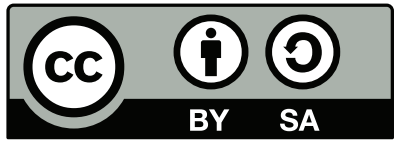
Algoritmi e Strutture Dati

Ricerca locale

Alberto Montresor

Università di Trento

2024/06/18



Sommario

Ricerca locale

Se si conosce una soluzione ammissibile (non necessariamente ottima) ad un problema di ottimizzazione, si può cercare una soluzione migliore nelle "vicinanze" di quella precedente.

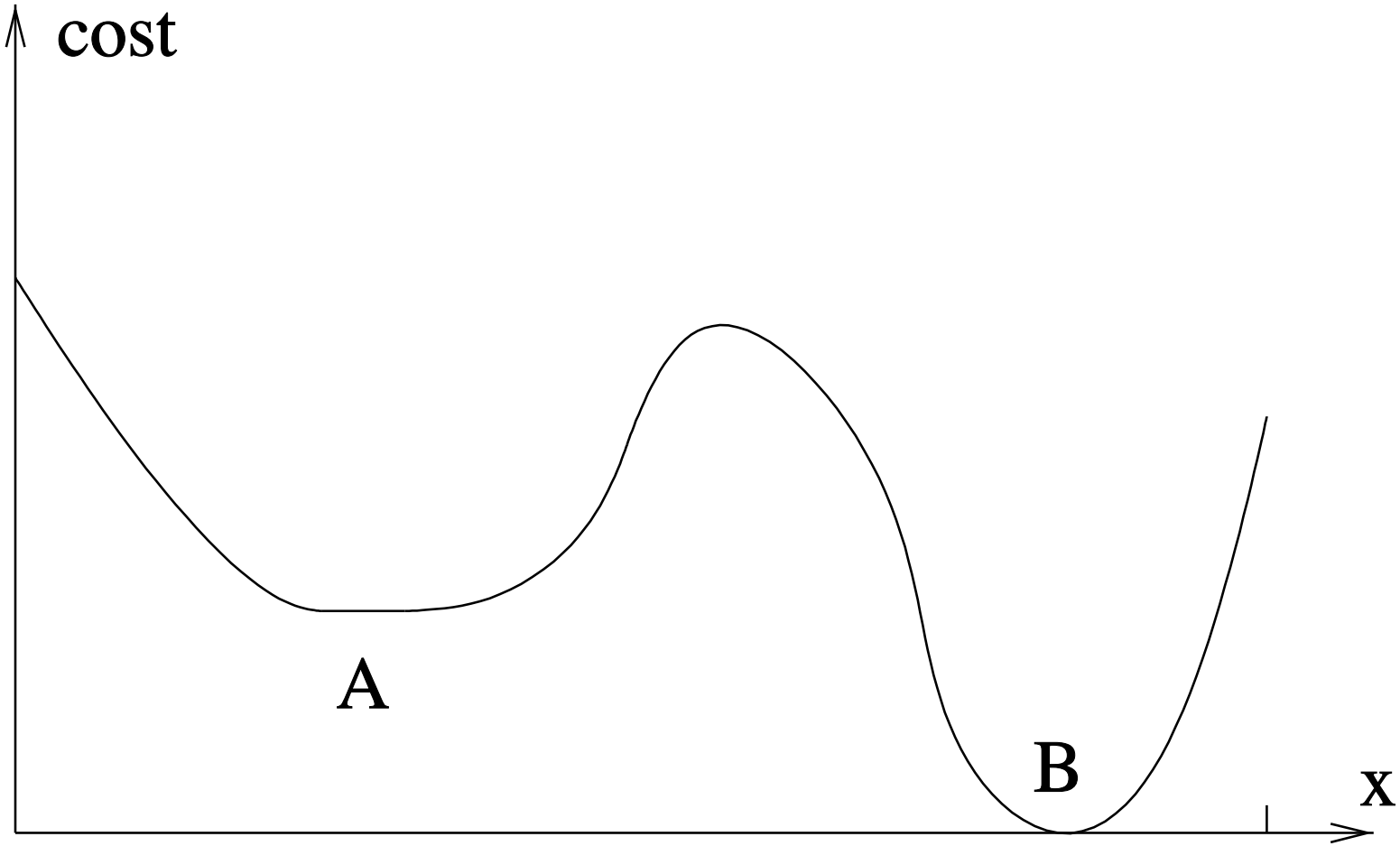
Si continua così fino a quando non si può più migliorare

---

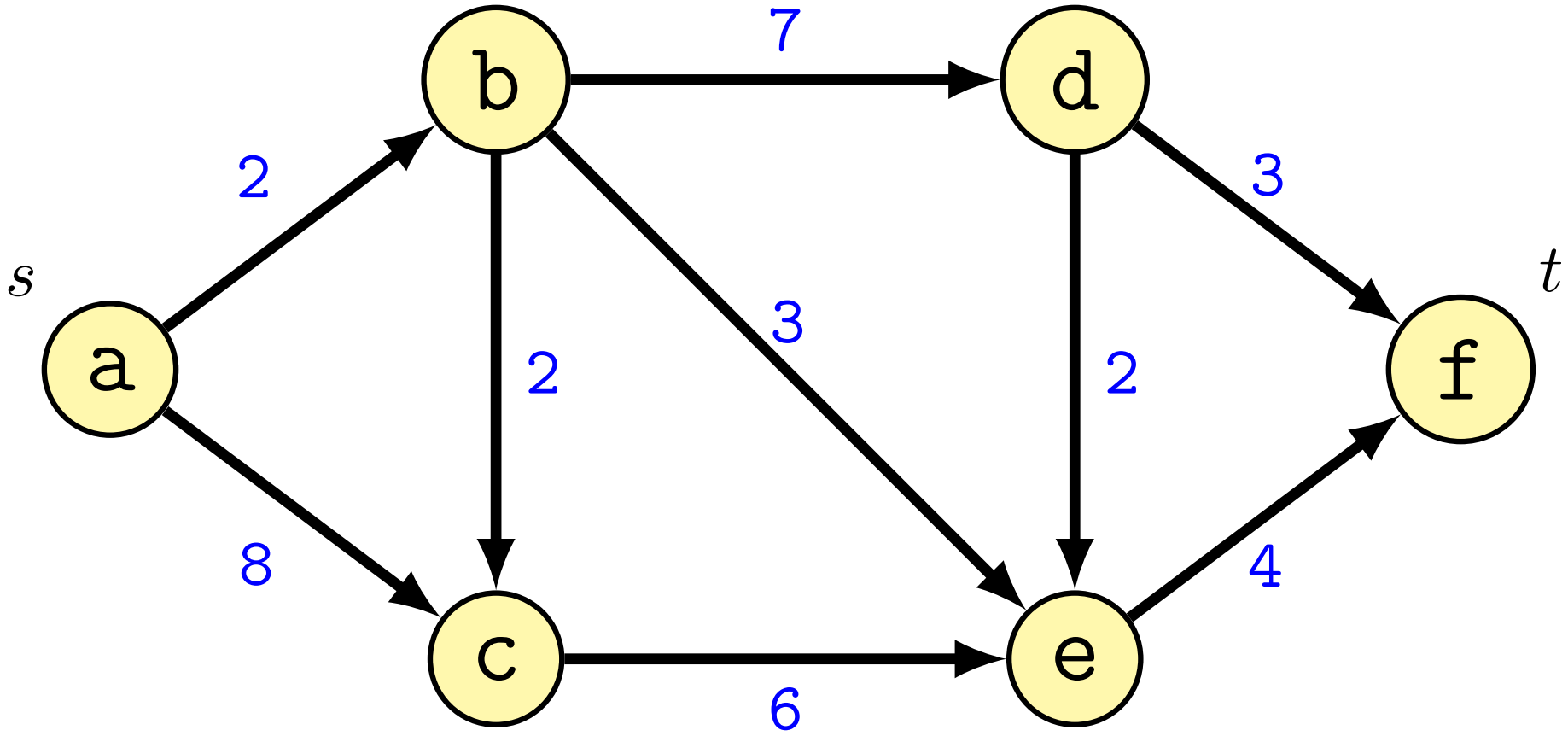
ricercaLocale()

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

Ricerca locale



Rete di flusso, sorgente, pozzo, capacità



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 che $(x, y) \notin E \Rightarrow c(x, y) = 0$.

Assunzioni

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

Flusso

Flusso

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

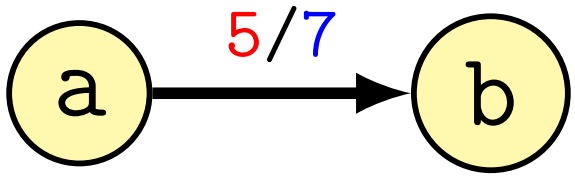
- Vincolo sulla capacità: $\forall x, y \in V,\ f(x, y) \leq c(x, y)$
- Antisimmetria: $\forall x, y \in V,\ f(x, y) = -f(y, x)$
- Conservazione del flusso: $\forall x \in V - \{s, t\},\ \sum_{y \in V} f(x, y) = 0$

Flusso

Vincolo sulla capacità

Il flusso non deve eccedere la capacità sull'arco.

$$\forall x, y \in V : f(x, y) \leq c(x, y)$$

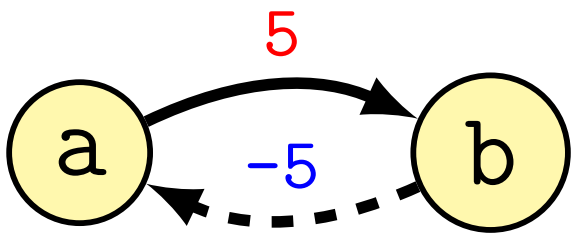


Flusso

Simmetria opposta

Il flusso che attraversa un arco in direzione (x, y) è l'opposto del flusso che attraversa l'arco in direzione (y, x).

$$\forall x, y \in V : f(x, y) = -f(y, x)$$



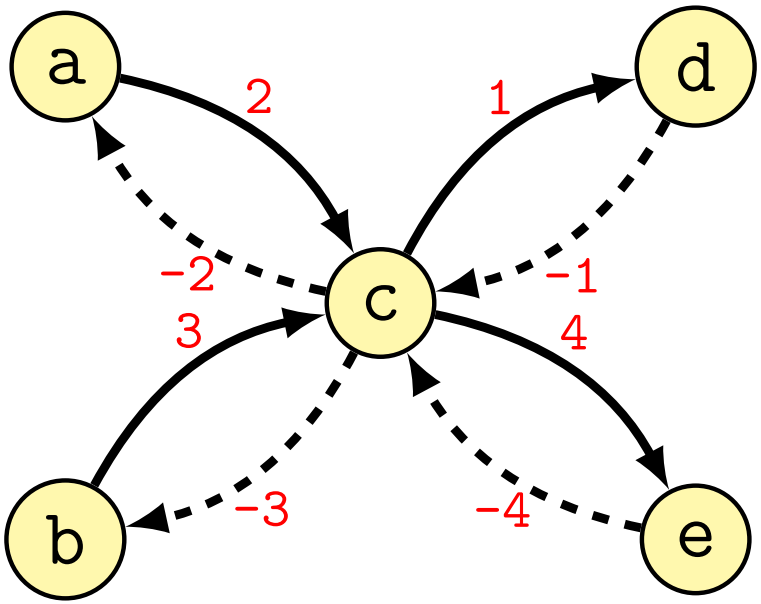
Il flusso viene definito in questo modo per semplificare la proprietà successiva e altre regole

Flusso

Conservazione del flusso

Per ogni nodo diverso da sorgente e pozzo, la somma dei flussi entranti deve essere uguale alla somma dei flussi uscenti.

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



Definizioni

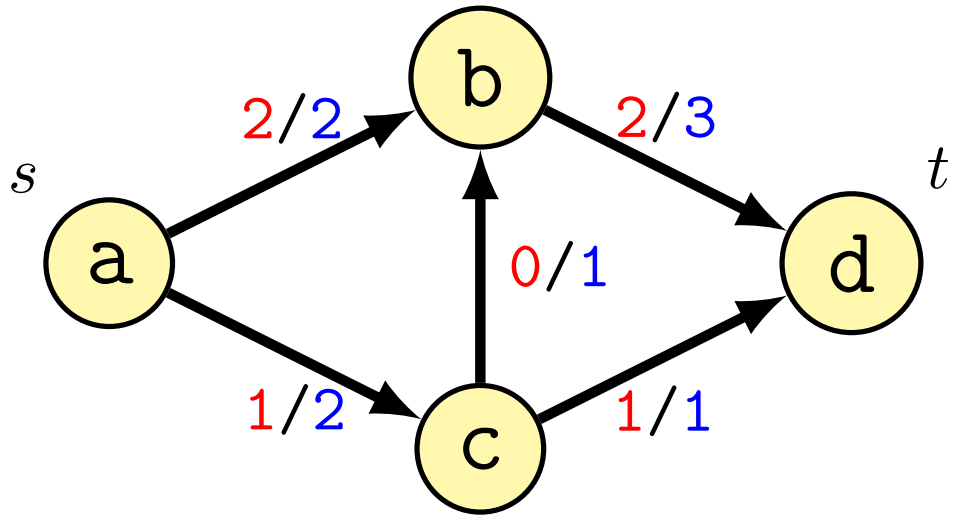
Valore del flusso

Il valore di un flusso f è definito come:

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

ovvero come la quantità di flusso uscente da s.

