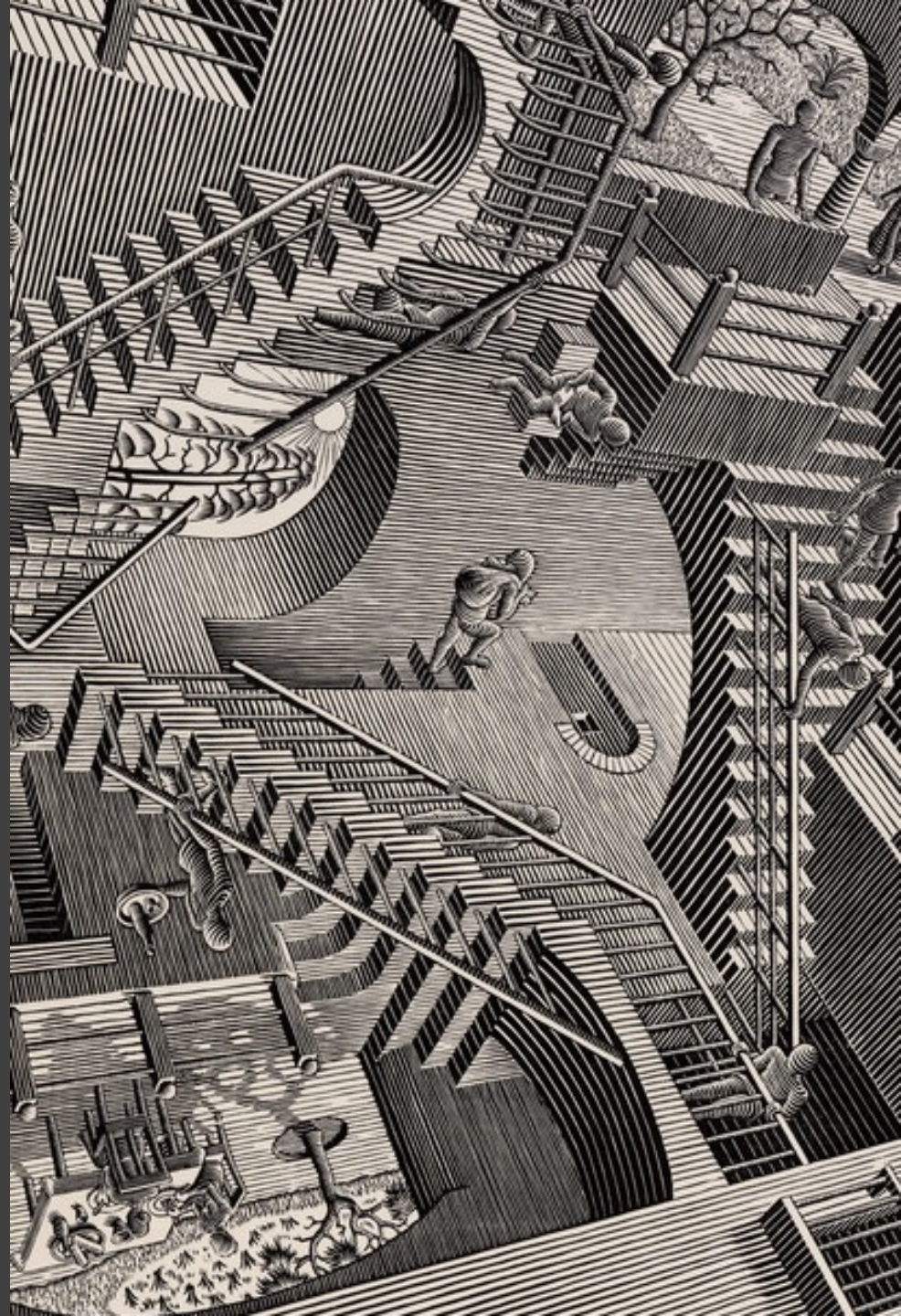
Università Roma Tre – Dipartimento di Matematica e Fisica
IN440 – Ottimizzazione Combinatoria
Prof. Marco Liverani

