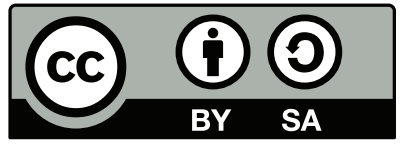
Algoritmi e Strutture Dati

Ricerca locale

Alberto Montresor

Università di Trento

2018/11/07



Sommario

Ricerca locale

Ricerca locale

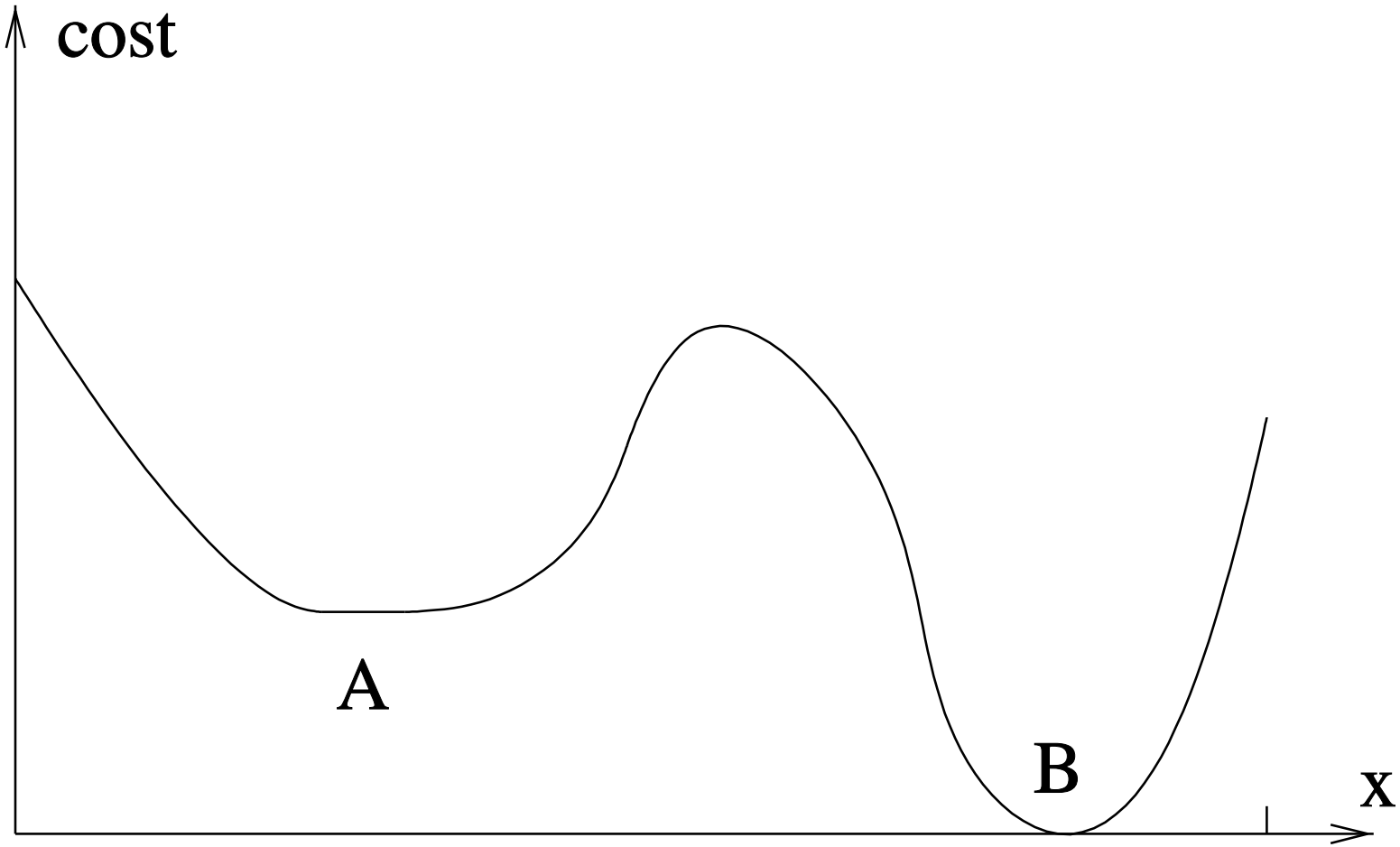
Se si conosce una soluzione ammissibile (non necessariamente ottima) ad un problema di ottimizzazione, si può cercare di trovare una soluzione migliore nelle "vicinanze" di quella precedente. Si continua in questo modo fino a quando non si è più in grado di migliorare

---

ricercaLocale()

$Sol =$ una soluzione ammissibile del problema
while $\exists S \in I(Sol)$ migliore di Sol **do**
    $Sol = S$
return Sol

Ricerca locale



Rete di flusso

Definizione

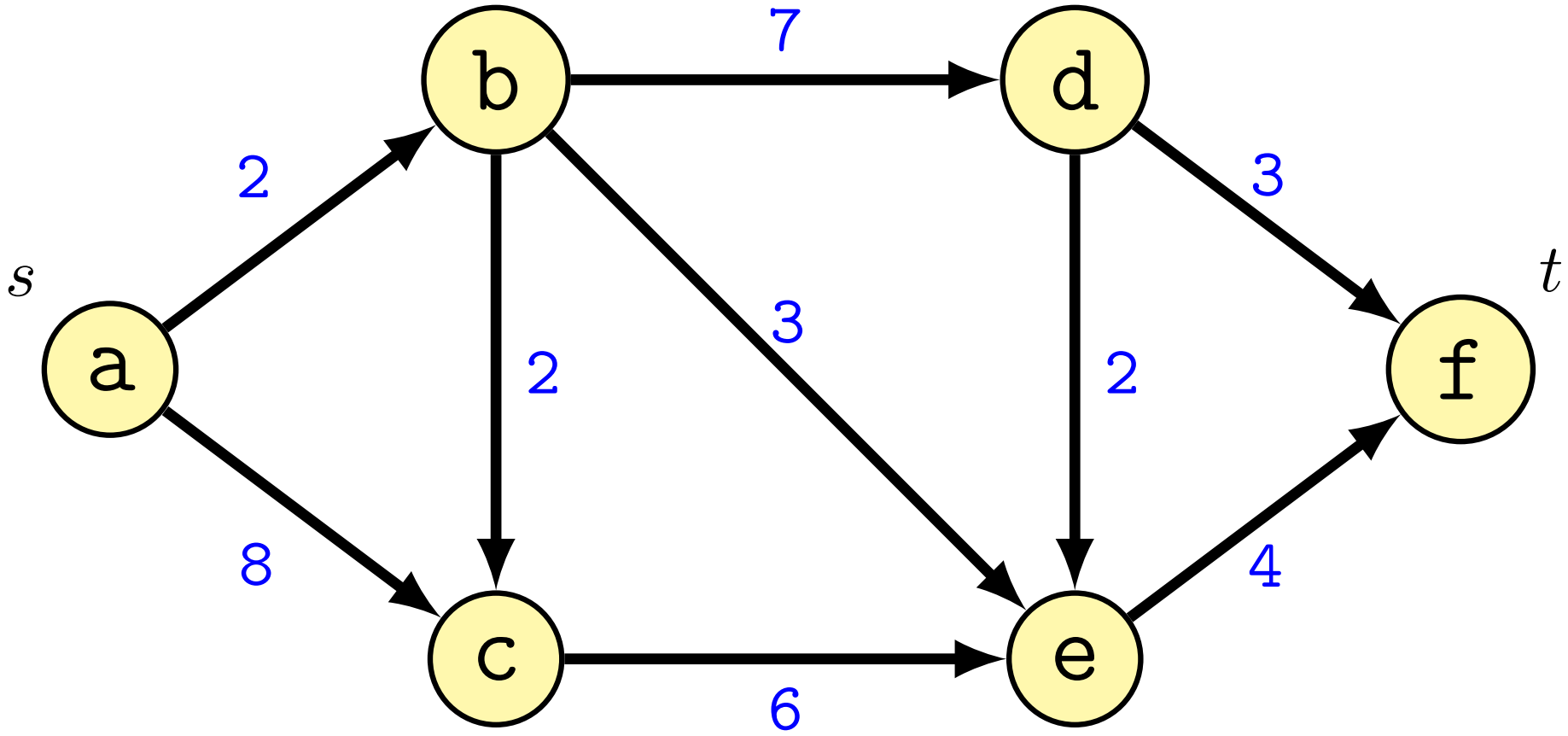
Una rete di flusso $G = (V, E, s, t, c)$ è data da:

- un grafo orientato $G = (V, E)$
- un nodo $s \in V$ detto sorgente
- un nodo $t \in V$ detto pozzo
- una funzione di capacità $c : V \times V \to \mathbb{R}^{\geq 0}$, tale per cui $(u, v) \notin E \Rightarrow c(u, v) = 0$.

Assunzioni

- Per ogni nodo $v \in V$, esiste un cammino $s \rightsquigarrow v \rightsquigarrow t$ da s a t che passa per v.
- Possiamo ignorare i nodi che non godono di questa proprietà

Rete di flusso



Flusso

Flusso

Un flusso in G è una funzione $f : V \times V \to \mathbb{R}$ che soddisfa le seguenti proprietà:

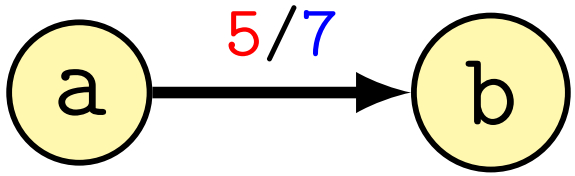
- Vincolo sulla capacità: $\forall u, v \in V,\ f(u, v) \leq c(u, v)$
- Antisimmetria: $\forall u, v \in V,\ f(u, v) = -f(v, u)$
- Conservazione del flusso: $\forall u \in V - \{s, t\}, \sum_{v \in V} f(u, v) = 0$

Flusso

Vincolo sulla capacità

Il flusso non deve eccedere la capacità sull'arco

$$\forall u, v \in V : f(u, v) \leq c(u, v)$$

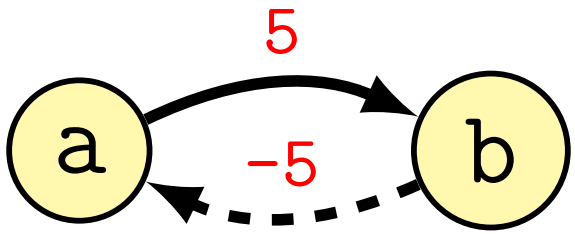


Flusso

Antisimmetria

$$\forall u, v \in V : f(u, v) = -f(v, u)$$

Il flusso viene definito in questo modo per semplificare la proprietà successiva e altre regole.