$|f| = 3$



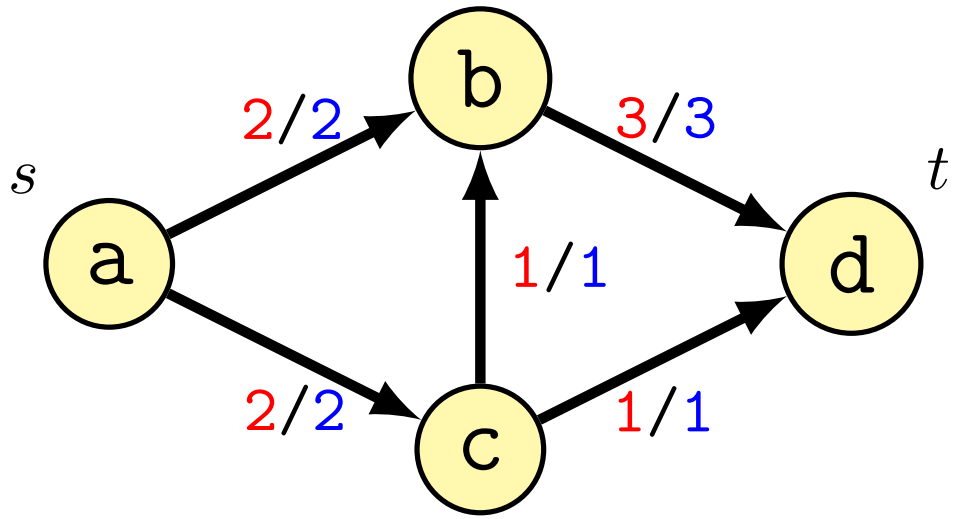
Problema

Flusso massimo

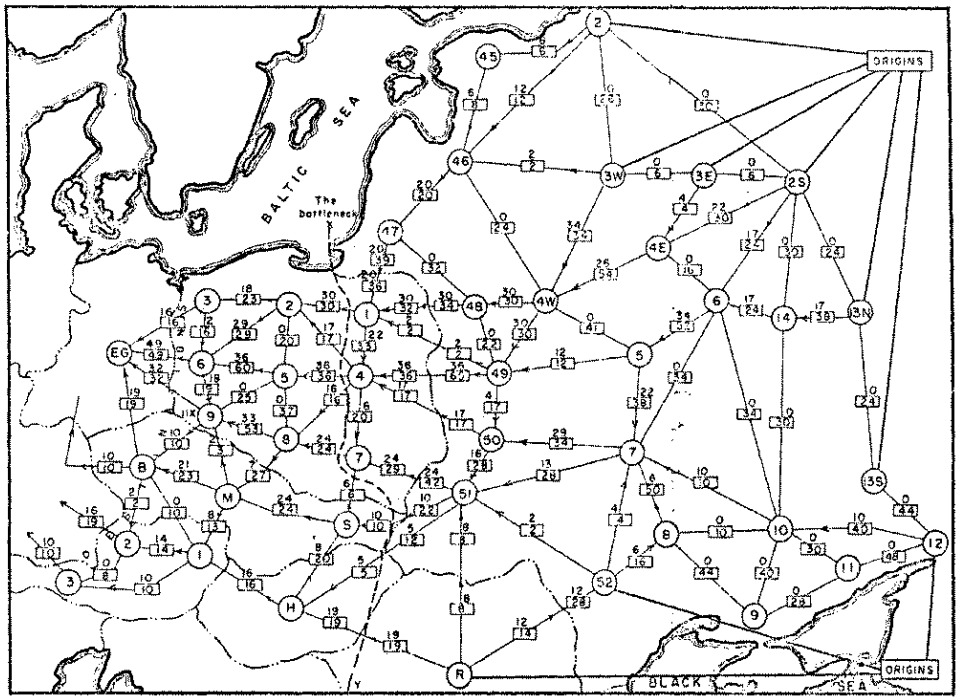
Data una rete $G = (V, E, s, t, c)$, trovare un flusso che abbia valore massimo fra tutti i flussi associabili alla rete.

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

$|f^*| = 4$



Flusso massimo nella guerra fredda

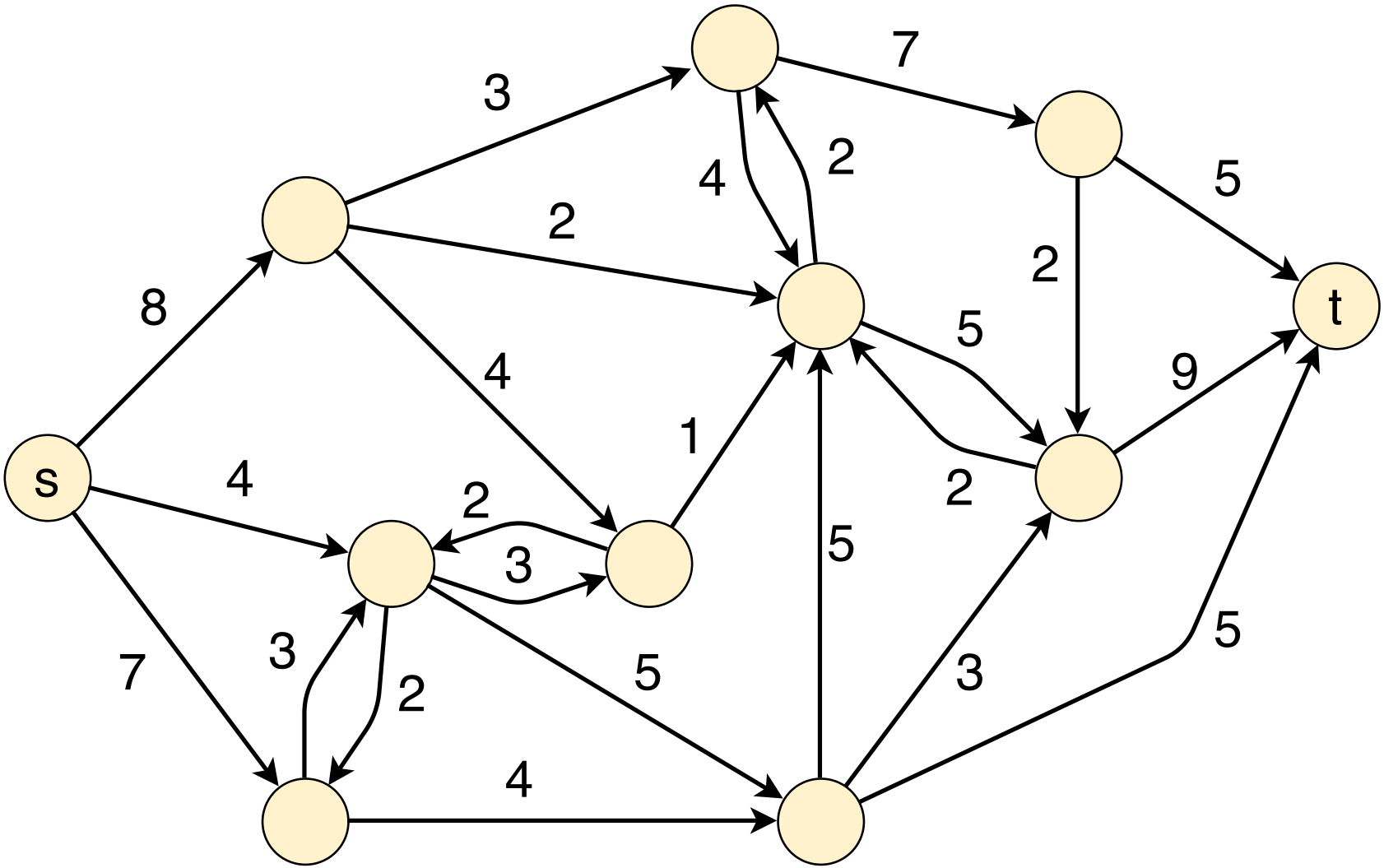


T. Harris, F. Ross, Fundamentals of a Method for Evaluating Rail Net Capacities. Research Memorandum RM-1573, The RAND Corporation, Santa Monica, California, 1955

Rete ferroviaria dell'Unione Sovietica occidentale e dell'Europa Orientale.
$|V| = 44, |E| = 105$.
Flusso massimo di $163 \cdot 10^6$ kg.

Alexander Schrijver. On the history of the transportation and maximum flow problems

Esercizio - Trovate il flusso massimo



https://disi.unitn.it/~montreso/asd/material/flow/flusso-esempio1.pdf

Metodo delle reti residue

Algoritmo, informale

- Si memorizza 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

È possibile dimostrare che questo approccio restituisce un flusso massimo

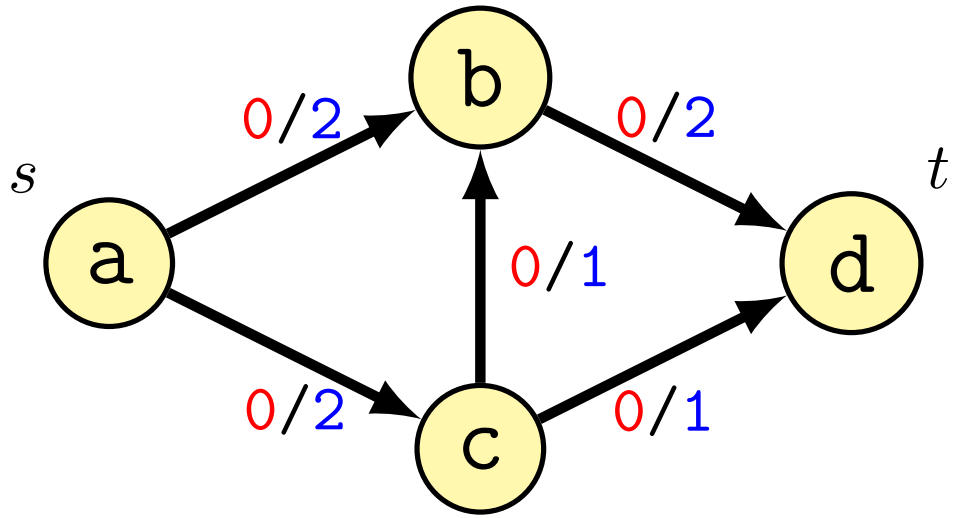
Definizioni

Flusso nullo

Definiamo flusso nullo la funzione $f_0 : V \times V \to \mathbf{R}^{\geq 0}$ tale che:

$$\forall x, y \in V : f(x, y) = 0$$

$|f| = 0$

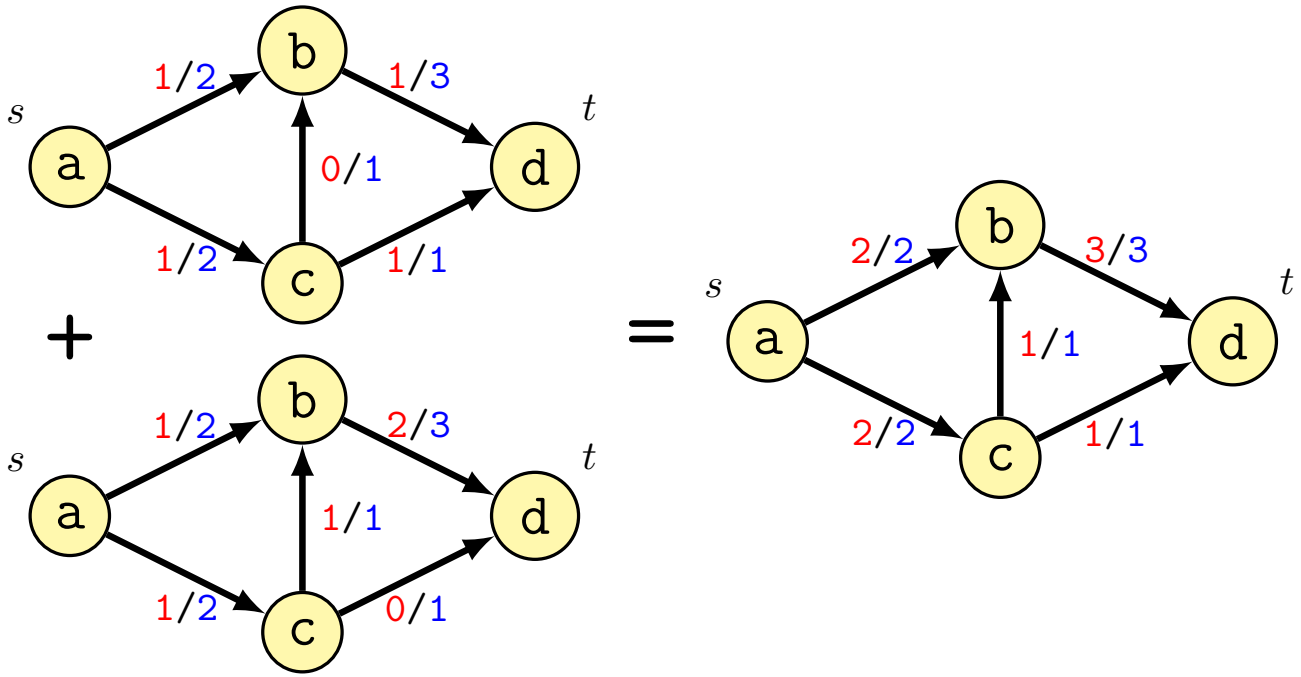


Definizioni

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 che

$$\forall x, y \in V : g(x, y) = f_1(x, y) + f_2(x, y)$$

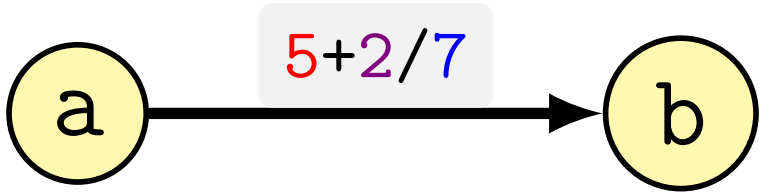


Definizioni

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

$$\forall x, y \in V : c_f(x, y) = c(x, y) - f(x, y)$$



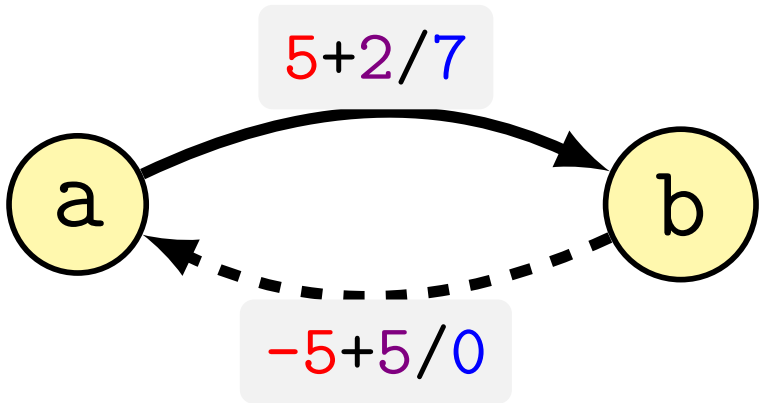
- Flusso in rosso
- Capacità residua in viola
- Capacità iniziale in blu