Flusso massimo su una rete

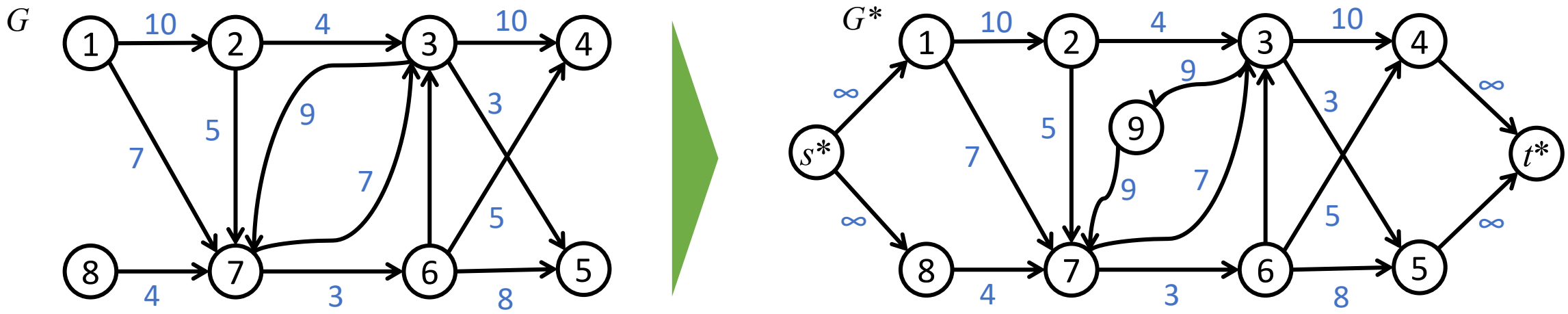


Reti di flusso

- Una **rete di flusso** è una quaterna (G, s, t, c), dove:
  - $G = (V, E)$ è un **grafo orientato**;
  - $s \in V(G)$ è una **sorgente** di G, ossia un vertice privo di spigoli entranti
  - $t \in V(G)$ è un **pozzo** di G, ossia un vertice privo di spigoli uscenti
  - $c\colon V \times V \longrightarrow \mathbb{R}^+$ è una **funzione capacità**, tale che se $(u, v) \notin E(G)$ risulta $c(u, v) = 0$
- La rete di flusso è tale che se $(u, v) \in E(G)$ allora $(v, u) \notin E(G)$
  - Se il grafo G ha sia lo spigolo (u, v) che lo spigolo (v, u), allora si introduce un nuovo vertice fittizio z e si sostituisce lo spigolo (v, u) con i due spigoli (v, z) e (z, u) assegnando ad entrambi la capacità $c(v, u)$
- Se per ogni $v \in V(G)$ esiste un cammino $p\colon s \rightsquigarrow v \rightsquigarrow t$ allora la rete è connessa e risulta $|E(G)| \geq |V(G)| - 1$
- Un **flusso** è una funzione $f\colon V \times V \longrightarrow \mathbb{R}$ tale che
  - **simmetria**: $f(u, v) = -f(v, u)$
  - **vincolo sulle capacità**: per ogni $u, v \in V(G)$ risulta $f(u, v) \leq c(u, v)$
  - **conservazione del flusso**: per ogni $u \in V(G) \setminus \{s, t\}$ risulta $\sum_{v \in V(G)} f(u, v) = \sum_{v \in V(G)} f(v, u)$

Reti di flusso

- Se la rete ha più di una sorgente si aggiunge una «*super-sorgente*» s^* con spigoli uscenti da s^* entranti nelle sorgenti, con capacità infinita; analogamente nel caso in cui sia presente più di un pozzo, si aggiunge un «*super pozzo*» t^* collegato ai pozzi con spigoli di capacità infinita



Flusso massimo su una rete

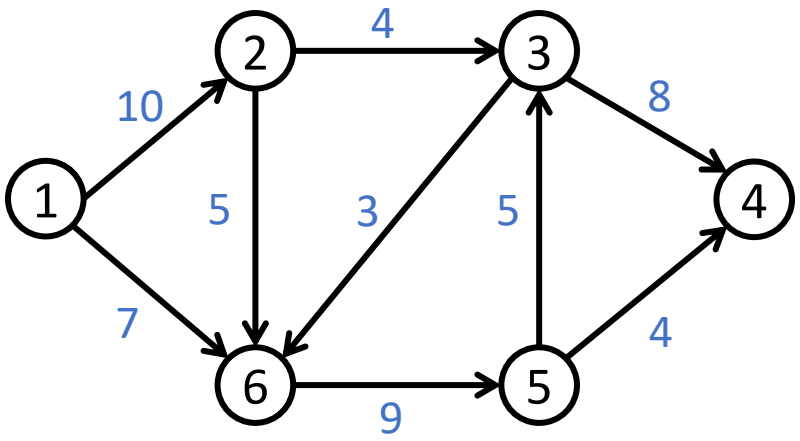
- Il **flusso della rete** è dato dal flusso totale che esce dalla sorgente:

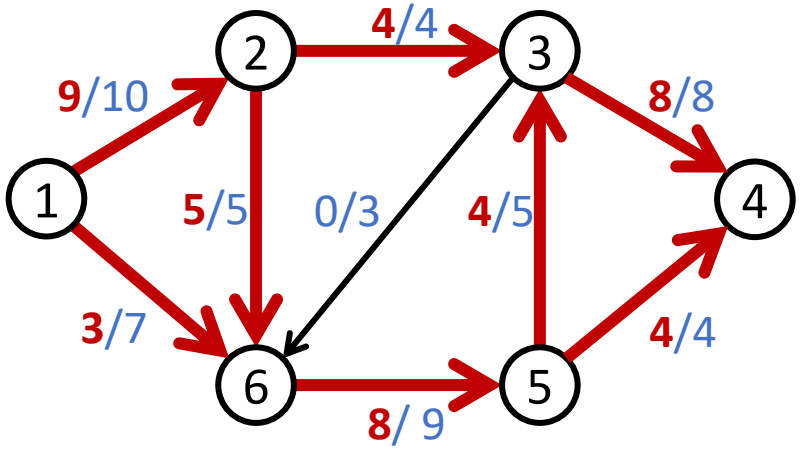
$$|f| = \sum_{v \in V(G)} f(s, v)$$

- Data una rete di flusso (G, s, t, c) il problema di ottimizzazione combinatoria che si intende risolvere chiede di individuare il massimo flusso che può essere trasferito nell'unità di tempo attraverso la rete dalla sorgente s al pozzo t
- Il problema è molto diverso dal problema del cammino di costo minimo dal vertice s al vertice t
  - nel caso del problema del cammino di costo minimo, si chiede di individuare **un solo cammino** (di costo minimo) da s a t, tra tutti i possibili cammini esistenti tra i due vertici
  - nel caso del problema di flusso massimo, si chiede di calcolare il massimo flusso che può passare attraverso la rete senza superare i limiti di capacità degli spigoli, utilizzando contemporaneamente **tutti i possibili cammini** da s a t