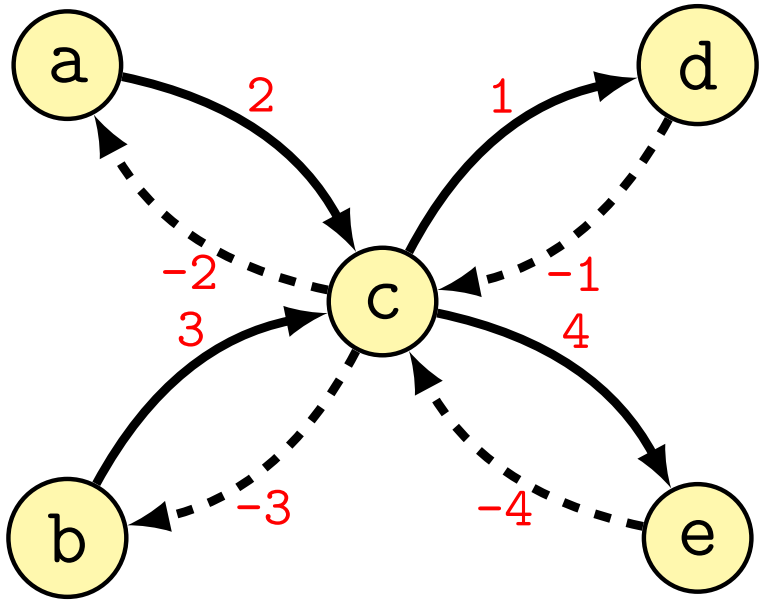


Flusso

Conservazione del flusso

Per ogni nodo, la somma dei flussi entranti deve essere uguale alla somma dei flussi uscenti.

$$\forall u \in V - \{s, t\} : \sum_{v \in V} f(u, v) = 0$$



Definizioni

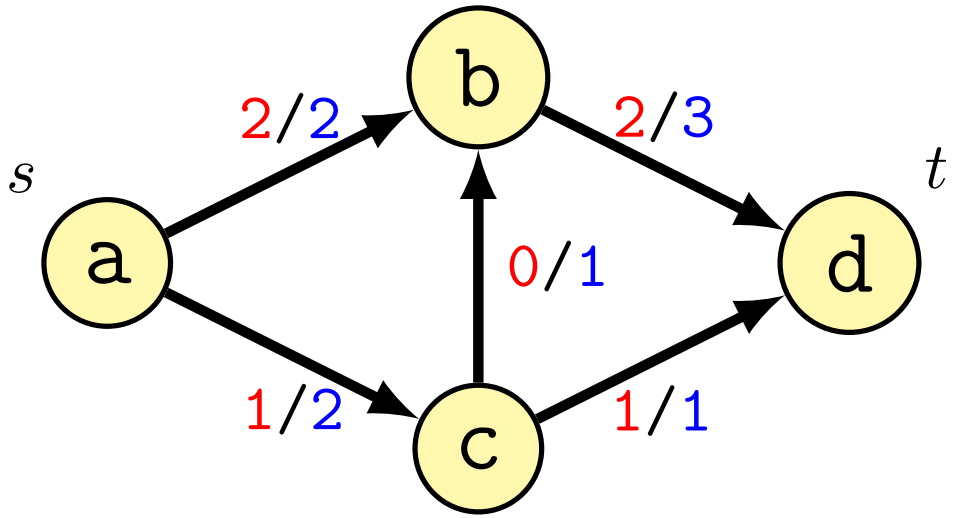
Valore del flusso

Il valore di un flusso f è definito come:

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

ovvero come la quantità di flusso uscente da s.

$|f| = 3$



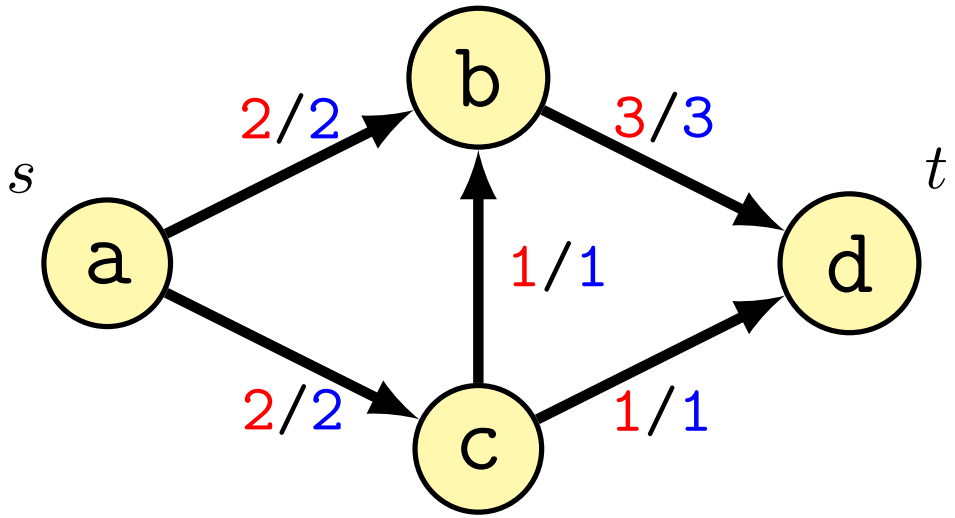
Problema

Flusso massimo

Data una rete $G = (V, E, s, t, c)$, trovare un flusso che abbia valore massimo fra tutti i flussi associabili alla rete.

$$|f^*| = \max\{|f|\}$$

$|f^*| = 4$



Metodo delle reti residue

Algoritmo, informale

- Lavoriamo partendo da un flusso "corrente" f, inizialmente nullo
- Si ripetono le operazioni seguenti:
  - Si "sottrae" il flusso attuale dalla rete iniziale, ottenendo una rete residua
  - Si cerca un flusso g all'interno della rete residua
  - Si somma g ad f

  fino a quando non è più possibile trovare un flusso positivo g

Output

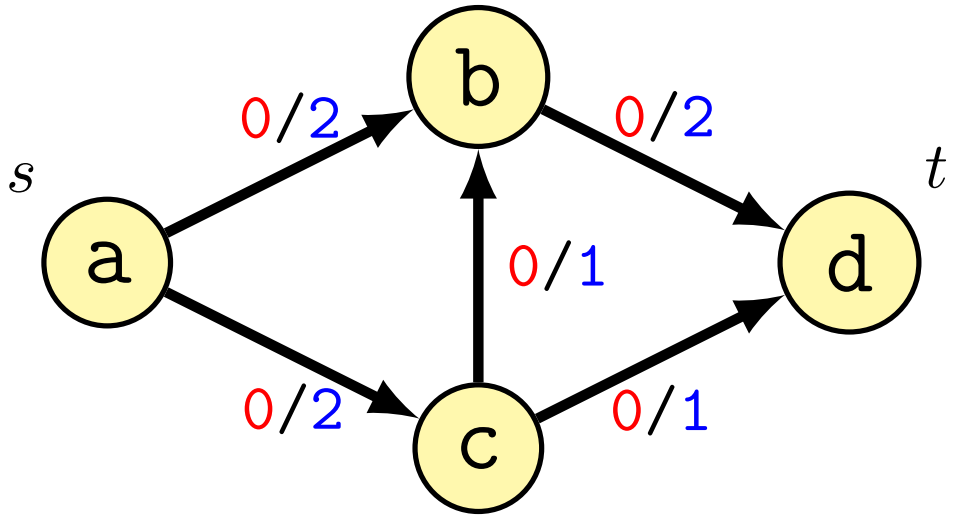
Se svolta correttamente, è possibile dimostrare che questo approccio restituisce un flusso massimo

Definizioni

Flusso nullo

Definiamo flusso nullo la funzione $f_0 : V \times V \to \mathbf{R}^{\geq 0}$ tale che $f(u, v) = 0$ per ogni $u, v \in V$.

