

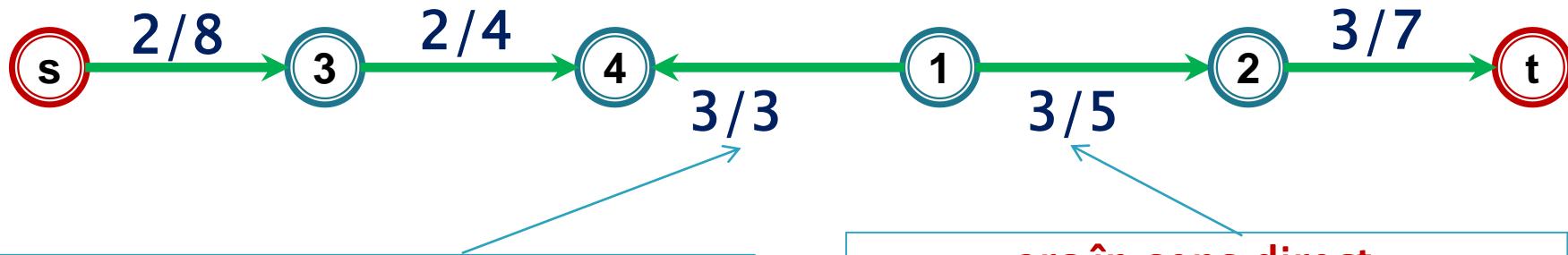
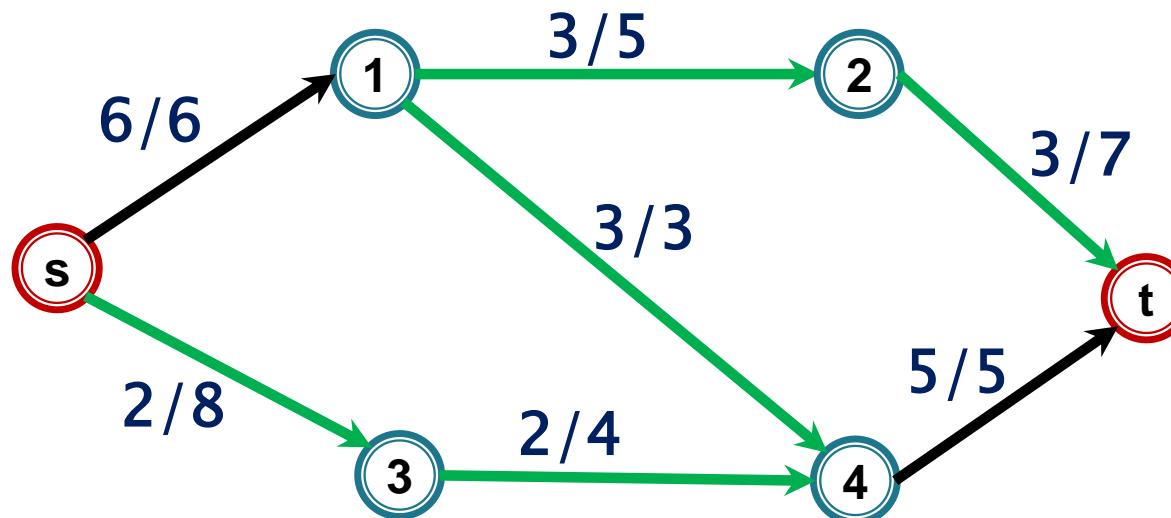
# **Algoritmul FORD–FULKERSON**

## **de determinare a unui flux maxim**

**+ a unei tăieturi minime**

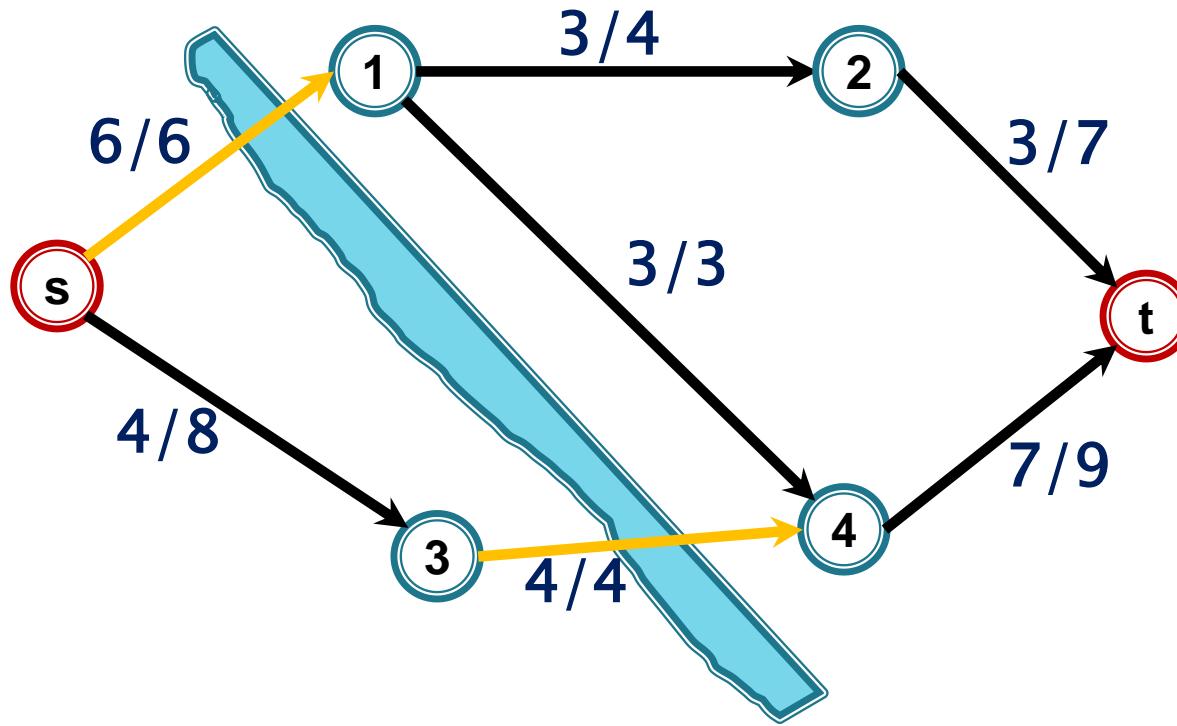
# Algoritmul Ford-Fulkerson

Amintim din exemplele anterioare:



arc în sens invers,  
putem trimite înapoi 3 unități de flux

arc în sens direct,  
mai putem trimite  $5-3=2$  unități



Fluxul este maxim – în mulțimea de arce evidențiată toate arcele au flux=capacitate și nu putem construi drumuri de la s la t care nu conțin arce din această mulțime (**s-t tăietură**)

# Algoritmul Ford-Fulkerson

Definim noțiunile necesare descrierii și studiului algoritmului:

- **s–t lanț f–nesaturat**
  - arc direct
  - arc invers
  - capacitate reziduală arc, lanț
- Operația de revizuire a fluxului de-a lungul unui s–t lanț *f–nesaturat*

Fie  $N = (G, \{s\}, \{t\}, I, c)$  o rețea

- ▶ Un **lanț** este o succesiune de vârfuri **distincte** și **arce** din  $G$

$$P = [v_0, e_1, v_1, \dots, v_{k-1}, e_k, v_k]$$

unde arcul  $e_i$  este fie  $v_{i-1}v_i$ , fie  $v_iv_{i-1}$

( $P$  este lanț elementar în graful neorientat asociat lui  $G$ )

Fie  $N = (G, \{s\}, \{t\}, I, c)$  o rețea

- ▶ Un **lanț** este o succesiune de vârfuri **distincte** și **arce** din  $G$

$$P = [v_0, e_1, v_1, \dots, v_{k-1}, e_k, v_k]$$

unde arcul  $e_i$  este fie  $v_{i-1}v_i$ , fie  $v_iv_{i-1}$

( $P$  este lanț elementar în graful neorientat asociat lui  $G$ )

Dacă

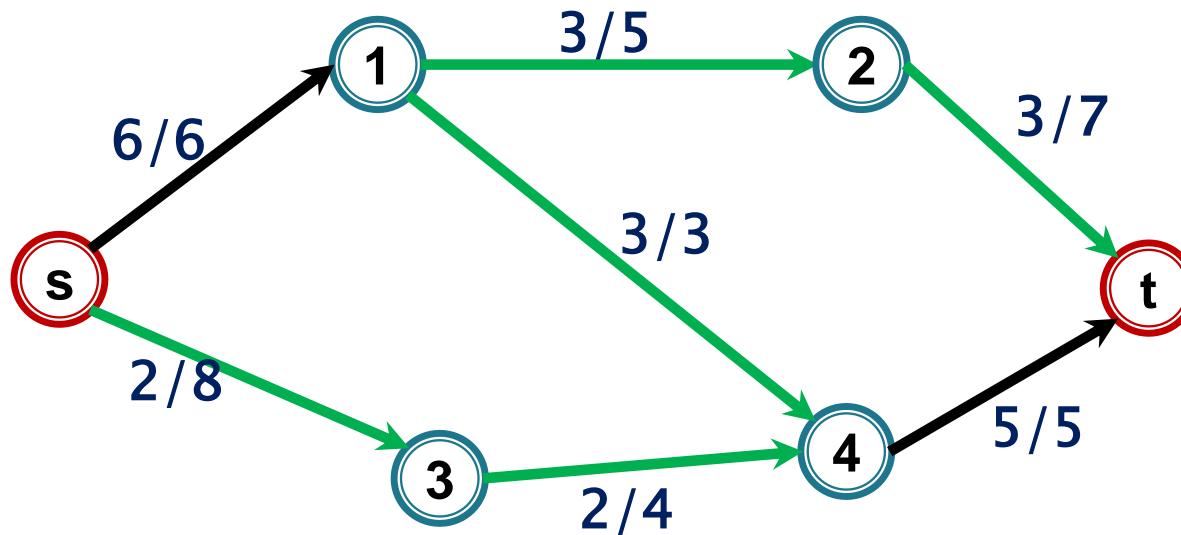
- $e_i = v_{i-1}v_i \in E(G)$ ,  $e_i$  s.n **arc direct (înainte)** în  $P$
- $e_i = v_iv_{i-1} \in E(G)$ ,  $e_i$  s.n **arc invers (înapoi)** în  $P$



- ▶ Dacă nu există confuzii vom omite arcele în scrierea lanțului P

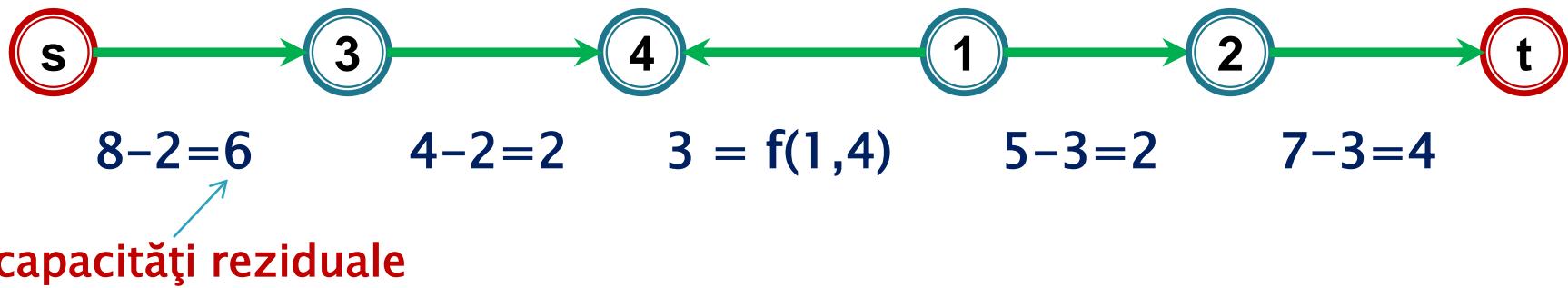
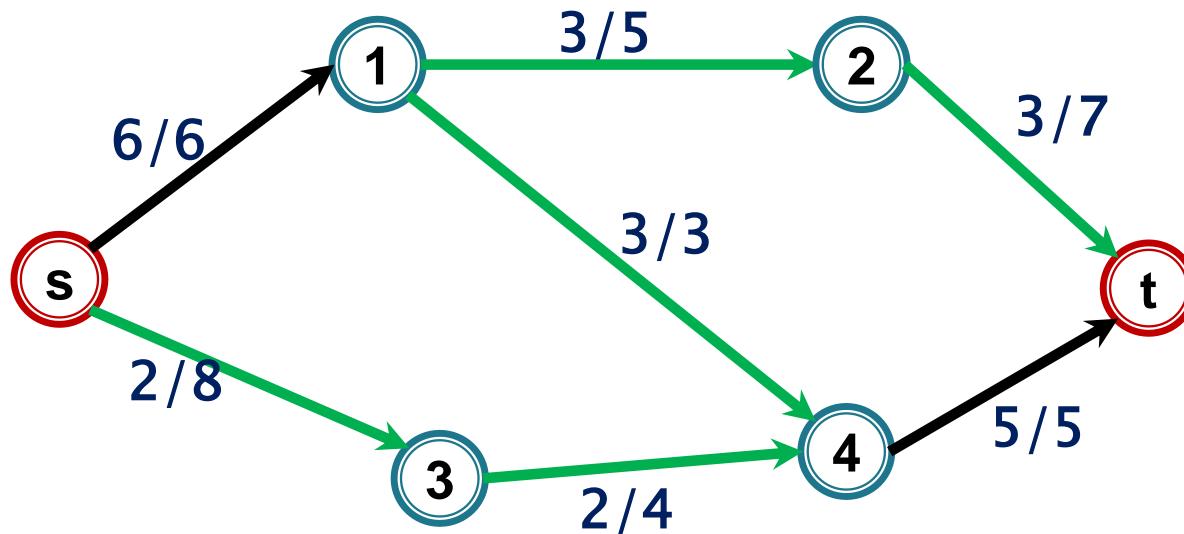
$$P = [v_0, v_1, \dots, v_{k-1}, v_k]$$

- ▶ Fie  $N$  rețea,  $f$  flux în  $N$ ,  $P$  un lanț
- ▶ Asociem fiecărui arc  $e$  din  $P$  o pondere, numită **capacitate reziduală** în  $P$



capacități reziduale?

- ▶ Fie  $N$  rețea,  $f$  flux în  $N$ ,  $P$  un lanț
- ▶ Asociem fiecărui arc  $e$  din  $P$  o pondere, numită **capacitate reziduală** în  $P$



- ▶ Fie  $N$  rețea,  $f$  flux în  $N$ ,  $P$  un lanț
- ▶ Asociem fiecărui arc  $e$  din  $P$  o pondere, numită **capacitate reziduală** în  $P$ :

$$i_P(e) = \begin{cases} c(e) - f(e), & \text{dacă } e \text{ este arc direct în } P \\ f(e), & \text{dacă } e \text{ este arc invers în } P \end{cases}$$

= cu cât mai poate fi modificat fluxul pe arcul  $e$ , de-a lungul lanțului  $P$

- ▶ Capacitatea reziduală a unui lanț P este

$$i(P) = \min\{i_P(e) \mid e \in E(P)\}$$

= cu cât mai poate fi modificat  
fluxul de-a lungul lanțului P

- ▶ Convenție: Dacă  $E(P) = \emptyset$ ,  $i(P) = \infty$

## ▶ Capacitatea reziduală a lanțului P



$$i(P) = ?$$

= cu cât putem revizui maxim fluxul de-a lungul lui P

## ▶ Capacitatea reziduală a lanțului P



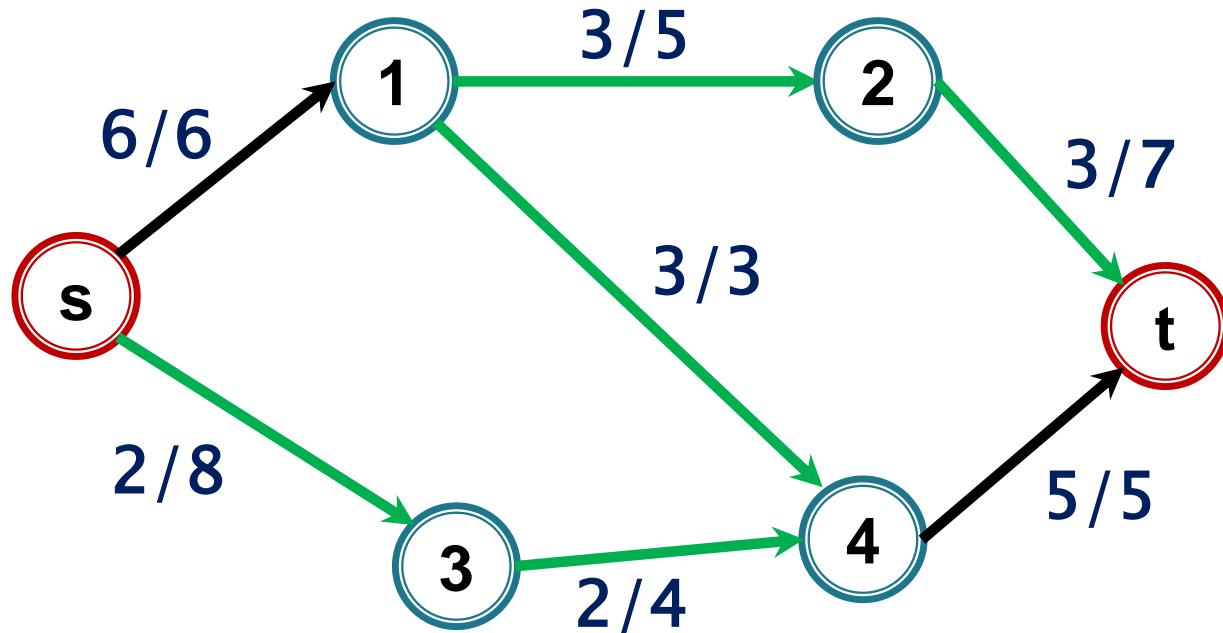
$$i(P) = \min\{6, 2, 3, 2, 4\} = 2$$

- ▶ Un s–t lanț P se numește
  - **f–nesaturat (f–drum de creștere)** **augmenting path**  
dacă  $i(P) \neq 0$
  - **f–saturat** dacă  $i(P) = 0$

# Fluxuri în rețele de transport

- ▶ Fie  $N$ - rețea,  $f$  flux în  $N$ ,  $P$  un  $s-t$  lanț **f-nesaturat**.
- ▶ Fluxul revizuit de-a lungul lanțului  $P$  se definește ca fiind  $\mathbf{f}_P : E \rightarrow \mathbb{N}$ ,

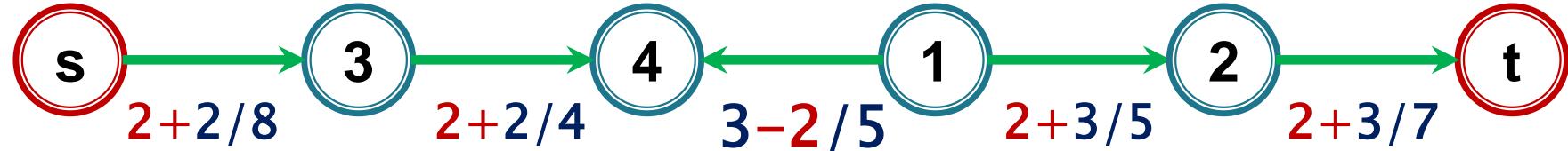
$$f_P(e) = \begin{cases} f(e) + i(P), & \text{dacă } e \text{ este arc direct în } P \\ f(e) - i(P), & \text{dacă } e \text{ este arc invers în } P \\ f(e), & \text{altfel} \end{cases}$$

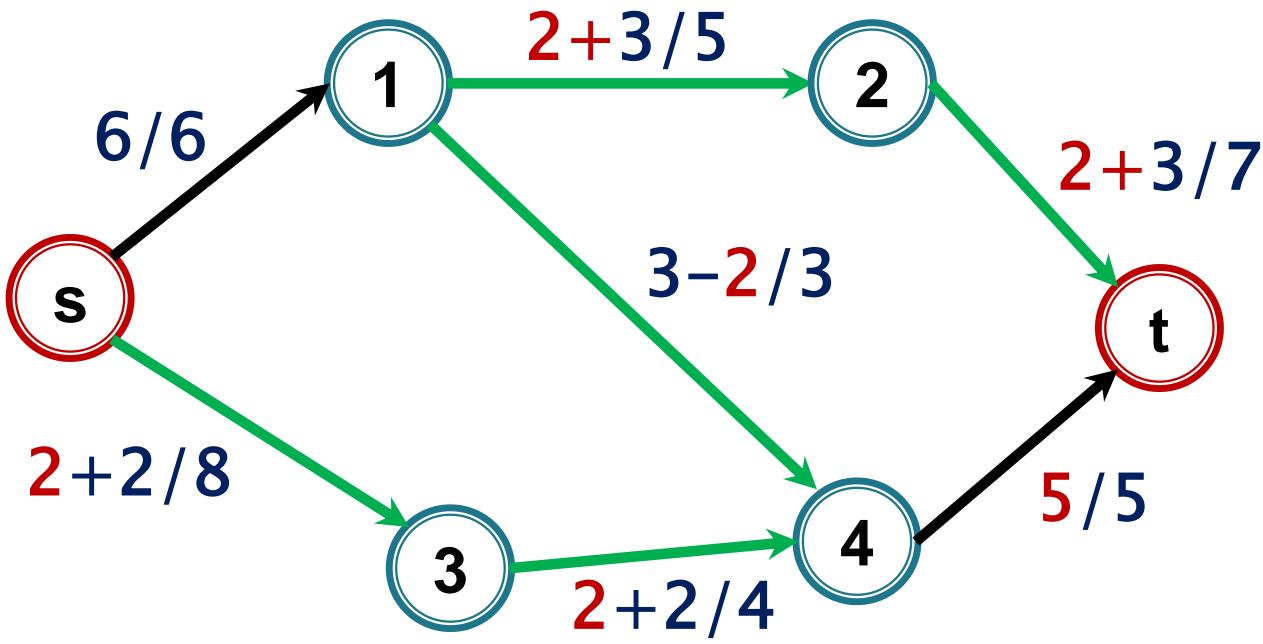


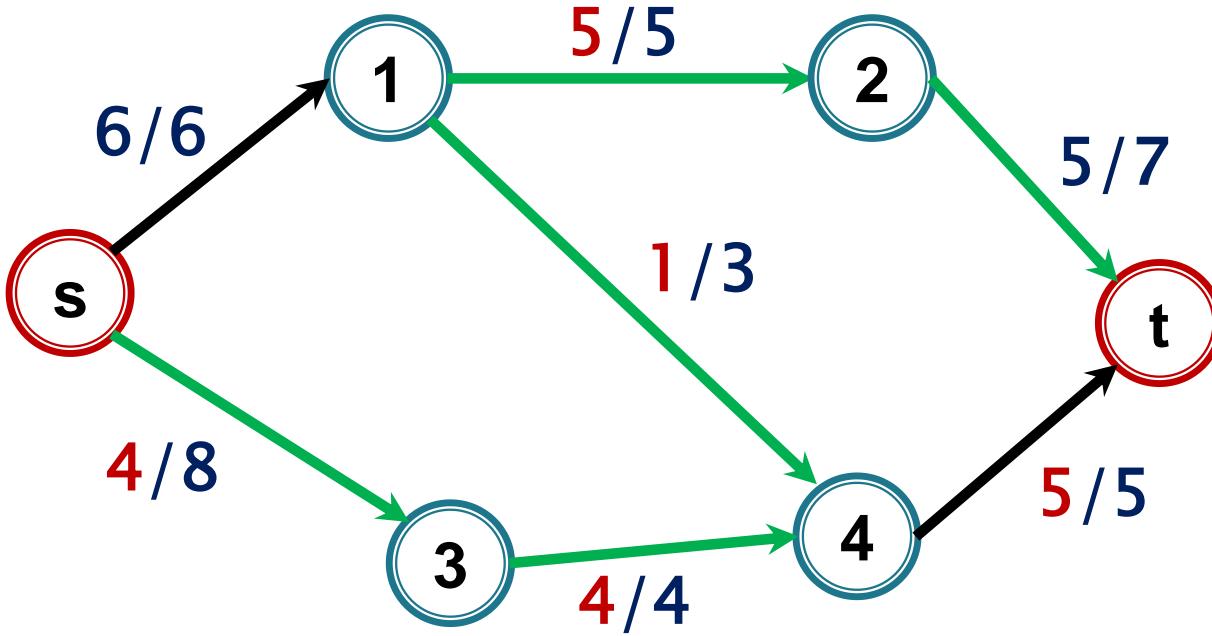
Considerăm s-t  
lanțul P evidențiat  
și revizuim fluxul



$$i(P) = 2$$







Fluxul după revizuirea de-a lungul lanțului P

## ▶ Proprietăți ale fluxului revizuit

## ▶ Proprietăți ale fluxului revizuit

Fie  $N = (G, \{s\}, \{t\}, I, c)$  o rețea și  $f$  flux în  $N$ .

Fie  $P$  un s-t lanț  $f$ -nesaturat în  $G$  și  $f_p$  fluxul revizuit de-a lungul lanțului  $P$ . Atunci

- $f_p$  este flux în  $G$

și

- $\mathbf{val}(f_P) = \mathbf{val}(f) + i(P) \geq \mathbf{val}(f) + 1$

(Temă)

# **Algoritmul FORD–FULKERSON**

## **Pseudocod**

# Algoritm generic de determinare a unui flux maxim – algoritmul FORD – FULKERSON

- Fie  $f$  un flux în  $N$  (de exemplu  $f \equiv 0$  fluxul vid:  
 $f(e) = 0, \forall e \in E$  ))

○

●

●

○

# Algoritm generic de determinare a unui flux maxim – algoritmul FORD – FULKERSON

- Fie  $f$  un flux în  $N$  (de exemplu  $f \equiv 0$  fluxul vid:  
 $f(e) = 0, \forall e \in E$  ))
- Cât timp există un s-t lanț  $f$ -nesaturat  $P$  în  $G$ 
  - determină un astfel de lanț  $P$
  - revizuieste fluxul  $f$  de-a lungul lanțului  $P$
- returnează  $f$

# Algoritm generic de determinare a unui flux maxim – algoritmul FORD – FULKERSON

- Pentru a determina și o s–t tăietură minimă, la finalul algoritmului considerăm

# Algoritm generic de determinare a unui flux maxim – algoritmul FORD – FULKERSON

- Pentru a determina și o s–t tăietură minimă, la finalul algoritmului considerăm

$X$  = multimea vârfurilor accesibile din  $s$  prin lanțuri  $f$ -nesaturate și

$$K = (X, V-X)$$

# Algoritmul FORD–FULKERSON

## Complexitate

# Algoritmul Ford–Fulkerson



- ▶ Algoritmul se termină?
- ▶ De ce este necesară ipoteza că fluxul are valori întregi?
- ▶ Care este numărul maxim de etape?
  - Cum determinăm un lanț f-nesaturat?
  - Criteriul după care construim lanțul f-nesaturat influențează numărul de etape (iterații cât timp)?