Flusso massimo su una rete

- Esempio:
  - il flusso passa attraverso gli spigoli senza mai superare la capacità
  - il flusso entrante in ogni spigolo è uguale al flusso uscente (ad eccezione della sorgente e del pozzo)
  - il valore del flusso massimo che passa attraverso la rete è $|f| = 12$





Capacità residua, rete residua, cammino aumentante

- Data una rete di flusso (G, s, t, c) e un flusso $|f|$ calcolato sugli spigoli di G, la **capacità residua** della rete $c_f(u, v)$ sugli spigoli di G è la capacità rimanente dopo aver fatto passare il flusso f sullo spigolo (u, v):

$$c_f(u, v) = c(u, v) - f(u, v)$$

- La **rete residua** di G indotta dal flusso f è una rete di flusso (G_f, s, t, c_f) dove:
  - c_f è la capacità residua della rete dopo aver fatto passare il flusso f su G
  - $G_f = (V, E_f)$, $E_f = \{(u, v) \in V \times V: c_f(u, v) > 0\}$

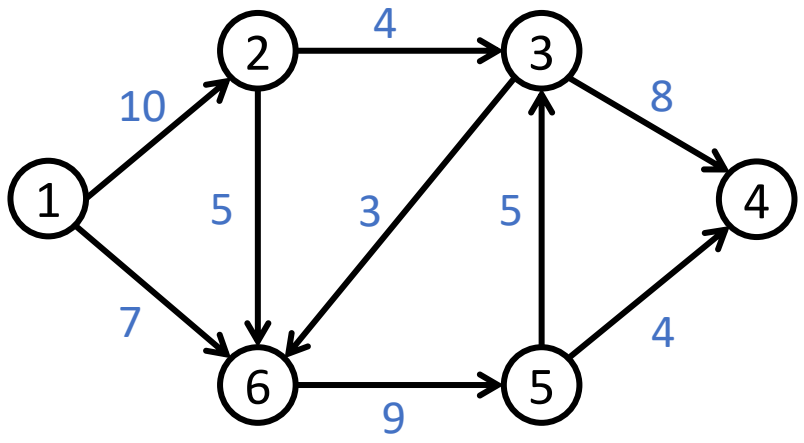
- Un **cammino aumentante** su G con flusso f è un cammino $p: s \rightsquigarrow t$ su G_f
- Ogni spigolo di p può accettare un flusso positivo senza superare il limite di capacità dello spigolo stesso
- La quantità massima di flusso di cui può aumentare ogni spigolo di p è la capacità residua di p:

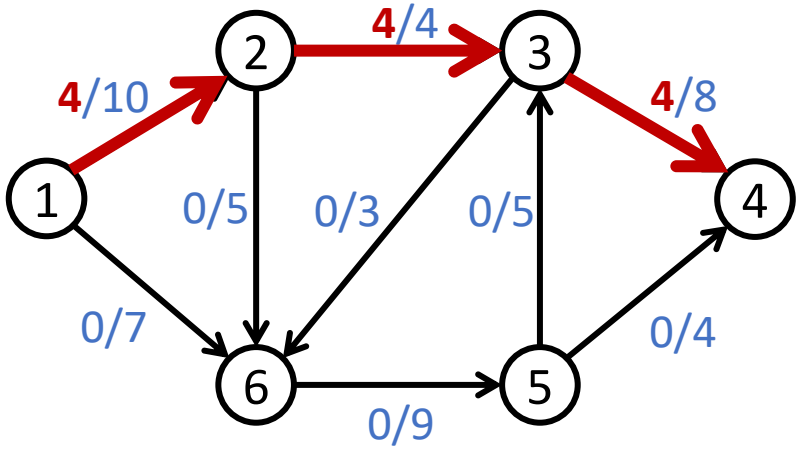
$$c_f(p) = \min\{c_f(u, v): (u, v) \in p\}$$

Capacità residua, rete residua, cammino aumentante

rete di flusso (G, s, t, c)

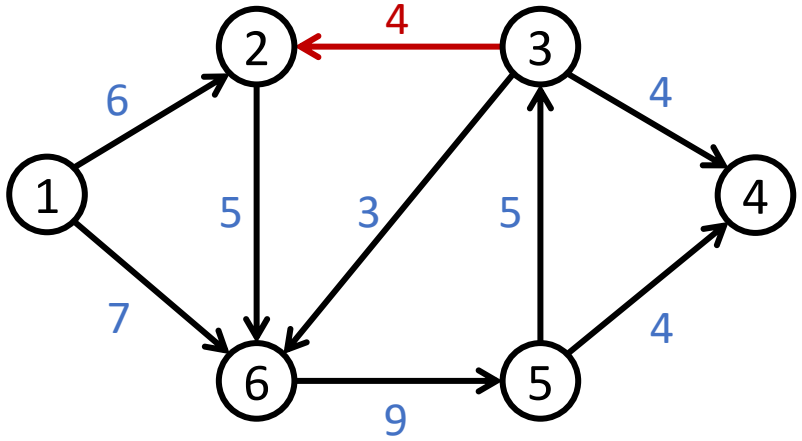
flusso iniziale $|f| = 0$





cammino aumentante p: $s \rightsquigarrow t$

trasporta un flusso $|f| = 4$



rete di flusso residua (G_f, s, t, c_f)

lo spigolo $(2,3)$ si è saturato e al suo posto è stato creato uno spigolo $(3,2)$ di capacità 4

Algoritmo di Ford e Fulkerson

- Data la rete di flusso (G, s, t, c) si vuole calcolare il massimo flusso $|f| = \sum_{v \in V(G)} f(s, v)$