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

$$\forall x, y \in V : c_f(x, y) = c(x, y) - f(x, y)$$



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

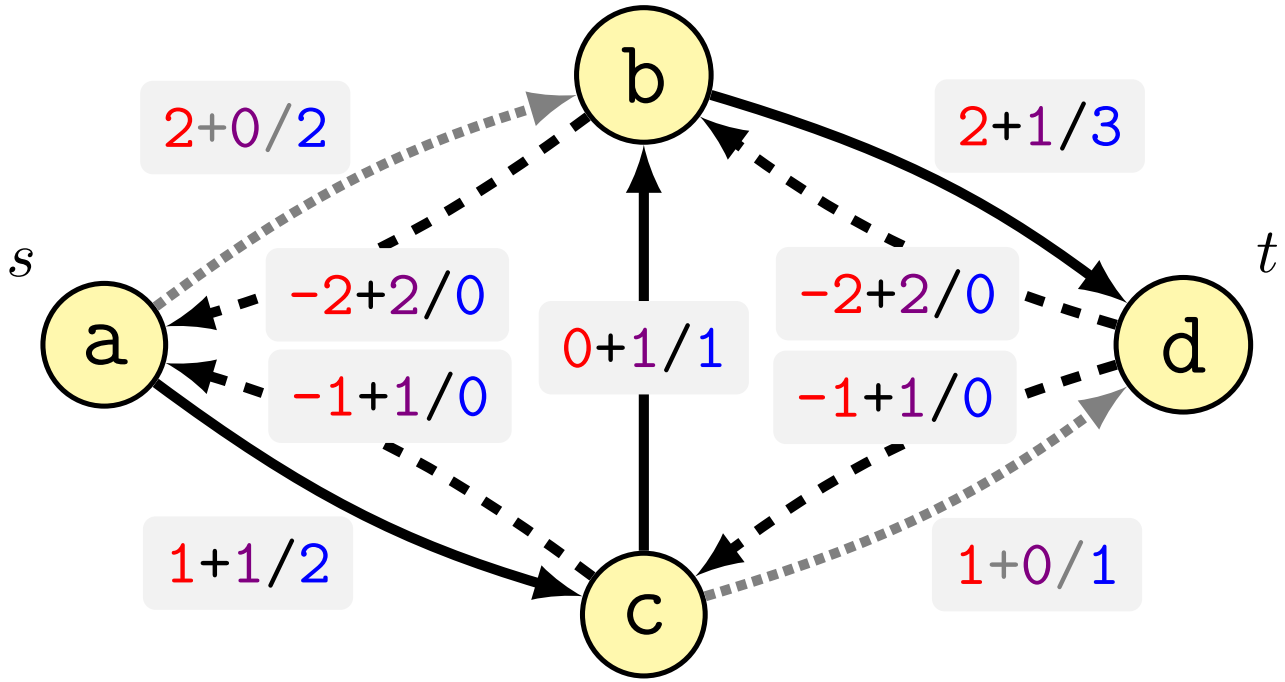
$$\begin{aligned} c_f(b, a) &= c(b, a) - f(b, a) \\ &= 0 - (-5) = 5 \end{aligned}$$

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:

$$\forall x, y \in V : (x, y) \in E_f \Leftrightarrow c_f(x, y) > 0$$



Algoritmo, schema generale

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

$f = f_0$ % Inizializza un flusso nullo

$c_f = c$ % Capacità iniziale

repeat

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

  $f = f + g$

  $c_f =$ Capacità 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.

Conservazione ($\forall x \in V - \{s, t\}$)

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

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 $(\forall x, y \in V)$

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

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.

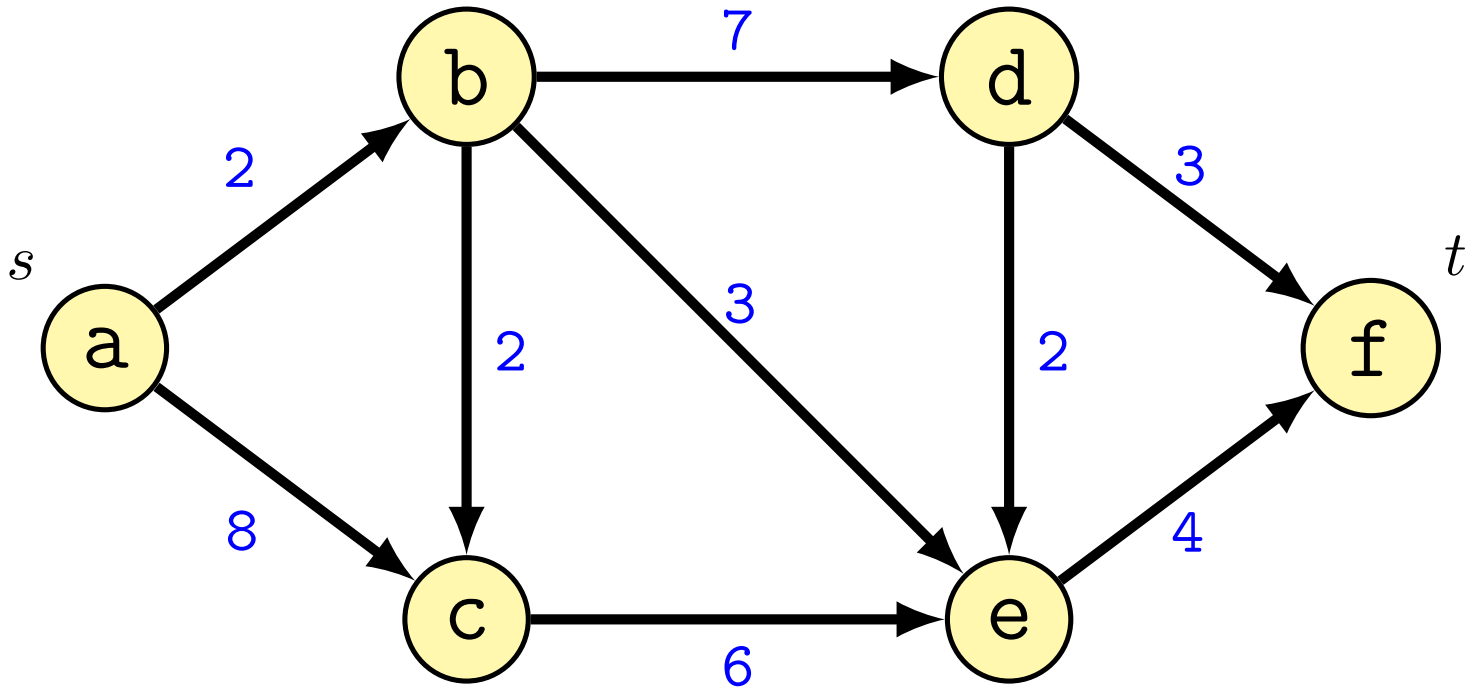
Vincolo sulla capacità ($\forall x, y \in V$)

$$
\begin{aligned}
g(x,y) &\leq c_f(x,y) && && g \text{ è un flusso in } G_f \\
f(x,y) + g(x,y) &\leq c_f(x,y) && + f(x,y) && \text{Aggiungo termine uguale} \\
(f+g)(x,y) &\leq c(x,y) - f(x,y) && + f(x,y) && \text{Sostituzione} \\
(f+g)(x,y) &\leq c(x,y) && && \text{Semplificazione}
\end{aligned}
$$

Flusso aggiuntivo

Domanda

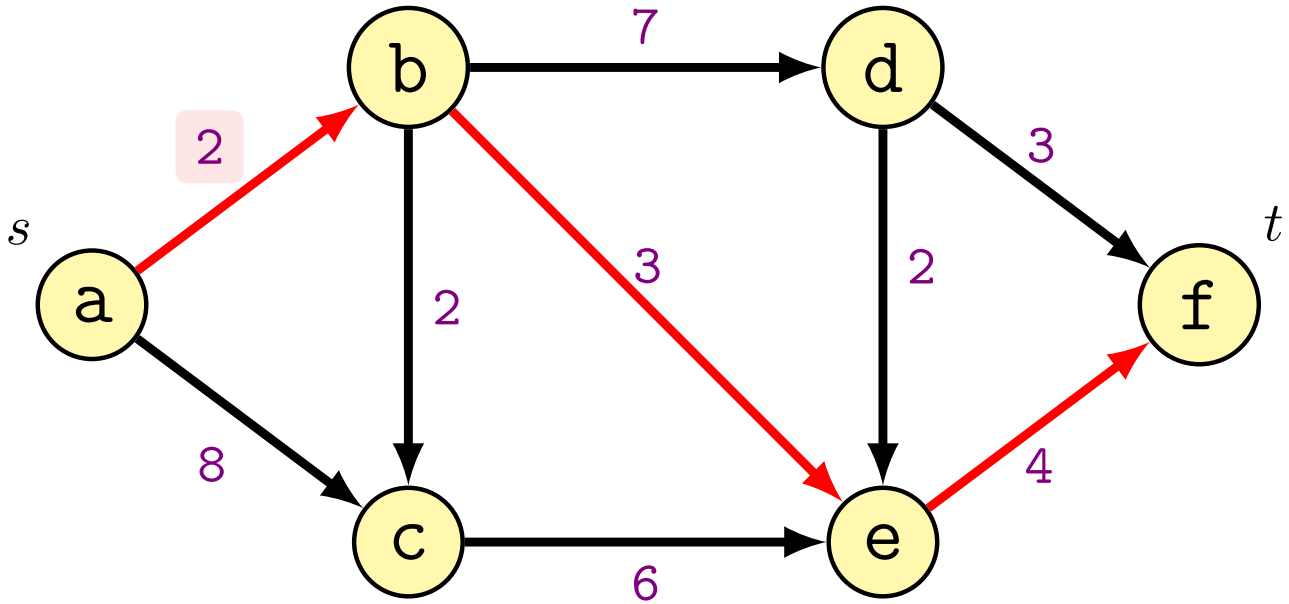
Il problema principale del metodo proposto è il seguente:
Come trovare un flusso aggiuntivo?



Cammini Aumentanti: Ford-Fulkerson, 1956

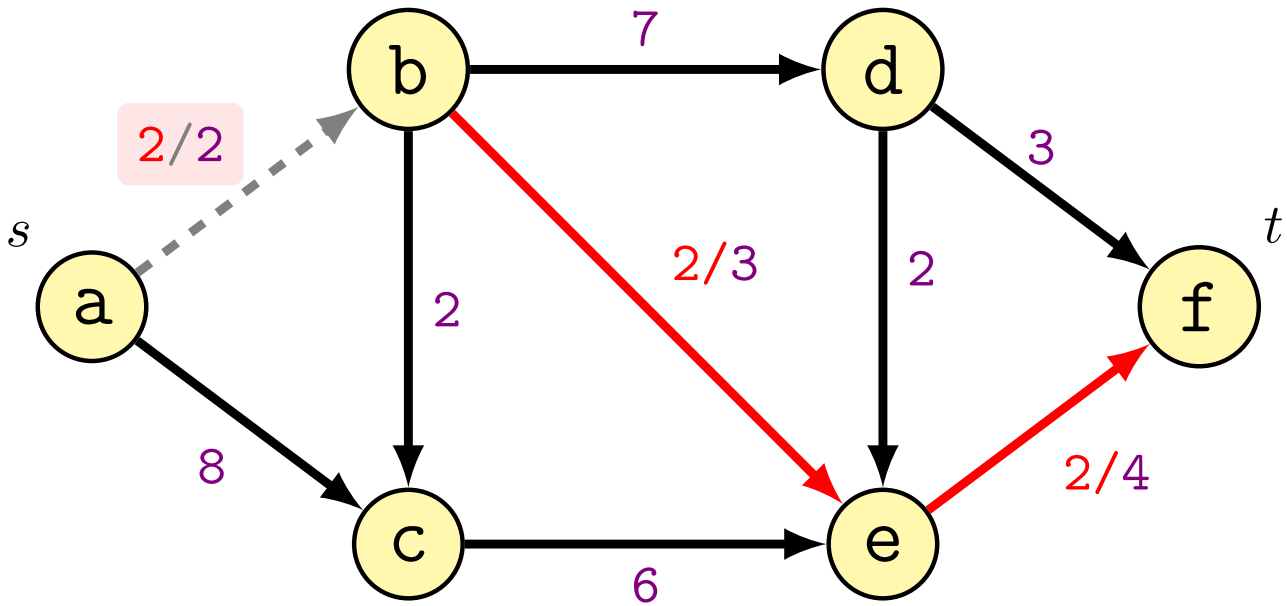
- Si trova un cammino $C = v_0, v_1, \ldots, v_n$, con $s = v_0$ e $t = v_n$ nella rete residua G_f;
- Si identifica la capacità del cammino, corrispondente alla minore capacità degli archi incontrati (collo di bottiglia):