# Algoritmul Ford–Fulkerson



► Algoritmul se termină? Complexitate?

# Algoritm FORD – FULKERSON

- Complexitate
  - La fiecare pas valoarea fluxului crește cu cel puțin 1 ( $i(P) \geq 1$  natural)

# Algoritm FORD – FULKERSON

- Complexitate
  - La fiecare pas valoarea fluxului crește cu cel puțin 1 ( $i(P) \geq 1$  natural)
  - Valoare flux  $\leq$  capacitate tăietură  $=>$ 

numărul de etape  $\leq$  capacitatea minimă a unei tăieturi

# Algoritm FORD – FULKERSON

## Complexitate

- $O(mL)$ , unde  $L$  este capacitatea minimă a unei căi

# Algoritm FORD – FULKERSON

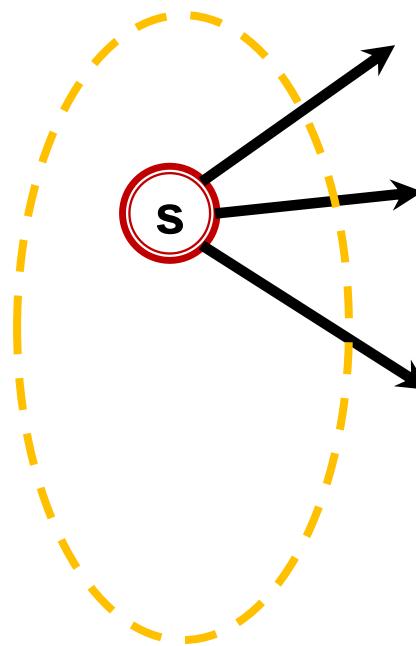
## Complexitate

- $O(mL)$ , unde  $L$  este capacitatea minimă a unei căi

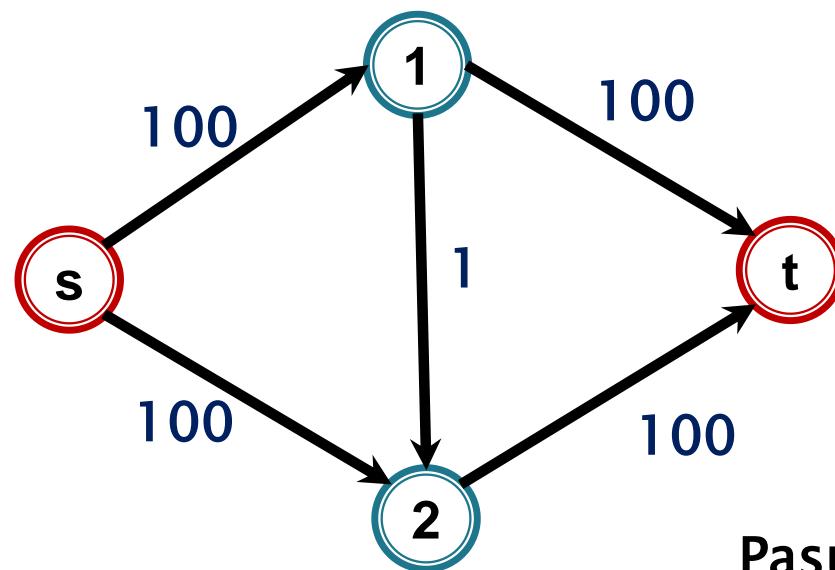
Avem  $L \leq c^+(s) \leq nC$ , unde  $C = \max\{c(e)|e \in E(G)\} \Rightarrow$

- $O(nmC)$  unde

$$C = \max\{c(e)|e \in E(G)\}$$



Criteriul după care construim lanțul f-nesaturat influențează numărul de etape



- Pasul 1:  $[s, 1, 2, t]$  avem  $i(P)=1$   
Pasul 2:  $[s, 2, 1, t]$  avem  $i(P)=1$   
Pasul 3:  $[s, 1, 2, t]$  avem  $i(P)=1$   
Pasul 4:  $[s, 2, 1, t]$  avem  $i(P)=1$

...

# Algoritmul Ford–Fulkerson



▶ Cum determinăm un lanț f-nesaturat?

# Algoritmul Ford–Fulkerson



Spre exemplu prin parcurgerea grafului pornind din vârful s și **considerând doar arce cu capacitatea reziduală pozitivă** (în raport cu lanțurile construite prin parcugere, memorate cu vectorul tata)  
= **s–t drum în graful rezidual**

# Algoritmul Ford–Fulkerson



Spre exemplu prin parcurgerea grafului pornind din vârful s și considerând doar arce cu capacitatea reziduală pozitivă (în raport cu lanțurile construite prin parcugere, memorate cu vectorul tata)

– Parcugerea BF  $\Rightarrow$

determinăm s–t lanțuri f–nesaturate de  
**lungime minimă**

$\Rightarrow$  **Algoritmul EDMONDS–KARP** = Ford–Fulkerson  
în care lanțul P ales la un pas are lungime minimă

# Algoritmul Ford–Fulkerson



Spre exemplu prin parcurgerea grafului pornind din vârful s și considerând doar arce cu capacitatea reziduală pozitivă (în raport cu lanțurile construite prin parcugere, memorate cu vectorul tata)

- Alte criterii de construcție lanț  $\Rightarrow$  alți algoritmi

# **Implementarea algoritmului FORD–FULKERSON**

**Varianta cu drumuri minime  
⇒ Algoritmul Edmonds–Karp**

# Implementare. Algoritmul Edmonds-Karp

Schema:

initializeaza\_flux\_nul()

cat timp (construieste\_s-t\_lant\_nesat\_BF()=true) executa

revizuieste\_flux\_lant()

afiseaza\_flux()

# Implementare. Algoritmul Edmonds–Karp

Schema:

initializeaza\_flux\_nul()

cat timp (construieste\_s-t\_lant\_nesat\_BF()=true) executa

revizuieste\_flux\_lant()

afiseaza\_flux()

Amintim: **a determina un s–t lanț nesaturat folosind BF în  $G \Leftrightarrow$  a determina un s–t drum folosind BF în graful rezidual  $G_f$**

# Varianta 1 de implementare

revizuirea fluxului folosind  
s-t lanțuri din G  
(fără a folosi graful rezidual)

# Implementare. Algoritmul Edmonds-Karp

**construieste\_s-t\_lant\_nesat\_BF()** – construiește un s–t lanț nesaturat prin parcurgerea BF din s

- sunt considerate în parcurgere doar arce pe care se poate modifica fluxul, adică având capacitate reziduală pozitivă
- Returnează **false** dacă un astfel de lanț nu există (și **true** dacă l-a putut construi)

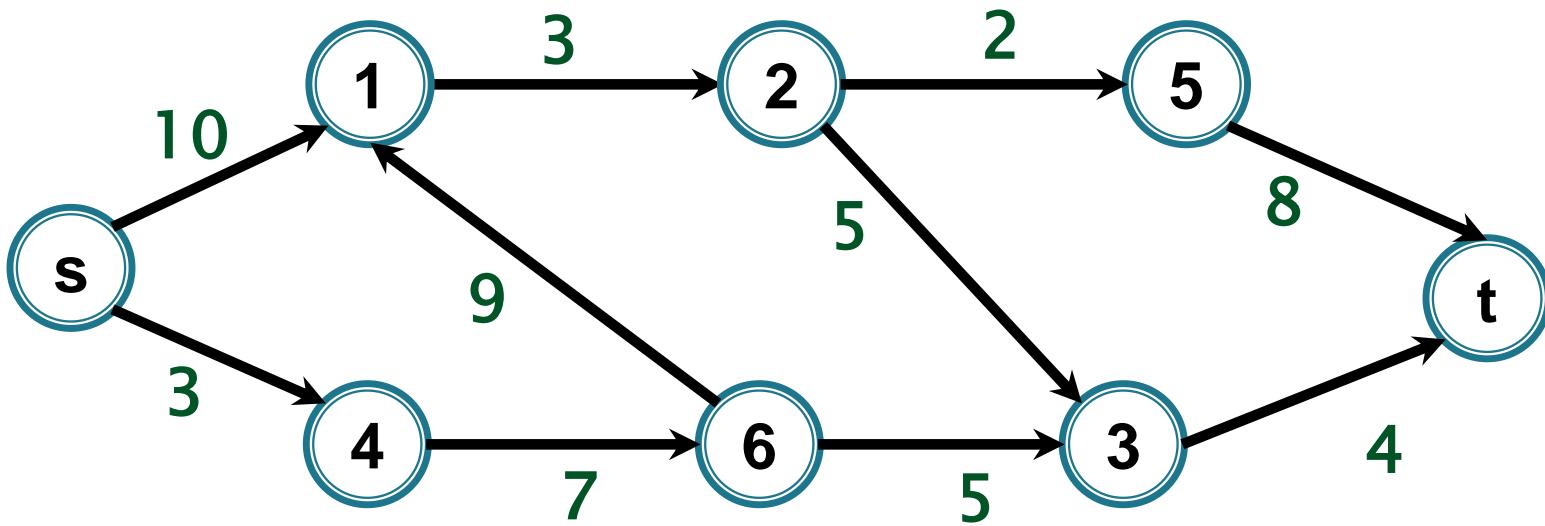
# Implementare. Algoritmul Edmonds-Karp

## revizuieste\_flux\_lant()

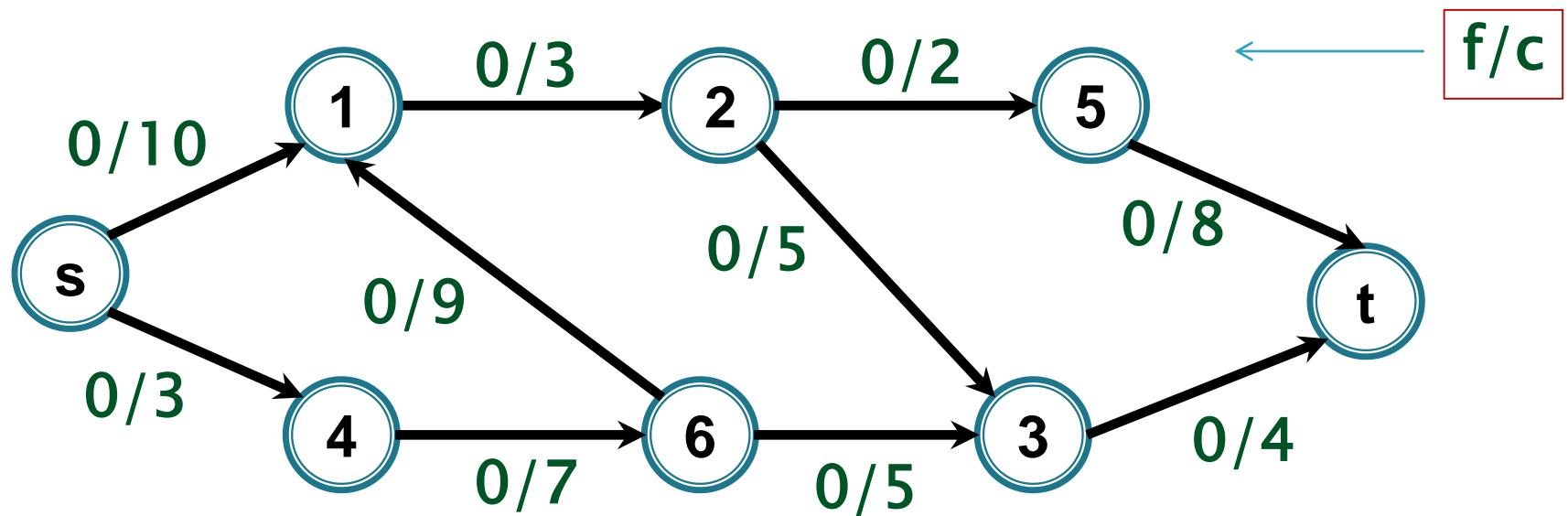
- fie  $P$  s-t lanțul găsit în `construieste_s-t_lant_nesat_BF()`
- calculăm  $i(P)$
- pentru fiecare arc  $e$  al lanțului  $P$ 
  - creștem cu  $i(P)$  fluxul pe  $e$  dacă este arc direct
  - scădem cu  $i(P)$  fluxul pe  $e$  dacă este arc invers

# Exemplu

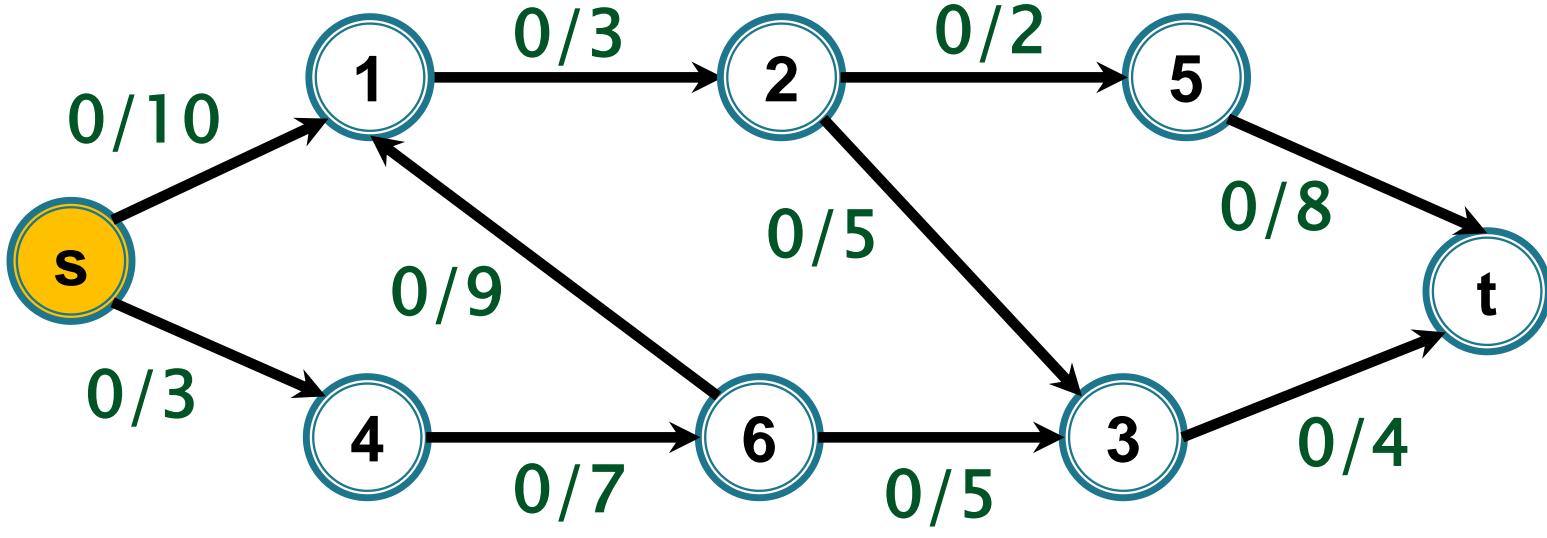
## Algoritmul EDMONDS-KARP

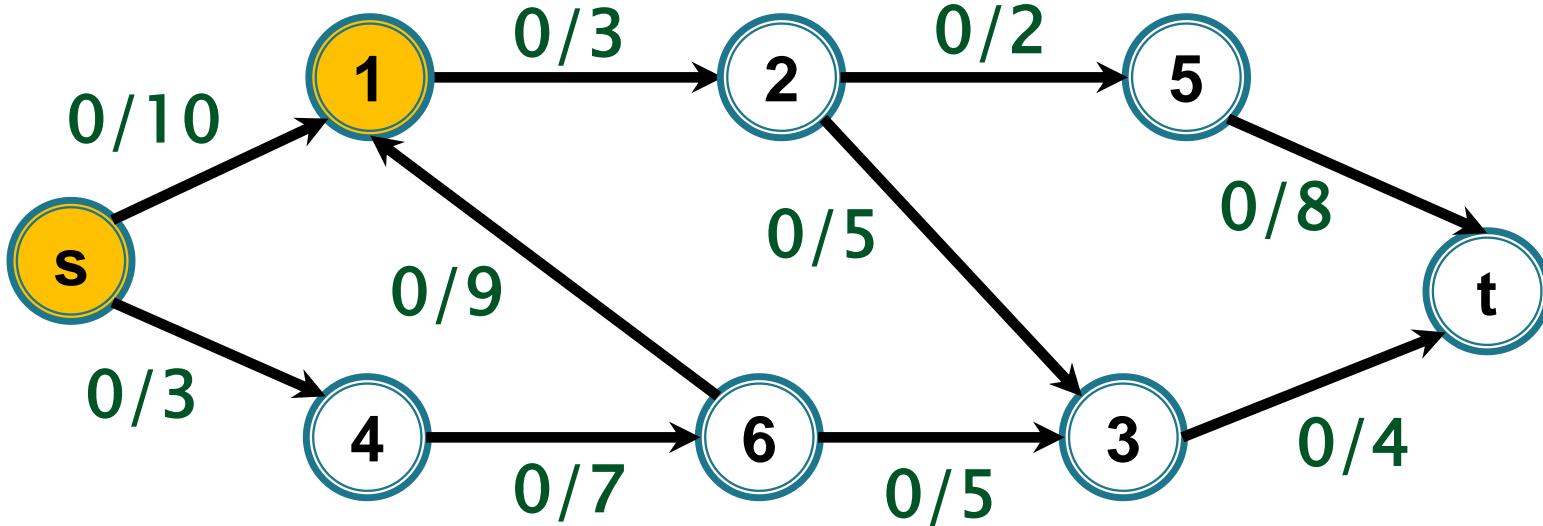


# **initializeaza\_flux\_nul**

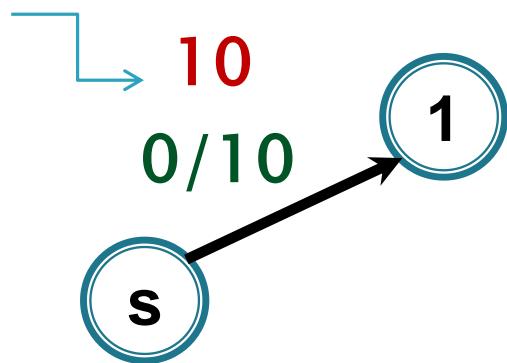


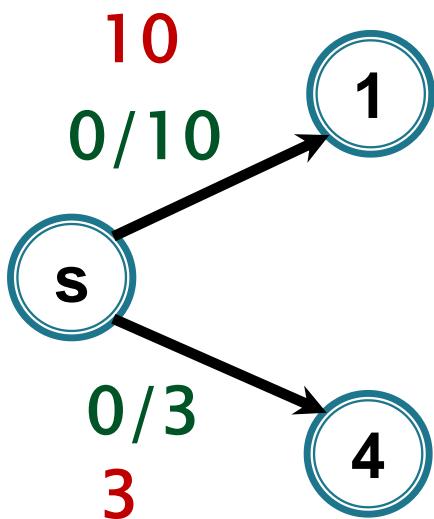
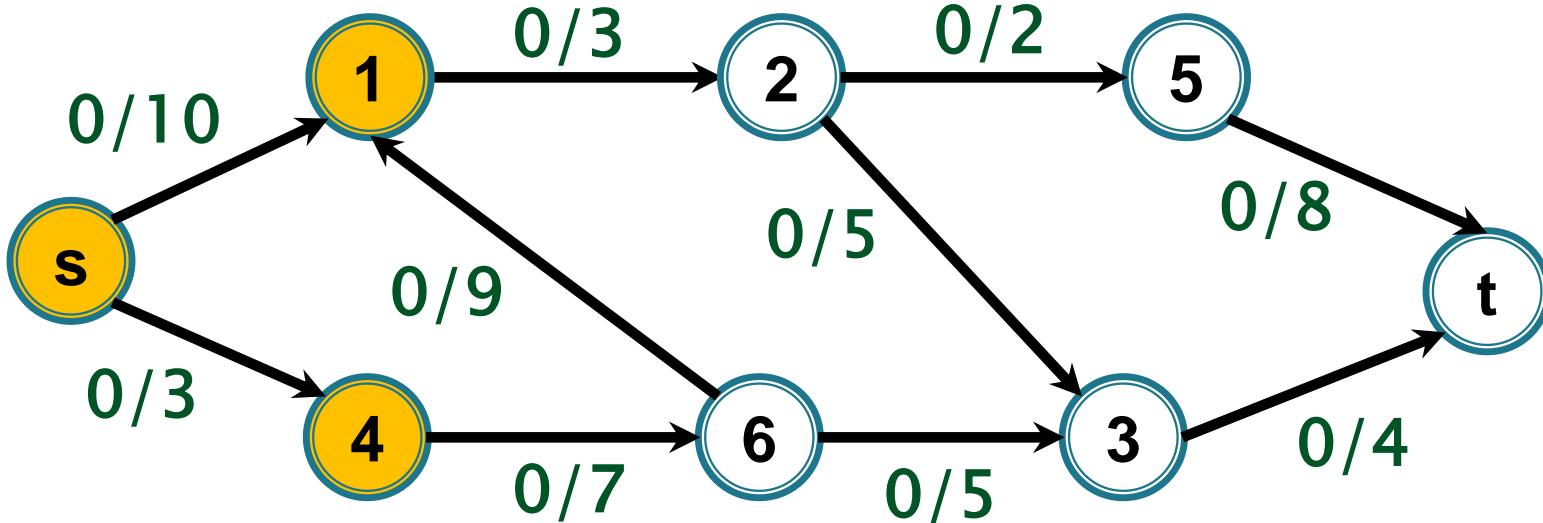
**construiste\_s-t\_lant\_nesat\_BF**

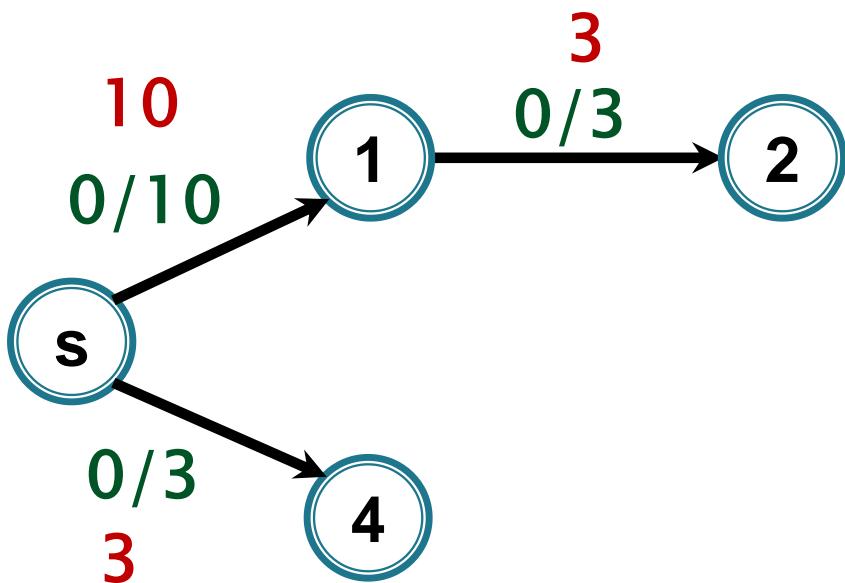
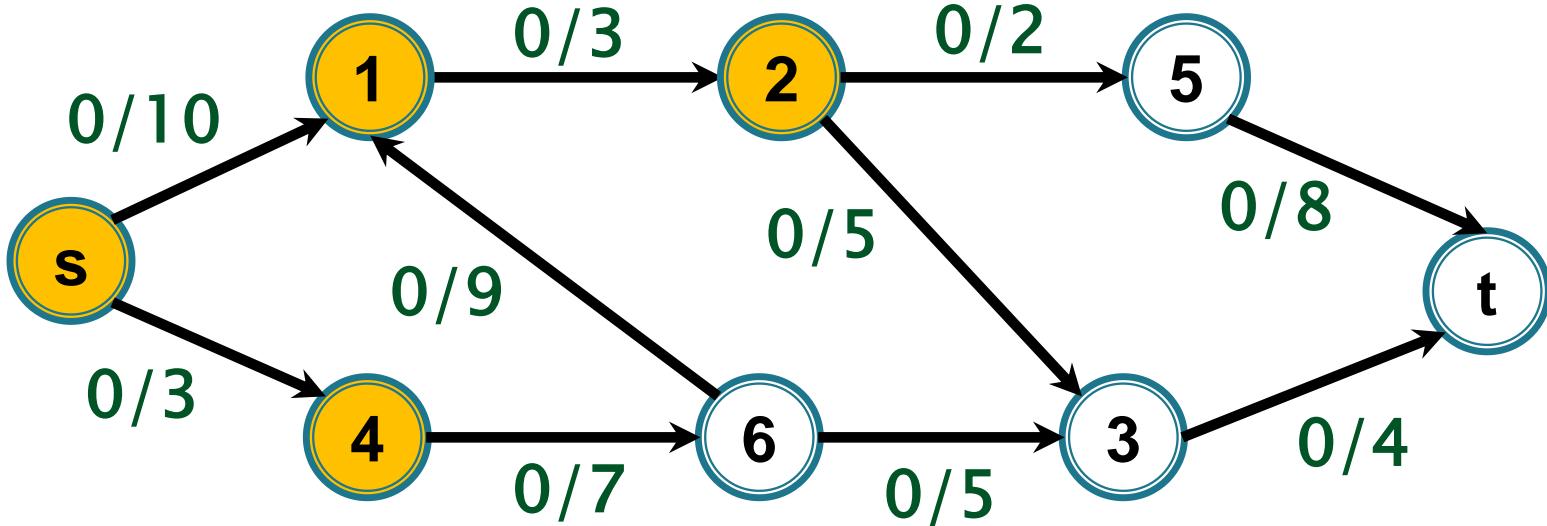