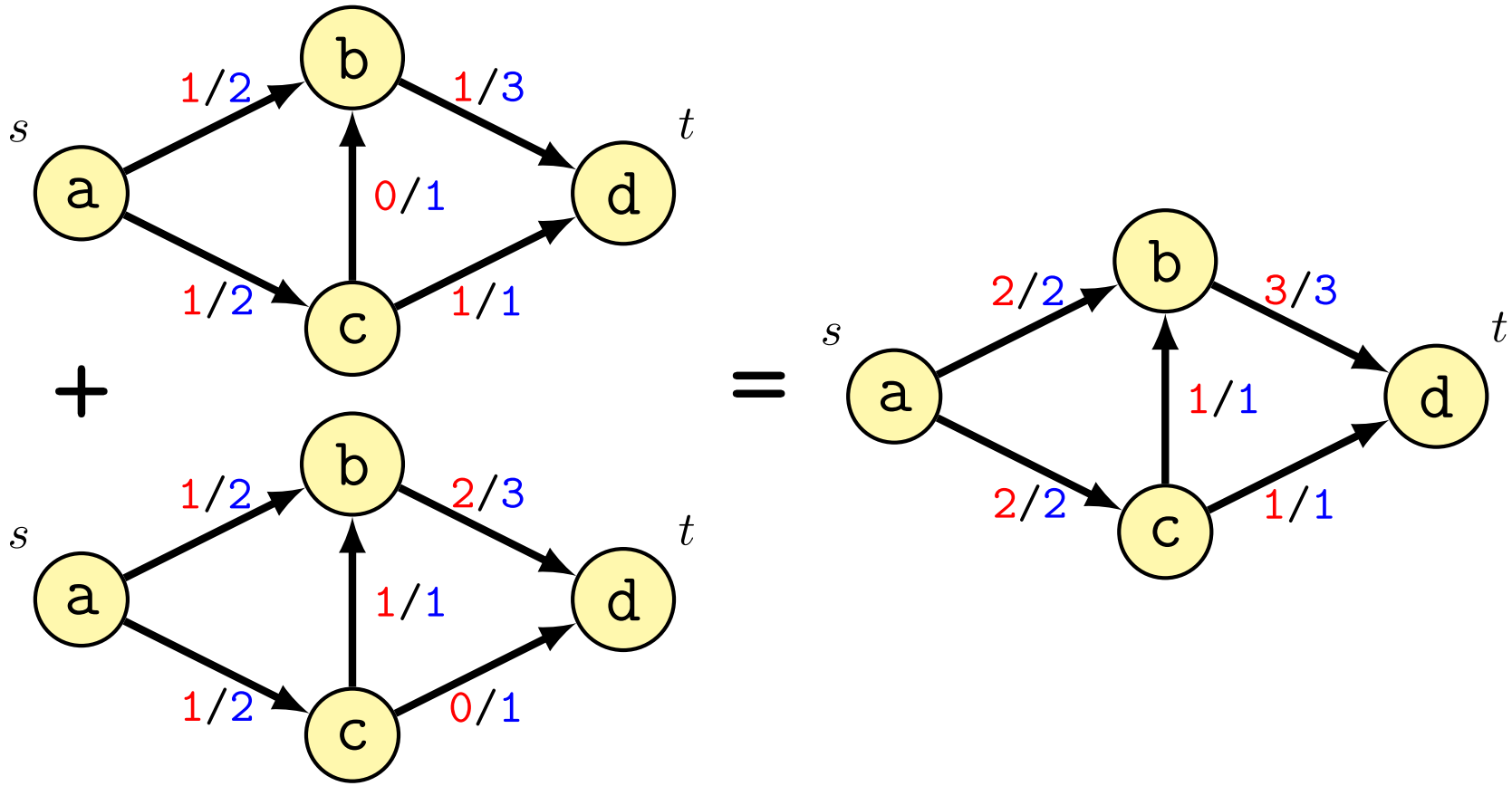
$|f| = 0$



Definizioni

Somma di flussi

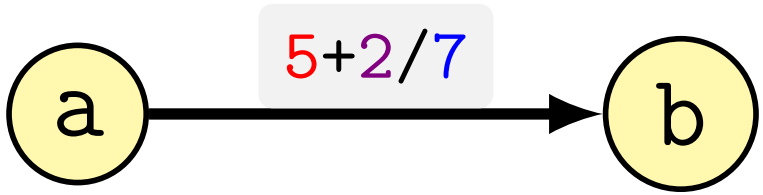
Per ogni coppia di flussi f_1 e f_2 in G, definiamo il flusso somma $g = f_1 + f_2$ come un flusso tale per cui $g(u, v) = f_1(u, v) + f_2(u, v)$.



Definizioni

Capacità residua

Definiamo capacità residua di un flusso f in una rete $G = (V, E, s, t, c)$ una funzione $c_f : V \times V \to \mathbf{R}^{\geq 0}$ tale che $c_f(u, v) = c(u, v) - f(u, v)$.

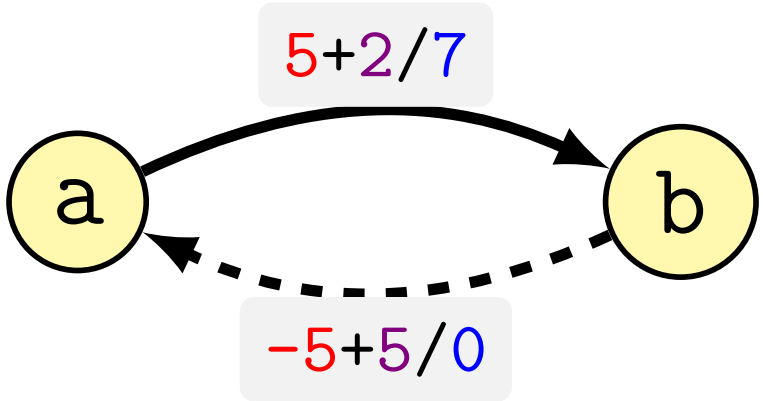


- Flusso in rosso
- Capacità residua in viola
- Capacità iniziale in blu

Definizioni

Capacità residua

Definiamo capacità residua di un flusso f in una rete $G = (V, E, s, t, c)$ una funzione $c_f : V \times V \to \mathbf{R}^{\geq 0}$ tale che $c_f(u, v) = c(u, v) - f(u, v)$.



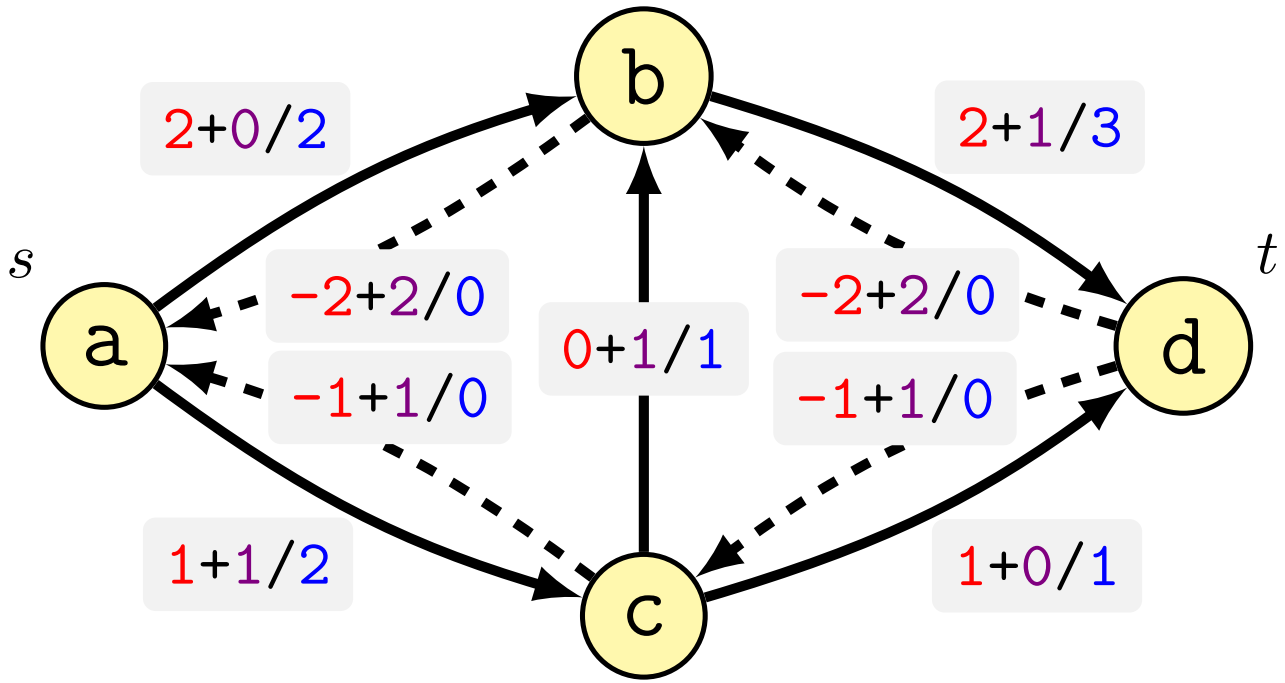
Per la definizione di capacità residua, si creano degli archi all'indietro:

$$\begin{aligned} c_f(b, a) &= c(b, a) - f(b, a) \\ &= 0 - (-5) = 5 \end{aligned}$$

Definizioni

Reti residue

Data una rete di flusso $G = (V, E, s, t, c)$ e un flusso f su G, possiamo costruire una rete residua $G_f = (V, E_f, s, t, c_f)$, tale per cui $(u, v) \in E_f$ se e solo se $c_f(u, v) > 0$.



Algoritmo, schema generale

int[][] maxFlow(Graph G, Node s, Node t, **int**[][] c)

$f = f_0$ % Inizializza un flusso nullo

$r = c$ % Capacità iniziale

repeat

  $g =$ trova un flusso in r tale che $|g| > 0$, altrimenti f_0

  $f = f + g$

  $r =$ Rete di flusso residua del flusso f in G

until $g = f_0$

return f

Dimostrazione correttezza

Lemma

Se f è un flusso in G e g è un flusso in G_f, allora $f + g$ è un flusso in G.

Vincolo sulla capacità

$$g(u,v) \leq c_f(u,v) \qquad g \text{ è un flusso in } G_f$$

$$f(u,v) + g(u,v) \leq c_f(u,v) + f(u,v) \qquad \text{Aggiungo termine uguale}$$

$$(f+g)(u,v) \leq c(u,v) - f(u,v) + f(u,v) \qquad \text{Sostituzione}$$

$$(f+g)(u,v) \leq c(u,v) \qquad \text{Semplificazione}$$

Dimostrazione correttezza

Lemma

Se f è un flusso in G e g è un flusso in G_f, allora $f + g$ è un flusso in G.

Antisimmetria

$$f(u,v) + g(u,v) = -f(v,u) - g(v,u) \qquad \text{Antisimmetria } f,\ g$$
$$f(u,v) + g(u,v) = -(f(v,u) + g(v,u)) \qquad \text{Raccolta segno } -$$
$$(f+g)(u,v) = -(f+g)(v,u) \qquad \text{Sostituzione}$$

Dimostrazione correttezza

Lemma

Se f è un flusso in G e g è un flusso in G_f, allora $f + g$ è un flusso in G.

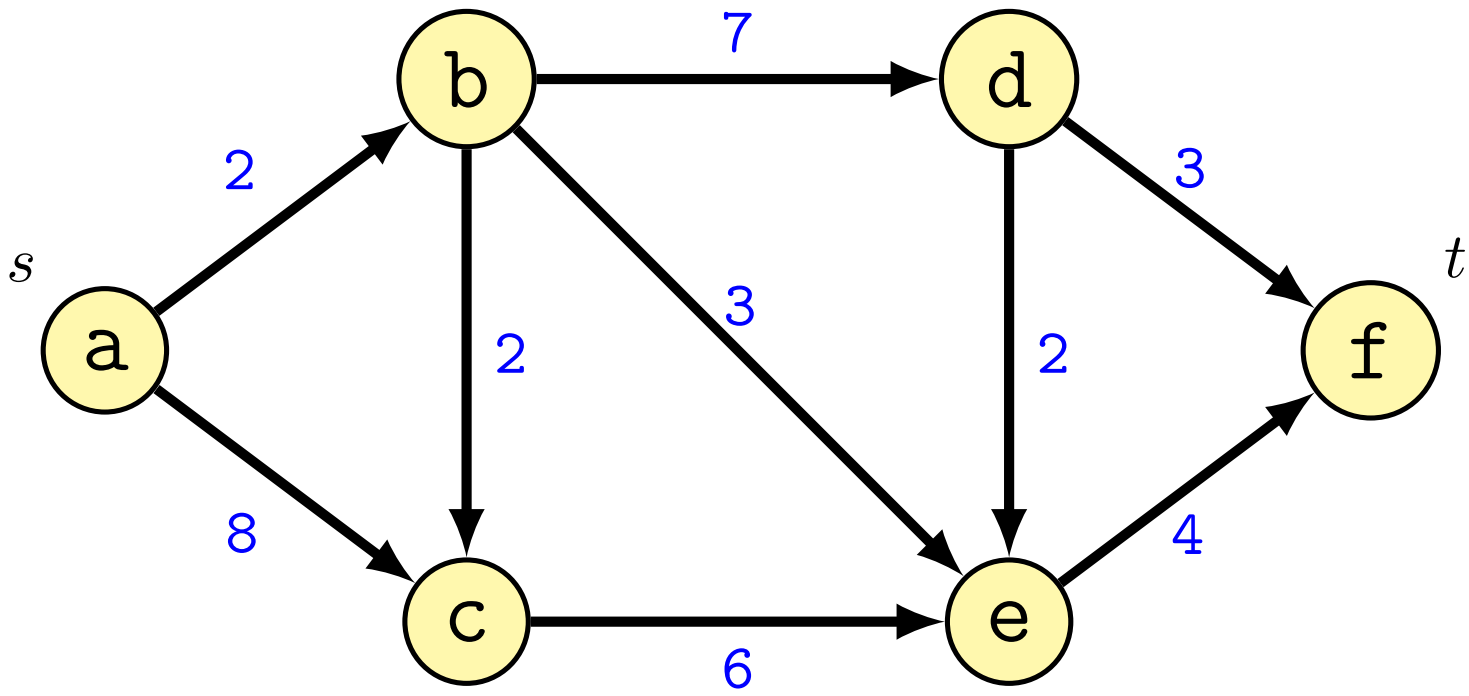
Conservazione

$$\begin{aligned}\sum_{v \in V}(f+g)(u,v) &= \sum_{v \in V}(f(u,v)+g(u,v)) \\ &= \sum_{v \in V} f(u,v) + \sum_{v \in V} g(u,v) \\ &= 0\end{aligned}$$

Metodo dei cammini aumentanti

Domanda

Il punto principale del metodo precedente è il seguente: come trovare un flusso aggiuntivo?