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



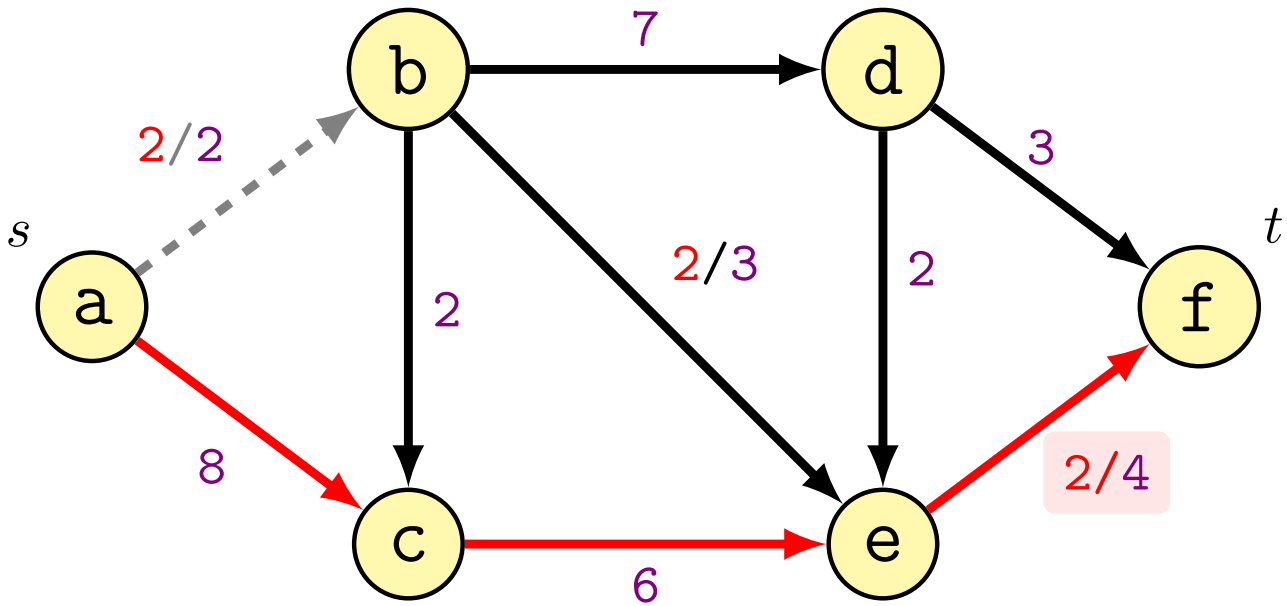
Cammini Aumentanti: Ford-Fulkerson, 1956

- 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)



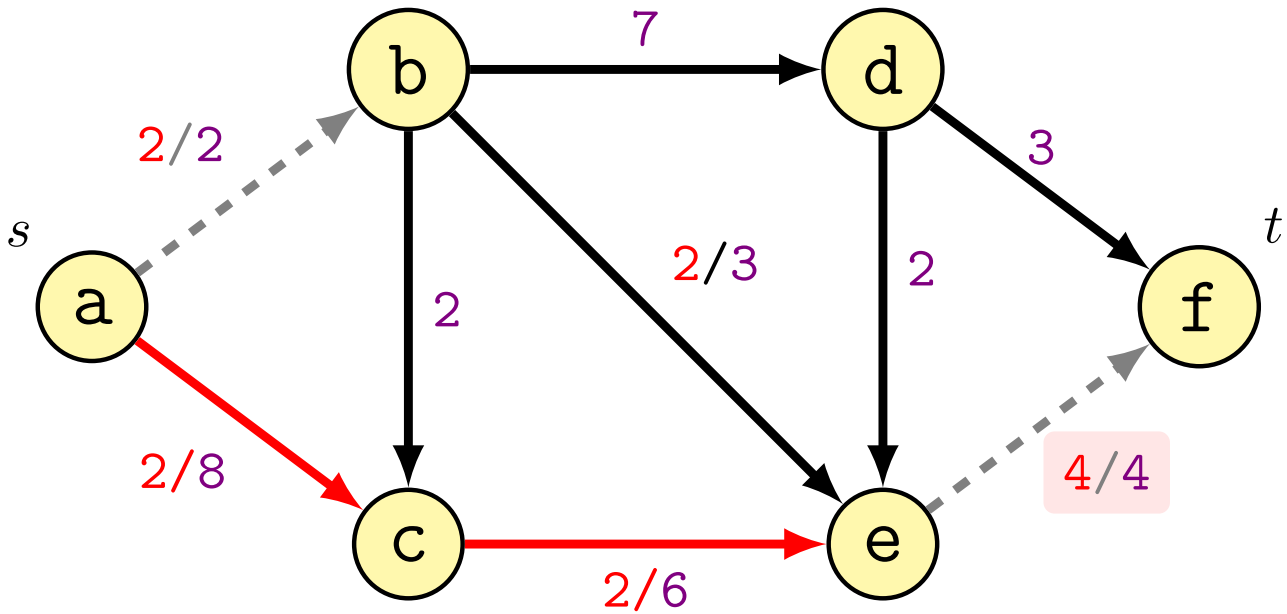
Cammini Aumentanti: Ford-Fulkerson, 1956

- 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)



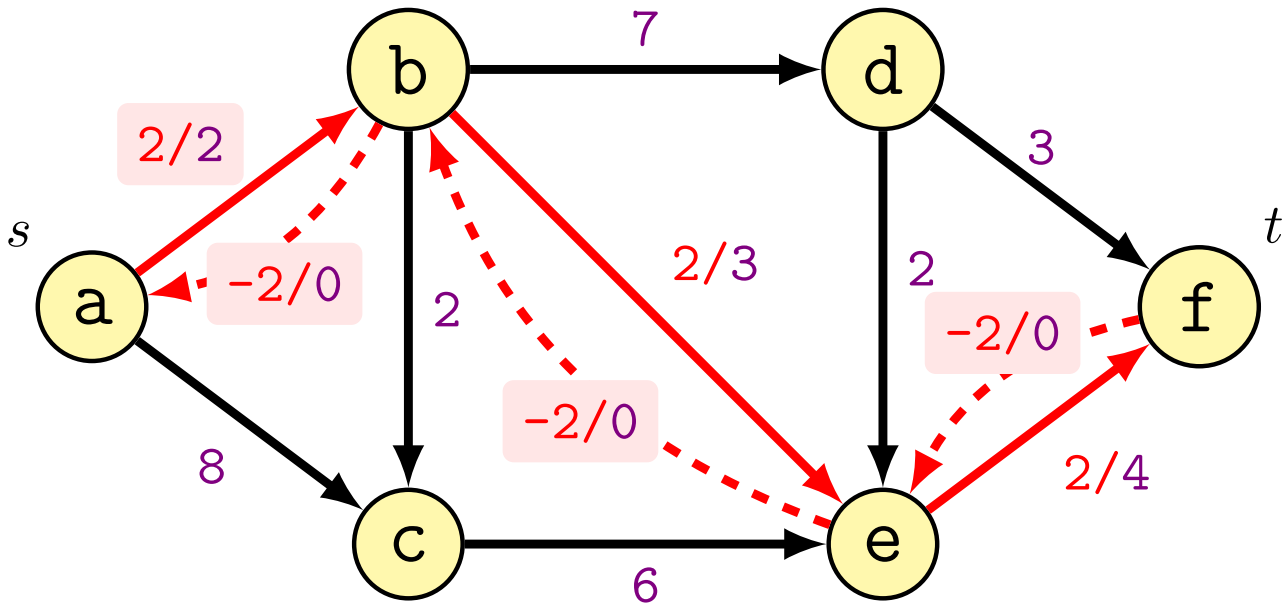
Cammini Aumentanti: Ford-Fulkerson, 1956

- 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)



Cammini Aumentanti: Ford-Fulkerson, 1956

- 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)



Cammini Aumentanti: Ford-Fulkerson, 1956

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

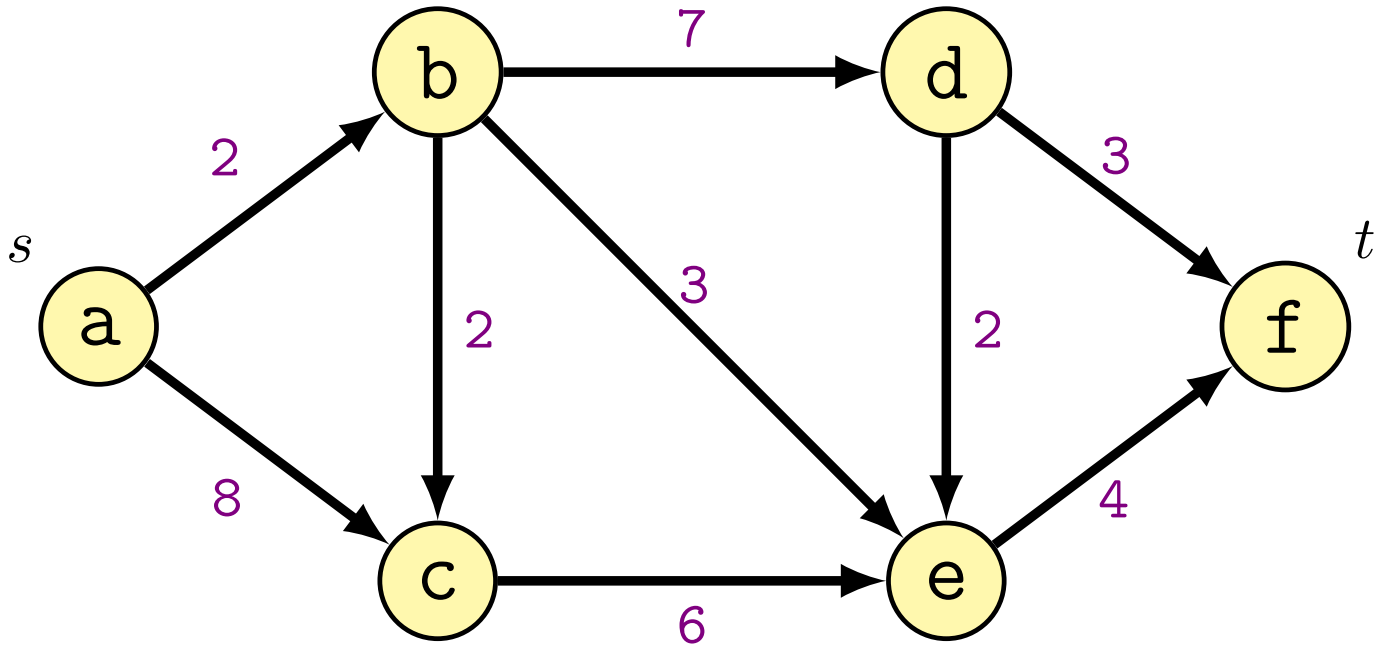
int[ ][ ] f = **new int**[ ][ ] % Flusso parziale
int[ ][ ] g = **new int**[ ][ ] % Flusso da cammino aumentante
int[ ][ ] c_f = **new int**[ ][ ] % Rete residua