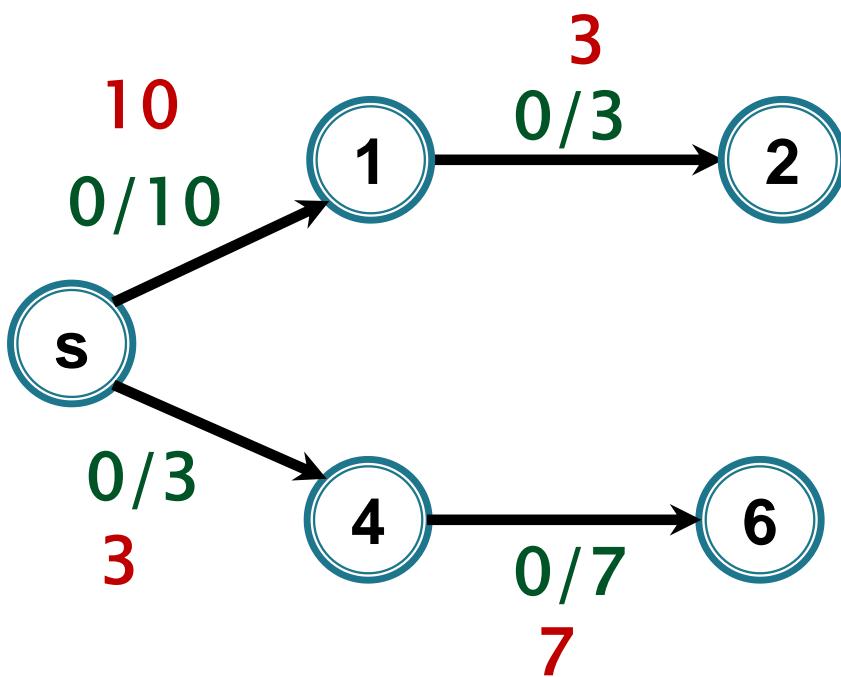
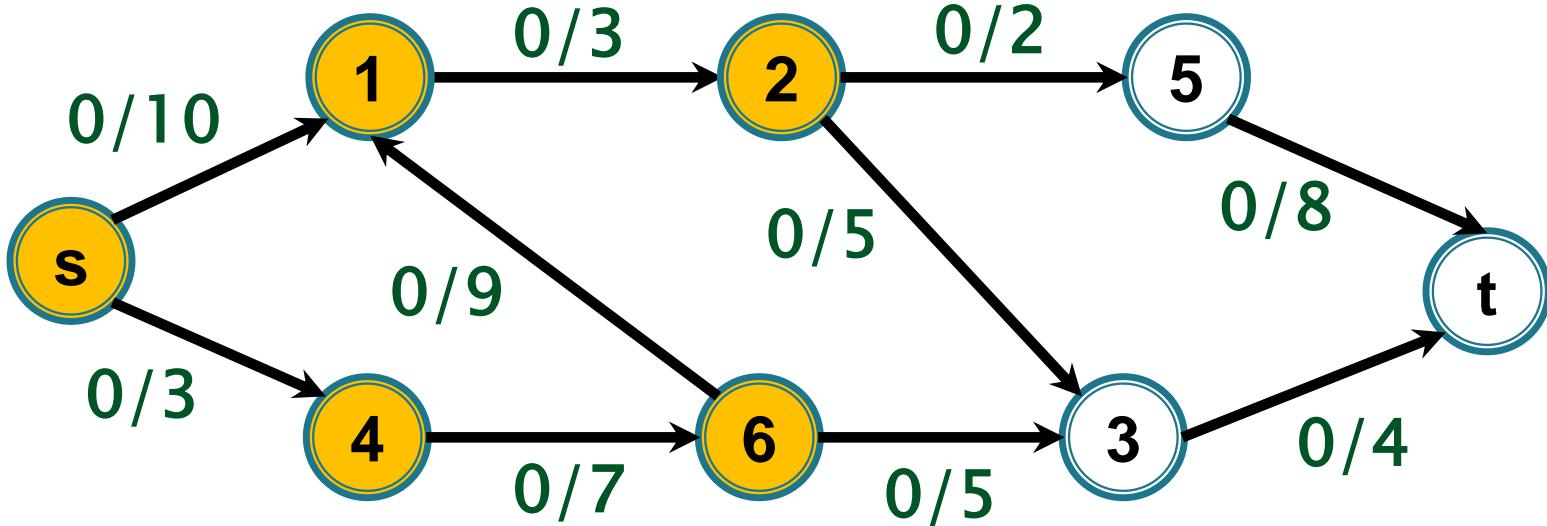


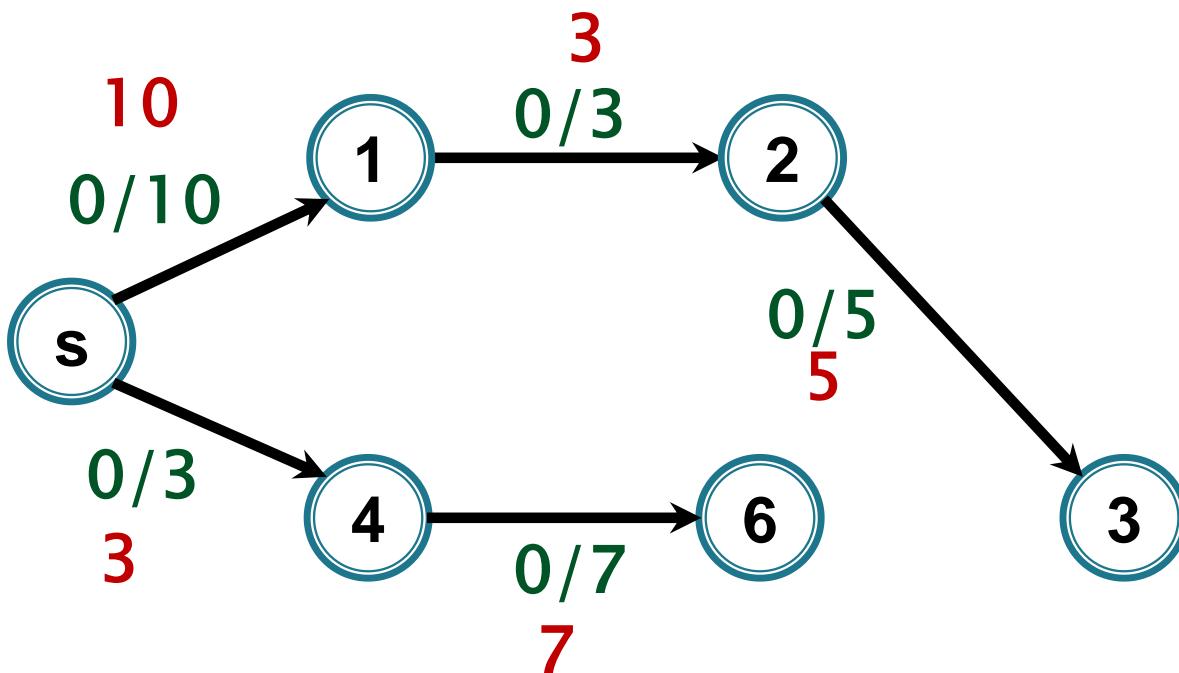
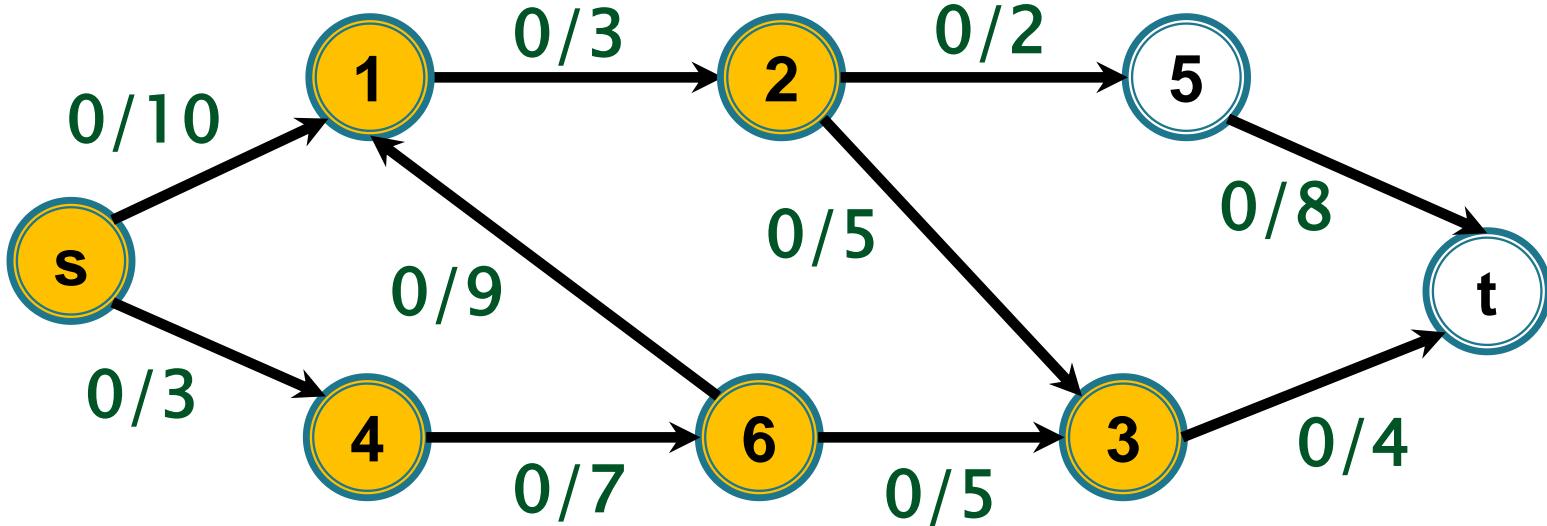
Capacitatea reziduală

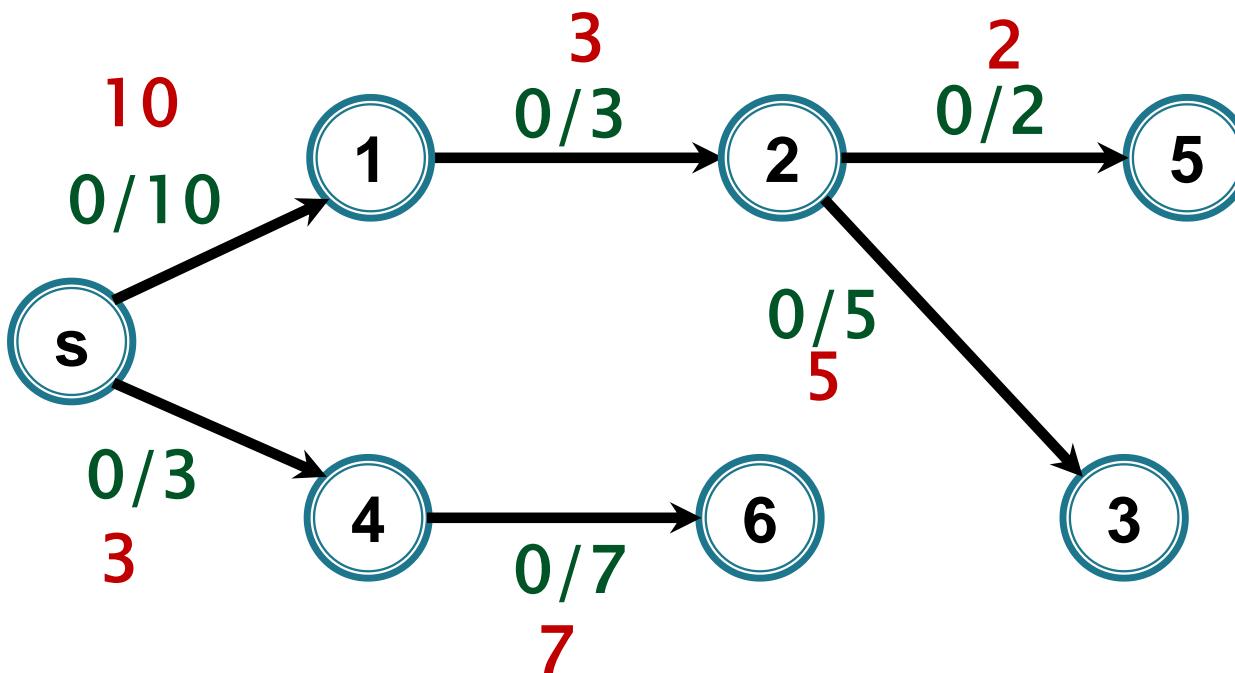
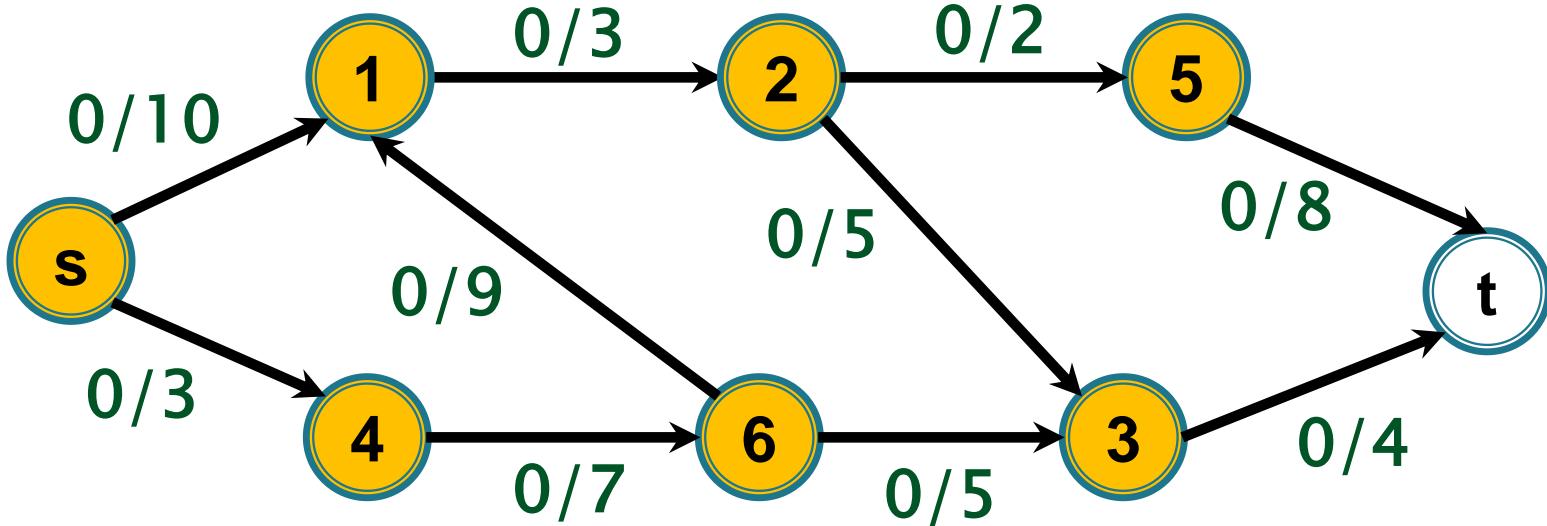


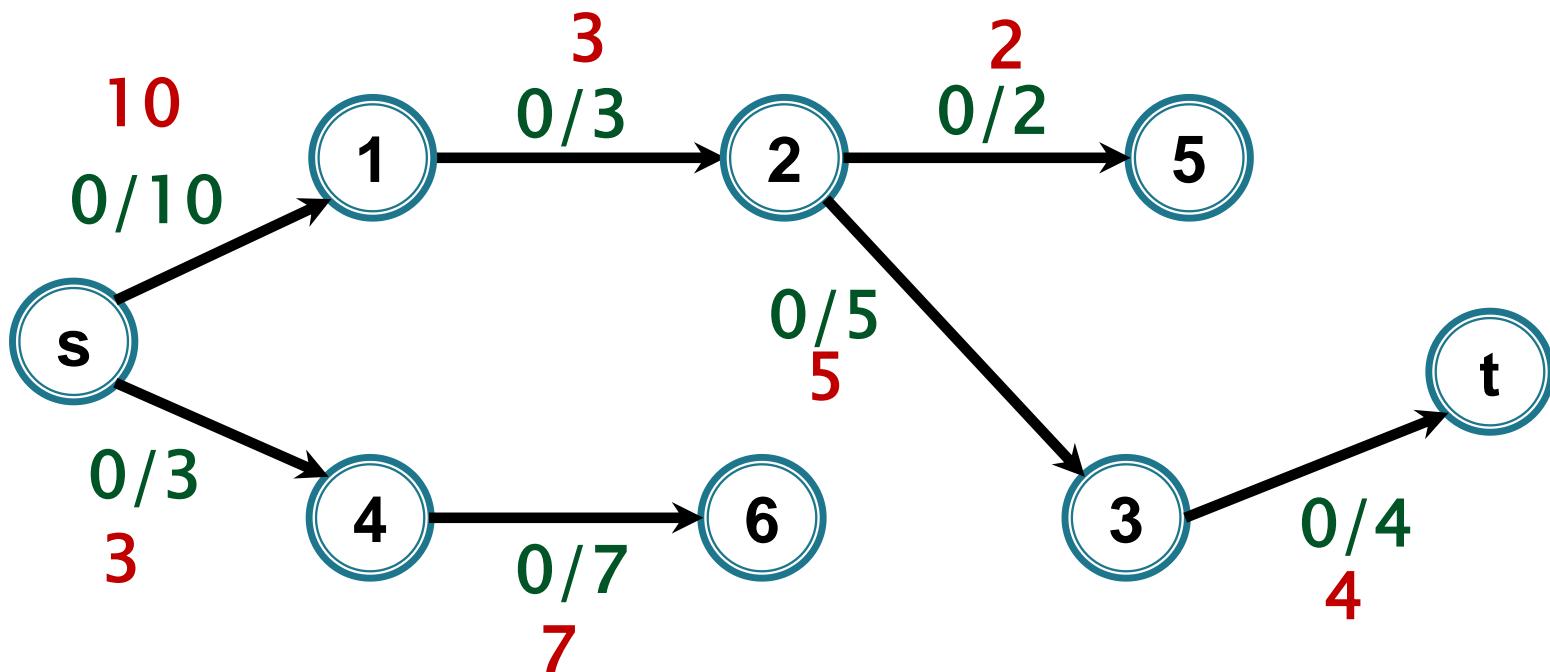
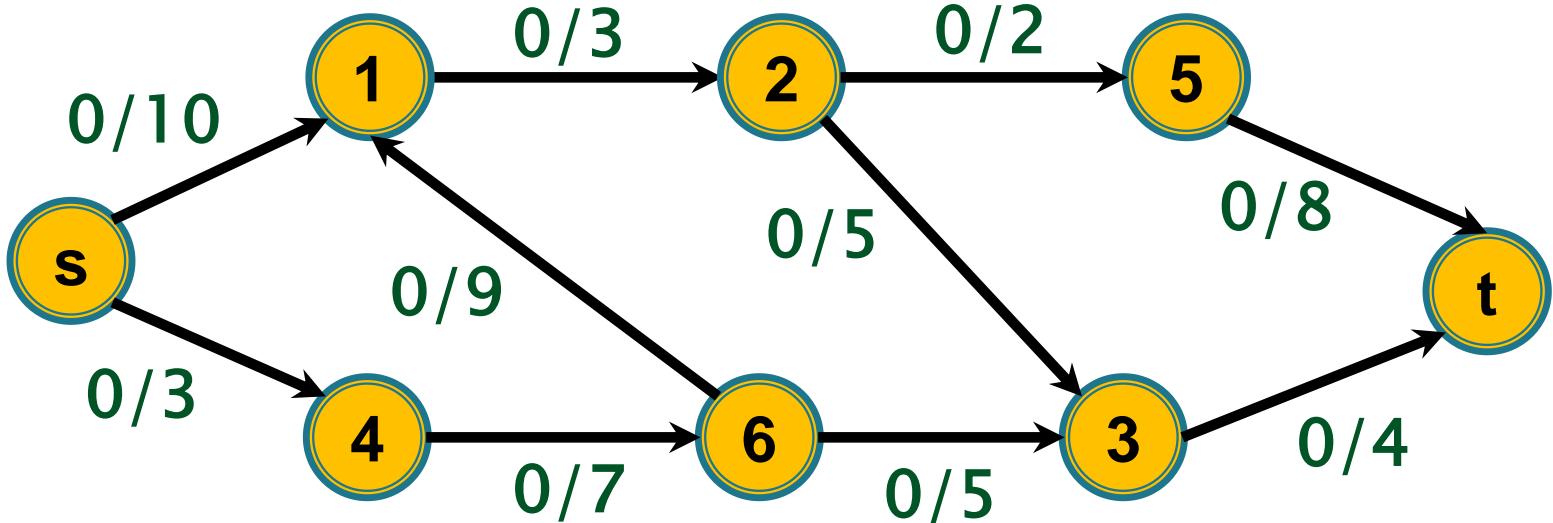


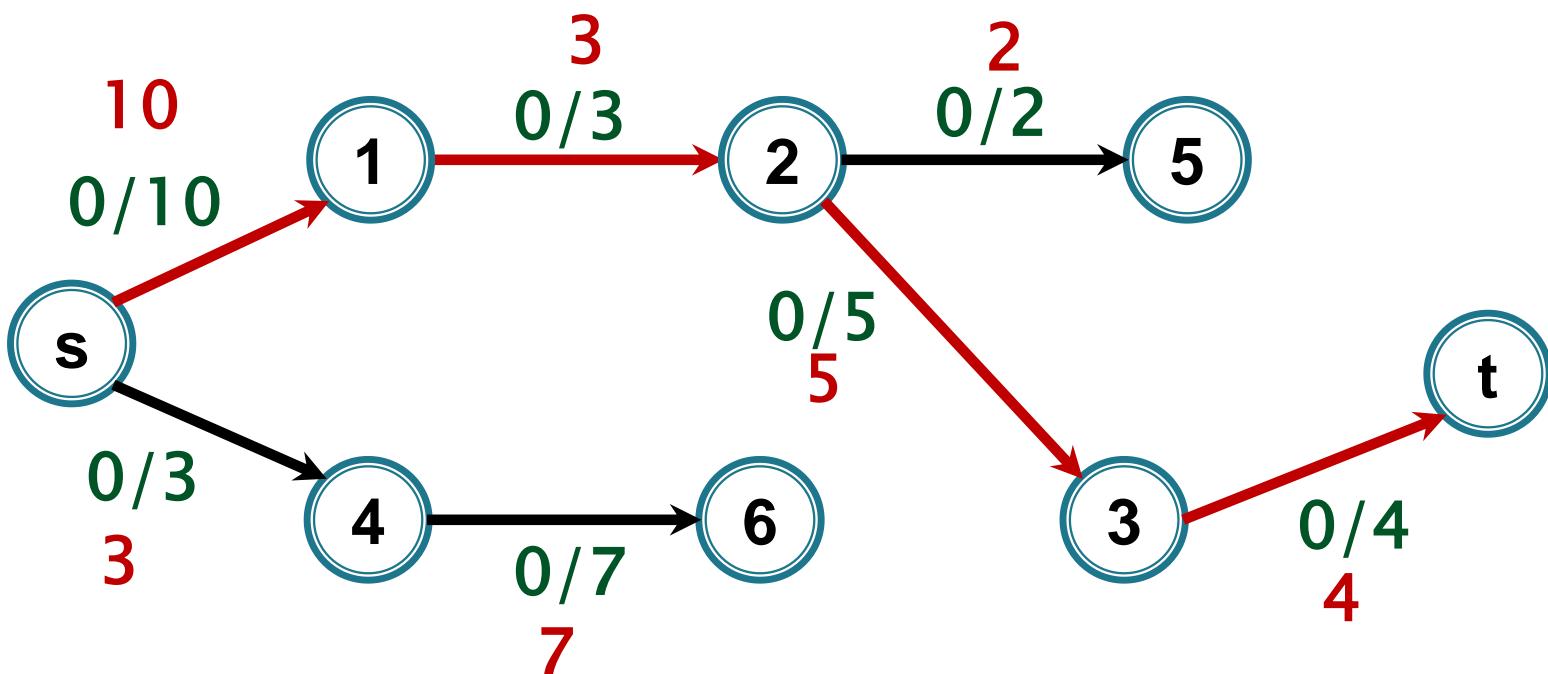
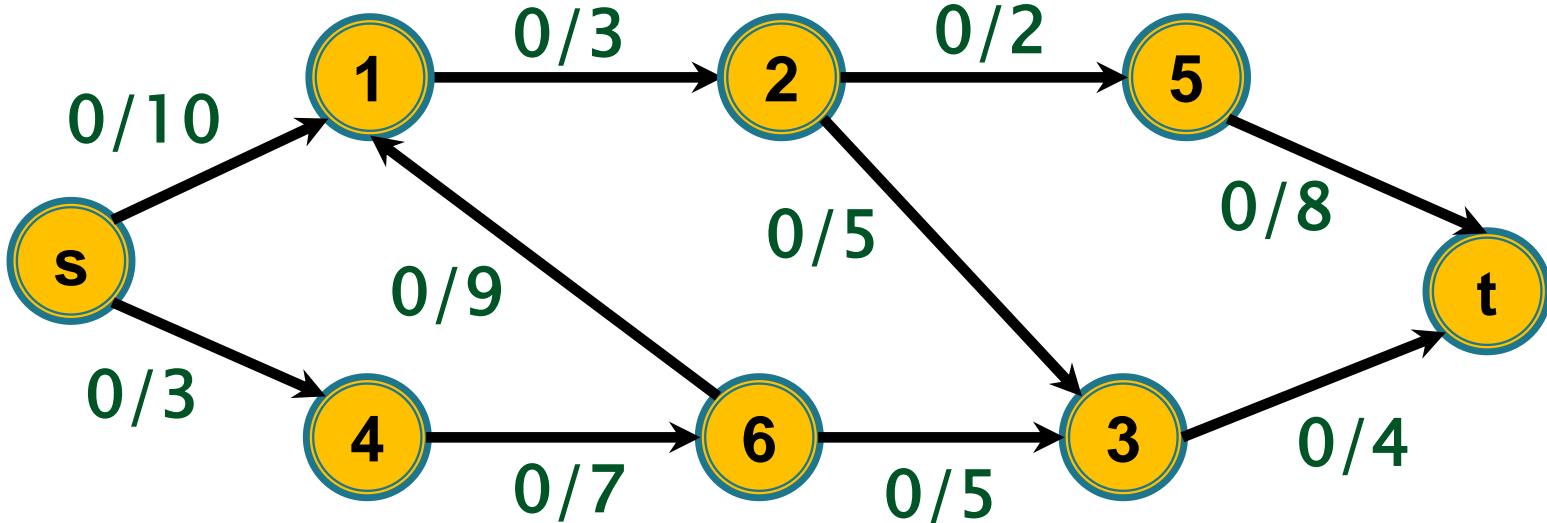