Ford-Fulkerson, 1956

- Si trova un cammino $C = v_0, v_1, \ldots, v_n$, con $s = v_0$ e $t = v_n$ nella rete residua G_f;
- Si identifica la capacità del cammino, corrispondente alla minore capacità degli archi incontrati (collo di bottiglia):

$$c_f(C) = \min_{i=2\ldots n} c_f(v_{i-1}, v_i)$$

- si crea un flusso addizionale g tale che
  - $g(v_{i-1}, v_i) = c_f(C)$;
  - $g(v_i, v_{i-1}) = -c_f(C)$ (per antisimmetria)
  - $g(x, y) = 0$ per tutte le altre coppie (x, y)

Ford-Fulkerson, 1956

int[ ][ ] maxFlow(GRAPH G, NODE s, NODE t, **int**[ ][ ] c)

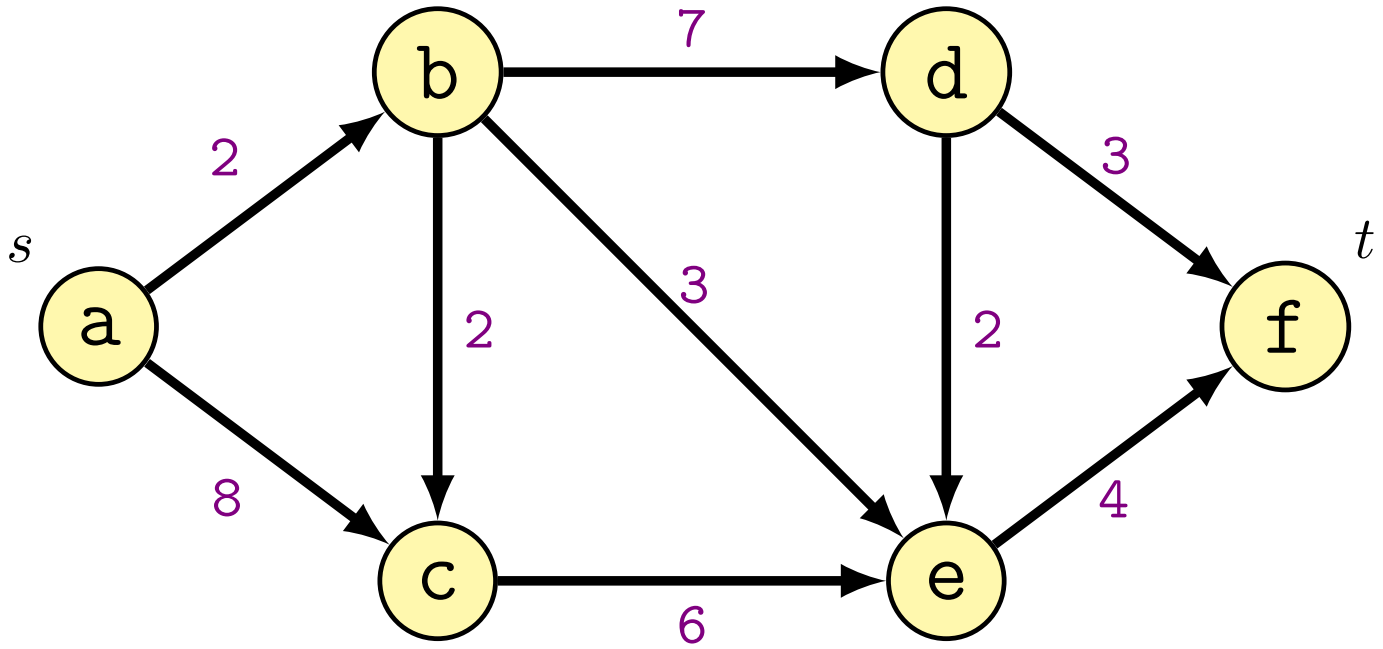
int[ ][ ] f = **new int**[ ][ ] % Flusso parziale
int[ ][ ] g = **new int**[ ][ ] % Flusso da cammino aumentante
int[ ][ ] r = **new int**[ ][ ] % Rete residua

foreach $u, v \in G.\mathsf{V}()$ **do**
    $f[u, v] = 0$ % Inizializza un flusso nullo
    $r[u, v] = c[u, v]$ % Copia c in r

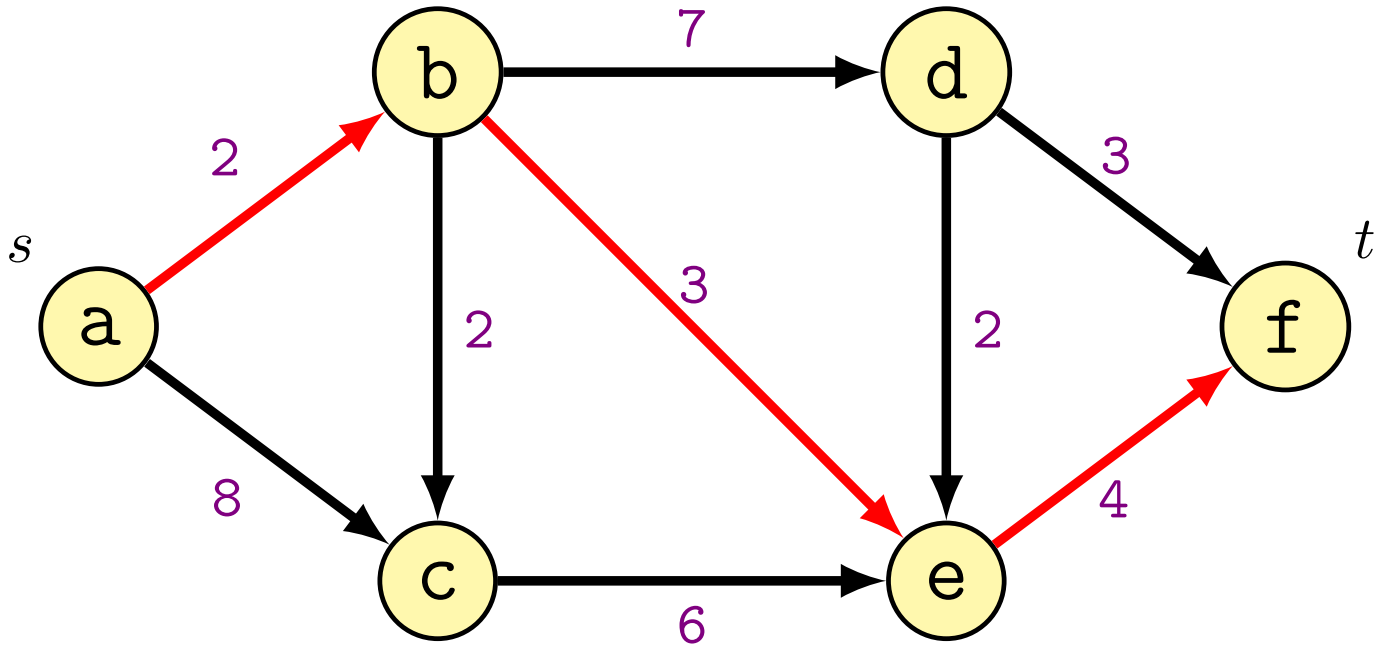
repeat
    $g =$
      flusso associato ad un cammino aumentante in r; altrimenti, f_0
    foreach $u, v \in G.\mathsf{V}()$ **do**
        $f[u, v] = f[u, v] + g[u, v]$ % $f = f + g$
        $r[u, v] = c[u, v] - f[u, v]$ % Calcola c_f