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

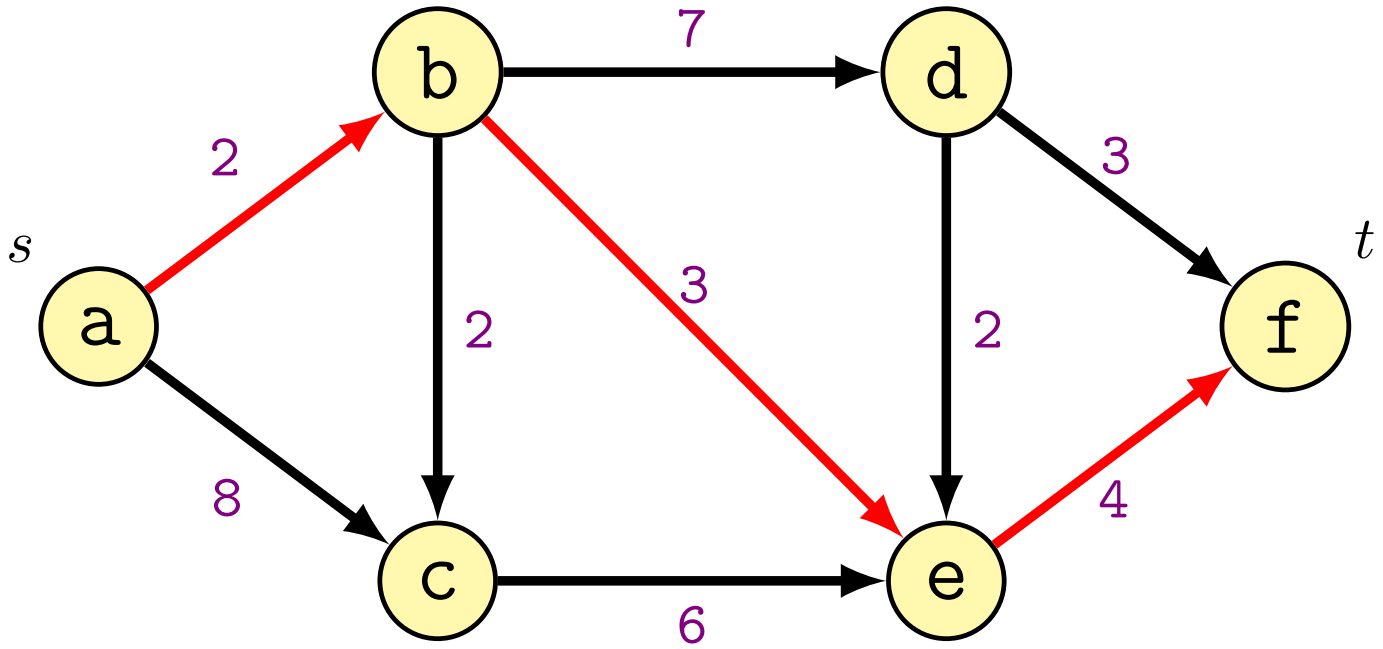
repeat
    g = flusso associato ad un cammino aumentante in r, oppure f_0
    foreach $x, y \in G.\mathsf{V}()$ **do**
        $f[x][y] = f[x][y] + g[x][y]$ % $f = f + g$
        $c_f[x][y] = c[x][y] - f[x][y]$ % Calcola c_f

until $g = f_0$
return f

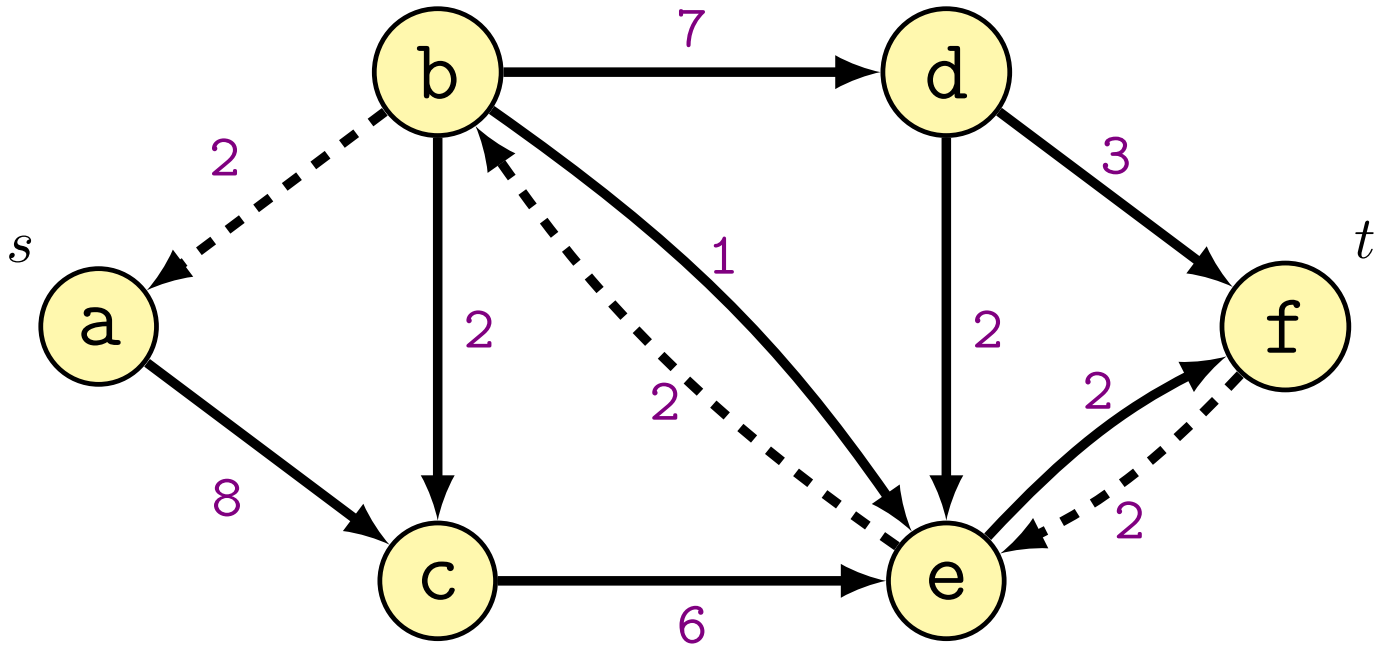
Esecuzione



Esecuzione

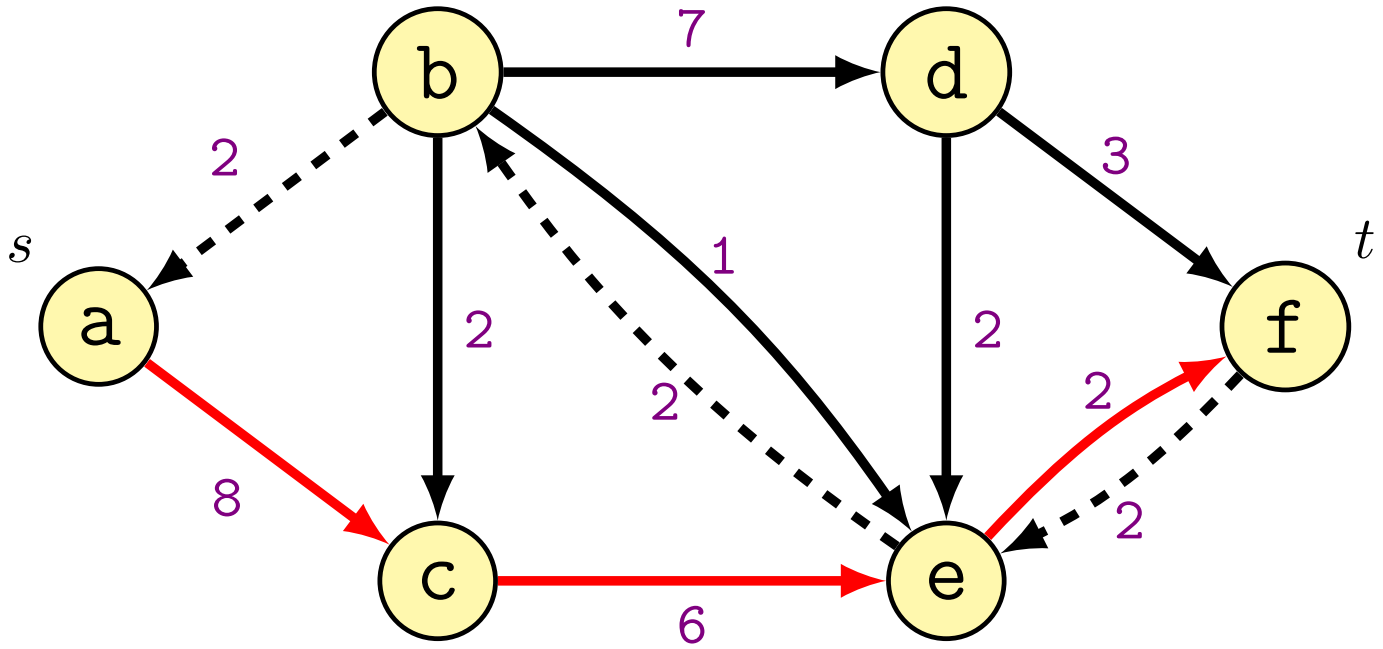


Esecuzione



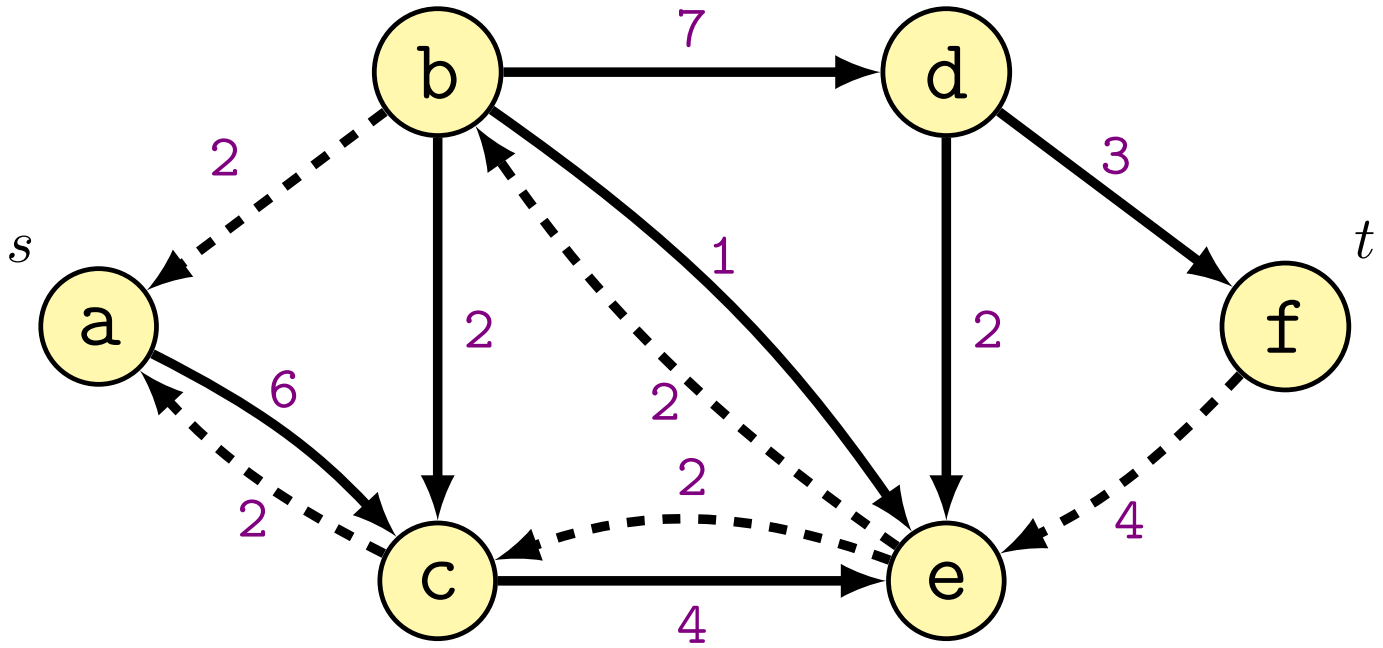
$f(\mathtt{a},\mathtt{b}) = 2$
$f(\mathtt{b},\mathtt{e}) = 2$
$f(\mathtt{e},\mathtt{f}) = 2$

Esecuzione



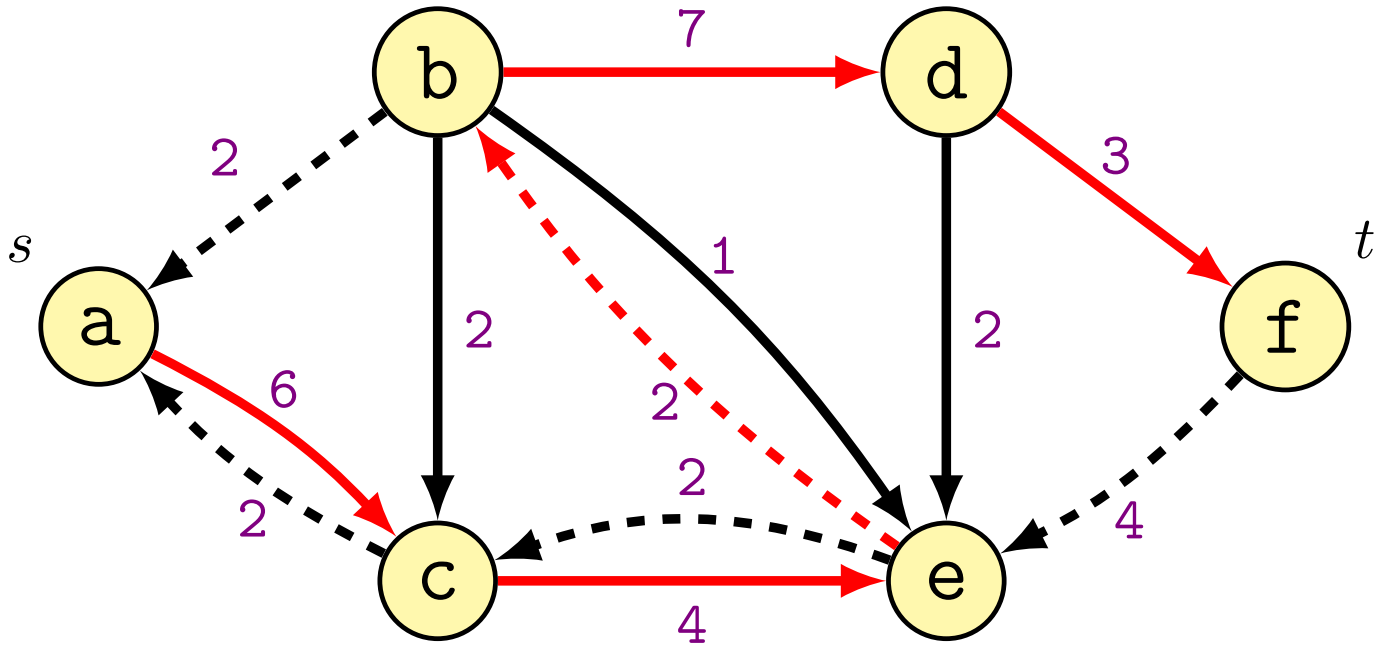
$f(\mathtt{a},\mathtt{b}) = 2$
$f(\mathtt{b},\mathtt{e}) = 2$
$f(\mathtt{e},\mathtt{f}) = 2$