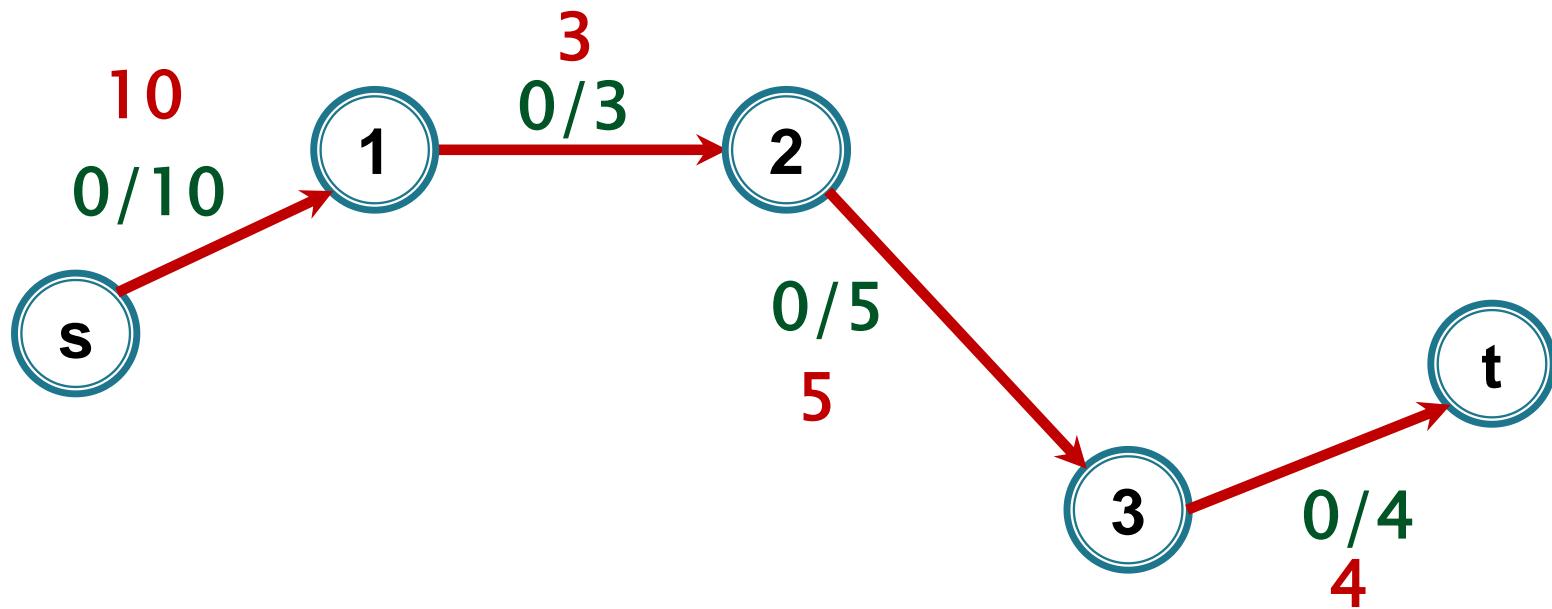
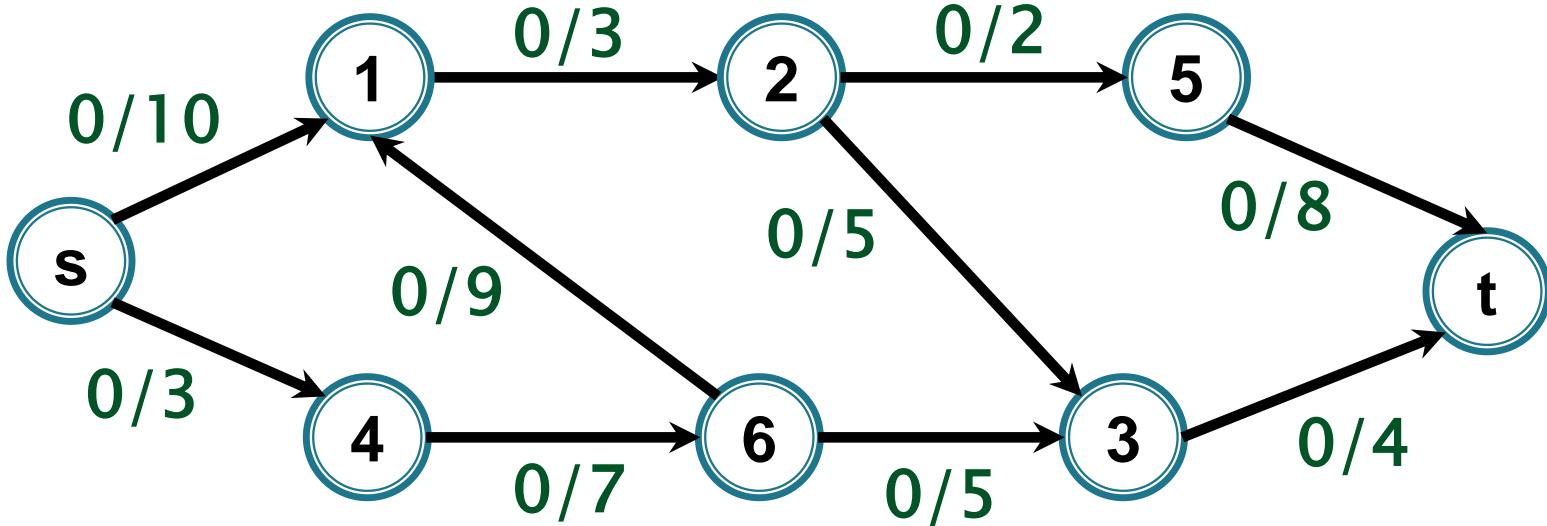




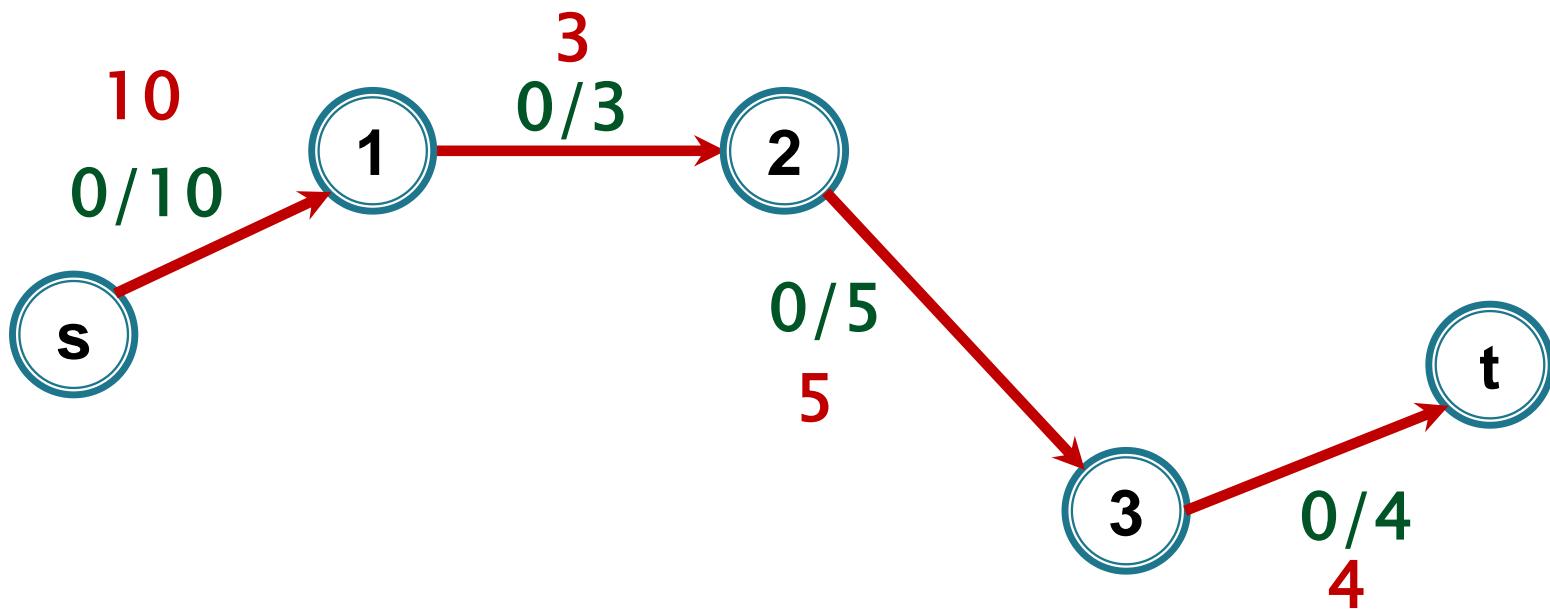
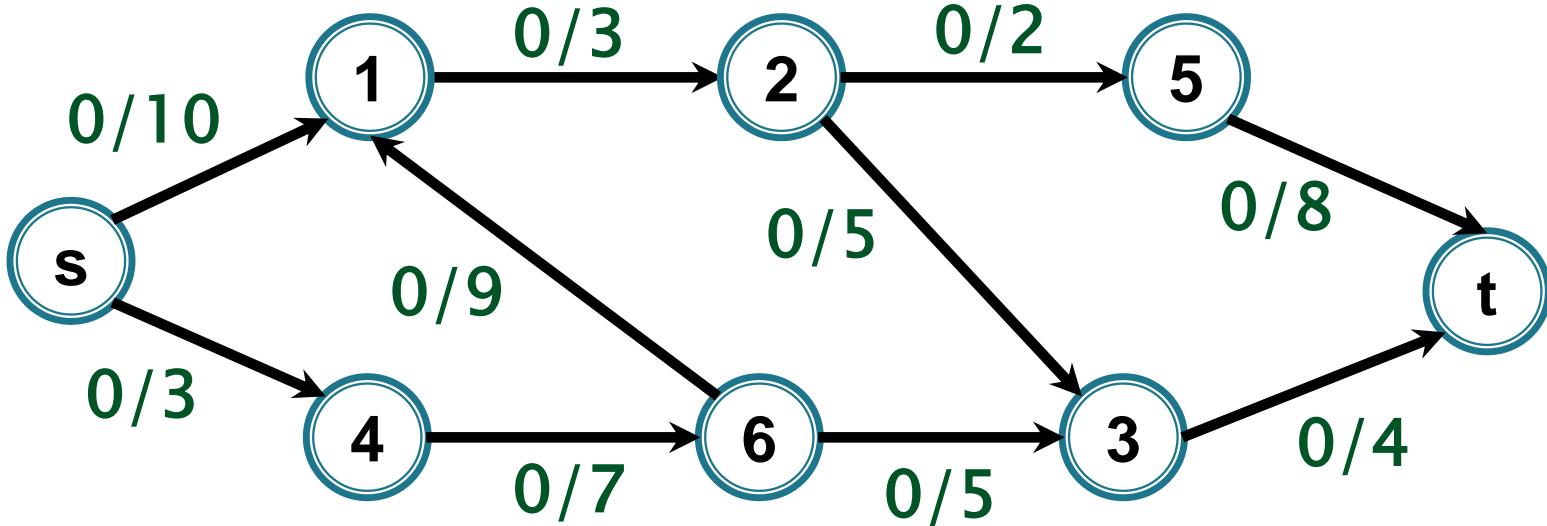




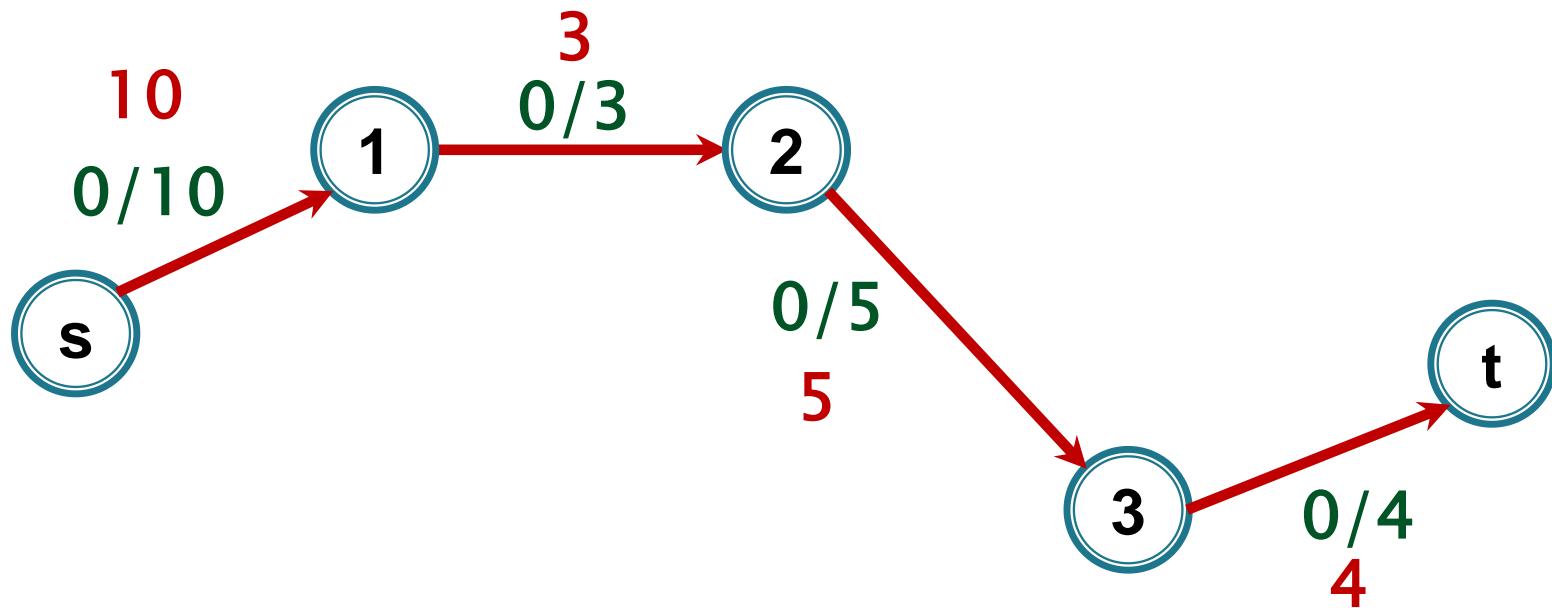
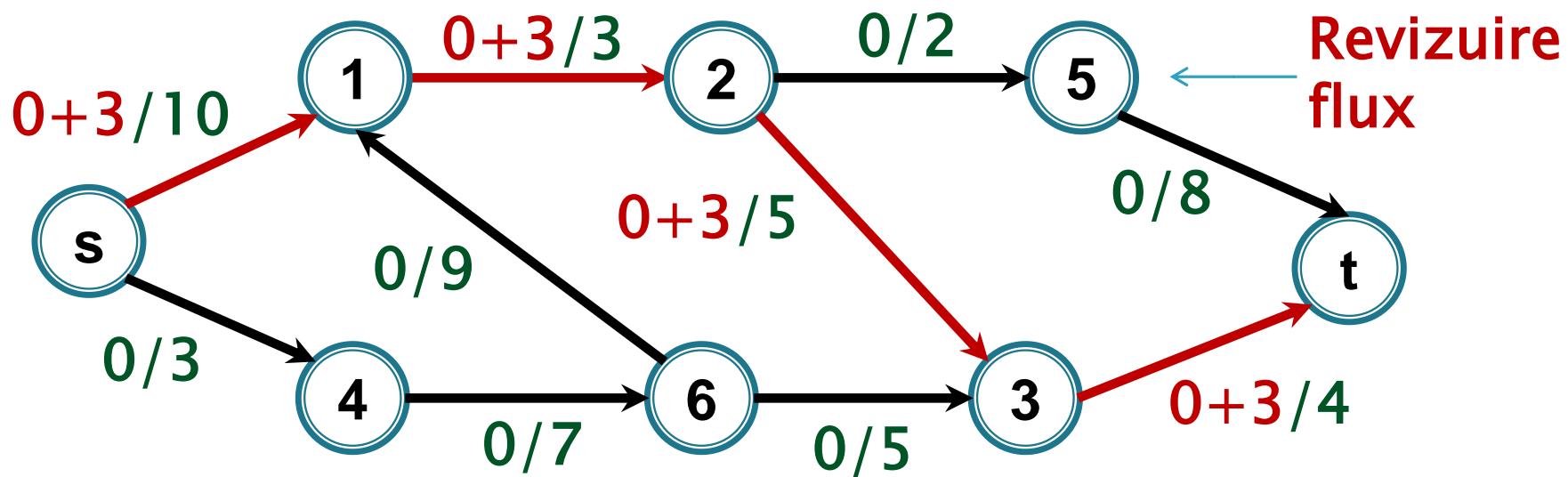
**revizuieste\_flux\_lant**



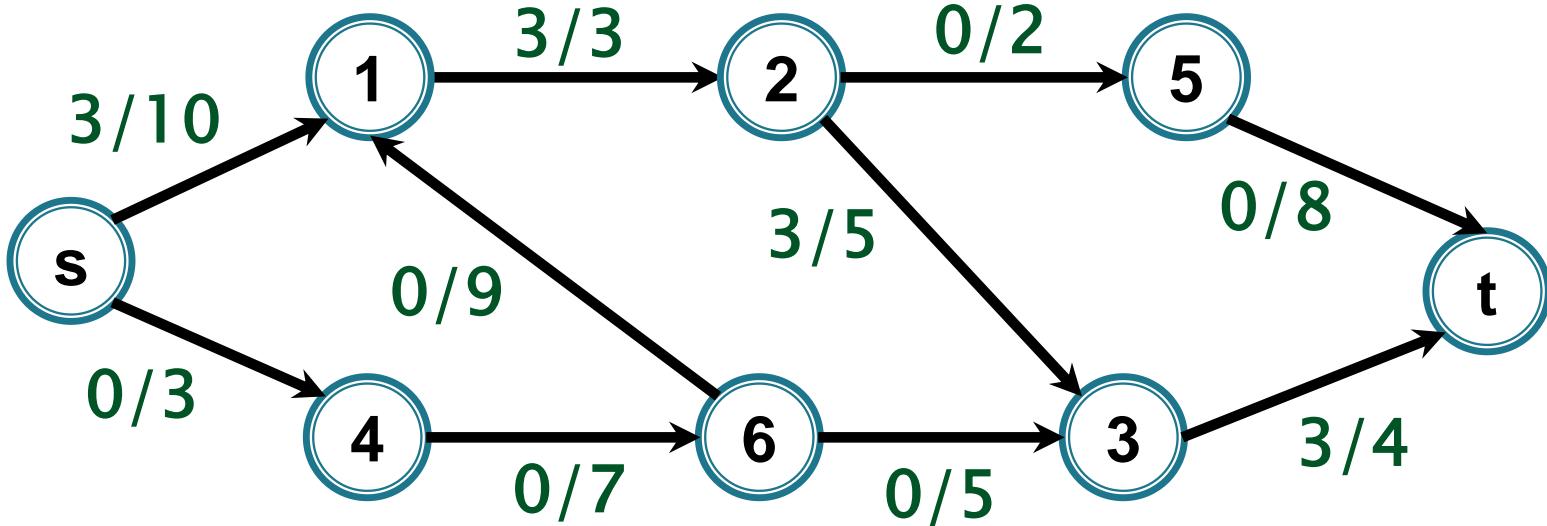
$$i(P) = ?$$



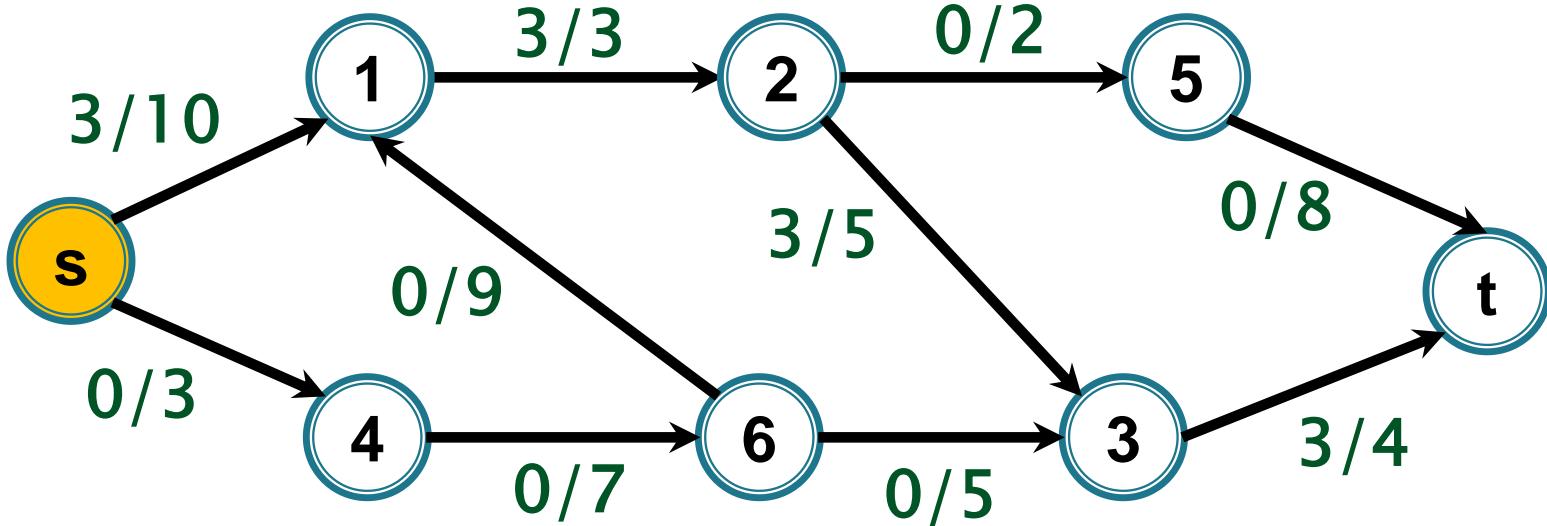
$$i(P) = \min \{10, 3, 5, 4\} = 3$$

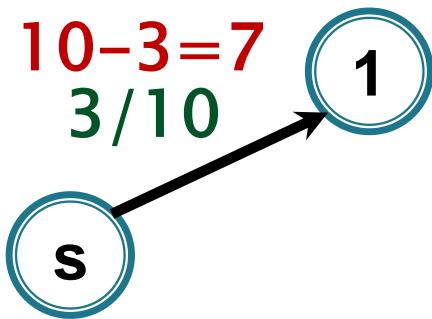
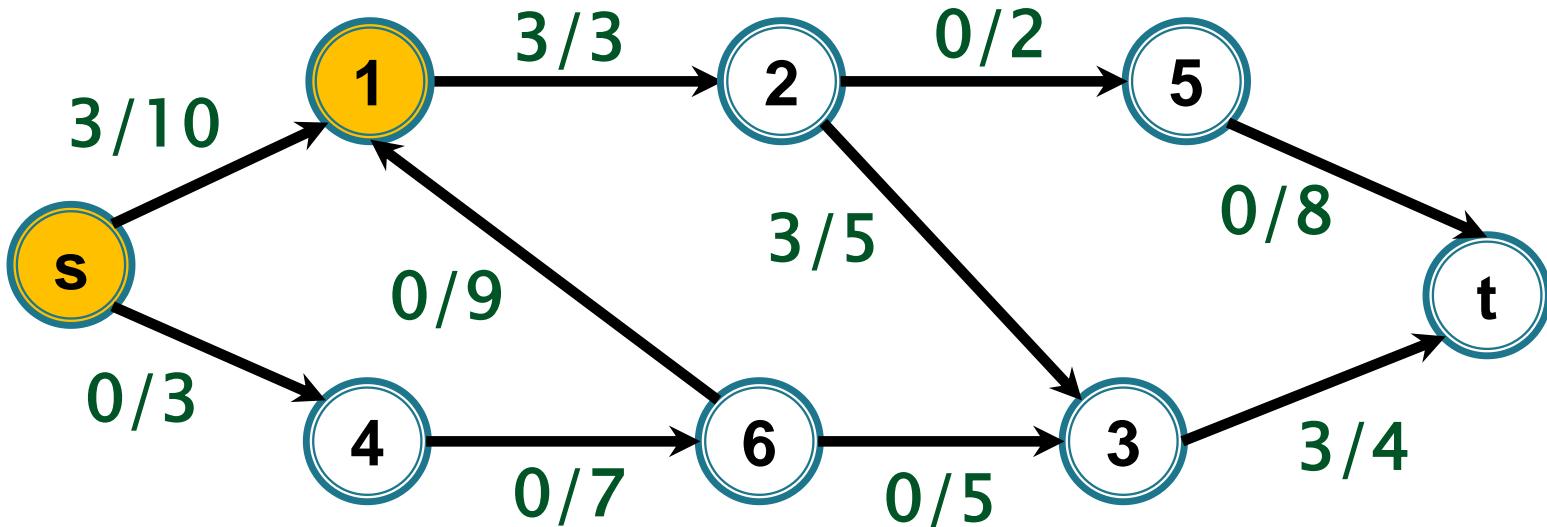


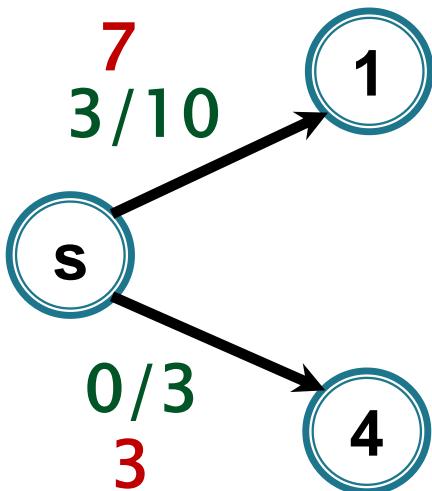
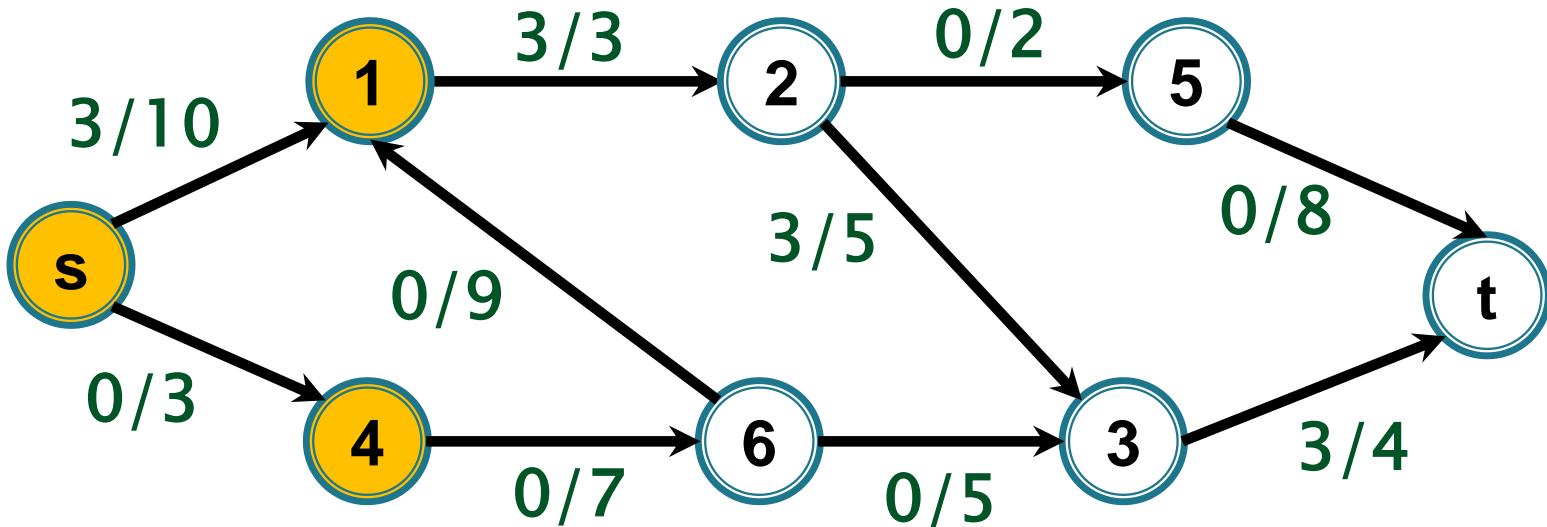
$$i(P) = \min \{10, 3, 5, 4\} = 3$$