until $g = f_0$
return f

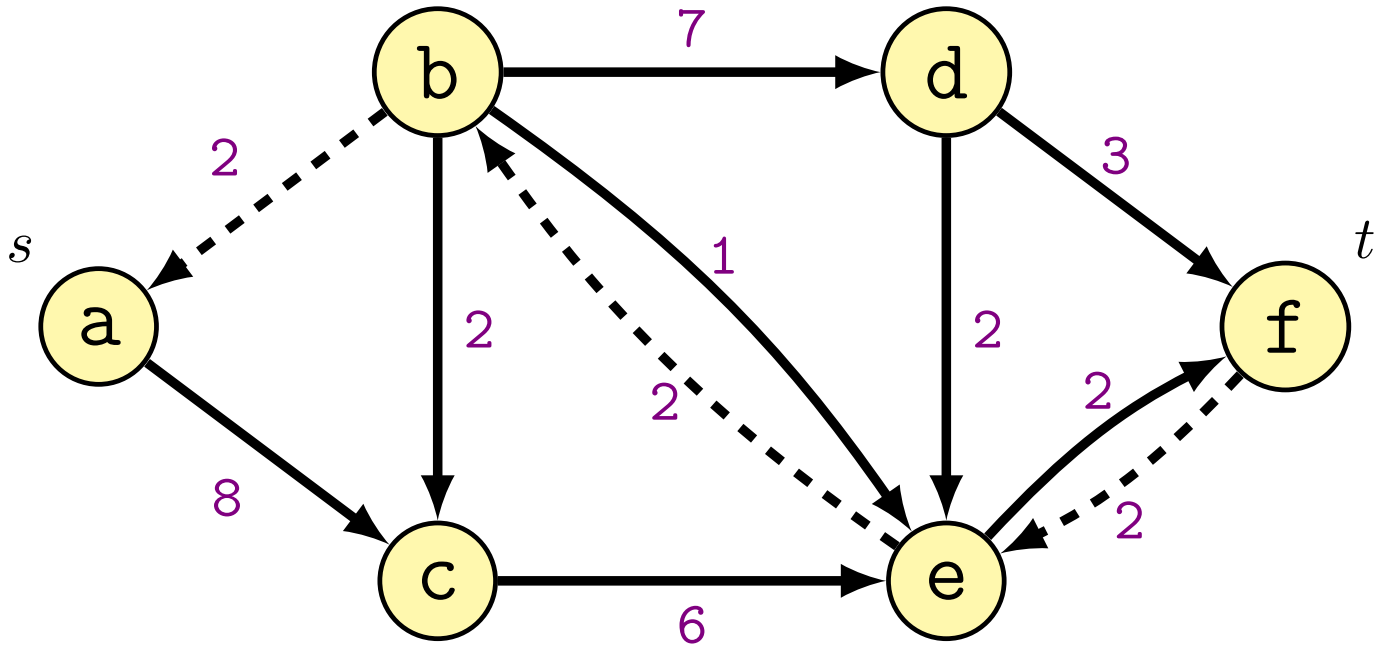
Esecuzione



Esecuzione

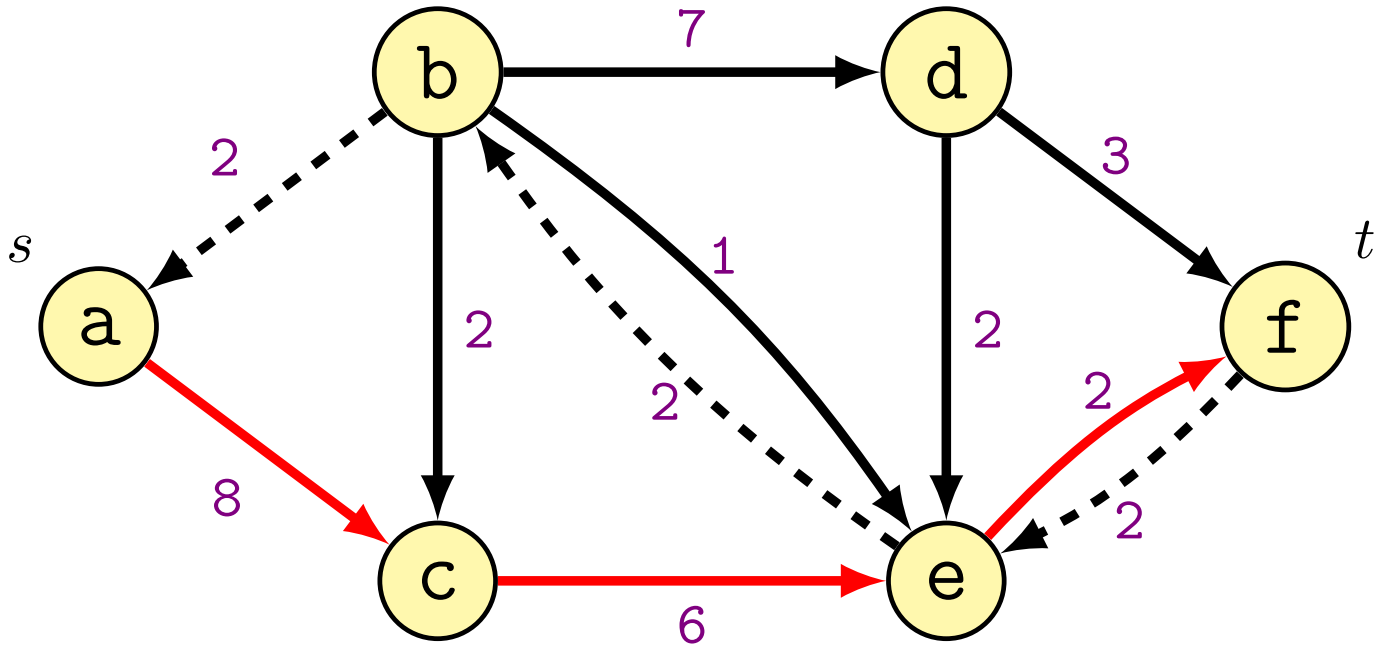


Esecuzione



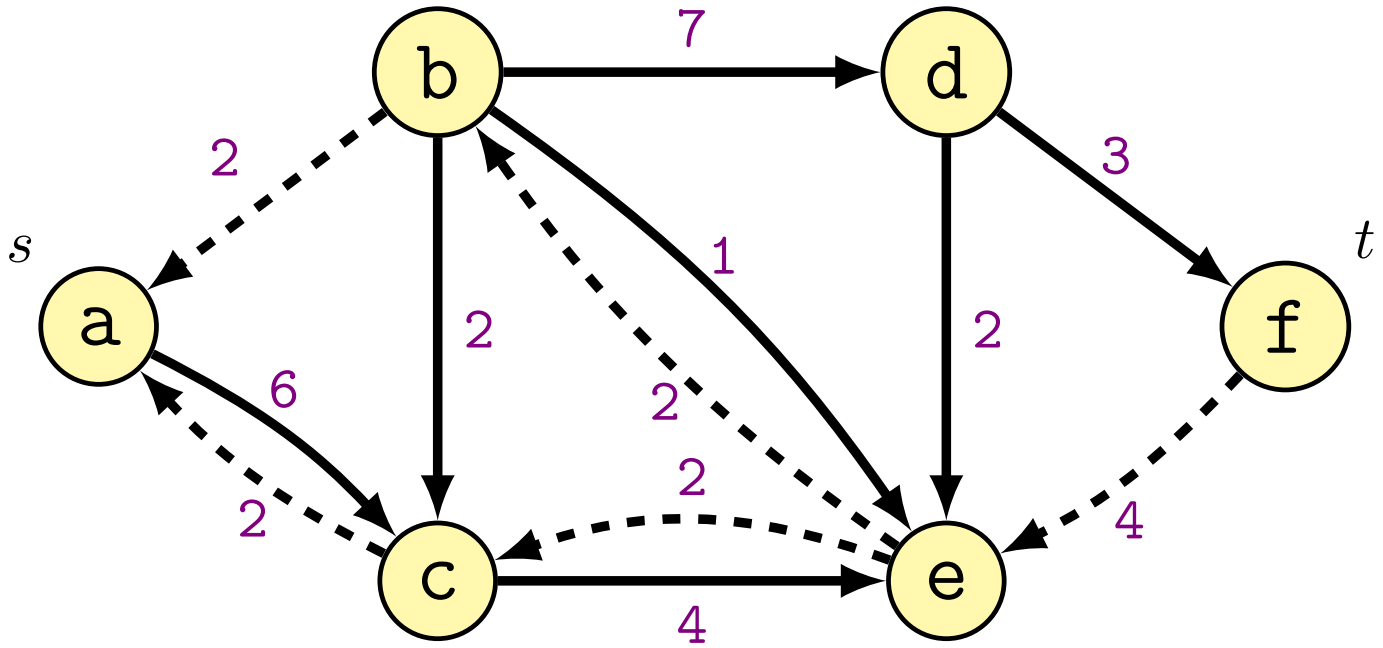
$f(\mathtt{a},\mathtt{b}) = 2$
$f(\mathtt{b},\mathtt{e}) = 2$
$f(\mathtt{e},\mathtt{f}) = 2$

Esecuzione



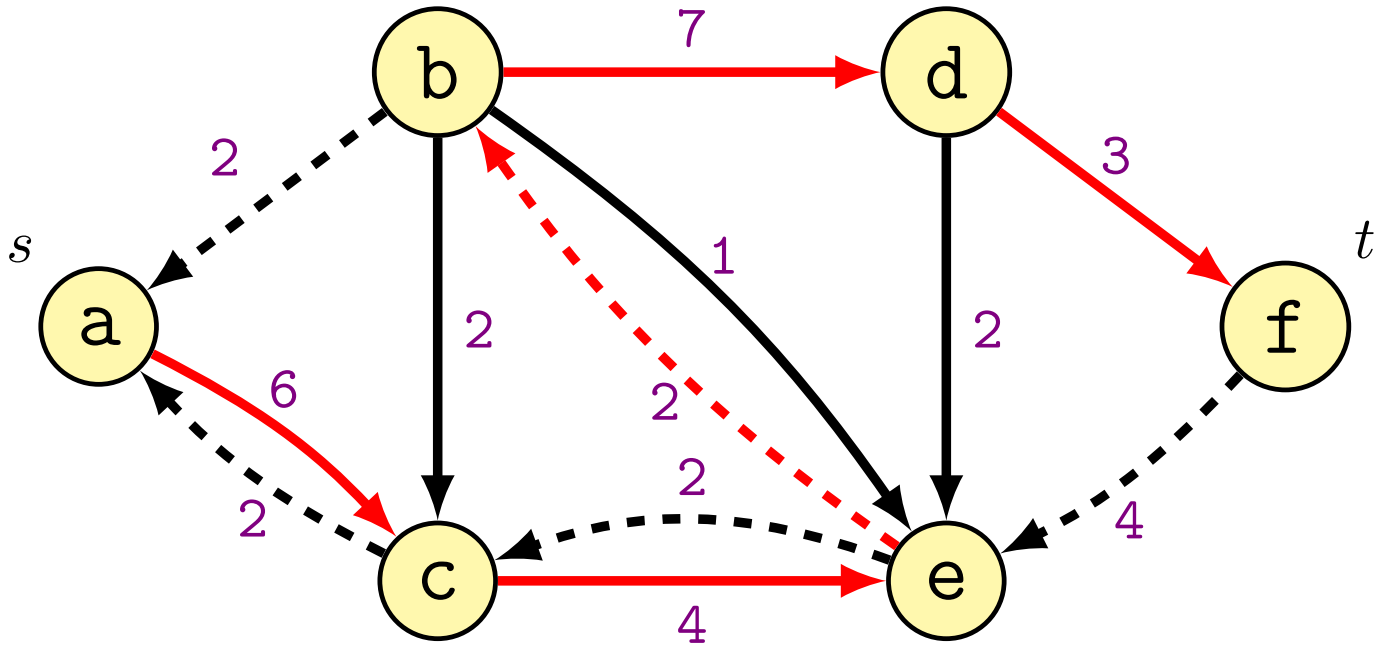
$f(\mathtt{a},\mathtt{b}) = 2$
$f(\mathtt{b},\mathtt{e}) = 2$
$f(\mathtt{e},\mathtt{f}) = 2$

