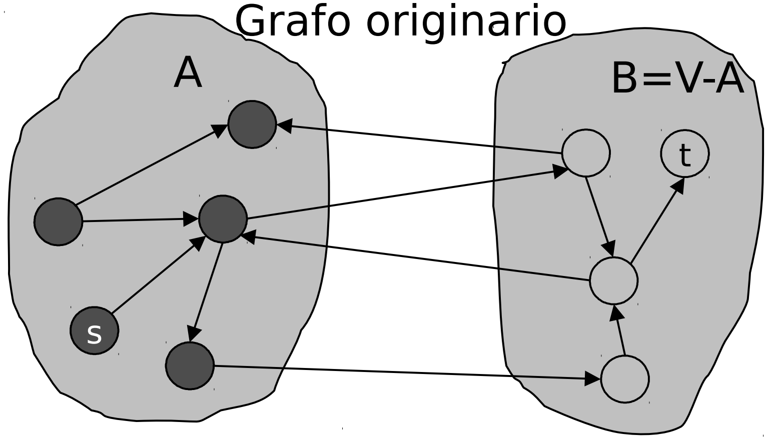
  - Se esistesse un percorso aumentante in G_f allora potremmo aumentare il flusso spingendo altro flusso lungo il cammino aumentante e di conseguenza f non sarebbe massimo. Infatti il teorema precedente implica che il nuovo flusso avrebbe valore = f + bottleneck(P,f) > f

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 dei 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 e` raggiungibile da s e di conseguenza t $\notin$ A. Quindi (A,B) con B=V-A e ` un taglio s-t.
  - Possiamo allora applicare il lemma del valore del taglio al flusso f e al taglio (A,B)

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



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(v,u)>0. Questo vuol dire che $c_f(u,v)>0$ e quindi (u,v) e` in $G_f$ e di conseguenza v e` in anch'esso in A. Contraddizione!

Conseguenze del teorema

- Un flusso f e` massimo se e solo se non ci sono cammini aumentanti

- Il valore del massimo flusso e` uguale alla capacita` 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 capacita` intere

- Assunzione. Tutte le capacita` sono interi tra uno e C.
  → somma capacita` archi uscenti da s ≤n C → valore massimo flusso $v(f^*) \leq nC$

- Invariante. Ciascun valore del flusso f(e) e ciascuna capacita` residua $c_f(e)$ rimane un intero durante l'intera esecuzione dell'algoritmo di Ford-Fulkerson

- Teorema. L'algoritmo termina in $v(f^*)$ iterazioni.
  Dim. Ogni iterazione aumenta il flusso di uno

Tempo di esecuzione nel caso di capacita` intere

- Teorema. Sia f^* la funzione di massimo flusso e supponiamo che la capacita` degli archi sia compresa tra 1 e C. L'algoritmo di Ford-Fulkerson ha tempo di esecuzione $O(m \times v(f^*)) = O(mnC)$

- Dim.
- Ogni iterazione del while impiega
  - Tempo $O(n+m)=O(m)$ per trovare il cammino aumentante con BFS o DFS.
  - Tempo $O(n)=O(m)$ per Augment
  - Tempo $O(m)$ per costruire il nuovo grafo residuo.
- Dal teorema nella pagina precedente si ha che il numero di iterazioni del while e` al piu` $v(f^*)$ per cui il tempo di esecuzione del while e` $O(m \times v(f^*))$. Abbiamo osservato che $v(f^*) \leq nC$ per cui tale tempo e` $O(m \times nC)$.

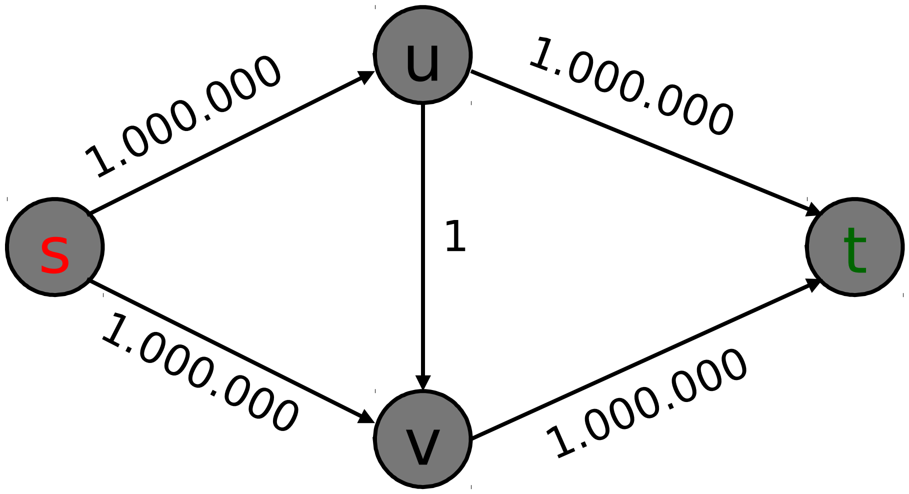
- Corollario. Se C=1, Ford-Fulkerson ha tempo di esecuzione $O(mn)$.

Tempo di esecuzione nel caso di capacita` intere

Teorema. Se tutte le capacita` sono intere allora il valore del flusso max v(f*) e` intero ed esiste una funzione flusso f con valore v(f*) tale che f(e) e` un intero per ogni arco e

- Dim. Basta considerare l'algoritmo di Ford-Fulkerson:
  - l'algoritmo termina quando non ci sono piu` cammini aumentanti e, per il teorema del massimo flusso e minimo taglio, il flusso restituito e` max.
  - abbiamo osservato che, in presenza di capacita` intere, il valore del flusso restituito dall'algoritmo di Ford-Fulkerson e` intero e assegna ad ogni arco e un valore f(e) intero.
  - I due punti precedenti implicano che la funzione di flusso restituita in output dall'algoritmo di Ford-Fulkerson ha valore max v(f*) ed e` tale che f(e) e` un intero per ogni arco e

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 → 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 $O(nm^2)$