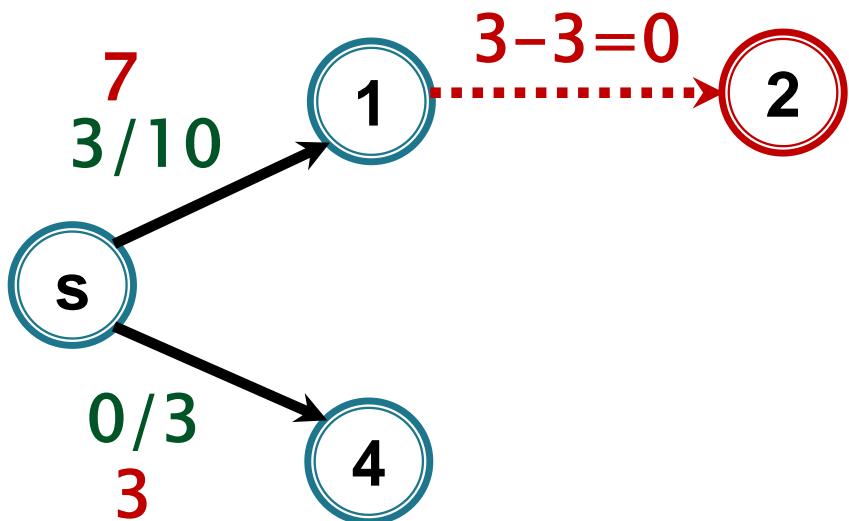
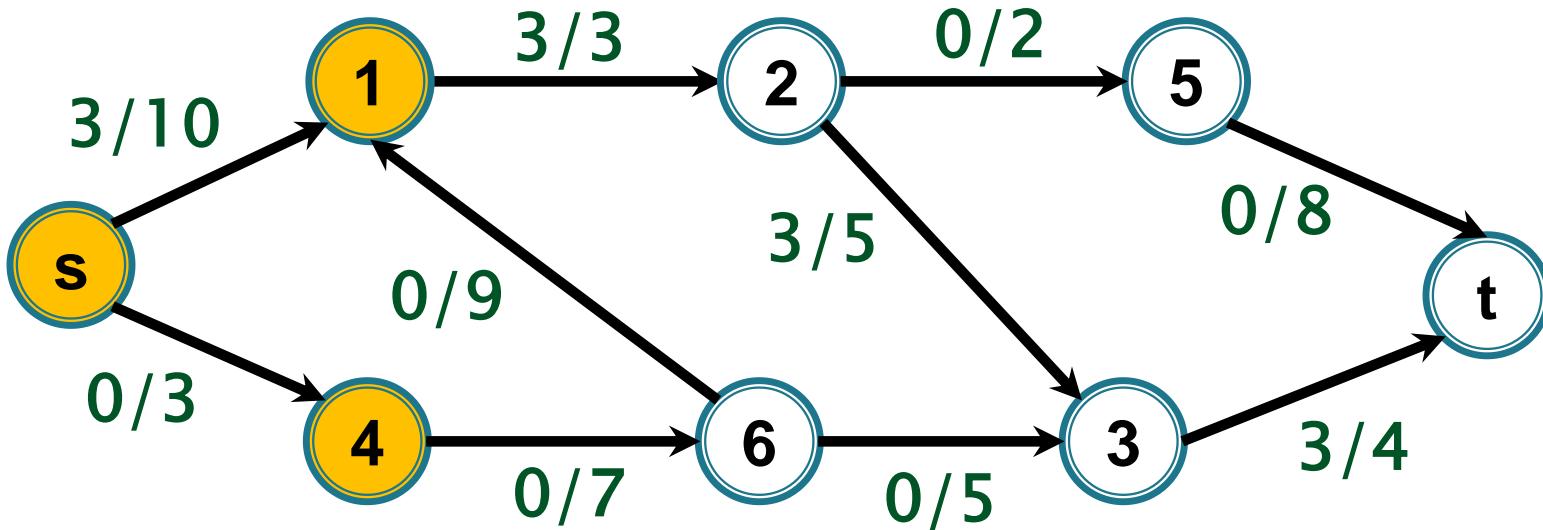


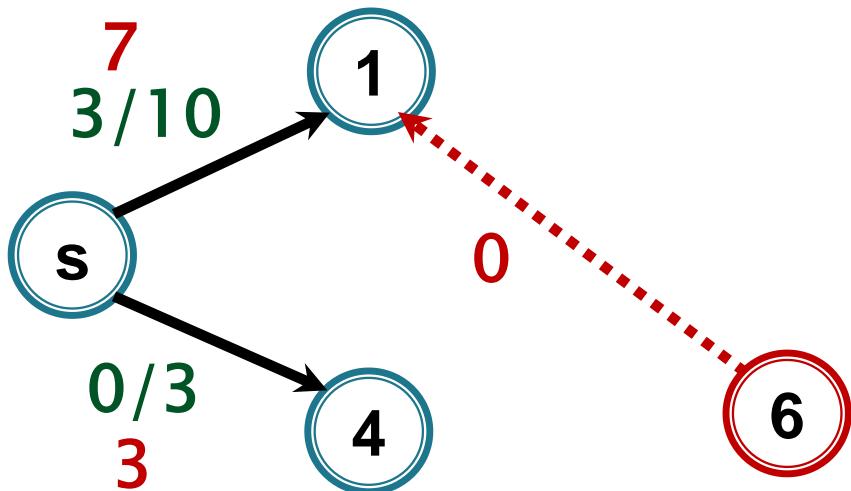
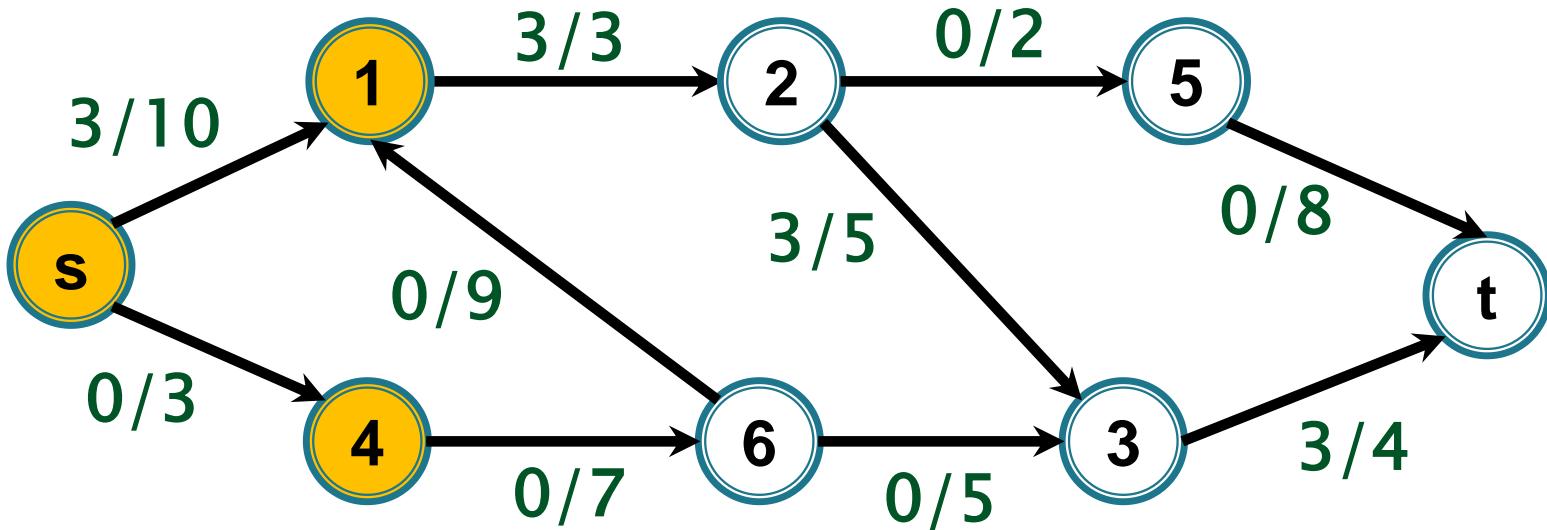
**construieste\_s-t\_lant\_nesat\_BF**

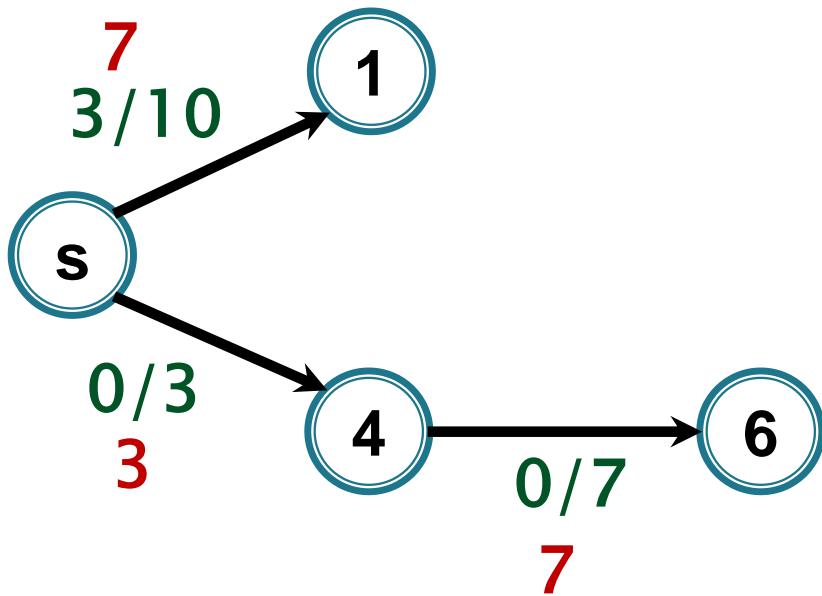
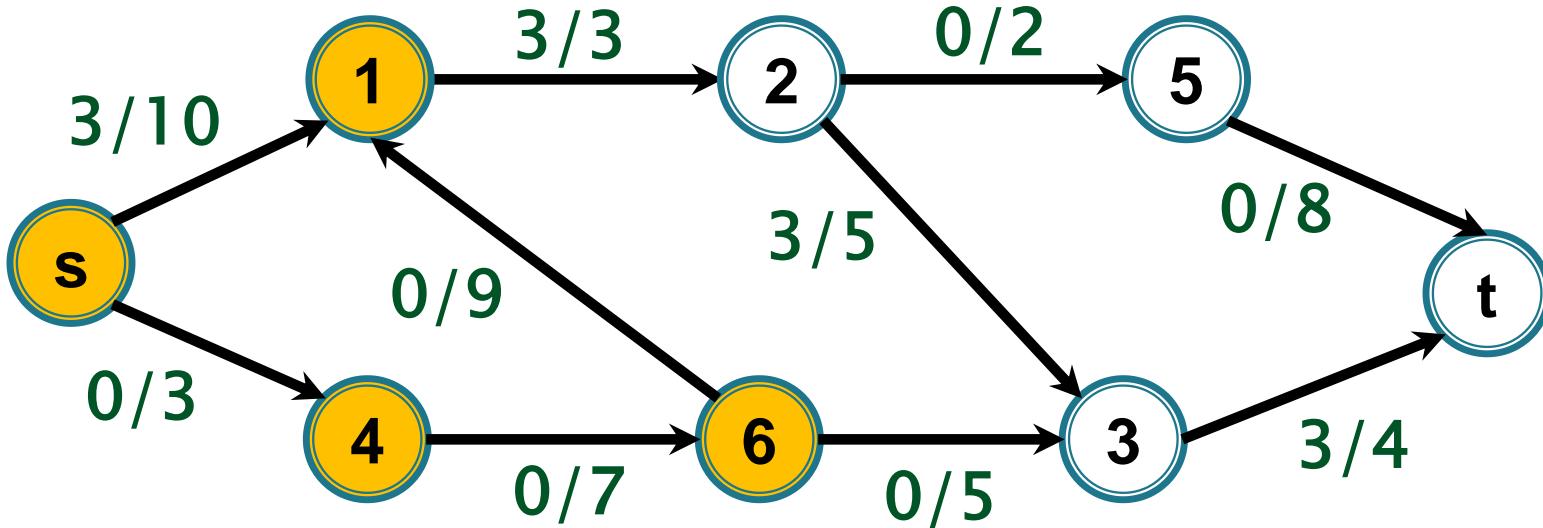


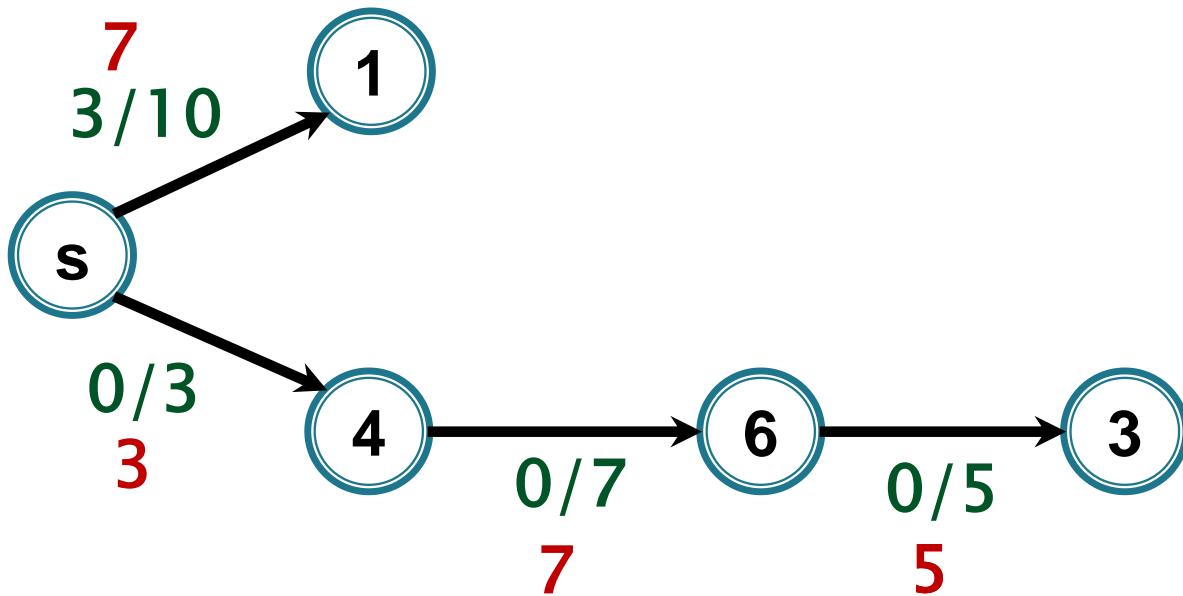
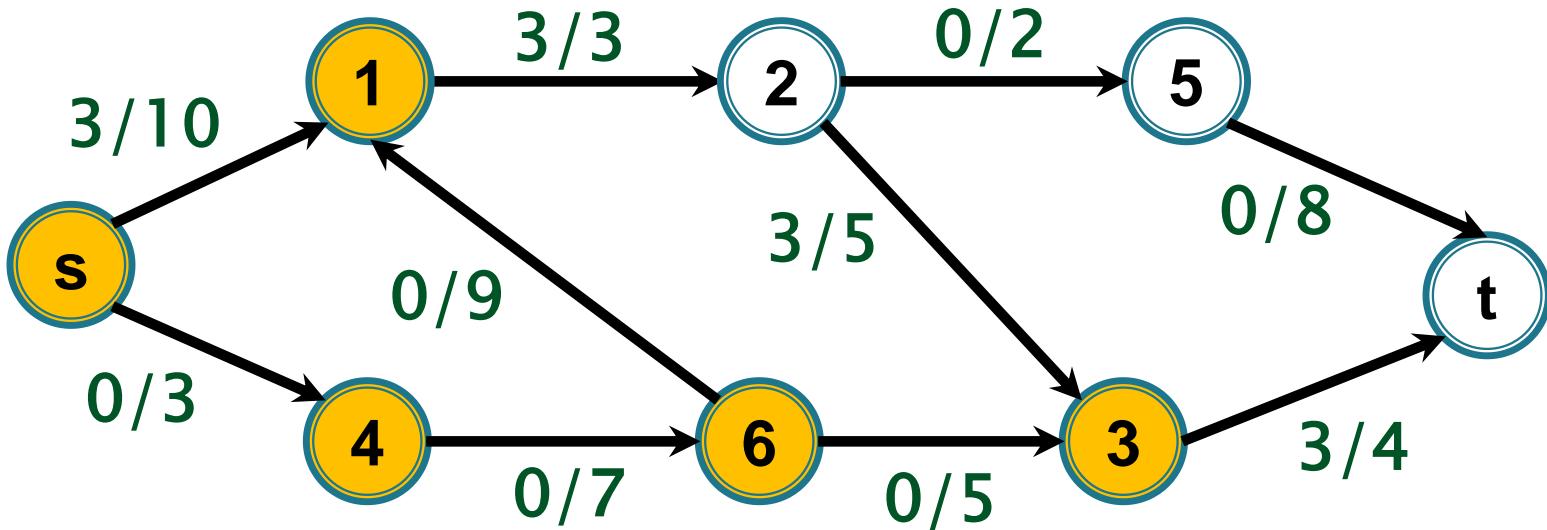


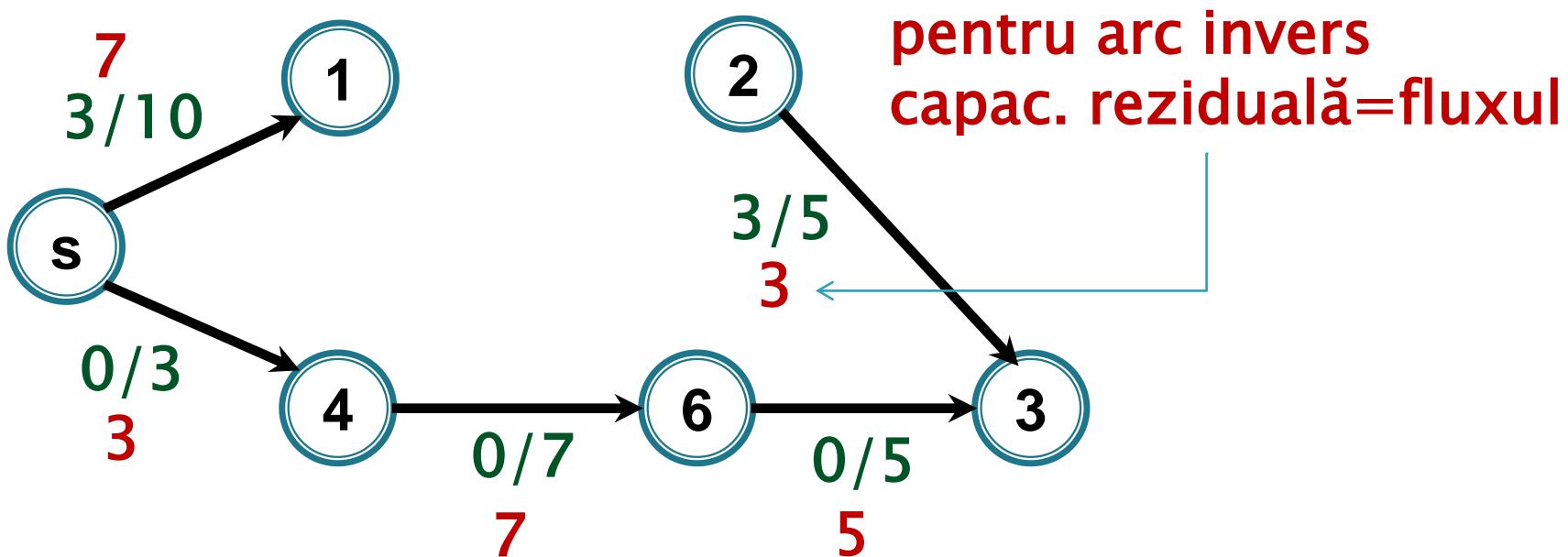
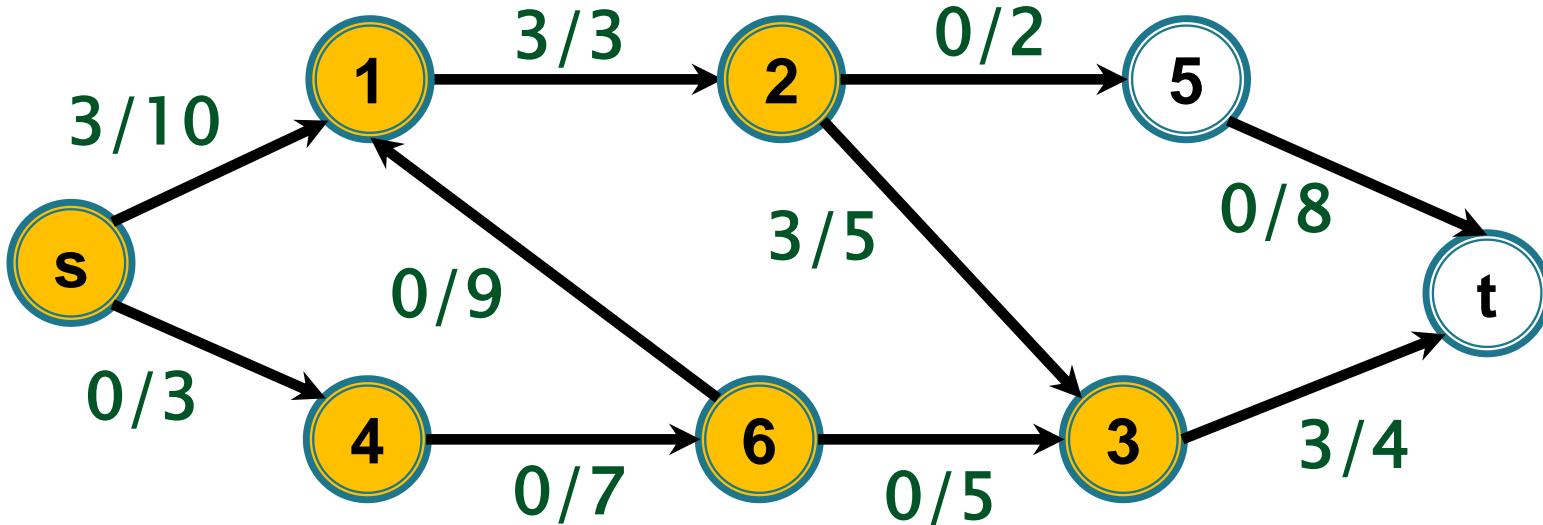


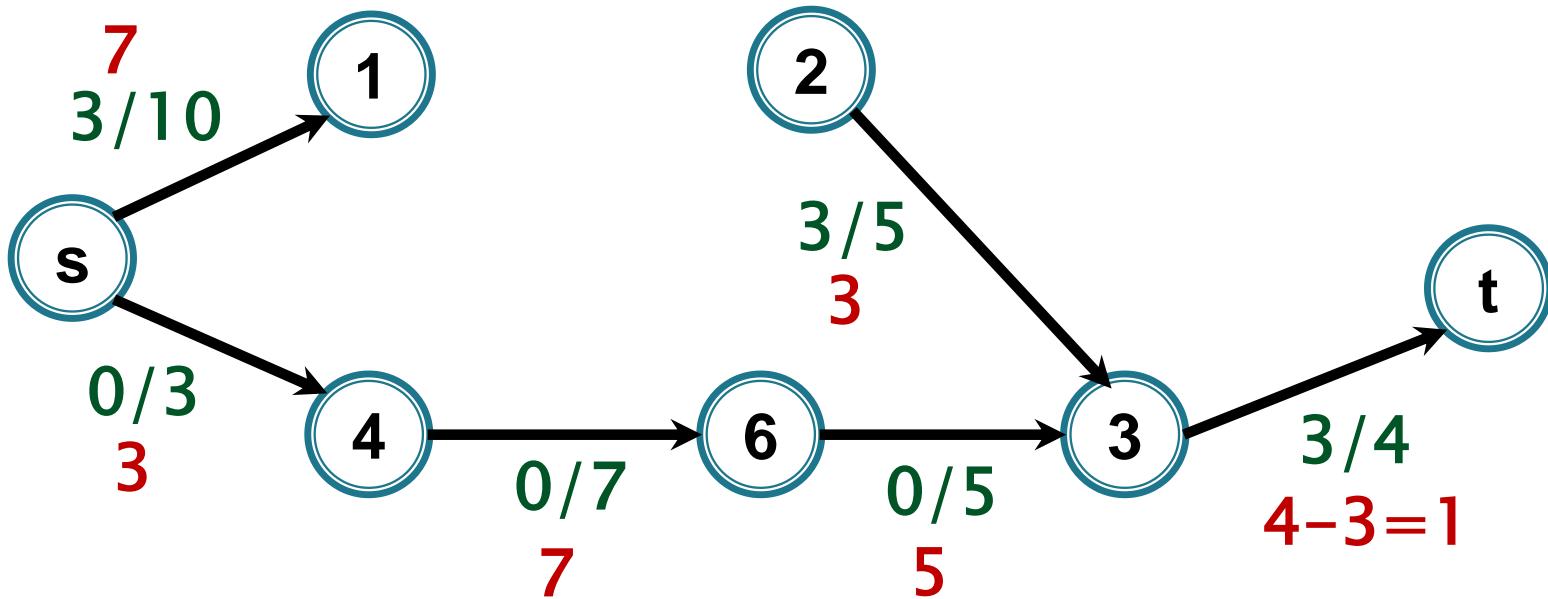
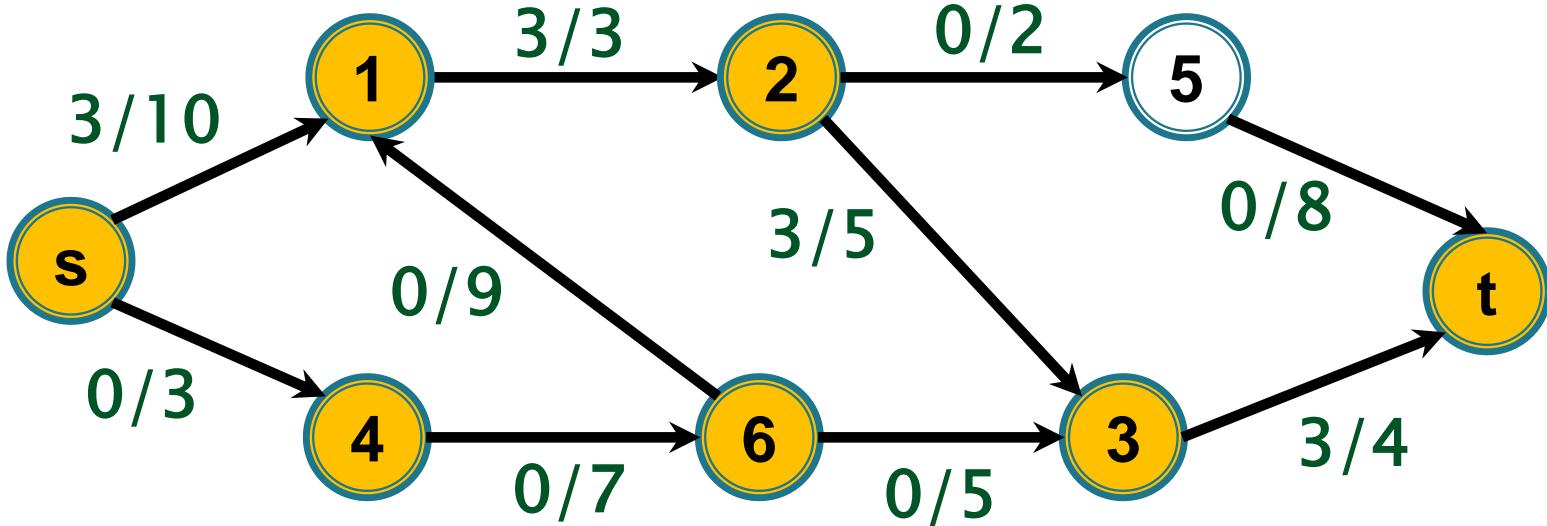