Esecuzione



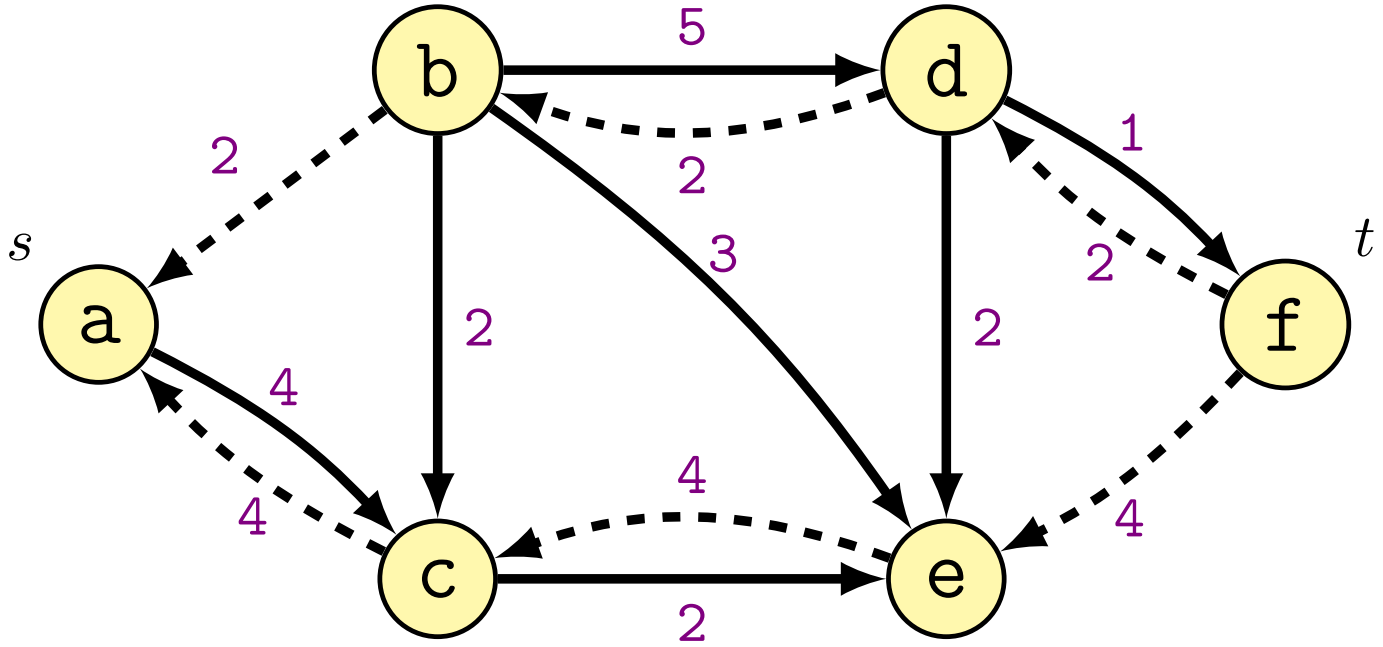
$f(\mathtt{a},\mathtt{b}) = 2$
$f(\mathtt{b},\mathtt{e}) = 2$
$f(\mathtt{e},\mathtt{f}) = 2 \not{4}$
$f(\mathtt{a},\mathtt{c}) = 2$
$f(\mathtt{c},\mathtt{e}) = 2$

Esecuzione



$f(\mathtt{a},\mathtt{b}) = 2$
$f(\mathtt{b},\mathtt{e}) = 2$
$f(\mathtt{e},\mathtt{f}) = 2 \not{4}$
$f(\mathtt{a},\mathtt{c}) = 2$
$f(\mathtt{c},\mathtt{e}) = 2$

Esecuzione



$f(\mathtt{a},\mathtt{b}) = 2$
$f(\mathtt{b},\mathtt{e}) = 2 \not{2}\ 0$
$f(\mathtt{e},\mathtt{f}) = 2 \not{2}\ 4$
$f(\mathtt{a},\mathtt{c}) = 2 \not{2}\ 4$
$f(\mathtt{c},\mathtt{e}) = 2 \not{2}\ 4$
$f(\mathtt{b},\mathtt{d}) = 2$
$f(\mathtt{d},\mathtt{f}) = 2$

Ricerca del cammino

Ford e Fulkerson suggerirono una semplice visita del grafo, in profondità oppure in ampiezza.

Edmonds e Karp suggerirono di utilizzare una visita in ampiezza.

Costo della visita: $O(V + E)$

Ricerca del cammino

```
/**
 * Compute the max-flow using the Ford-Fulkerson algorithm
 * @param C the capacity matrix
 * @param s the source node
 * @param t the sink node
 * @return the flow matrix
 */
private static int[][] flow(int[][] C, int s, int t) {
  // Create an empty flow
  int[][] F = new int[C.length][C.length];
  // Visited array to perform DFS, initially empty
  boolean[] visited = new boolean[C.length];
  // Repeat until there is no path
  while (dfs(C, F, s, t, visited, Integer.MAX_VALUE) > 0) {
    Arrays.fill(visited, false);
  }
  return F;
}
```

Ricerca del cammino

```
/**
 * Performs a DFS starting from node i and trying to reach node t.
 * @param C the capacity matrix; if capacity[x][y]>0, there is a edge from x to y
 * @param F the flow matrix to be computed
 * @param i the current node,
 * @param t the sink node
 * @param visited the boolean set containing the nodes that have been visited
 * @param min the smallest capacity found during the visit.
 * @returns the value of the additional flow found during the DFS
 */
private static int dfs(int[][] C, int[][] F, int i, int t, boolean[] visited, int min) {
  if (i==t) return min;              // If sink has been reached, terminate
  visited[i] = true;
  for (int j=0; j < C.length; j++) {
    if (C[i][j] > 0 && !visited[j]) {    // Non-visited neighbor
      int val = dfs(C, F, j, t, visited, Math.min(min, C[i][j]));
      if (val > 0) {
        C[i][j] = C[i][j]-val; C[j][i] = C[j][i]+val;
        F[i][j] = F[i][j]+val; F[j][i] = F[j][i]-val;
        return val;
      }
    }
  }
  return 0;                    // The sink has not been found
}
```

Complessità

Complessità, limite superiore – Versione Ford-Fulkerson

Se le capacità della rete sono intere, l'algoritmo di Ford e Fulkerson ha complessità $O((V + E)|f^*|)$ o $O(V^2|f^*|)$ nel caso pessimo.

- L'algoritmo parte dal flusso nullo e termina quando il valore totale del flusso raggiunge $|f^*|$
- Ogni incremento è positivo, nel senso che il valore del flusso viene incrementato di almeno uno dal cammino aumentante
- Ogni ricerca di un cammino richiede una visita del grafo, con costo $O(V + E)$ / $O(V^2)$;
- La somma dei flussi e il calcolo della rete residua può essere effettuato in tempo $O(V + E)$ / $O(V^2)$.

Complessità

Complessità, limite superiore – Edmonds e Karp

Se le capacità della rete sono intere, l'algoritmo di Edmonds e Karp ha complessità $O(VE^2)$ nel caso pessimo.

Come si conciliano i due limiti superiori?

- $O(VE^2)$ vs $O((V+E)|f^*|)$
- Sono entrambi limiti superiori
- Sono entrambi validi
- Si deve quindi prendere il più basso fra i due

Dimostrazione complessità

Teorema

La complessità dell'algoritmo di Edmonds-Karp è $O(VE^2)$.

- Vengono eseguiti $O(VE)$ aumenti di flusso, ognuno dei quali richiede una visita in ampiezza $O(V+E)$.
- $O(VE(V+E)) = O(VE^2)$

Lemma - Aumenti di flusso

Il numero totale di aumenti di flusso eseguiti dall'algoritmo di Edmonds e Karp è $O(VE)$.

Dimostrazione complessità

Lemma - Monotonia

Sia $\delta_f(s, v)$ la distanza minima da s a v in una rete residua G_f. Sia $f' = f + g$ un flusso nella rete iniziale, con g flusso non nullo derivante da un cammino aumentante. Allora $\delta_{f'}(s, v) \geq \delta_f(s, v)$.

- Quando viene aggiunto un cammino, alcuni archi si "spengono": hanno una capacità residua di 0
- Questi archi erano utilizzati nei cammini minimi (BFS)
- La sparizione di questi archi non può rendere più corto un cammino minimo

Dimostrazione complessità

Lemma - Aumenti di flusso

Il numero totale di aumenti di flusso eseguiti dall'algoritmo di Edmonds e Karp è $O(VE)$.

- Sia G_f una rete residua
- Sia C un cammino aumentante di G_f.
- (u, v) è un arco critico (collo di bottiglia) in C se

$$c_f(u, v) = \min_{(x,y \in C)} \{c_f(x, y)\}$$

- In ogni cammino esiste almeno un arco critico
- Una volta aggiunto il flusso associato a C, l'arco critico scompare dalla rete residua.

Dimostrazione complessità

- Poiché i cammini aumentanti sono cammini minimi, abbiamo che:
$$\delta_f(s,v) = \delta_f(s,u) + 1$$
- L'arco (u,v) potrà ricomparire se e solo se il flusso lungo l'arco diminuirà, ovvero se (v,u) appare in un cammino aumentante
- Sia g il flusso quando questo accade; come sopra, abbiamo:
$$\delta_g(s,u) = \delta_g(s,v) + 1$$
- Per Lemma 1, abbiamo anche che $\delta_f(s,v) \leq \delta_g(s,v)$; quindi:
$$\begin{aligned} \delta_g(s,u) &= \delta_g(s,v) + 1 \\ &\geq \delta_f(s,v) + 1 \\ &\geq \delta_f(s,u) + 2 \end{aligned}$$

Dimostrazione complessità

- Dal momento in cui un nodo è critico al momento in cui può tornare ad essere critico , il cammino minimo si è allungato almeno di due passi.
- La lunghezza massima del cammino fino a u, tenuto conto che poi si deve ancora seguire l'arco (u, v), è $V - 2$.
- Quindi un arco può diventare critico al massimo $(V - 2)/2 = V/2 - 1$ volte.
- Poiché ci sono $O(E)$ archi che possono diventare critici $O(V)$ volte, abbiamo che il numero massimo di flussi aumentanti è $O(VE)$.