Esecuzione



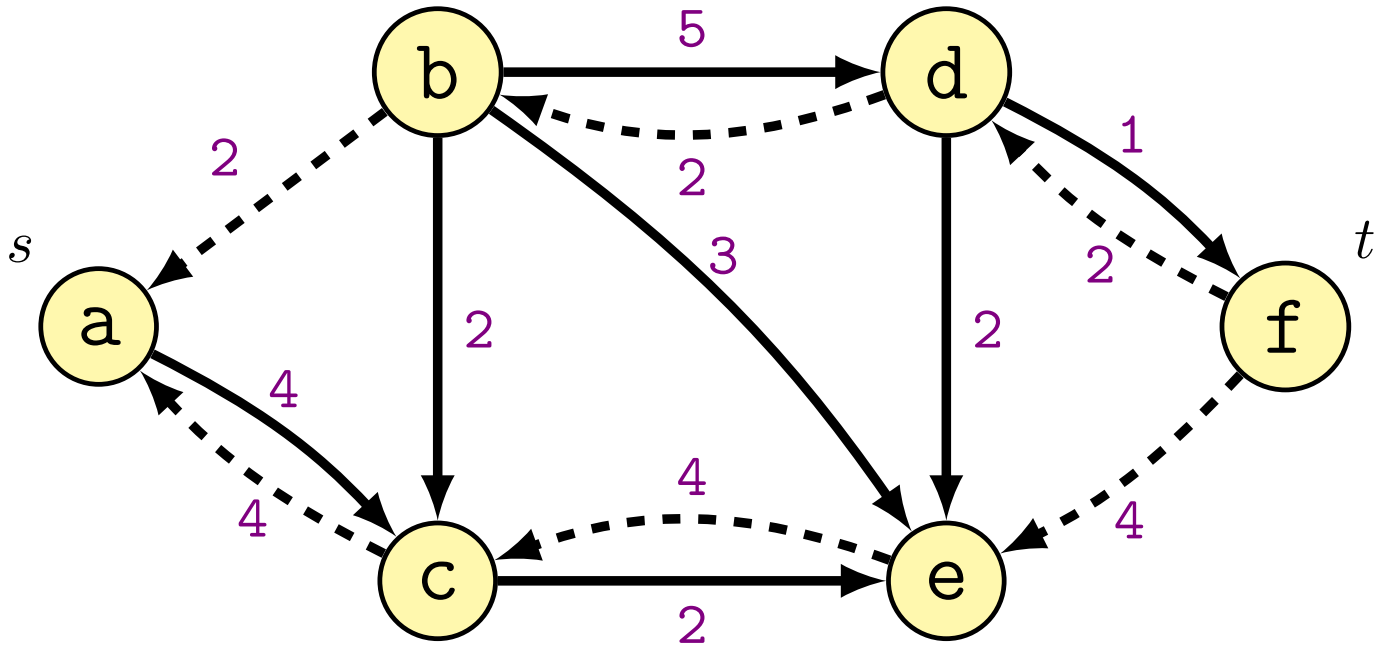
$f(\mathtt{a},\mathtt{b}) = 2$
$f(\mathtt{b},\mathtt{e}) = 2$
$f(\mathtt{e},\mathtt{f}) = 2 \not{2}\ 4$
$f(\mathtt{a},\mathtt{c}) = 2$
$f(\mathtt{c},\mathtt{e}) = 2$

Esecuzione



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

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 (1956) parlavano genericamente di una visita per trovare un cammino.