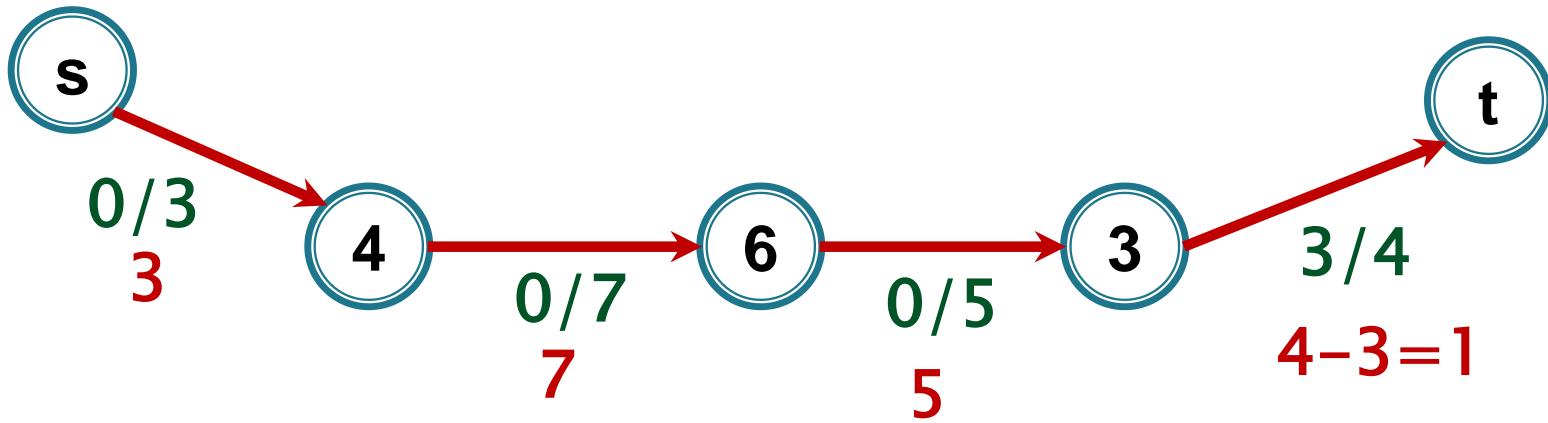
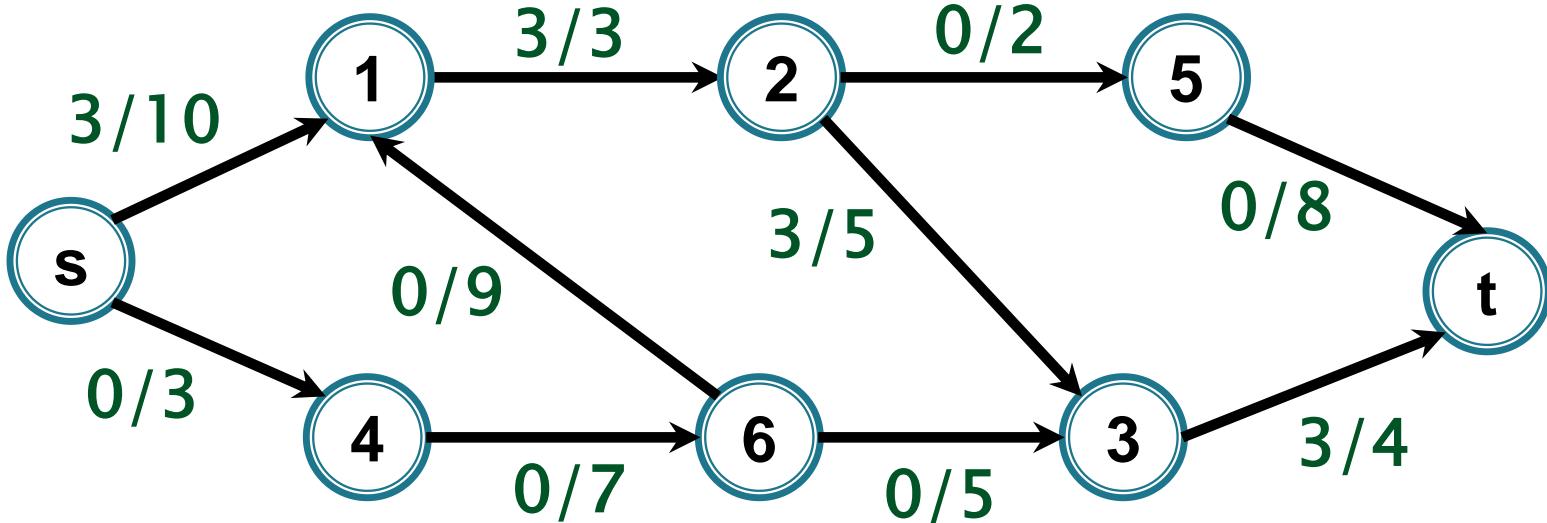


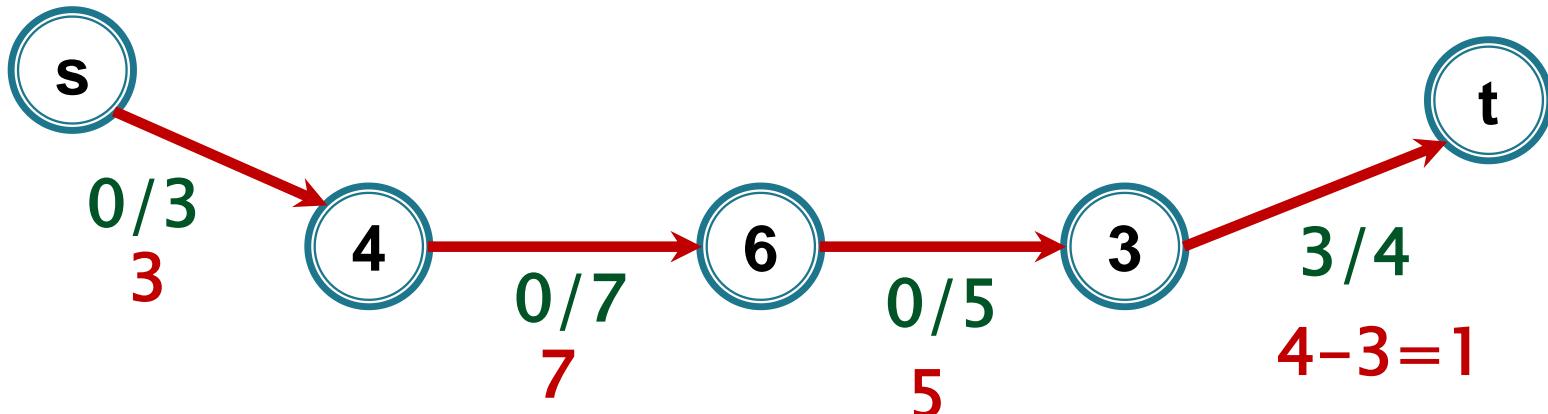
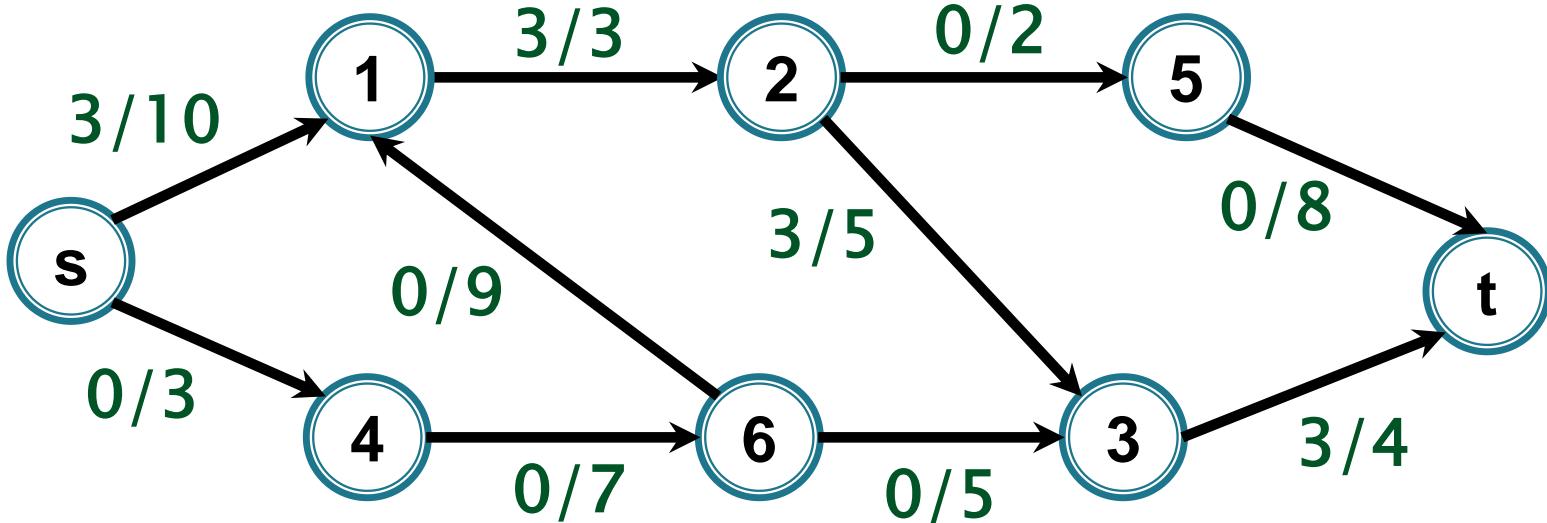




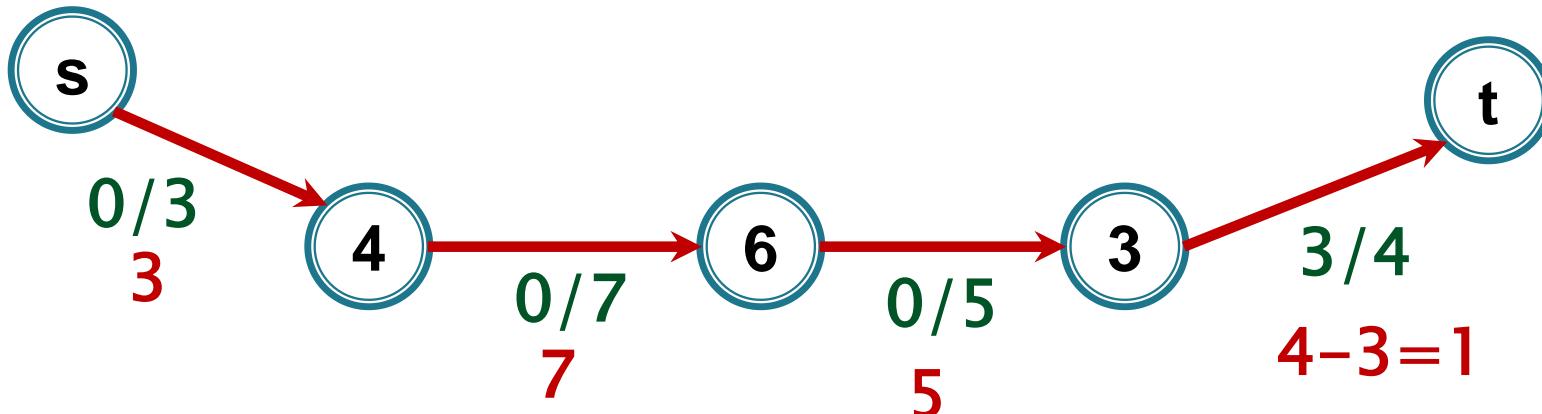
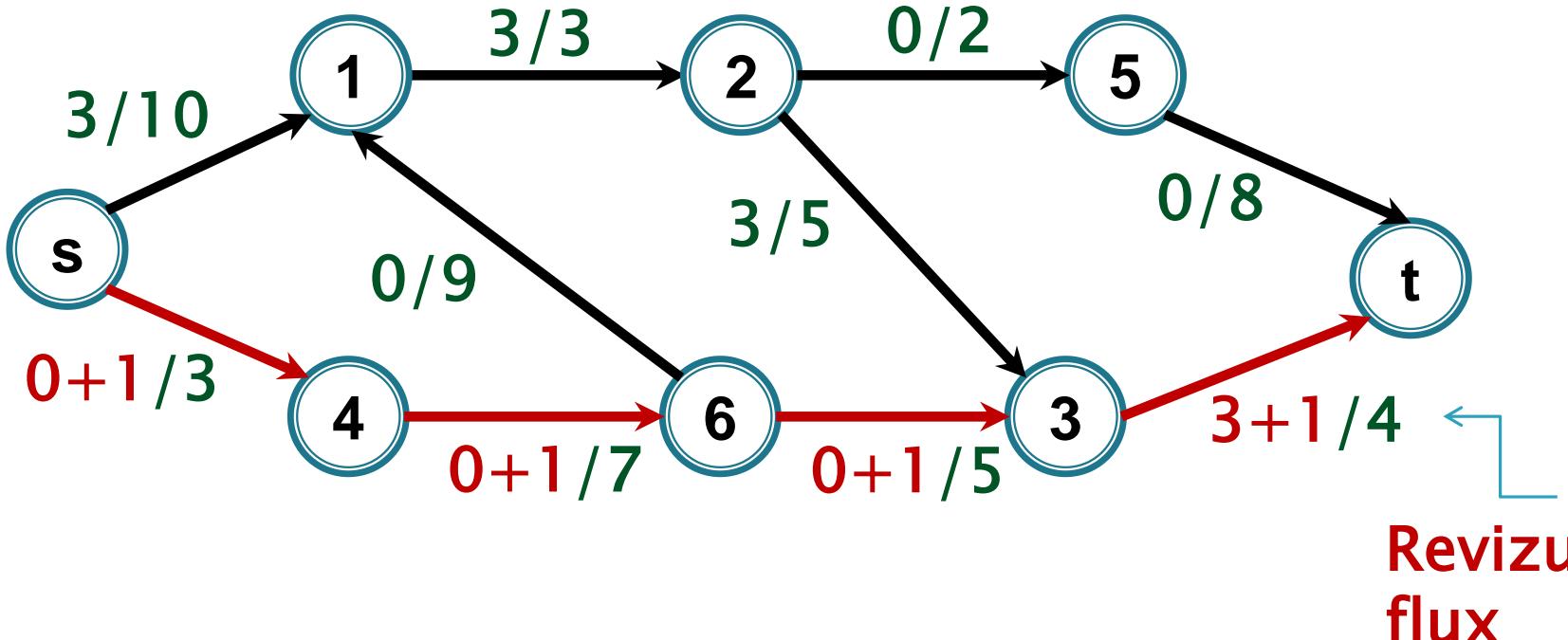


**revizuieste\_flux\_lant**

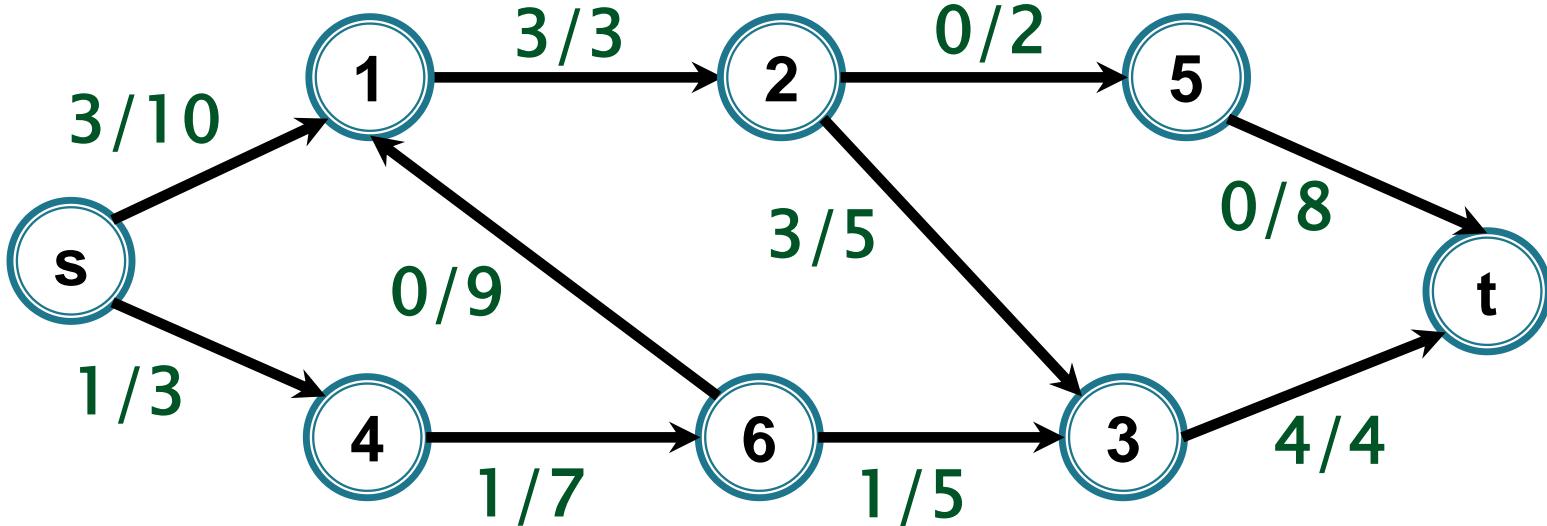




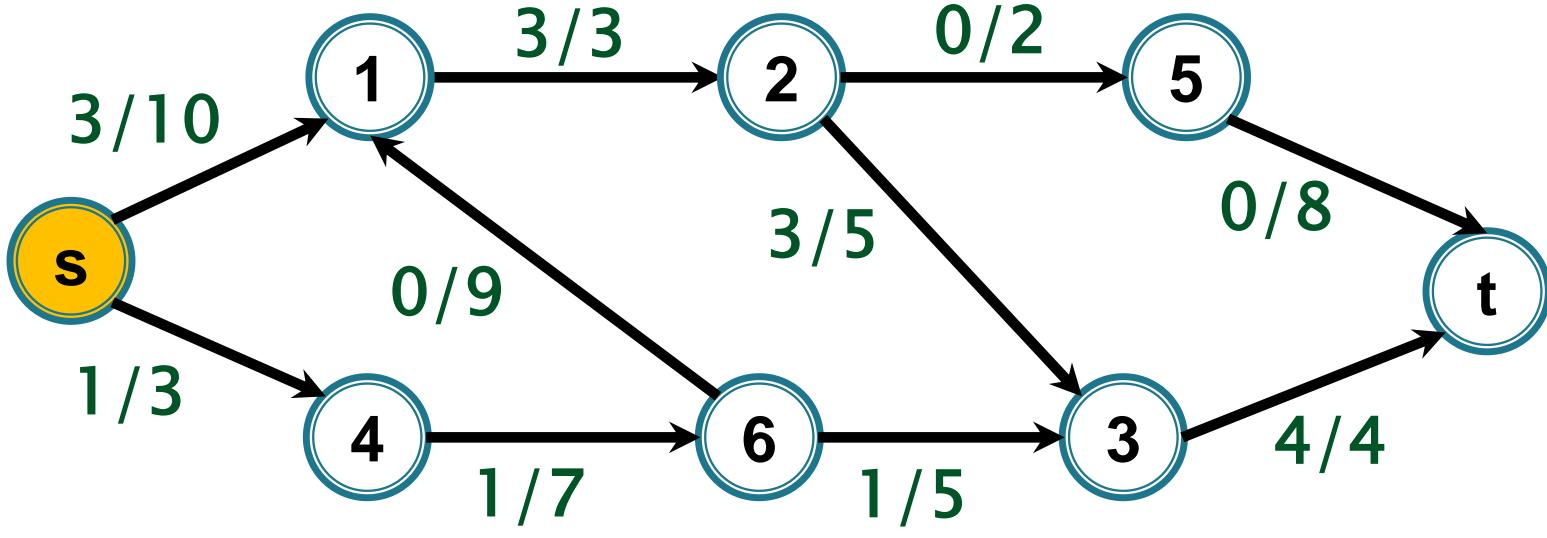
$$i(P) = \min\{ 3, 7, 5, 1 \} = 1$$

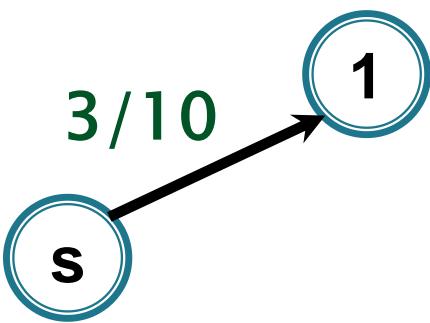
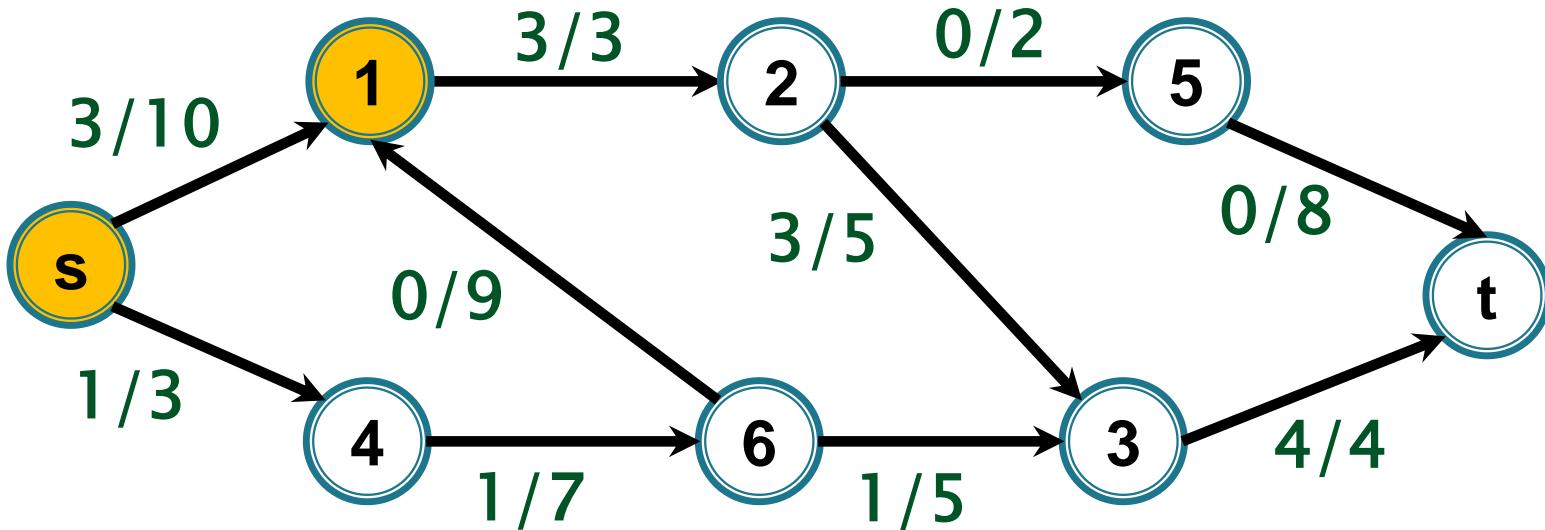


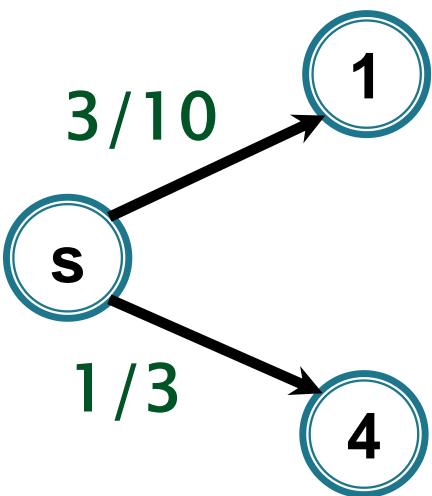
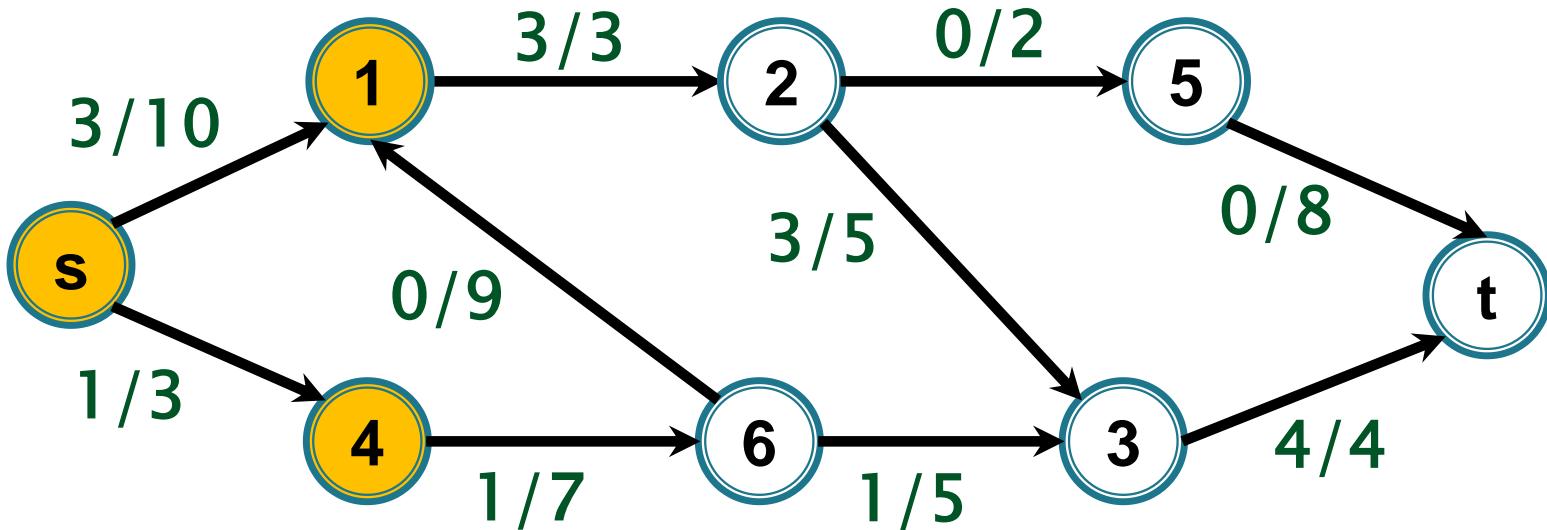
$$i(P) = \min\{ 3, 7, 5, 1 \} = 1$$