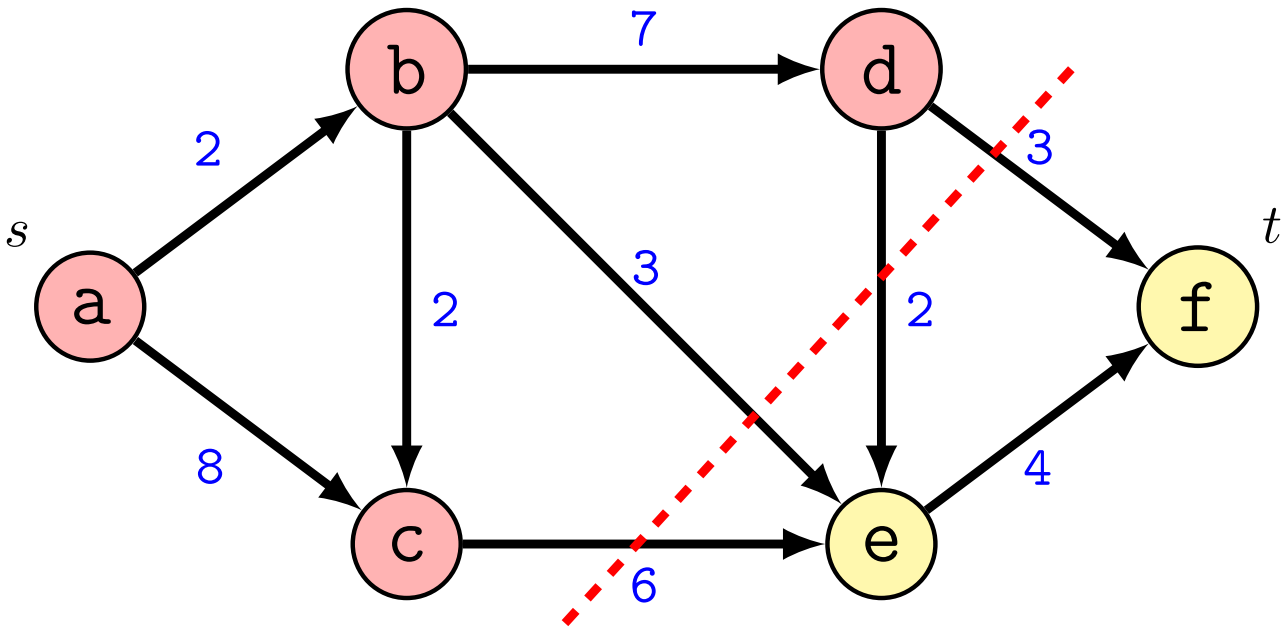
Complessità – Altre versioni

Nome	**Complessità**	**Note**
Ford-Fulkerson	$O(E\|f^*\|)$	Converge con valori razionali
Edmonds-Karp	$O(VE^2)$	Specializzazione basata su BFS
Dinic, blocking flow	$O(V^2E)$	In alcune reti particolari, $O((V^{2/3}, E^{1/2})E)$
MPM	$O(V^3)$	Solo su DAG
Dinic	$O(VE \log V)$	Struttura dati Dynamic trees
Goldberg e Rao	$O(\min(n^{2/3}, m^{1/2})m \log(n^2/m+2) \log C)$	$C = \max_{(u,v)\in E} c(u,v)$
Orlin + King,Rao,Tarjan	$O(VE)$	Pubblicato nel 2013!

Dimostrazione correttezza – Definizioni

Taglio

Un taglio (S, T) della rete di flusso $G = (V, E, s, t, c)$ è una partizione di V in S e $T = V - S$ tale che $s \in S$ e $t \in T$.



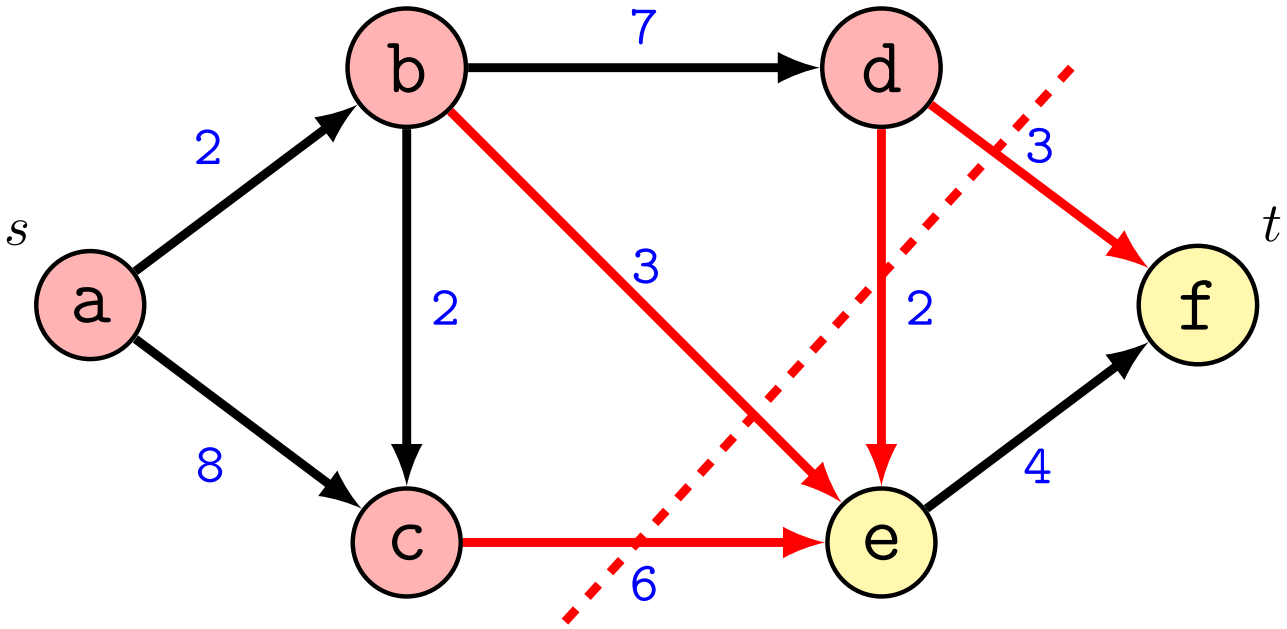
$S = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$
$T = \{\mathtt{e}, \mathtt{f}\}$

Dimostrazione correttezza – Definizioni

Capacità di un taglio

La capacità $c(S,T)$ attraverso il taglio (S,T) è pari a:

$$c(S,T) = \sum_{u \in S, v \in T} c(u,v)$$



$S = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$
$T = \{\mathtt{e}, \mathtt{f}\}$

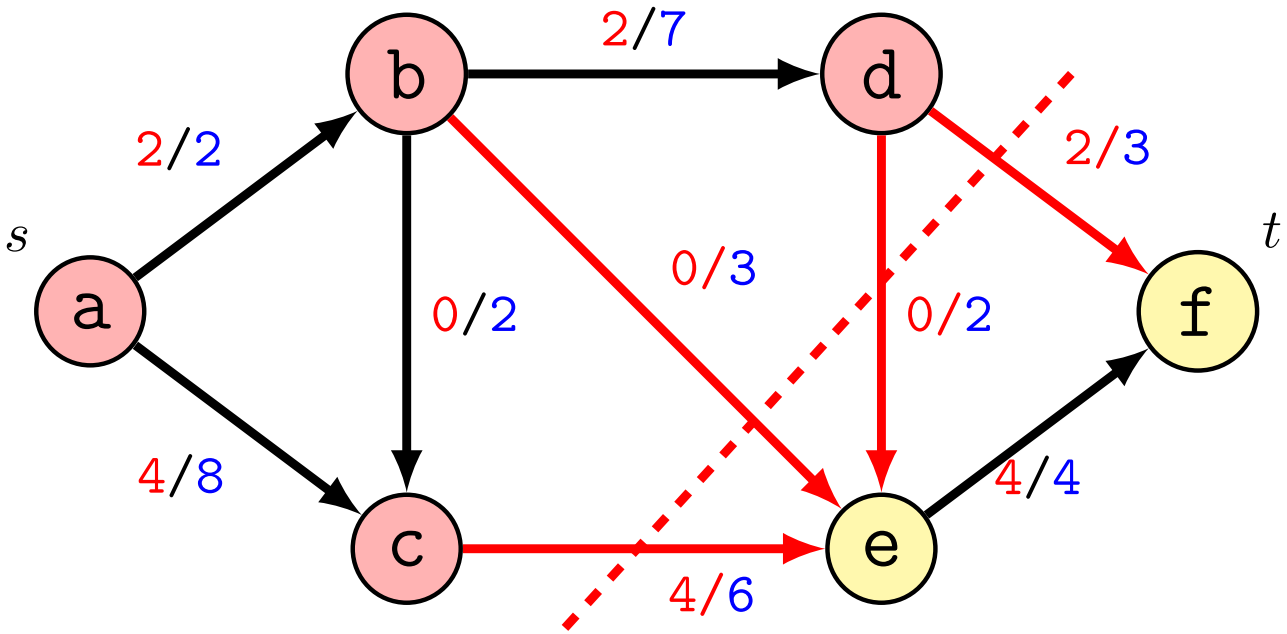
$C(S,T) = 14$

Dimostrazione correttezza – Definizioni

Flusso di un taglio

Se f è un flusso in G, il *flusso netto* $F(S,T)$ attraverso (S,T) è pari a:

$$F(S,T) = \sum_{u \in S, v \in T} f(u,v)$$



$S = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$
$T = \{\mathtt{e}, \mathtt{f}\}$

$C(S,T) = 14$

$F(S,T) = 6$

Dimostrazione correttezza – Lemma

Lemma

Dato un flusso f e un taglio (S,T), la quantità di flusso $F(S,T)$ che attraversa il taglio è uguale a $|f|$.

$$
\begin{aligned}
F(S,T) &= \sum_{u\in S, v\in T} f(u,v) \qquad\qquad T = V - \Sigma \\
&= \sum_{u\in S, v\in V} f(u,v) - \sum_{u\in S, v\in S} f(u,v) \\
&= \sum_{u\in S-\{s\}, v\in V} f(u,v) + \sum_{v\in V} f(s,v) - \sum_{u\in S, v\in S} f(u,v) \\
&= 0 + |f| + 0
\end{aligned}
$$

dove la prima uguaglianza a zero è vera per la conservazione del flusso, mentre la seconda è vera per antisimmetria.

Dimostrazione correttezza – Rapporto fra flusso e taglio

Un taglio definisce un limite superiore al flusso massimo che può essere presente nella rete.