Lester Ford jr.
(1927 – 2017)

Algoritmo 28 FORD-FULKERSONGENERICO(G, s, t, c)

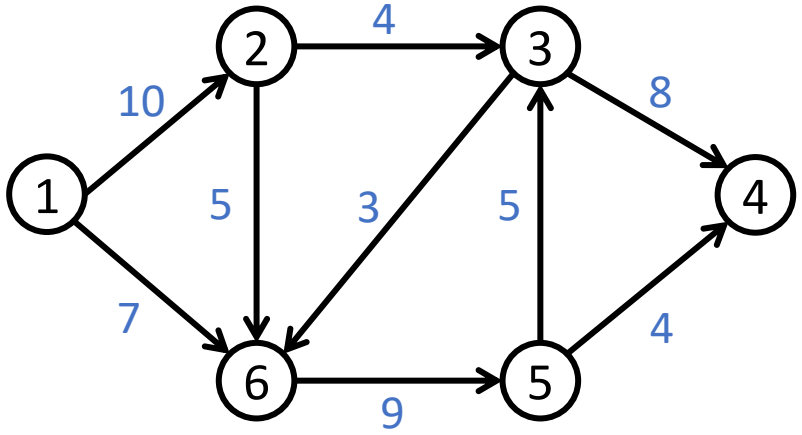
Input: Una rete di flusso (G, s, t, c)
Output: Il valore del massimo flusso $|f|$ per G

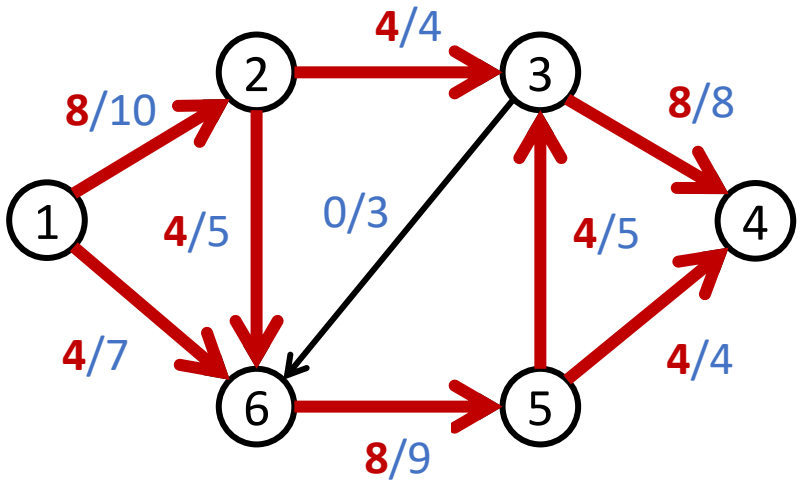
1: inizializza il flusso $f = 0$
2: **fintanto che** esiste un cammino aumentante p **ripeti**
3: aumenta il flusso f lungo p
4: **fine-ciclo**

Delbert R. Fulkerson
(1924 – 1976)

- Il cammino aumentante al passo 2 viene individuato con una visita in profondità del grafo G_f a partire dal vertice s
- *Complessità:* $O(m\,|f|)$

Algoritmo di Ford e Fulkerson





$|f| = 12$ ⇒ **Flusso massimo**: non esistono più cammini aumentanti da s a t

Algoritmo 29 FORD-FULKERSON(G, s, t, c)

Input: Una rete di flusso (G, s, t, c)

Output: Il valore del massimo flusso $|f|$ per G

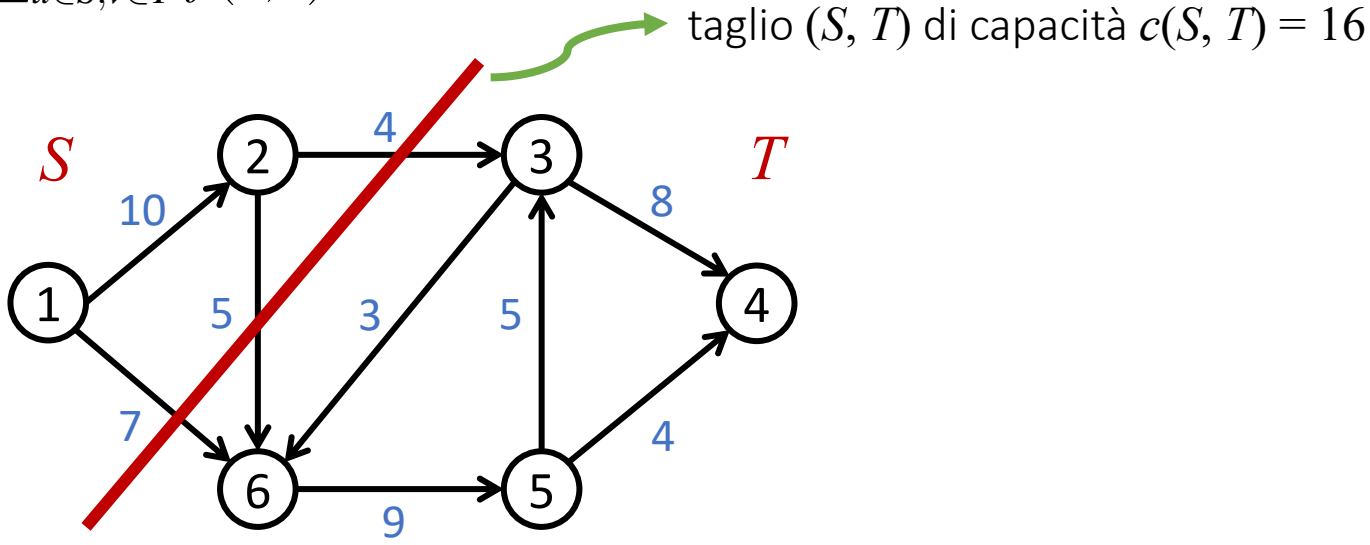
1: **per ogni** $(u, v) \in E(G)$ **ripeti**
2: $f(u, v) := 0,\ f(v, u) := 0$
3: **fine-ciclo**
4: **fintanto che** $\exists$ un cammino aumentante $p : s \rightsquigarrow t$ in G_f **ripeti**
5: $c_f(p) := \min\{c_f(u, v)$ tale che $(u, v) \in p\}$ dove:

$$c_f(u, v) = \begin{cases} c(u, v) - f(u, v) \text{ se } (u, v) \in E(G) \\ f(v, u) \text{ altrimenti} \end{cases}$$

6: **per ogni** spigolo $(u, v) \in p$ **ripeti**
7: **se** $(u, v) \in E(G)$ **allora**
8: $f(u, v) := f(u, v) + c_f(p)$
9: **altrimenti**
10: $f(v, u) := f(v, u) - c_f(p)$
11: **fine-condizione**
12: **fine-ciclo**
13: **fine-ciclo**
14: $|f^*| := 0$
15: **per ogni** $v \in V(G)$ **ripeti**
16: $|f^*| := |f^*| + f(s, v)$
17: **fine-ciclo**

Flusso massimo e taglio minimo

- Un **taglio** di una rete di flusso (G, s, t, c) è una coppia (S, T) tali che $S, T \subset V(G)$, $s \in S$, $t \in T$, $T = V \setminus S$
- La **capacità del taglio** (S, T) è $c(S, T) = \sum c(u, v)$ per $(u, v) \in E(G)$ tali che $u \in S$ e $v \in T$
- Un **taglio minimo** è un taglio di capacità minima per la rete di flusso
- Flusso netto del taglio: $f(S,T) = \sum_{u \in S, v \in T} f(u,v)$



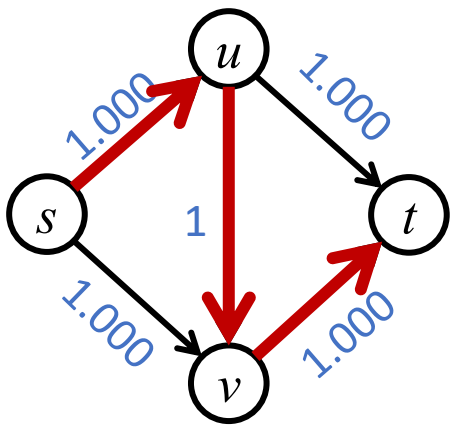
- **Osservazione**: *il valore di un flusso è limitato superiormente dalla capacità di un taglio qualsiasi della rete; quindi il flusso massimo sulla rete è limitato superiormente dalla capacità del taglio minimo per la stessa rete*

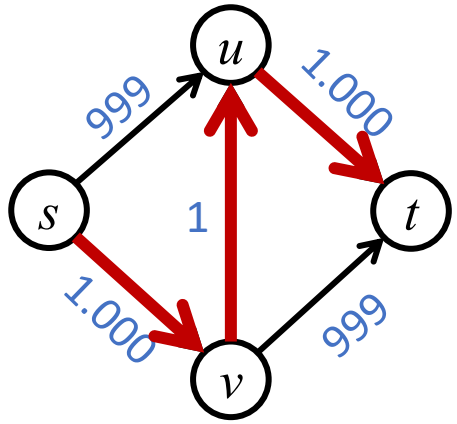
Flusso massimo e taglio minimo

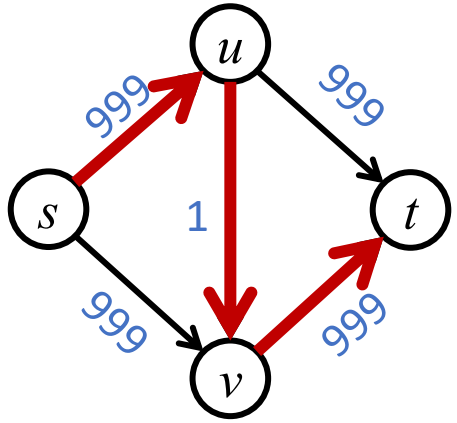
- **Teorema del flusso massimo e del taglio minimo** (Ford e Fulkerson). Se f è un flusso per la rete (G, s, t, c) allora le seguenti affermazioni sono equivalenti:
  1. $|f|$ è un flusso massimo
  2. La rete residua G_f non contiene cammini aumentanti
  3. $|f| = c(S, T)$ per qualche taglio (S, T) di G
- *Dimostrazione*:
  - 1 ⇒ 2: Se per assurdo esiste un cammino aumentante in G_f (un cammino p da s a t in G_f) allora il flusso su quel cammino è f_p e risulta $f + f_p > f$ e dunque f non è massimo
  - 2 ⇒ 3: Se G_f non ha cammini aumentanti, quindi non esiste un cammino da s a t in G_f, sia $S = \{u \in V(G)$: in G_f esiste un cammino p: $s \rightsquigarrow u\}$ e $T = V \setminus S$; (S, T) è un taglio e risulta $f(u, v) = c(u, v)$ per ogni $(u,v) \in E(G)$ tale che $u \in S$ e $v \in T$; dunque $|f| = f(S, T) = c(S, T)$
  - 3 ⇒ 1: $|f| \leq c(S, T)$ per ogni taglio (S, T); dunque se $|f| = c(S, T)$ allora f è un flusso massimo ■