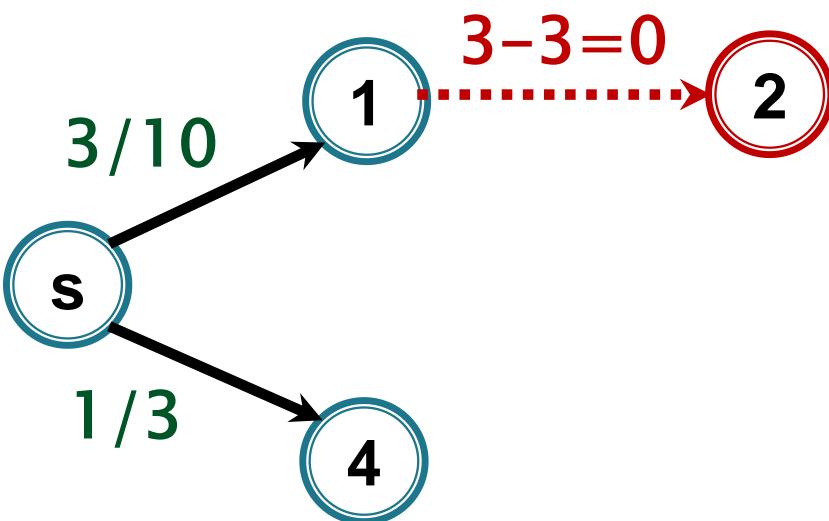
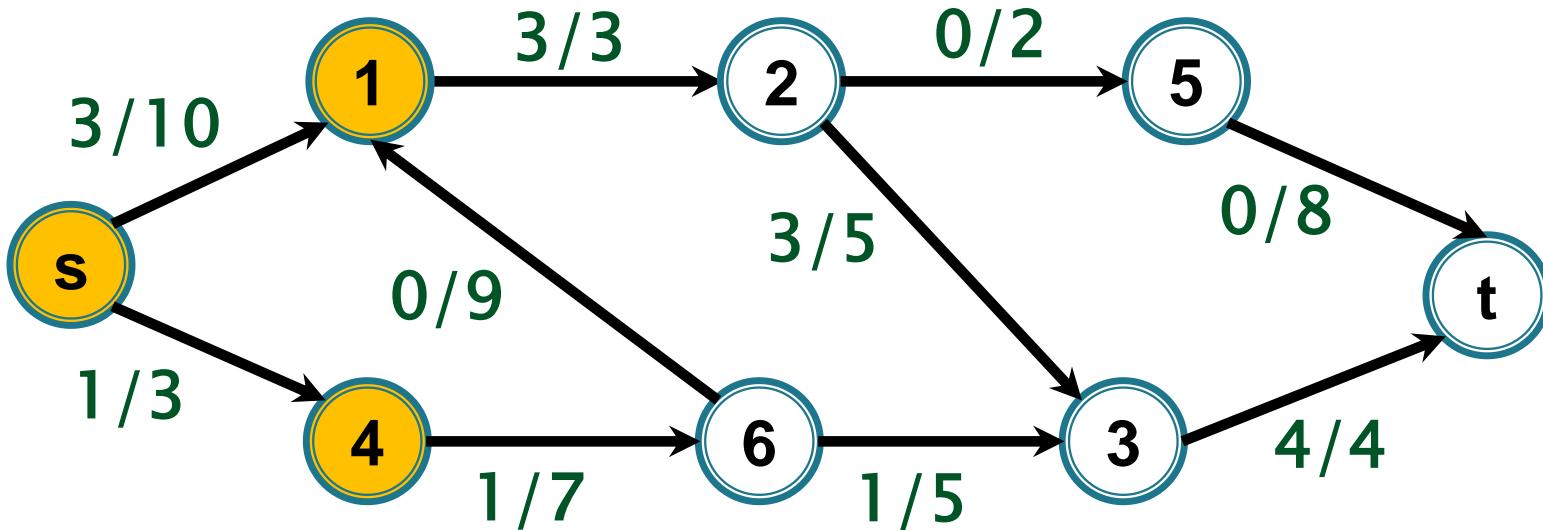


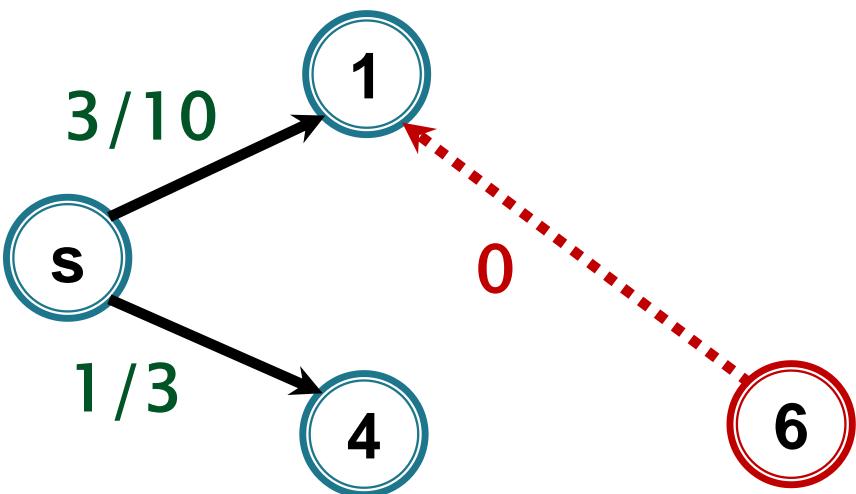
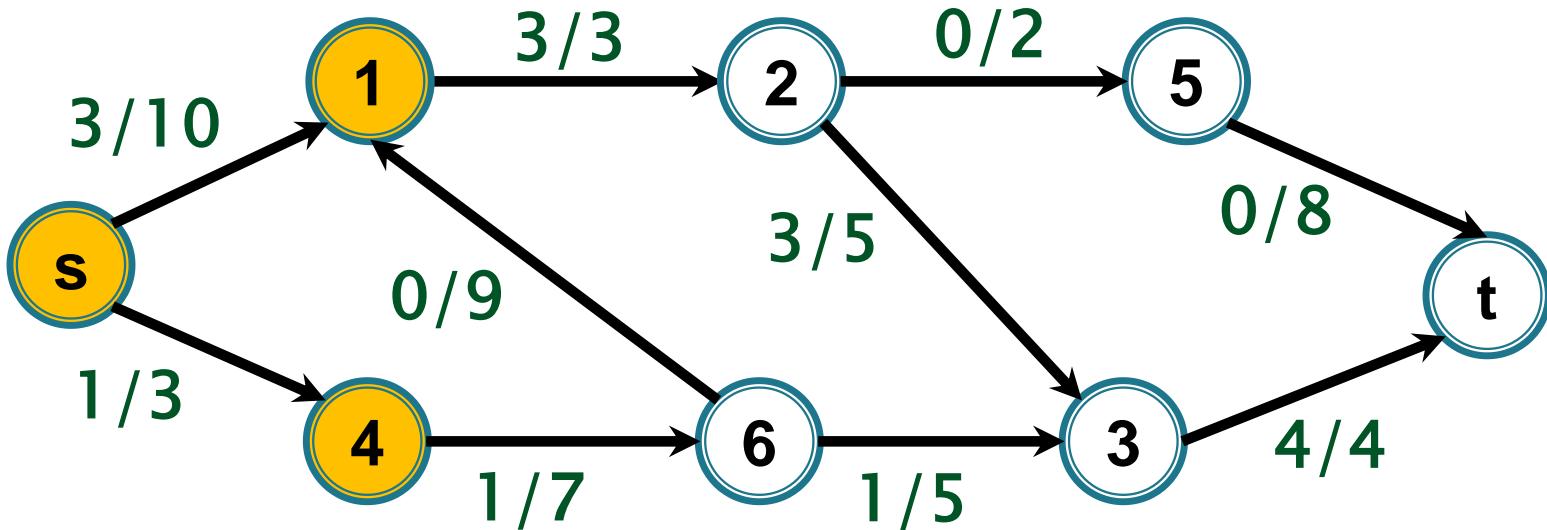
**construieste\_s-t\_lant\_nesat\_BF**

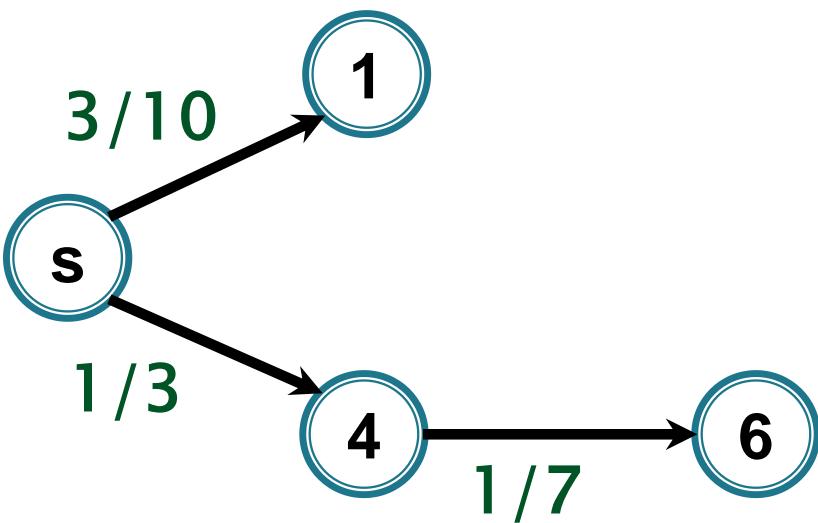
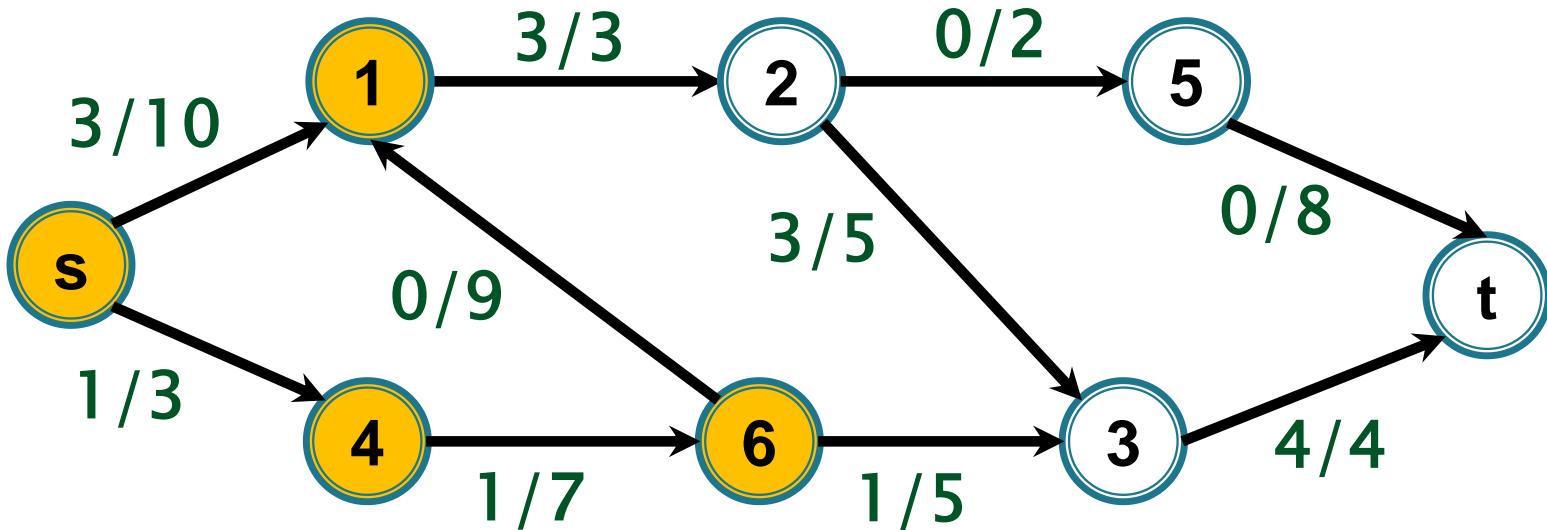


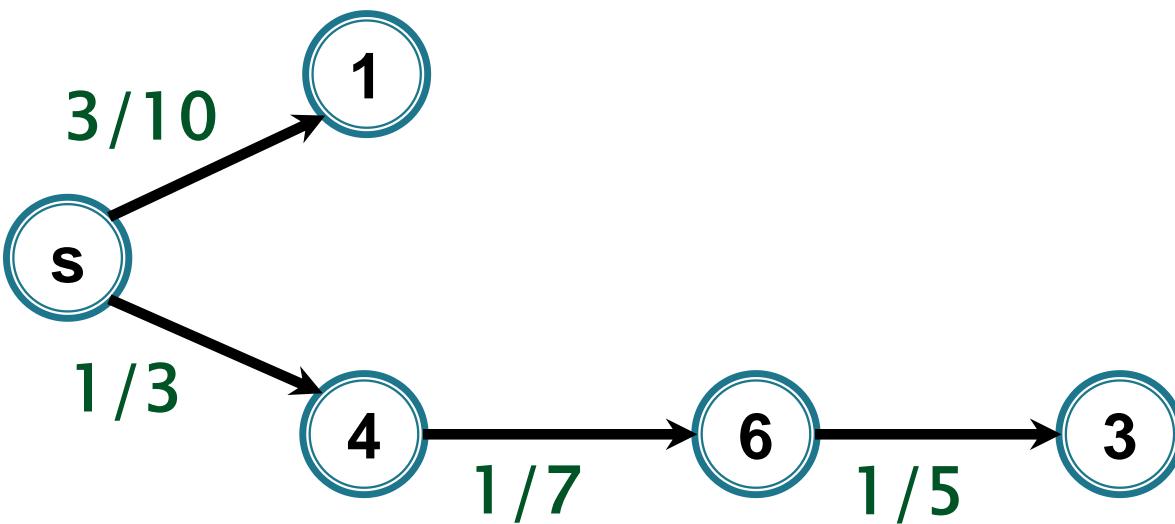
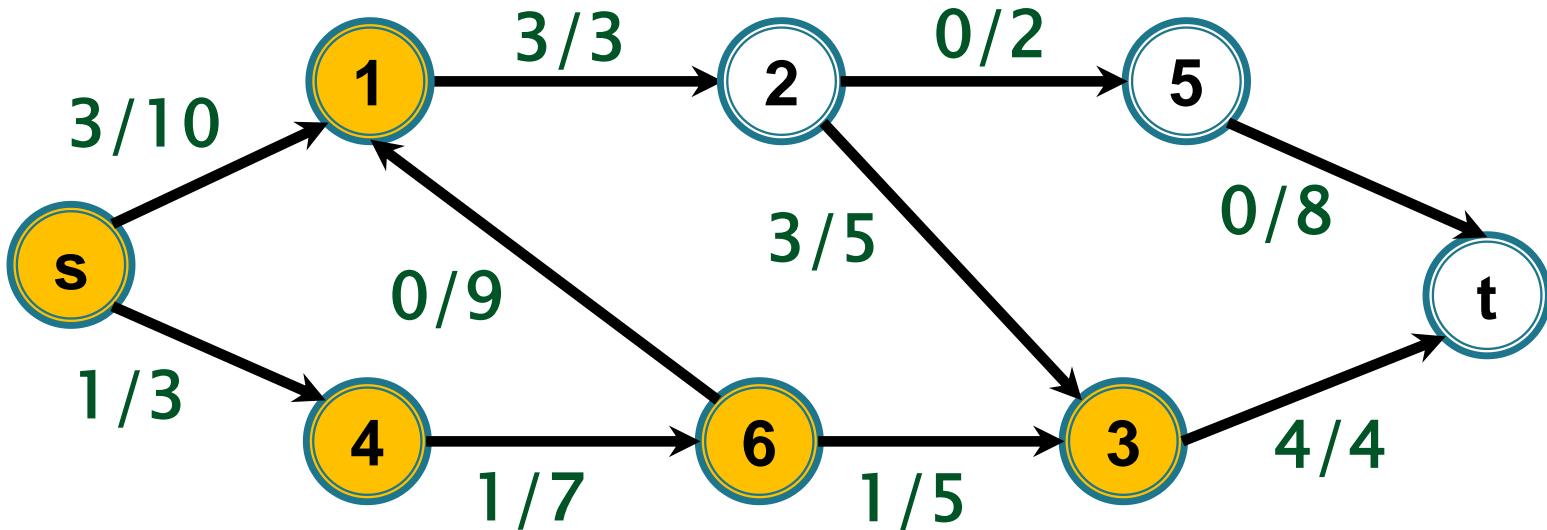


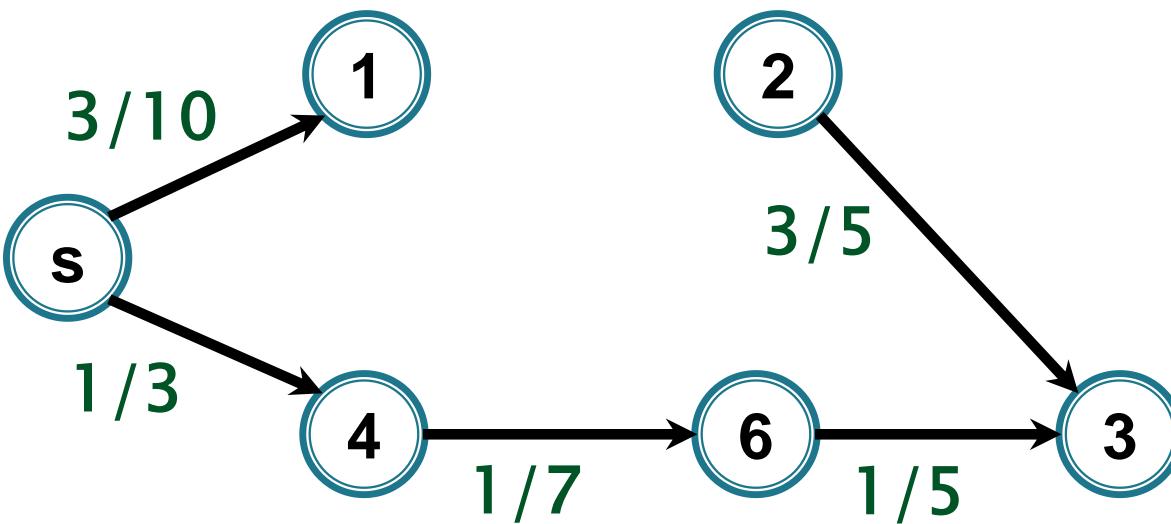
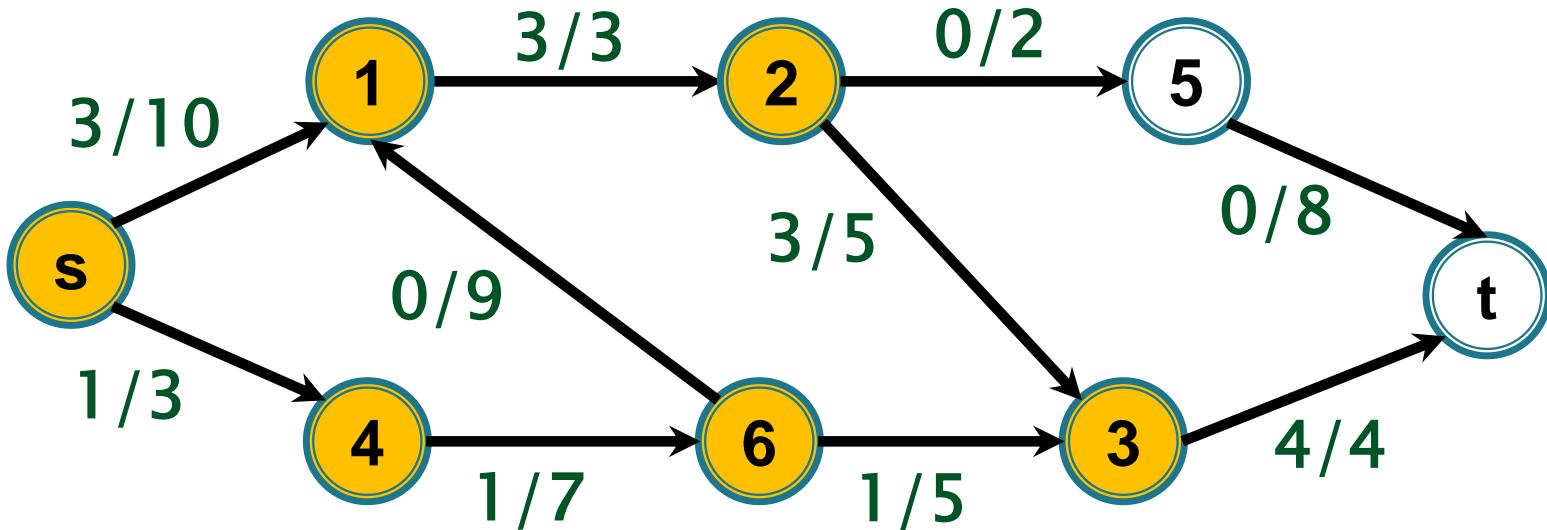


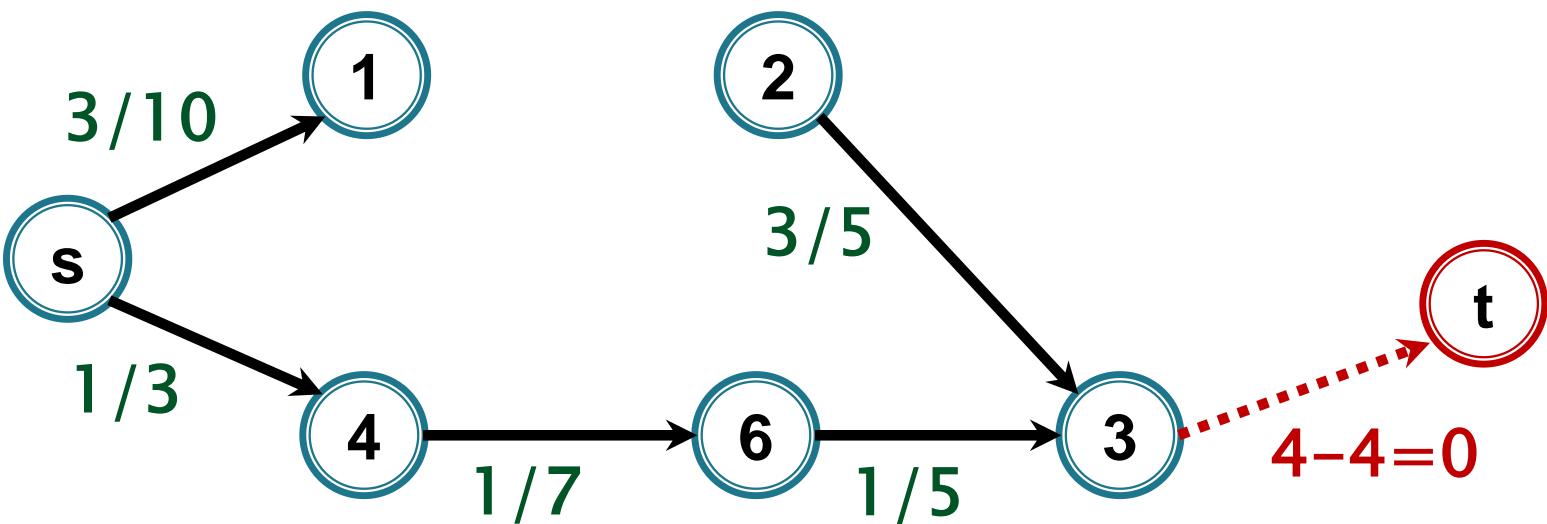
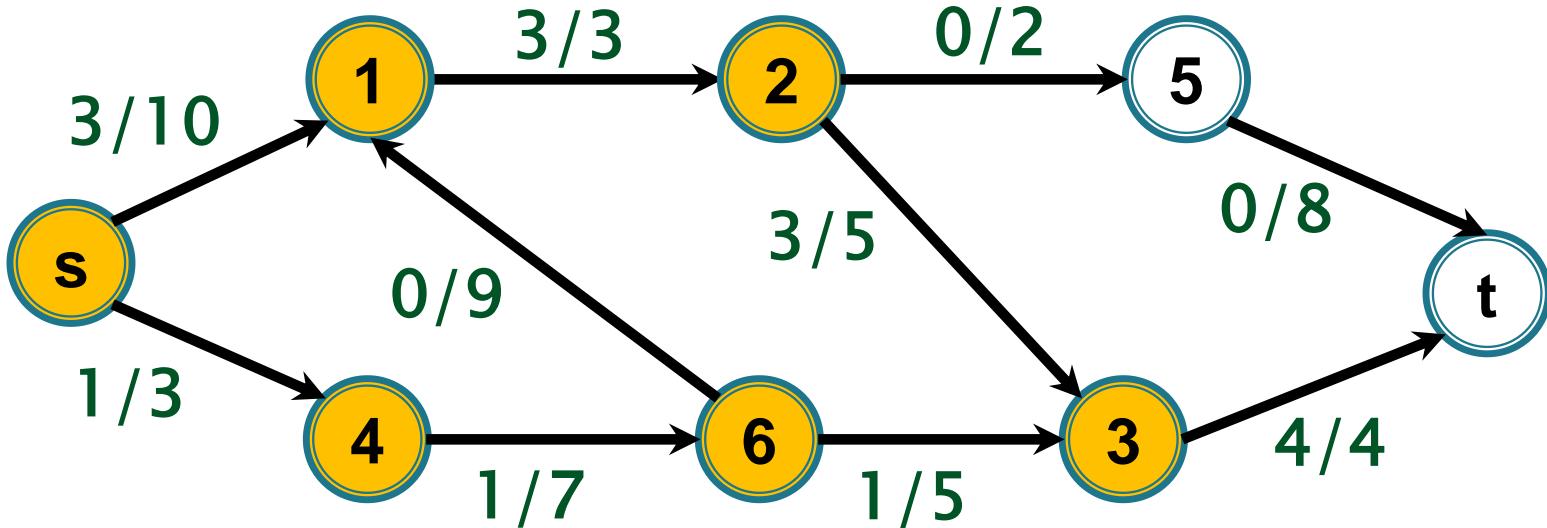