Edmonds e Karp (1972) 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
}
```

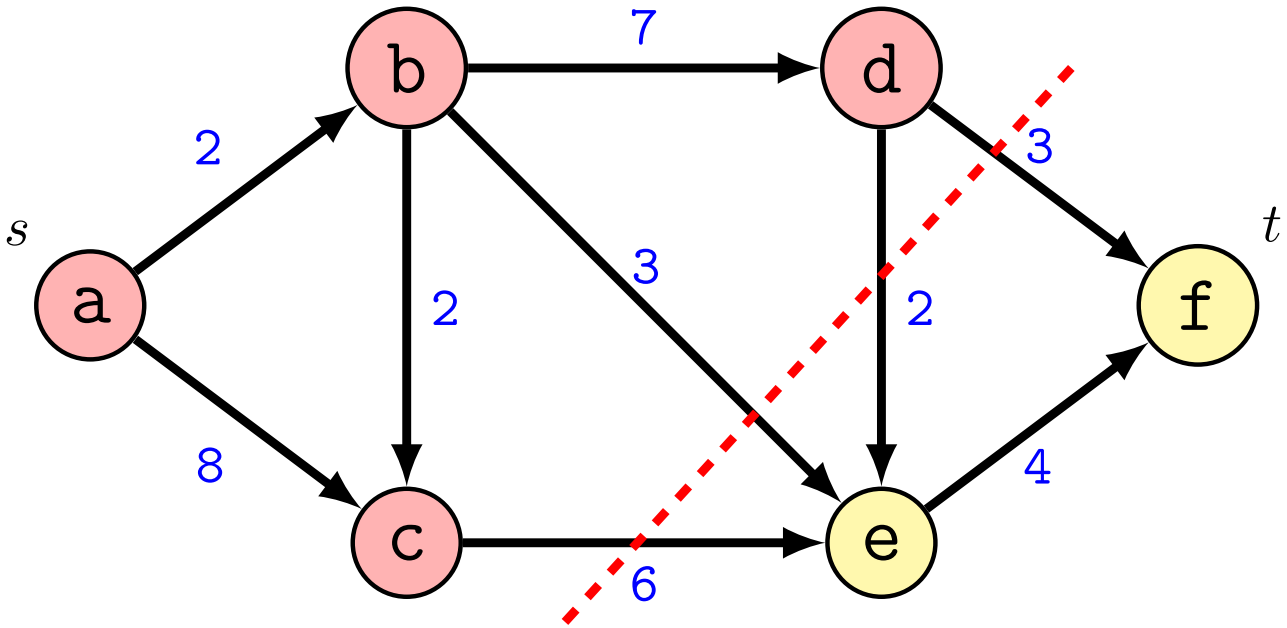
Dimostrazione correttezza – Definizioni

Taglio

Un taglio (S, T) della rete di flusso $G = (V, E, s, t, c)$ è una partizione di V in due sottoinsiemi disgiunti S, T tali che:

$$S = V - T$$

$$s \in S \wedge t \in T$$



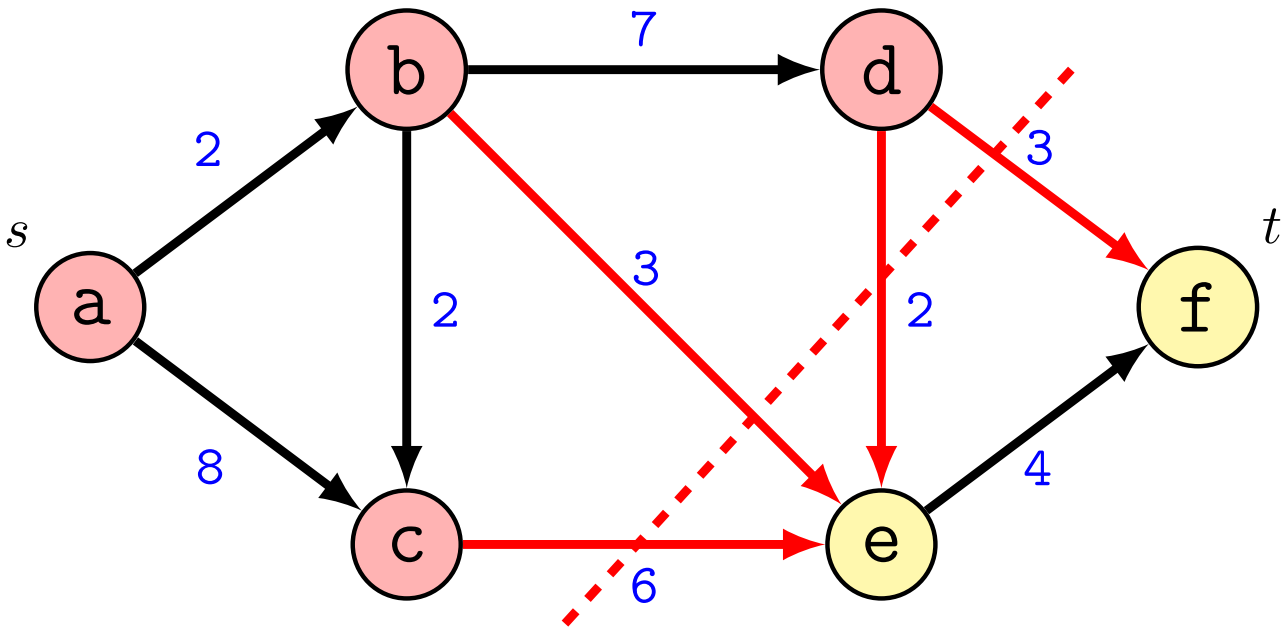
$S = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$
$T = \{\mathtt{e}, \mathtt{f}\}$

Dimostrazione correttezza – Definizioni

Capacità di un taglio

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

$$C(S,T) = \sum_{x \in S, y \in T} c(x,y)$$



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

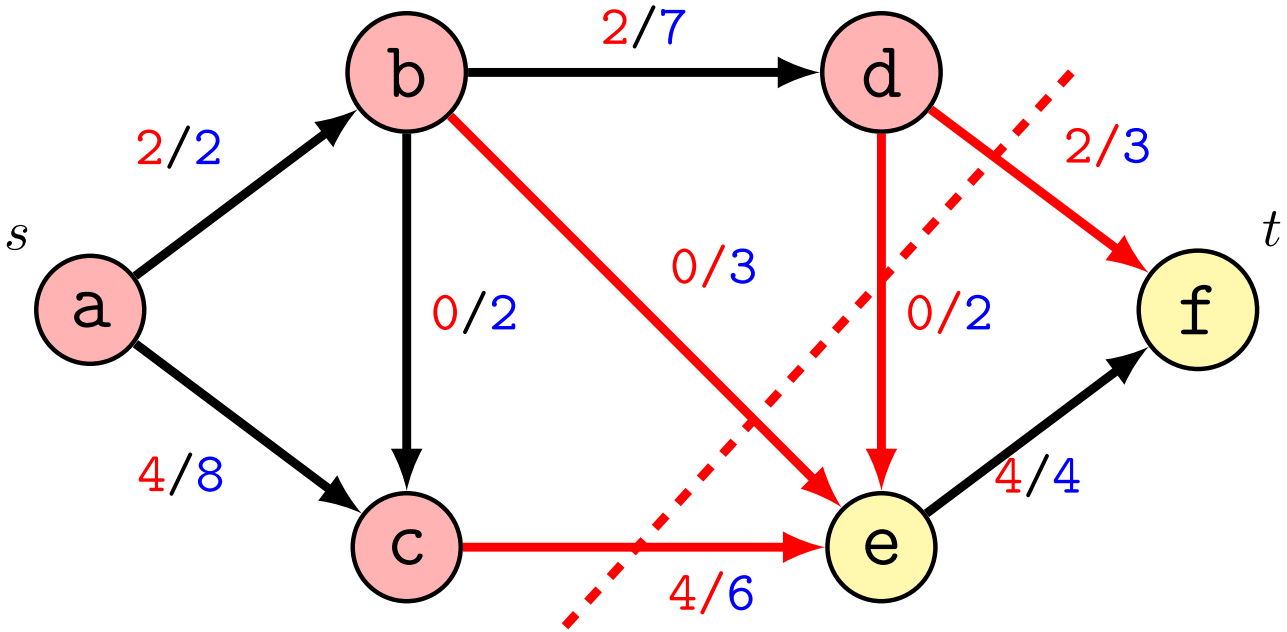
$C(S,T) = 14$

Dimostrazione correttezza – Definizioni

Flusso di un taglio

Se f è un flusso in G, il flusso netto $F_f(S,T)$ attraverso (S,T) è:

$$F_f(S,T) = \sum_{x \in S, y \in T} f(x,y)$$



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

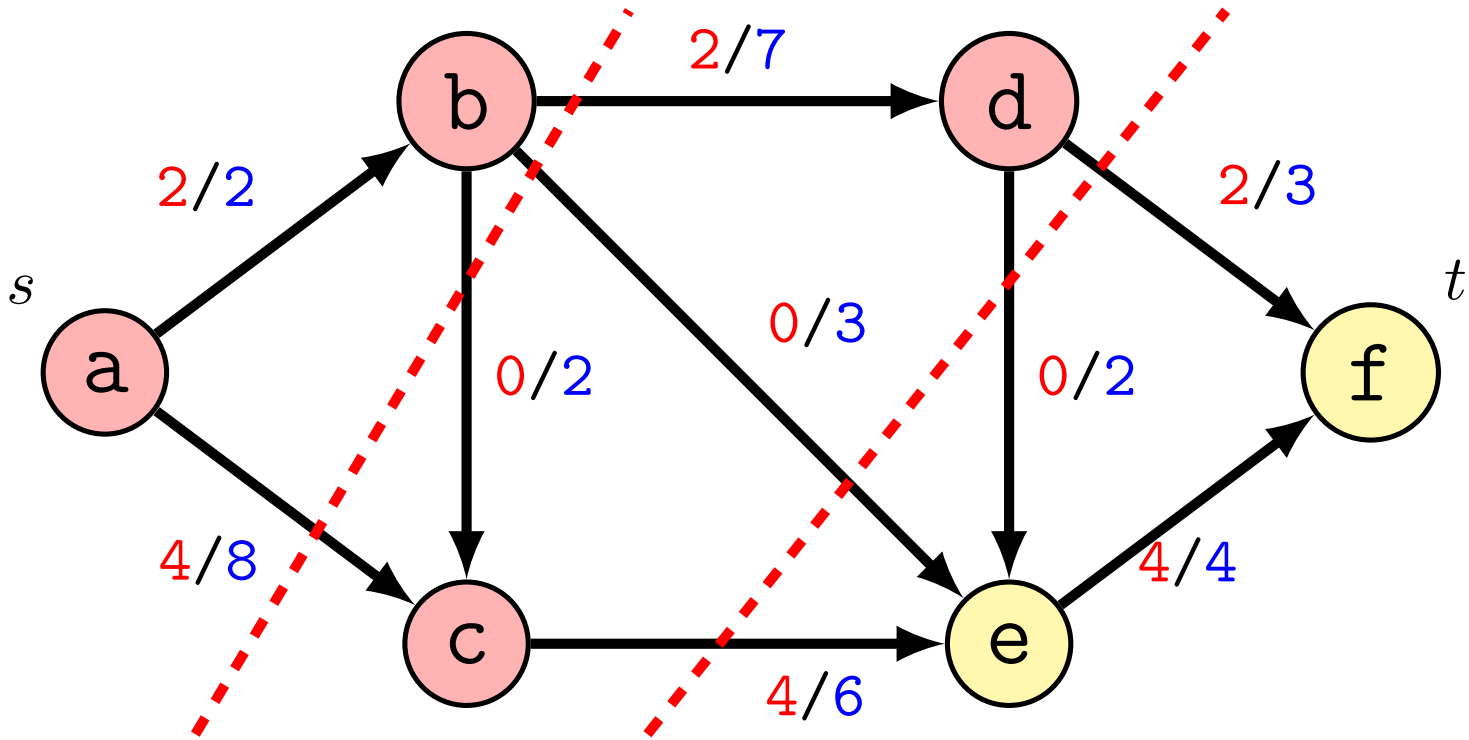
$C(S,T) = 14$

$F_f(S,T) = 6$

Dimostrazione correttezza – Lemma valore del flusso

Lemma – Valore del flusso di un taglio

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



Dimostrazione correttezza – Lemma valore del flusso

Lemma – Valore del flusso di un taglio

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

$$
\begin{aligned}
F_f(S,T) &= \sum_{x \in S, y \in T} f(x,y) \\
&= \sum_{x \in S, y \in V} f(x,y) - \sum_{x \in S, y \in S} f(x,y) && T = V - S \\
&= \sum_{x \in S, y \in V} f(x,y) - 0 && \text{Antisimmetria} \\
&= [\ldots]
\end{aligned}
$$

Dimostrazione correttezza – Lemma valore del flusso

Lemma – Valore del flusso di un taglio

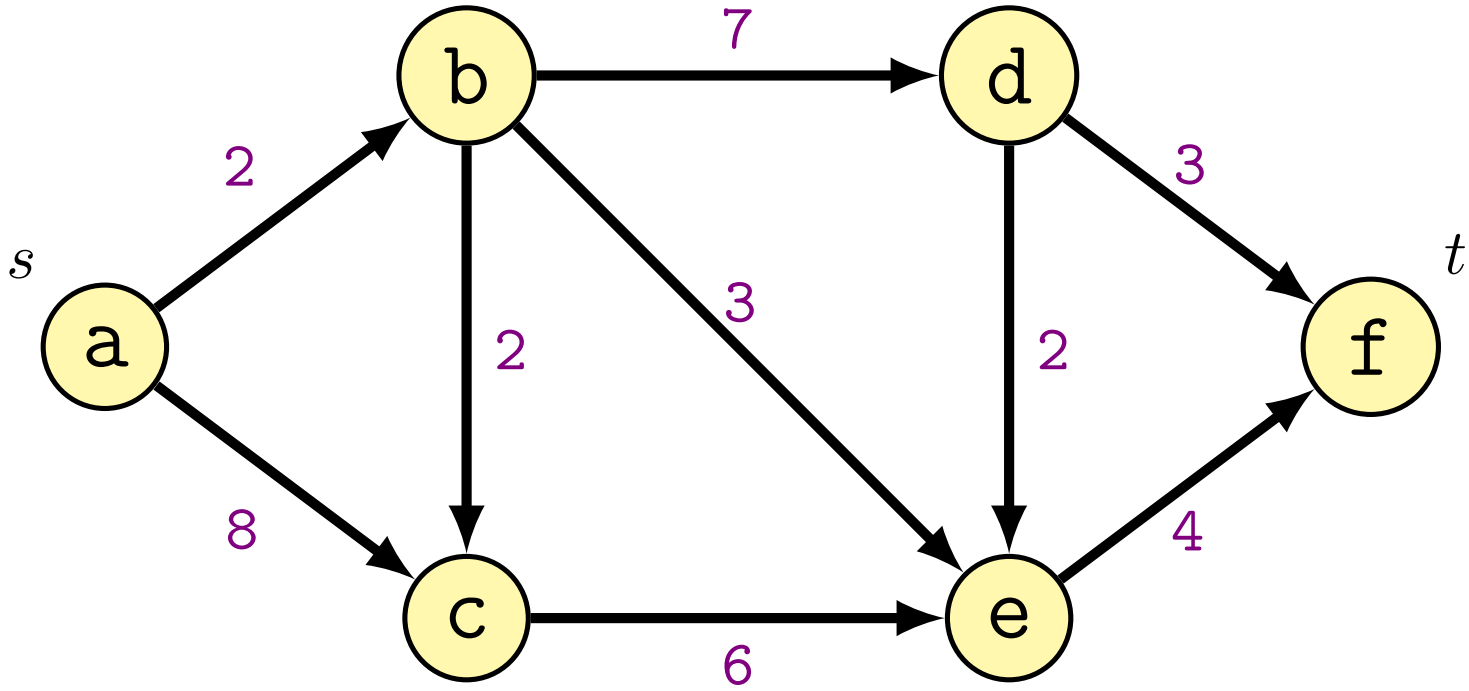
Dato un flusso f e un taglio (S, T), la quantità di flusso $F_f(S, T)$ che attraversa il taglio è uguale a $|f|$.

$$
\begin{aligned}
F_f(S,T) &= \sum_{x \in S, y \in V} f(x,y) \\
&= \sum_{x \in S - \{s\}, y \in V} f(x,y) + \sum_{y \in V} f(s,y) && s \in S \\
&= \sum_{x \in S - \{s\}} \sum_{y \in V} f(x,y) + \sum_{y \in V} f(s,y) && \text{Sommatoria} \\
&= \sum_{x \in S - \{s\}} 0 + \sum_{y \in V} f(s,y) && \text{Conservazione flusso} \\
&= \sum_{y \in V} f(s,y) = |f| && \text{Definizione valore flusso}
\end{aligned}
$$

Dimostrazione correttezza – Lemma capacità del taglio

Lemma – Capacità taglio

Il flusso massimo è limitato superiormente dalla capacità del taglio minimo, ovvero il taglio la cui capacità è minore fra tutti i tagli.

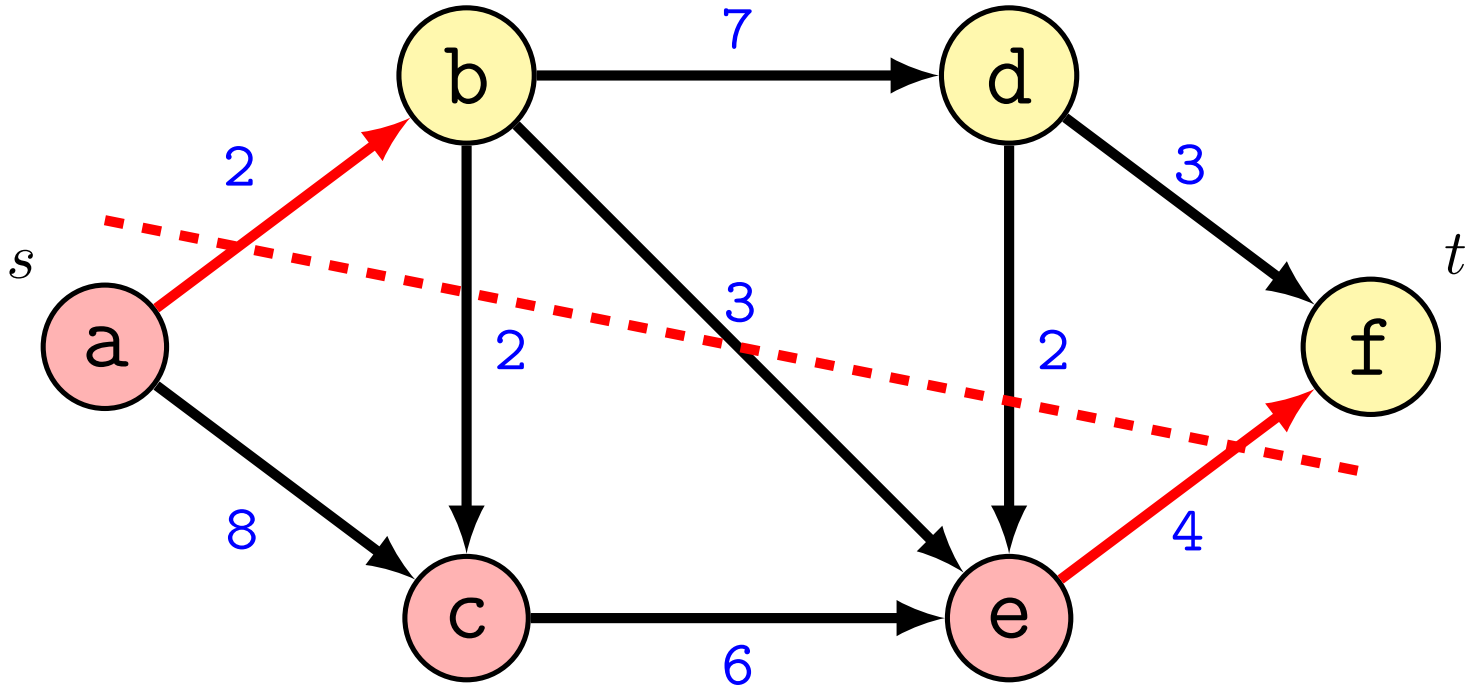


Qual è il taglio minimo?

Dimostrazione correttezza – Lemma capacità del taglio

Lemma – Capacità taglio

Il flusso massimo è limitato superiormente dalla capacità del taglio minimo, ovvero il taglio la cui capacità è minore fra tutti i tagli.



Dimostrazione correttezza – Lemma capacità del taglio

Lemma – Capacità taglio

Il flusso massimo è limitato superiormente dalla capacità del taglio minimo, ovvero il taglio la cui capacità è minore fra tutti i tagli.

- Nessun flusso attraverso un taglio supera la capacità del taglio

$$\forall f : F_f(S,T) \leq C(S,T) \qquad \forall (S,T) \text{ taglio di } V$$

Dimostrazione:

$$\forall f : F_f(S,T) = \sum_{x \in S, y \in T} f(x,y) \leq \sum_{x \in S, y \in T} c(x,y) = C(S,T)$$

(Vincolo sulla capacità)

Dimostrazione correttezza – Lemma capacità del taglio

Lemma – Capacità taglio

Il flusso massimo è limitato superiormente dalla capacità del taglio minimo, ovvero il taglio la cui capacità è minore fra tutti i tagli.

- Nessun flusso attraverso un taglio supera la capacità del taglio

$$\forall f : F_f(S,T) \leq C(S,T) \qquad \forall (S,T) \text{ taglio di } V$$

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

$$\forall f : |f| = F_f(S,T) \qquad \forall (S,T) \text{ taglio di } V$$

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

$$\forall f : |f| \leq C(S,T) \qquad \forall (S,T) \text{ taglio di } V$$

Teorema del taglio minimo / flusso massimo

Teorema

Le seguenti tre affermazioni sono equivalenti:

1. f è un flusso massimo
2. non esiste alcun cammino aumentante nella rete residua G_f
3. esiste un taglio minimo (S, T) tale che $C(S, T) = |f|$

Dimostreremo circolarmente:

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

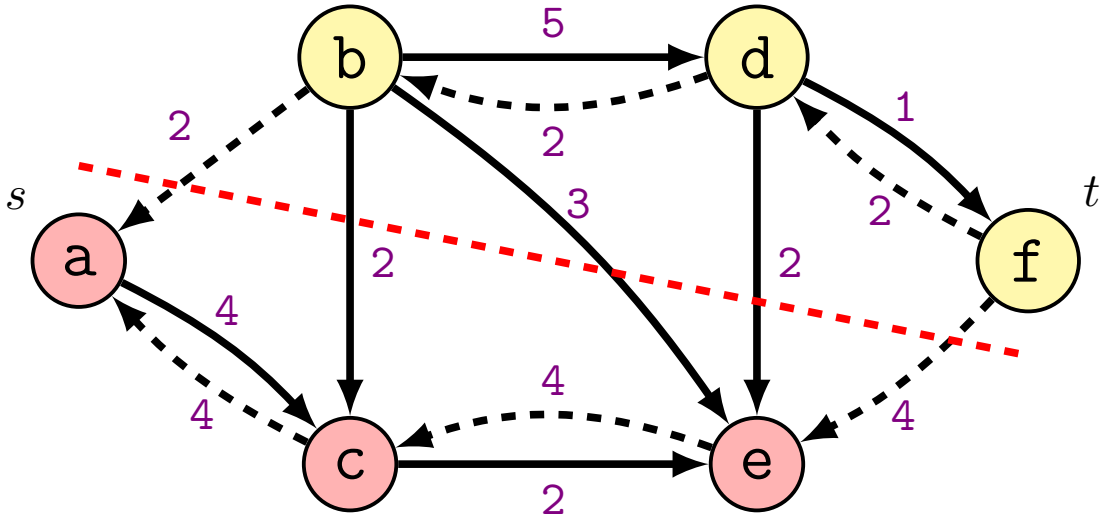
Dimostrazione correttezza – (1) $\Rightarrow$ (2)

f è un flusso massimo $\Rightarrow$
non esiste nessun cammino aumentante nella rete residua G_f

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

Dimostrazione correttezza – (2) ⇒ (3)

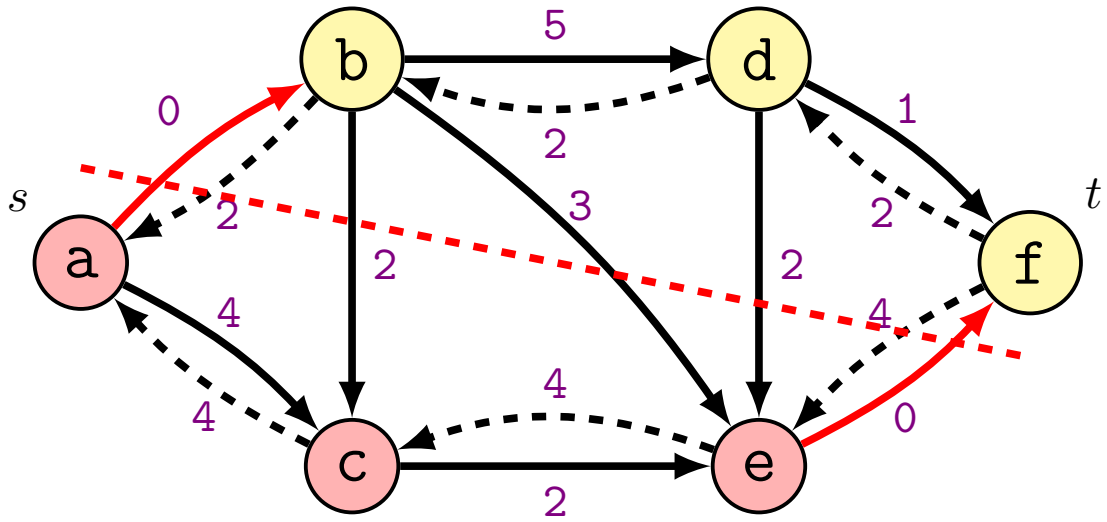
Non esiste nessun cammino aumentante nella rete residua G_f ⇒ esiste un taglio minimo (S, T) tale che $C(S, T) = |f|$



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