Algoritmo di Edmonds-Karp

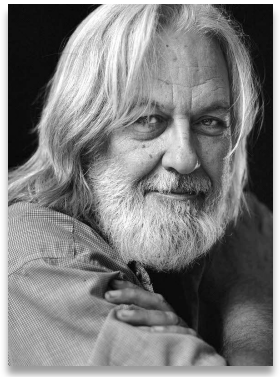
- L'algoritmo di Ford e Fulkerson può rivelarsi assai inefficiente in alcuni casi







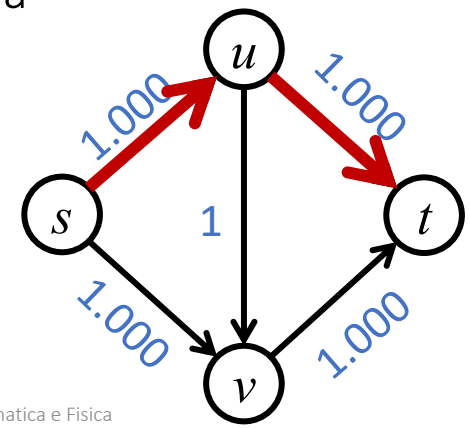
. . .

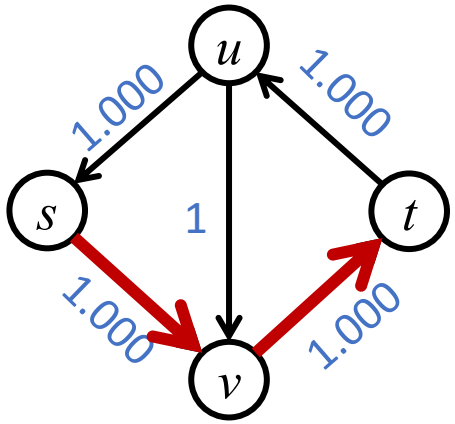


Jack Edmonds
(1934)

Richard Karp
(1935)

- Il problema è nella selezione del cammino aumentante: Edmonds e Karp proposero un algoritmo simile, ma più efficiente, che seleziona i cammini aumentanti eseguendo la **visita in ampiezza** della rete residua



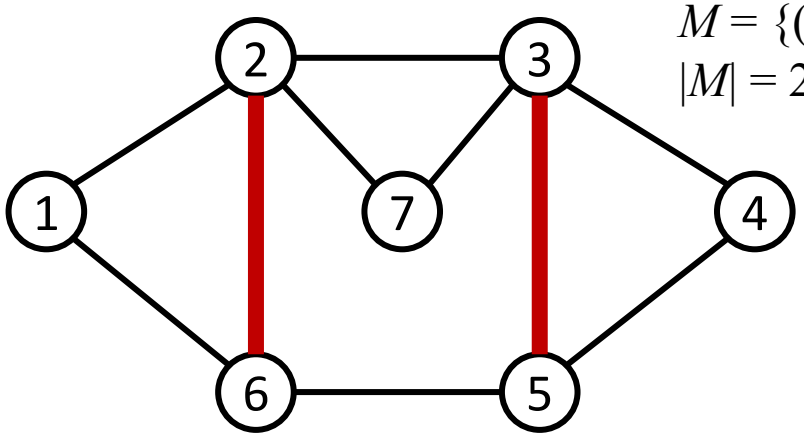


$|f| = 2.000$

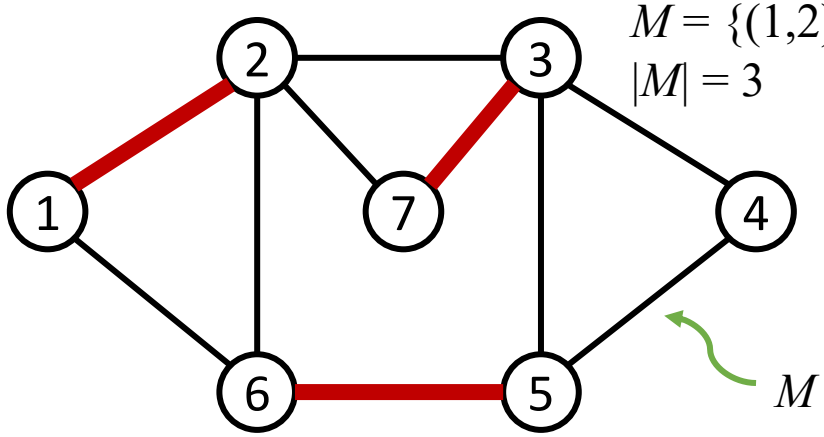
Complessità: $O(n\, m^2)$

Matching massimale

- Dato un grafo $G = (V, E)$, un ***matching*** (accoppiamento) è un sottoinsieme $M \subseteq E$ tale che per ogni $v \in V(G)$ esiste al massimo uno spigolo $(u, v) \in M$
- Un matching è ***massimale*** se $|M|$ è massima