- Nessun flusso attraverso un taglio supera la capacità del taglio

$$F(S,T) \leq C(S,T) \qquad \forall S \subseteq V, T = V - S$$

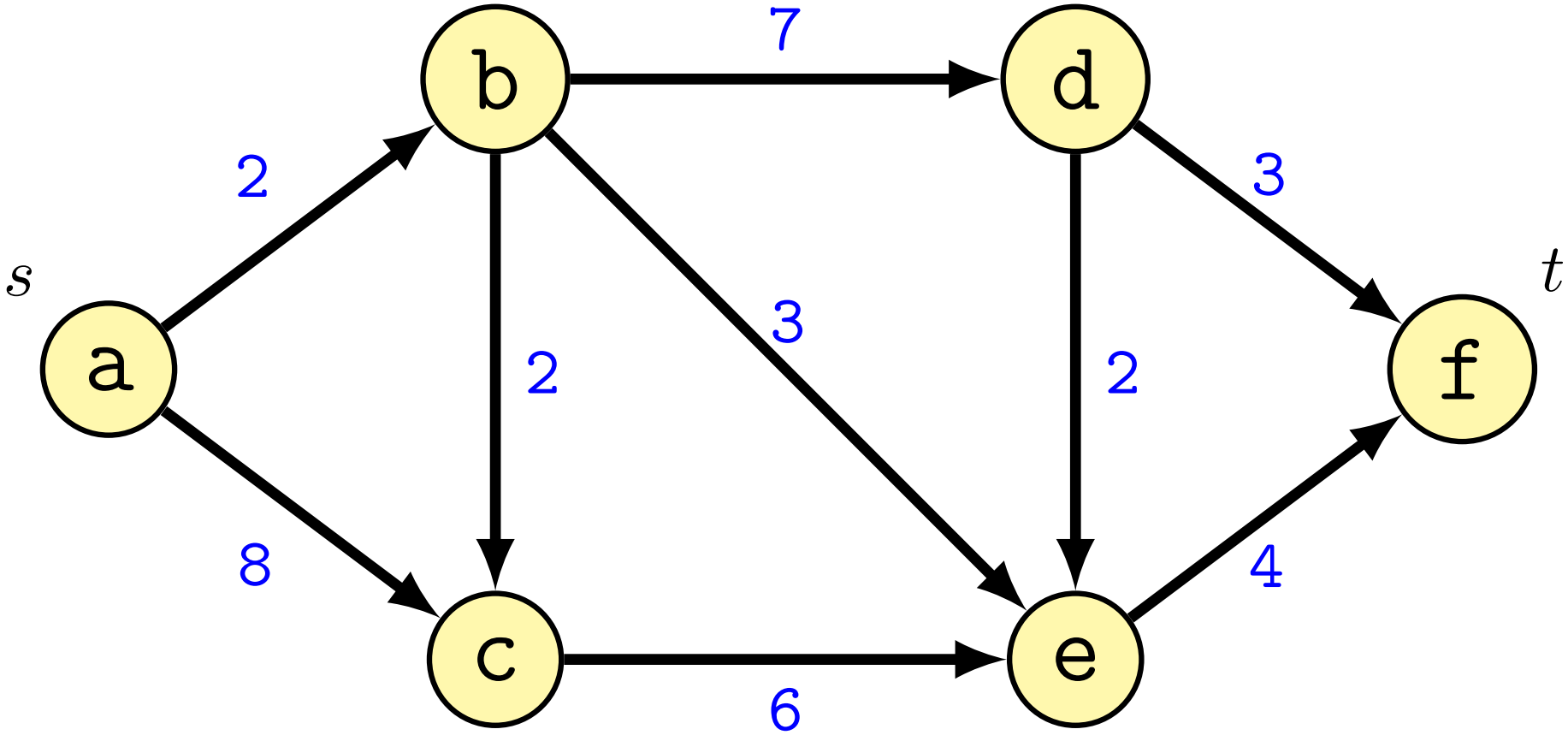
- Il flusso che attraversa un taglio è uguale al valore del flusso

$$F(S,T) = |f| \qquad \forall S \subseteq V, T = V - S$$

- Quindi, il valore del flusso è limitato superiormente dalla capacità di tutti i possibili tagli.

$$|f| \leq C(S,T) \qquad \forall S \subseteq V, T = V - S$$

Dimostrazione di correttezza



Teorema del taglio minimo / flusso massimo

Teorema

Le seguenti tre affermazioni sono equivalenti:

1. f è un flusso massimo
2. non esiste nessun cammino aumentante per G
3. esiste un taglio minimo (S, T) tale che $c(S, T) = |f|$

Dimostreremo circolarmente:

- $(1) \Rightarrow (2)$
- $(2) \Rightarrow (3)$
- $(3) \Rightarrow (1)$

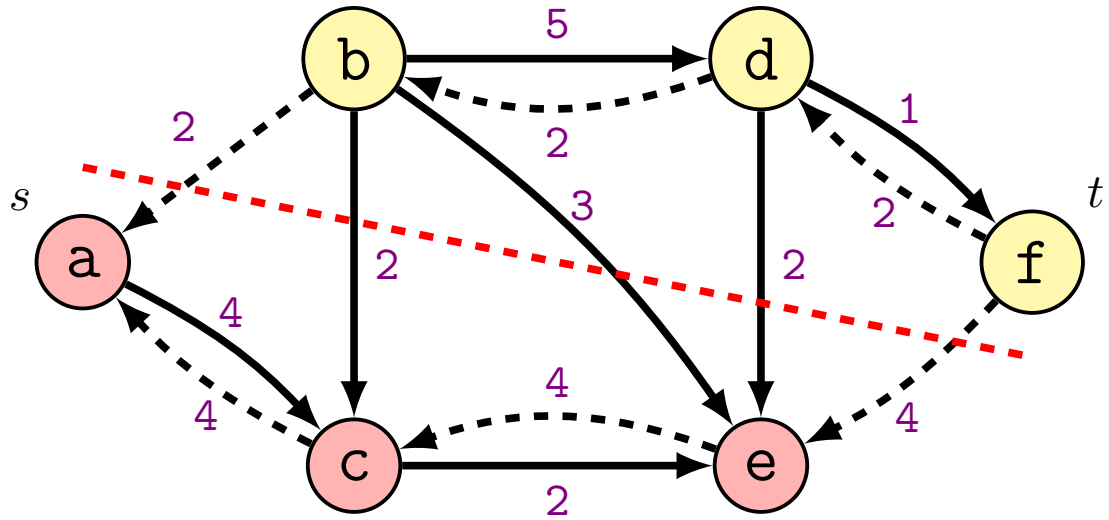
Dimostrazione correttezza – (1) $\Rightarrow$ (2)

f è un flusso massimo $\Rightarrow$ non esiste nessun cammino aumentante per G

- Se esistesse un cammino aumentante, il flusso potrebbe essere aumentato e quindi non sarebbe massimo (assurdo).

Dimostrazione correttezza – (2) ⇒ (3)

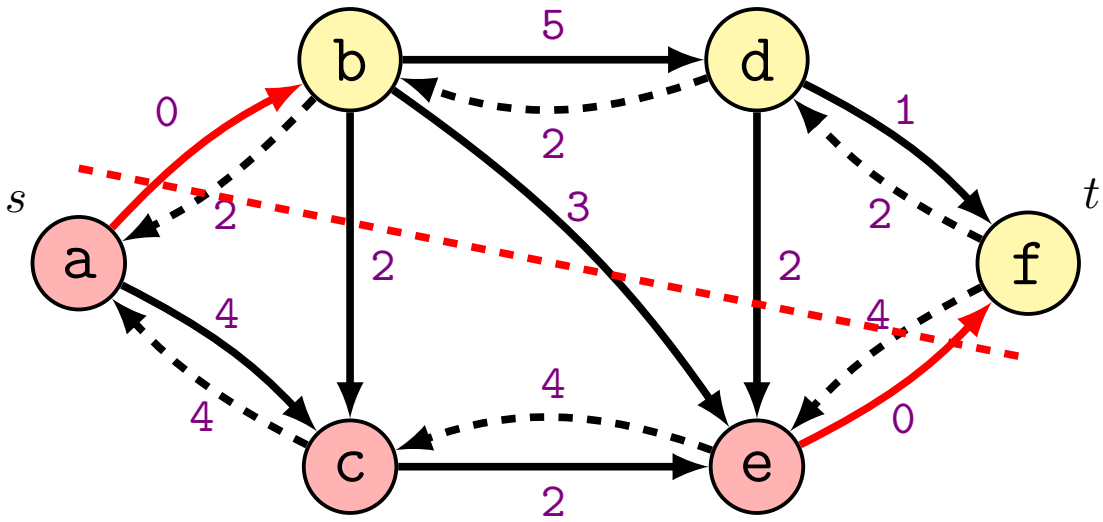
Non esiste nessun cammino aumentante per G ⇒
esiste un taglio minimo (S, T) tale che $c(S, T) = |f|$



- Poiché non esiste nessun cammino aumentante per f, non esiste nessun cammino da s a t nella rete residua G_f
- Sia S l'insieme dei vertici raggiungibili da s; $T = V - S$
- Ovviamente $s \in S$ e $t \in T$, quindi (S, T) è un taglio

Dimostrazione correttezza – (2) ⇒ (3)

Non esiste nessun cammino aumentante per G ⇒
esiste un taglio minimo (S, T) tale che $c(S, T) = |f|$



- Poiché t non è raggiungibile da s in G_f, tutti gli archi (u, v) con $u \in S$ e $v \in T$ sono saturati; ovvero, $f(u, v) = c(u, v)$.

Dimostrazione correttezza – (2) $\Rightarrow$ (3)

Non esiste nessun cammino aumentante per $G \Rightarrow$
esiste un taglio minimo (S, T) tale che $c(S, T) = |f|$

- Per il lemma precedente,

$$|f| = \sum_{u \in S, v \in T} f(u, v)$$

- Ne segue che:

$$|f| = \sum_{u \in S, v \in T} f(u, v) = \sum_{u \in S, v \in t} c(u, v) = C(S, T)$$

Dimostrazione correttezza – (3) $\Rightarrow$ (1)

Esiste un taglio (S, T) tale che $c(S, T) = |f| \Rightarrow$
f è un flusso massimo

Poiché per un qualsiasi flusso f e un qualsiasi taglio (S, T) vale la relazione $|f| \leq c(S, T)$, il flusso che soddisfa $|f| = c(S, T)$ deve essere massimo.

Applicazioni

- Bipartite matching
- Data mining
- Project selection
- Airline scheduling
- Baseball elimination
- Image segmentation
- Network connectivity
- Network reliability
- Distributed computing
- Egalitarian stable matching
- Security of statistical data
- Network intrusion detection
- Multi-camera scene reconstruction
- Gene function prediction

Abbinamento (matching) massimo nei grafi bipartiti

Problema - Job Assignment - Input

- Un insieme J contenente n job
- Un insieme W contenente m worker
- Una relazione $R \subseteq J \times W$, tale per cui $(j, w) \in R$ se e solo se il job j può essere useguito dal worker w

Problema - Job Assignment - Output

- Il più grande sottoinsieme $O \subseteq R$, tale per cui:
  - ogni job venga assegnato al più ad un worker
  - ad ogni worker venga assegnato al più un job

Esempio