Dimostrazione correttezza – (2) ⇒ (3)

Non esiste nessun cammino aumentante nella rete residua G_f ⇒ 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 (x, y) con $x \in S$ e $y \in T$ sono saturati; ovvero, $f(x, y) = c(x, y)$.

Dimostrazione correttezza – (2) ⇒ (3)

Non esiste nessun cammino aumentante nella rete residua G_f ⇒ esiste un taglio minimo (S, T) tale che $C(S, T) = |f|$

- Per il Lemma – Valore del flusso di un taglio,

$$|f| = \sum_{x \in S, y \in T} f(x, y)$$

- Ne segue che:

$$|f| = \sum_{x \in S, y \in T} f(x, y) = \sum_{x \in S, y \in T} c(x, y) = C(S, T)$$

- (S, T) è minimo perchè $|f| = C(S, T)$ e per ogni taglio (S', T'), abbiamo che $|f| \leq C(S', T')$ (per il lemma su Capacità taglio)

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.

Complessità

Complessità, limite superiore – Ford-Fulkerson

Se le capacità sono intere, l'algoritmo di Ford-Fulkerson ha complessità $O((V+E)|f^*|)$ (liste) o $O(V^2|f^*|)$ (matrice).

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

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 $O(VE^2)$.

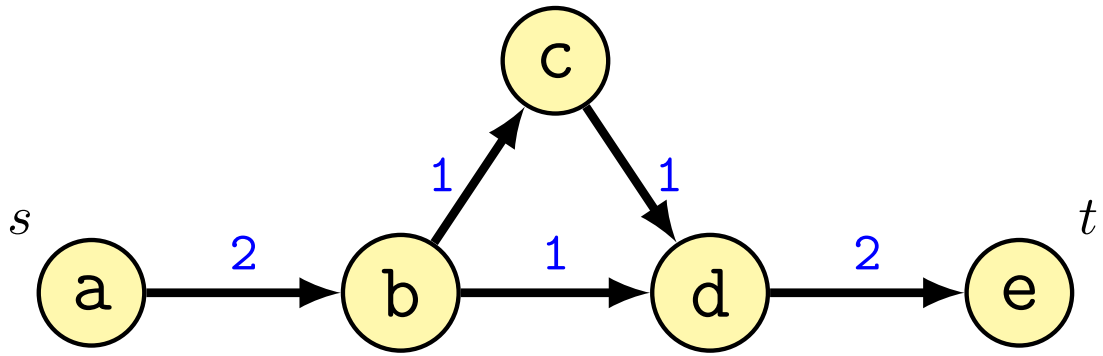
- Vengono eseguiti $O(VE)$ aumenti di flusso, ognuno dei quali richiede una visita in ampiezza $O(V+E)$.
- La complessità è quindi $O(VE(V+E))$.
- Poiché $E = \Omega(V)$ (ogni nodo diverso da sorgente, pozzo ha almeno un arco entrante e un arco uscente), possiamo semplificare scrivendo che la complessità è $O(VE^2)$

Dimostrazione complessità

Lemma - Monotonia

Sia $\delta_f(s,x)$ la distanza minima da s a x 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 $\delta_{f'}(s,x) \geq \delta_f(s,x)$.

- Quando viene aumentato il flusso, alcuni archi si "spengono" (capacità residua 0)
- Questi archi erano utilizzati nei cammini minimi (BFS)
- I cammini minimi non posso diventare più corti

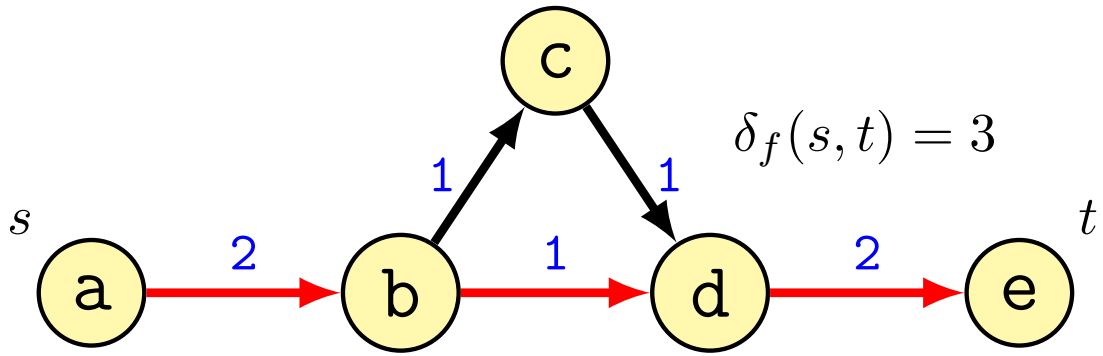


Dimostrazione complessità

Lemma - Monotonia

Sia $\delta_f(s,x)$ la distanza minima da s a x 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 $\delta_{f'}(s,x) \geq \delta_f(s,x)$.

- Quando viene aumentato il flusso, alcuni archi si "spengono" (capacità residua 0)
- Questi archi erano utilizzati nei cammini minimi (BFS)
- I cammini minimi non posso diventare più corti

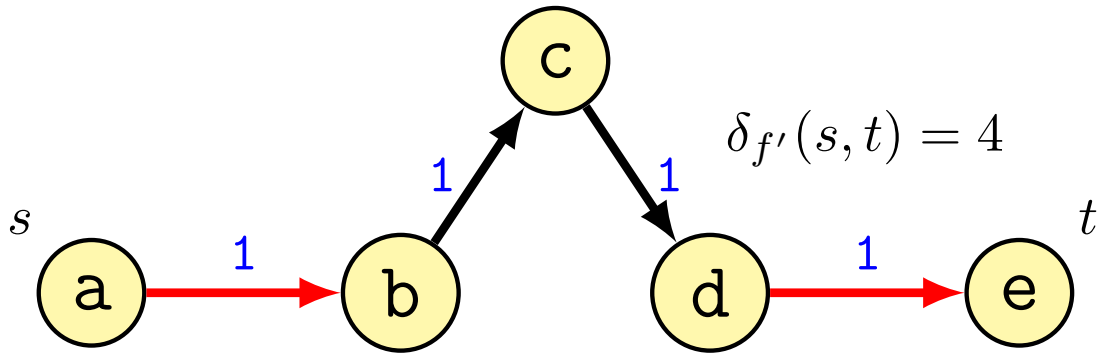


Dimostrazione complessità

Lemma - Monotonia

Sia $\delta_f(s,x)$ la distanza minima da s a x 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 $\delta_{f'}(s,x) \geq \delta_f(s,x)$.

- Quando viene aumentato il flusso, alcuni archi si "spengono" (capacità residua 0)
- Questi archi erano utilizzati nei cammini minimi (BFS)
- I cammini minimi non posso diventare più corti



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.
- (x, y) è un arco critico (collo di bottiglia) in C se

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

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

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 x, tenuto conto che poi si deve ancora seguire l'arco (x, y), 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
Dinitz, blocking flow	$O(V^2E)$	In alcune reti particolari, $O(\min(V^{2/3}, E^{1/2})E)$
MPM	$O(V^3)$	Solo su DAG
Dinitz	$O(VE \log V)$	Struttura dati Dynamic trees
Goldberg e Rao	$O(\min(V^{2/3}, E^{1/2})E \cdot \cdot \log(V^2/E + 2) \log C)$	$C = \max_{(x,y)\in E} c(x, y)$
Orlin + King,Rao,Tarjan	$O(VE)$	2013
Chen, Kyng, Liu at al.	$O(V + E^{1+o(1)})$	"Quasi-linear", 112 pagine, 2022!

Complessità – Letture

Andrew V. Goldberg, Robert E. Tarjan. Efficient Maximum Flow Algorithms. Communications of the ACM, 57(8):82–89. 2014.

Li Chen, Rasmus Kyng, Yang P. Liu, Richard Peng, Maximilian Probst Gutenberg, Sushant Sachdeva. Maximum Flow and Minimum-Cost Flow in Almost-Linear Time. 63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022.

Applicazioni

- Maximum Flows with Edge Demands (Circulation Problem)
  Flusso massimo con limiti inferiori e superiori per la capacità
- Node Supplies and Demands
  Il flusso può essere generato o consumato in ogni nodo
- Min-Cost Max Flows
  Al flusso è associato un costo, fra due flussi massimali scegliamo quello con costo minimo

Jeff Erickson. Lecture 18: Extensions of Maximum Flow (8 pp.)

David P. Williamson. Network Flow Algorithms.
Cambridge University Press, 2019 (273 pp.)

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

Jeff Erickson. Lecture 11: Applications of Flows and Cuts.

Abbinamento (matching) massimo nei grafi bipartiti

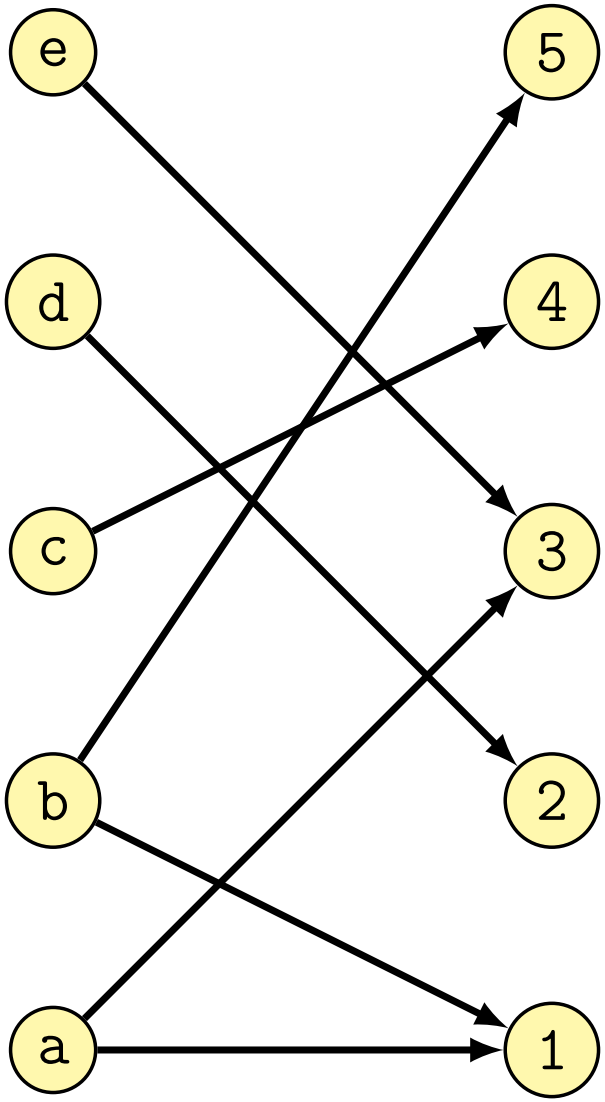
Problema - Job Assignment - Input

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

Problema - Job Assignment - Output

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

Esempio



Esempio