$M = \{(2,6), (3,5)\}$
$|M| = 2$

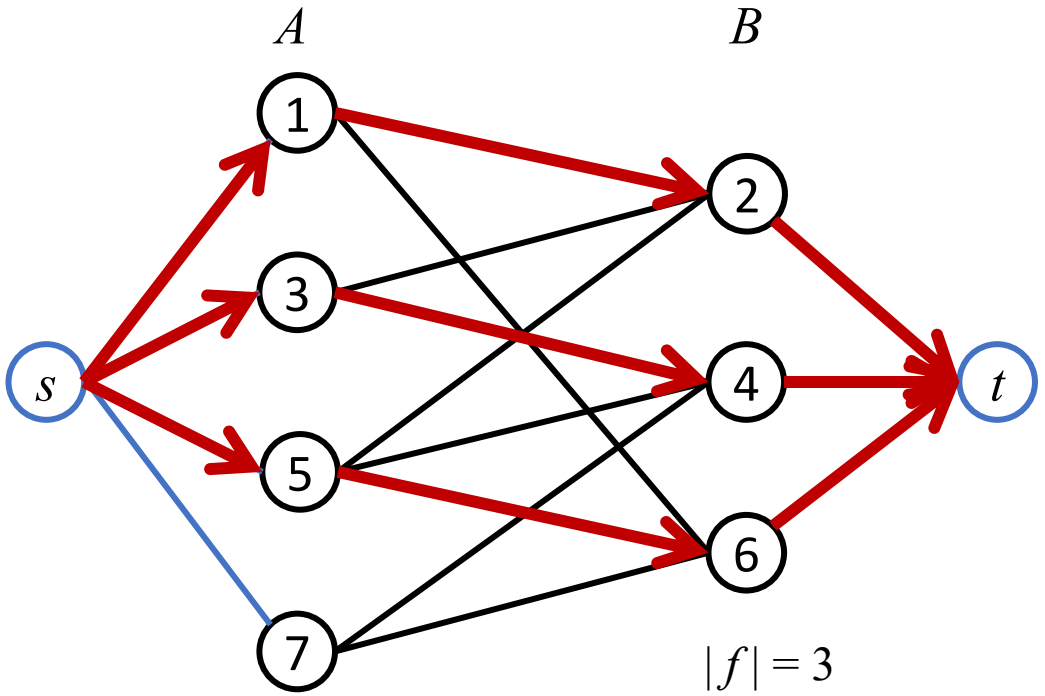


$M = \{(1,2), (3,7), (5,6)\}$
$|M| = 3$

M è un matching massimale

Matching massimale su un grafo bipartito

- Sia $G = (V, E)$ un **grafo bipartito** con $A, B \subset V$ tali che $A \cup B = V$, $A \cap B = \emptyset$ e per ogni $(u, v) \in E(G)$ risulta $u \in A$ e $v \in B$



- Possiamo costruire una **rete di flusso** (G', s, t, c) associata a G aggiungendo una sorgente s e un pozzo t e collegando ad s tutti i vertici $u \in A$ e al pozzo t tutti i vertici $v \in B$ e definendo la capacità $c(u, v) = 1$ per ogni $(u, v) \in E'(G')$
- Un matching M su G identifica un flusso su G' con valore $|f| = \sum_{(u,v) \in M} c(u, v)$
- D'altra parte, ogni flusso da s a t su G' identifica un matching su G
- Dunque un flusso massimo su G' è anche un matching massimale su G e viceversa: pertanto un matching massimale su un grafo bipartito può essere calcolato applicando un algoritmo di flusso massimo alla rete di flusso (G', s, t, c)

Algoritmo Push-Relabel

Robert Endre Tarjan
(1948)

- Con l'obiettivo di costruire una nuova strategia risolutiva per il problema del massimo flusso su una rete, definiamo un concetto un po' diverso da quello del flusso, rilassando alcuni vincoli e definendo il **preflusso** come una funzione $f : V \times V \longrightarrow \mathbb{R}$ tale che valgano le seguenti proprietà:
  - antisimmetria: $f(u, v) = -f(v, u)$
  - vincolo di capacità: $f(u, v) \leq c(u, v)$
  - proprietà debole di conservazione del flusso: $f(V, u) \geq 0$ per ogni $u \in V \setminus \{s\}$

Andrew Vladislav Goldberg
(1960)

- Definiamo il **flusso in eccesso** come $e(u) = f(V, u)$; diciamo inoltre che se $e(u) > 0$ allora $u \in V \setminus \{s, t\}$ è **traboccante**

- Sia (G, s, t, c) una rete di flusso con $G = (V, E)$ una rete di flusso con sorgente s, pozzo t, preflusso f e capacità c; una funzione **altezza** h: $V \longrightarrow \mathbb{N}$ è definita ponendo:
  - $h(s) = |V|$
  - $h(t) = 0$
  - $h(u) \leq h(v) + 1$ per ogni $(u, v) \in E_f$

Algoritmo Push-Relabel

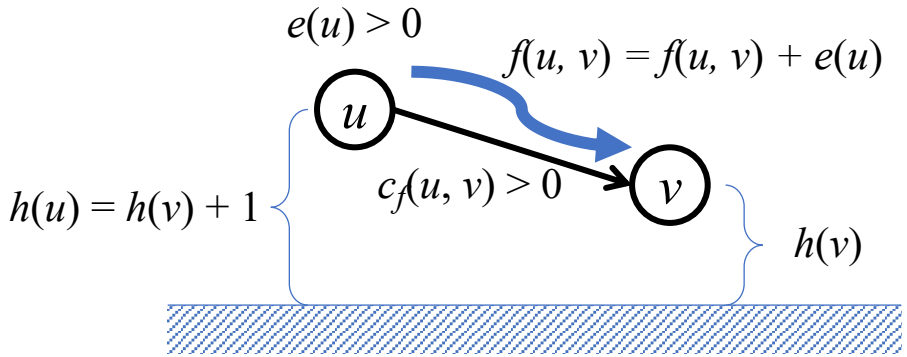
Definiamo le seguenti operazioni elementari:

- PUSH(u, v)

  si applica se u è traboccante e $h(u) = h(v) + 1$ per ridurre il flusso in eccesso in u scaricandolo verso v

Algoritmo 30 PUSH(u, v)

1: $x := \min\{e(u), c_f(u, v)\}$
2: $f(u, v) = f(u, v) + x$, $f(v, u) = -f(u, v)$
3: $e(u) = e(u) - x$, $e(v) = e(v) + x$

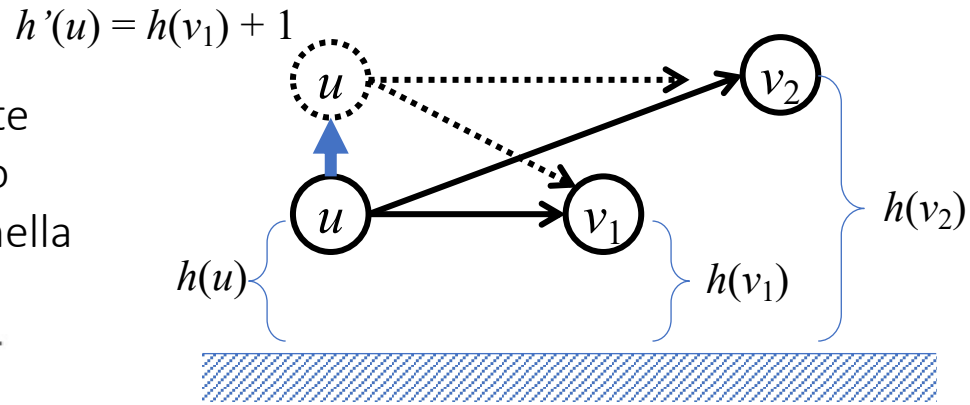


- RELABEL(u)

  si applica quando u è traboccante e tutti i vertici v adiacenti a u nella rete residua sono tali che $h(u) \leq h(v)$; allora si innalza u al minimo necessario per poter scaricare il flusso in eccesso verso un vertice v adiacente a u nella rete residua G_f eseguendo l'operazione PUSH(u, v)

Algoritmo 31 RELABEL(u)

1: $h(u) = 1 + \min\{h(v) : (u, v) \in E_f\}$



Algoritmo Push-Relabel

- L'algoritmo per il calcolo del flusso massimo richiede di **inizializzare il preflusso** sulla rete (G, s, t, c) inviando da s verso i vertici adiacenti $v \in N(s)$ tutto il flusso consentito dalla capacità degli spigoli $(s, v) \in E$

Algoritmo 32 INIZIALIZZAPREFLUSSO(G, s, t, c)

Input: Una rete di flusso (G, s, t, c)
Output: Un preflusso f su G

1: **per ogni** vertice $u \in V(G)$ **ripeti**
2: $h(u) = 0, e(u) = 0$
3: **fine-ciclo**
4: **per ogni** spigolo $(u, v) \in E(G)$ **ripeti**
5: $f(u, v) = f(v, u) = 0$
6: **fine-ciclo**
7: $h(s) = |V|$
8: **per ogni** $v \in N(s)$ **ripeti**
9: $f(s, v) = c(s, v), f(v, s) = -f(s, v)$
10: $e(v) = c(s, v), e(s) = e(s) - c(s, v)$
11: **fine-ciclo**

Algoritmo Push-Relabel

- L'algoritmo, dopo aver inizializzato il preflusso, ripete più volte le operazioni PUSH e RELABEL, finché la configurazione della rete non rende impossibile applicarle ulteriormente

Algoritmo 33 PUSHRELABEL(G, s, t, c)

1: INIZIALIZZAPREFLUSSO(G, s, t, c)
2: **fintanto che** sussistono le condizioni per applicare PUSH o RELABEL **ripeti**
3: esegue PUSH(u, v) se è applicabile, altrimenti esegue RELABEL(u)
4: **fine-ciclo**

- Al termine dell'algoritmo avremo finito per inviare verso il pozzo t la massima quantità di flusso possibile, calcolando così il flusso massimo $|f|$ per la rete (G, s, t, c)

Algoritmo Push-Relabel

- Possiamo scrivere in modo più dettagliato l'algoritmo come segue:
- La complessità computazionale dell'algoritmo è $O(n^2m)$

Algoritmo 34 PUSHRELABEL(G,s,t,c)

Input: Una rete di flusso (G,s,t,c)
Output: Un flusso massimo f su G

```
1: INIZIALIZZAPREFLUSSO(G,s,t,c)
2: ripeti
3:     flag = 0
4:     per ogni u ∈ V(G) ripeti
5:         se e(u) > 0 allora
6:             se esiste v ∈ N(u) in G_f tale che h(u) = h(v) + 1 allora
7:                 PUSH(u,v), flag = 1
8:             altrimenti se h(u) ≤ h(v) per ogni v ∈ N(u) in G_f allora
9:                 RELABEL(u), flag = 1
10:            fine-condizione
11:        fine-condizione
12:    fine-ciclo
13: fintanto che flag = 1
14: il flusso massimo è |f| = e(t)
```

Riferimenti bibliografici

- Cormen, Leiserson, Rivest, Stein, «*Introduzione agli algoritmi e strutture dati*», terza edizione, McGraw-Hill (Cap. 26)
- Robert E. Tarjan, «Data Structures and Network Algorithms», SIAM, 1983