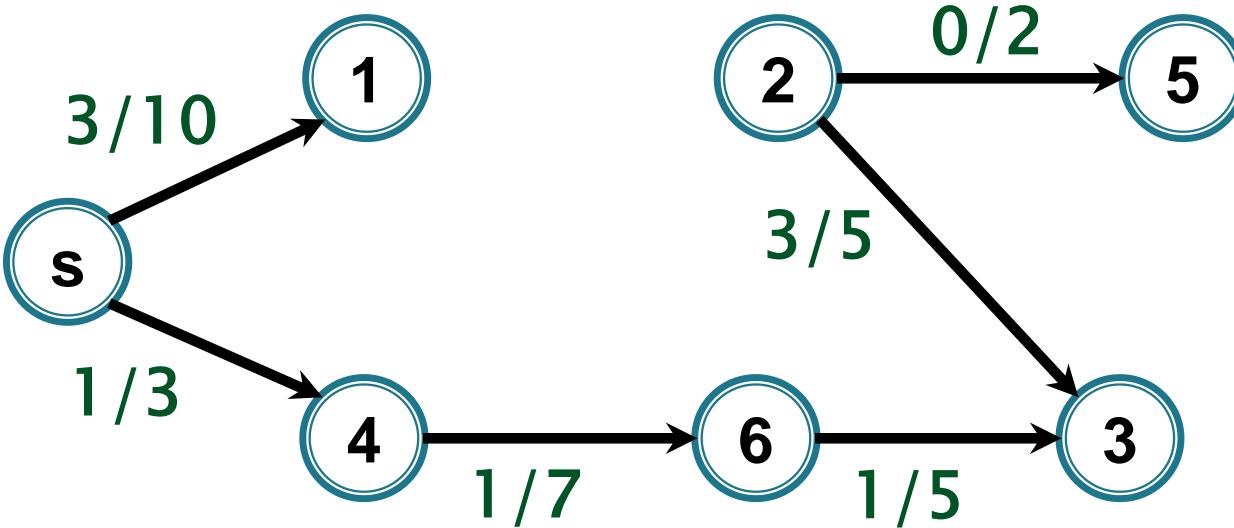
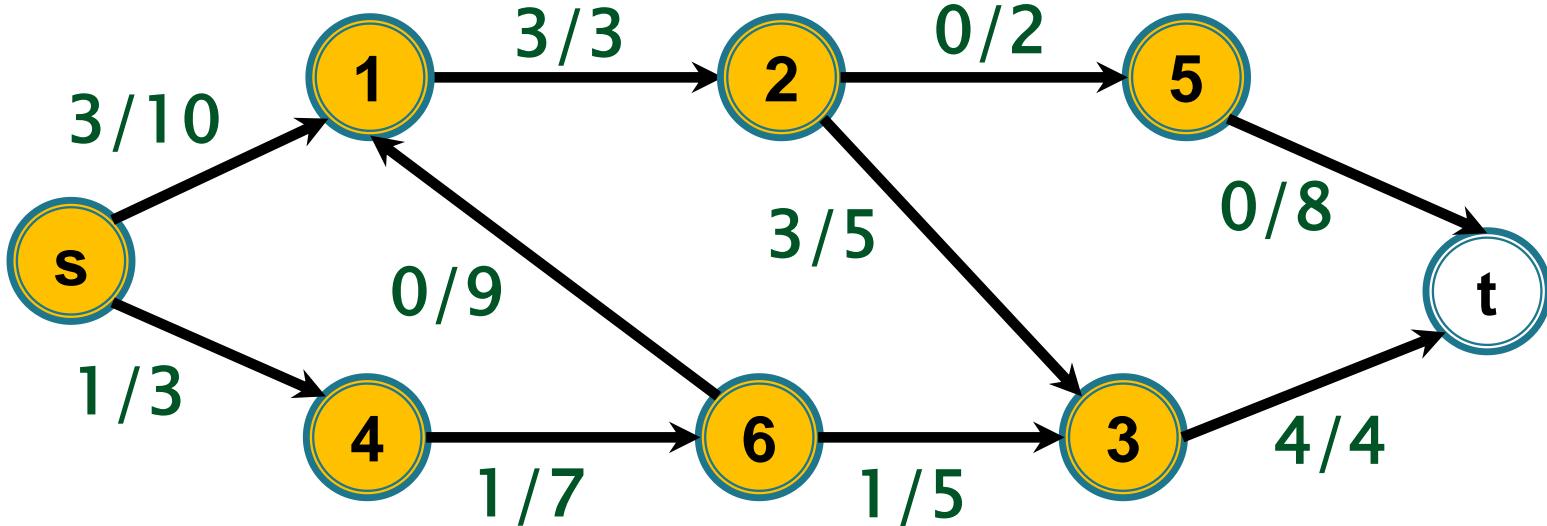


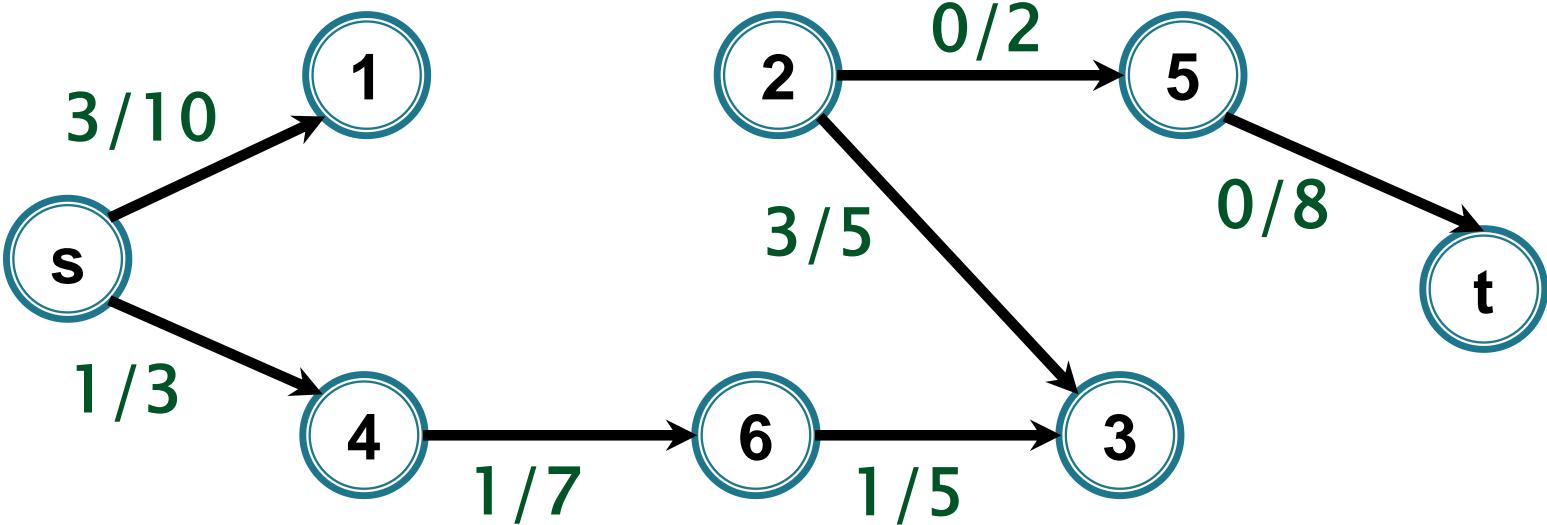
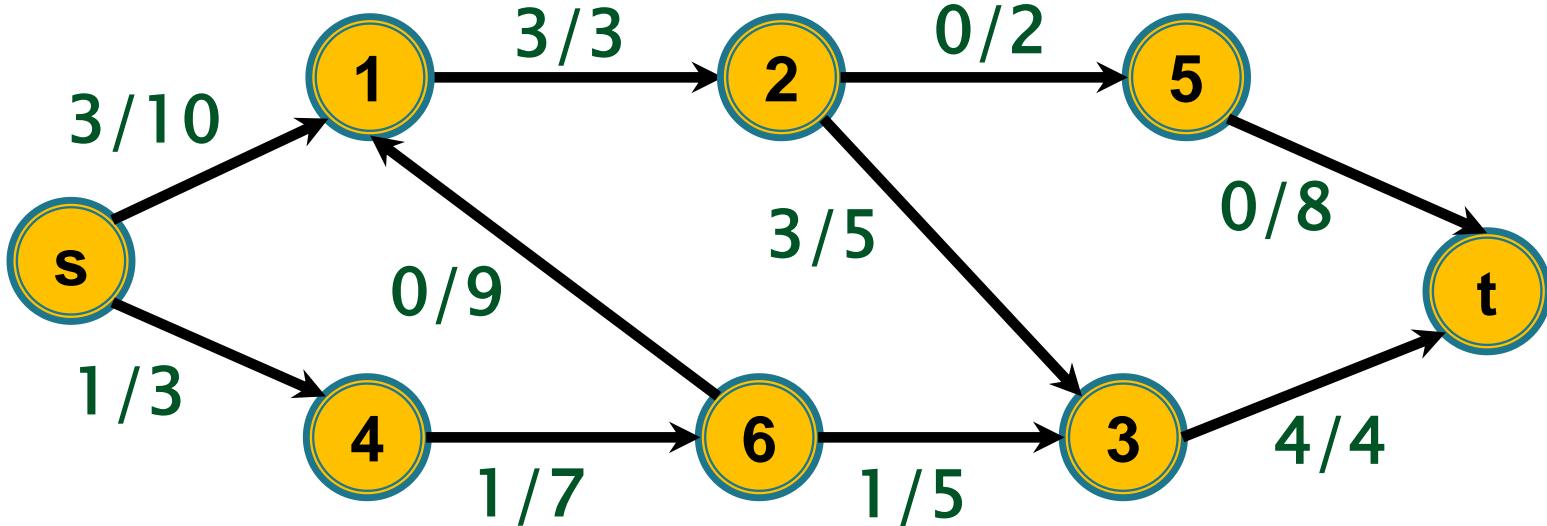




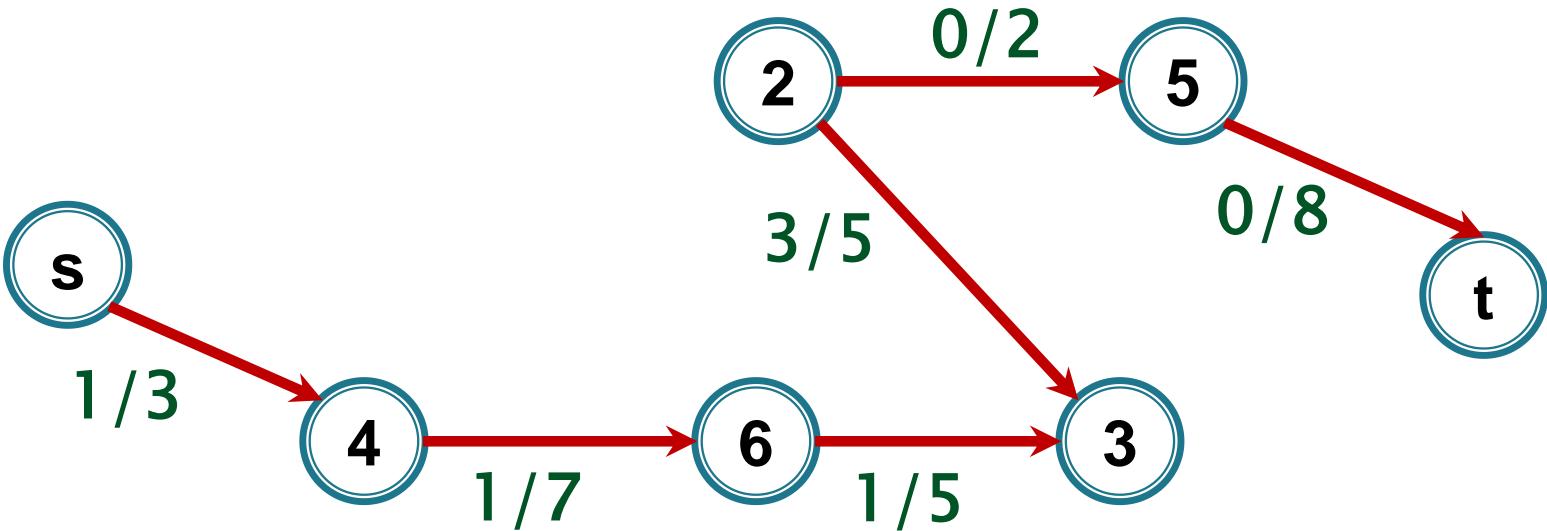
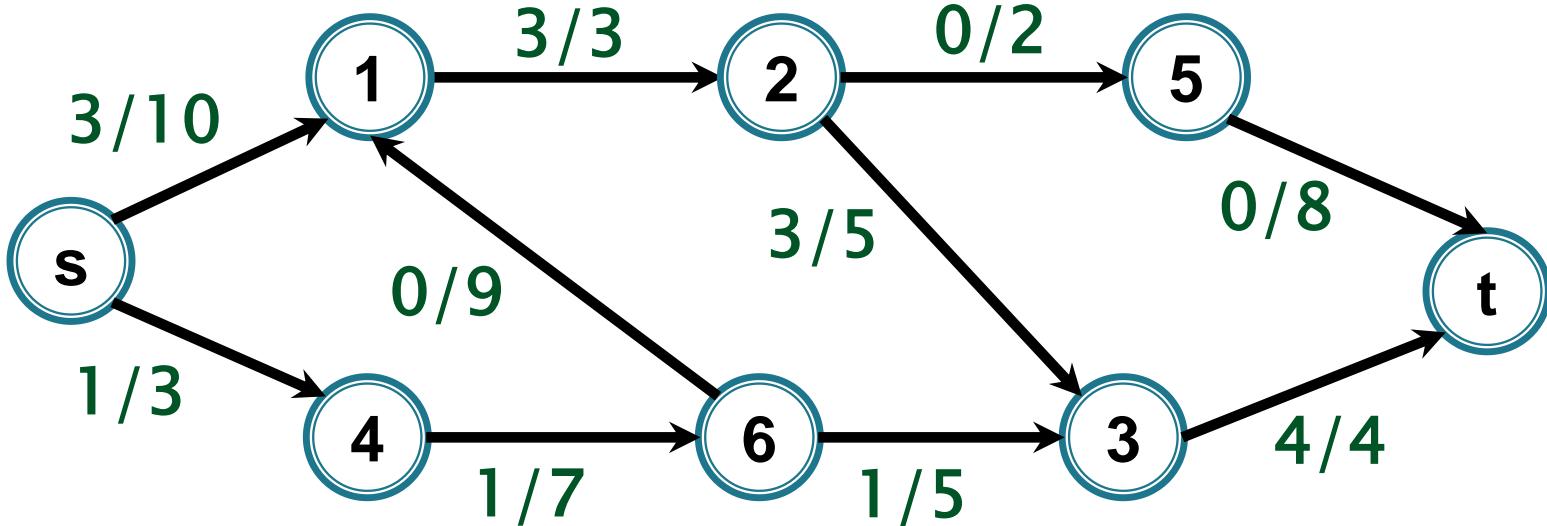


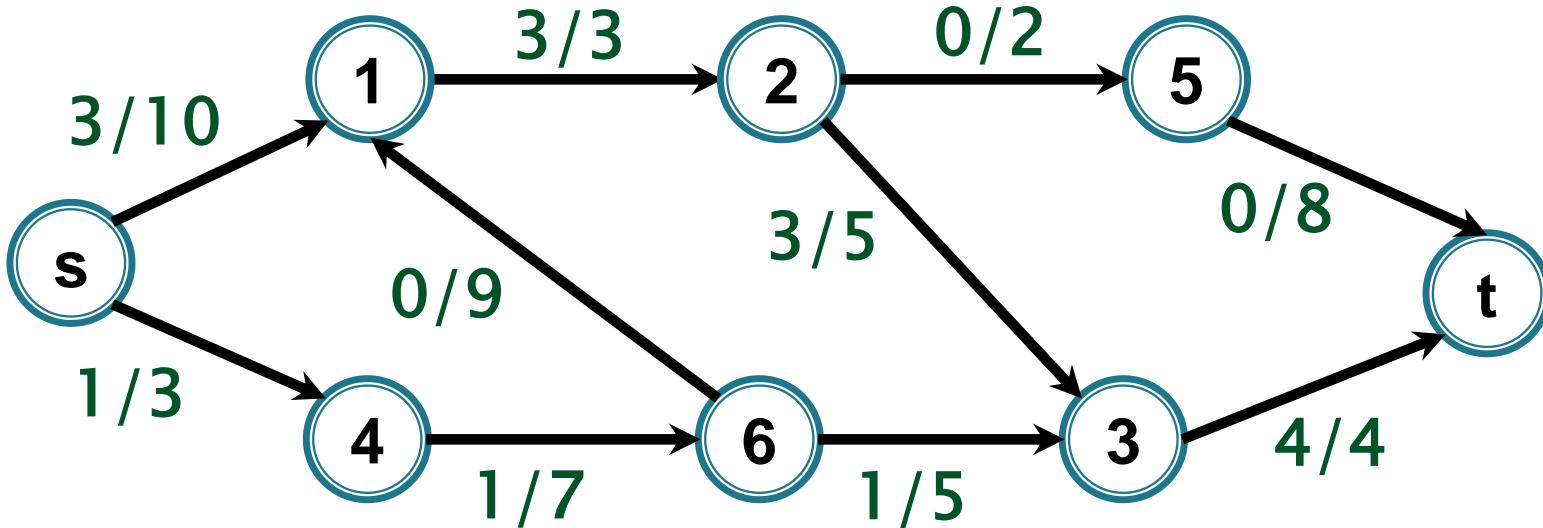




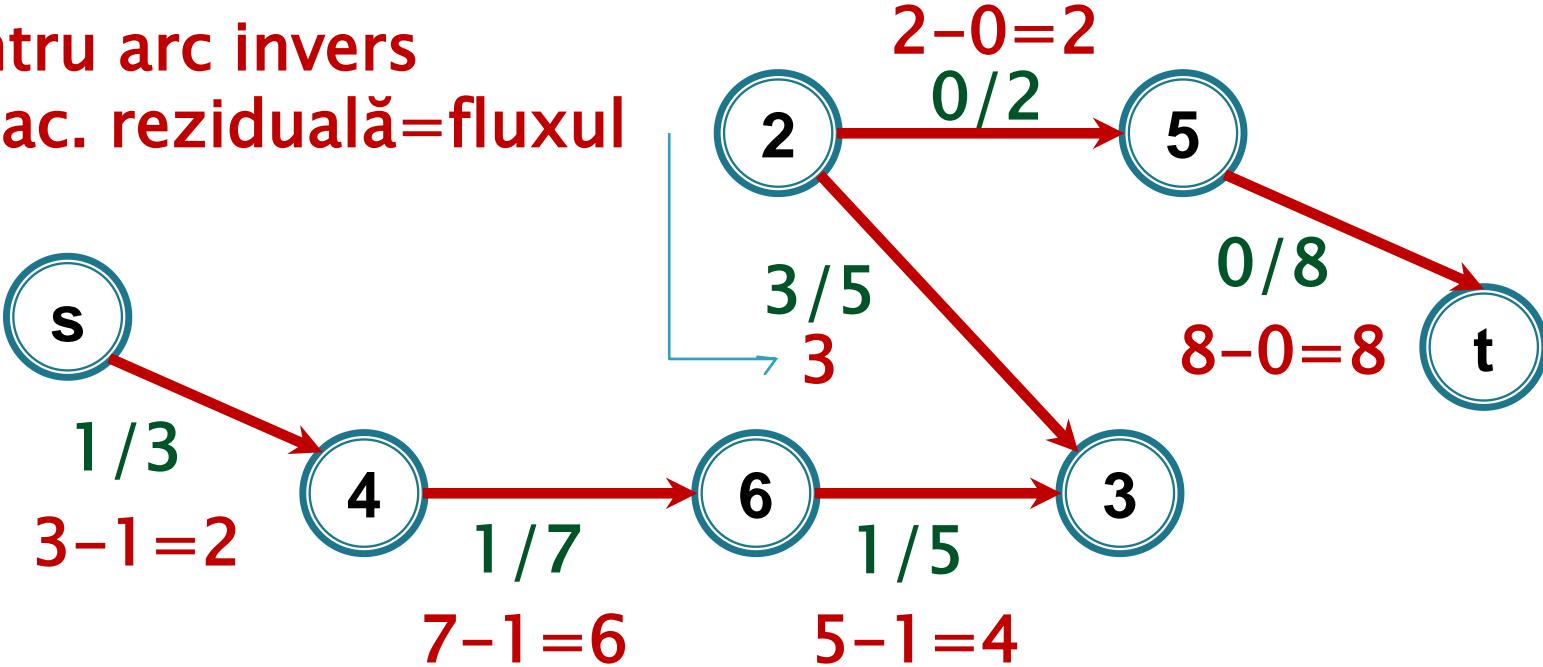


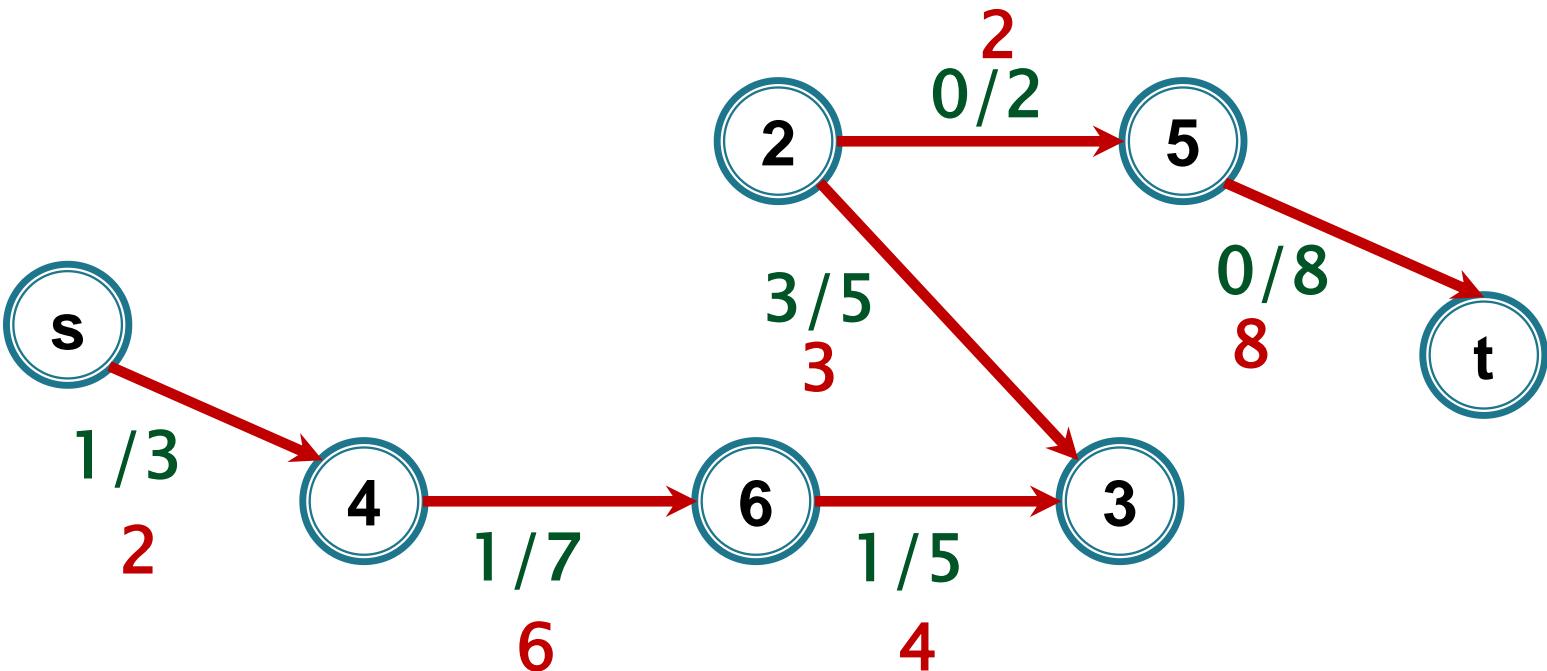
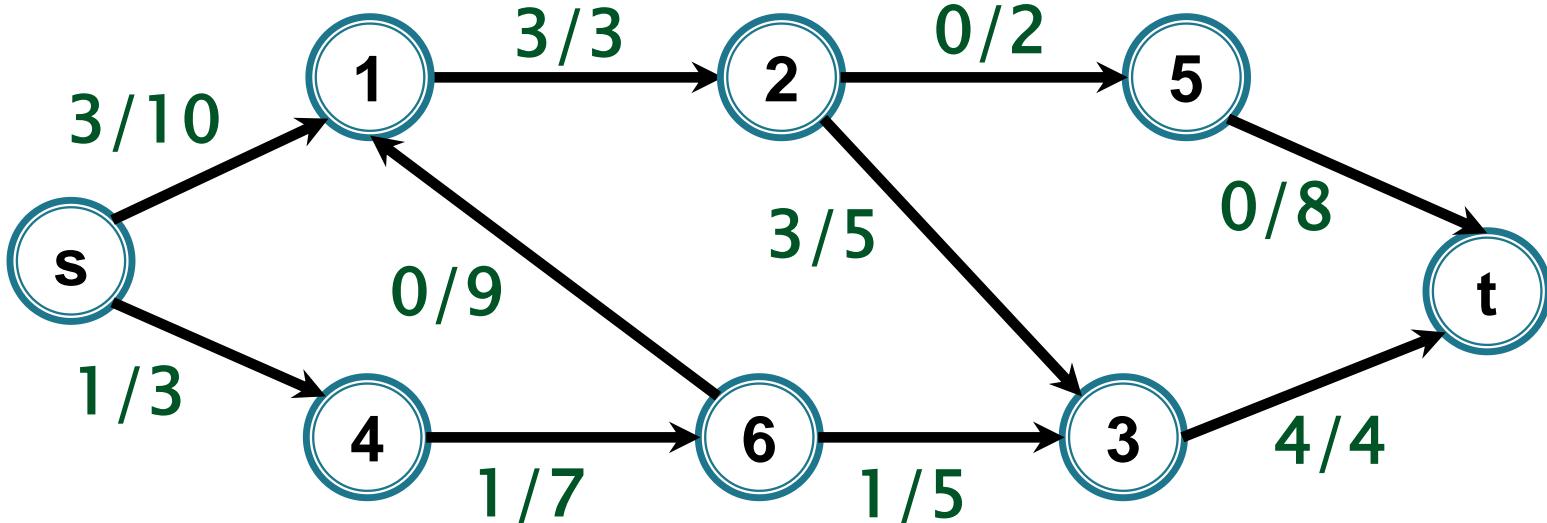
**revizuieste\_flux\_lant**



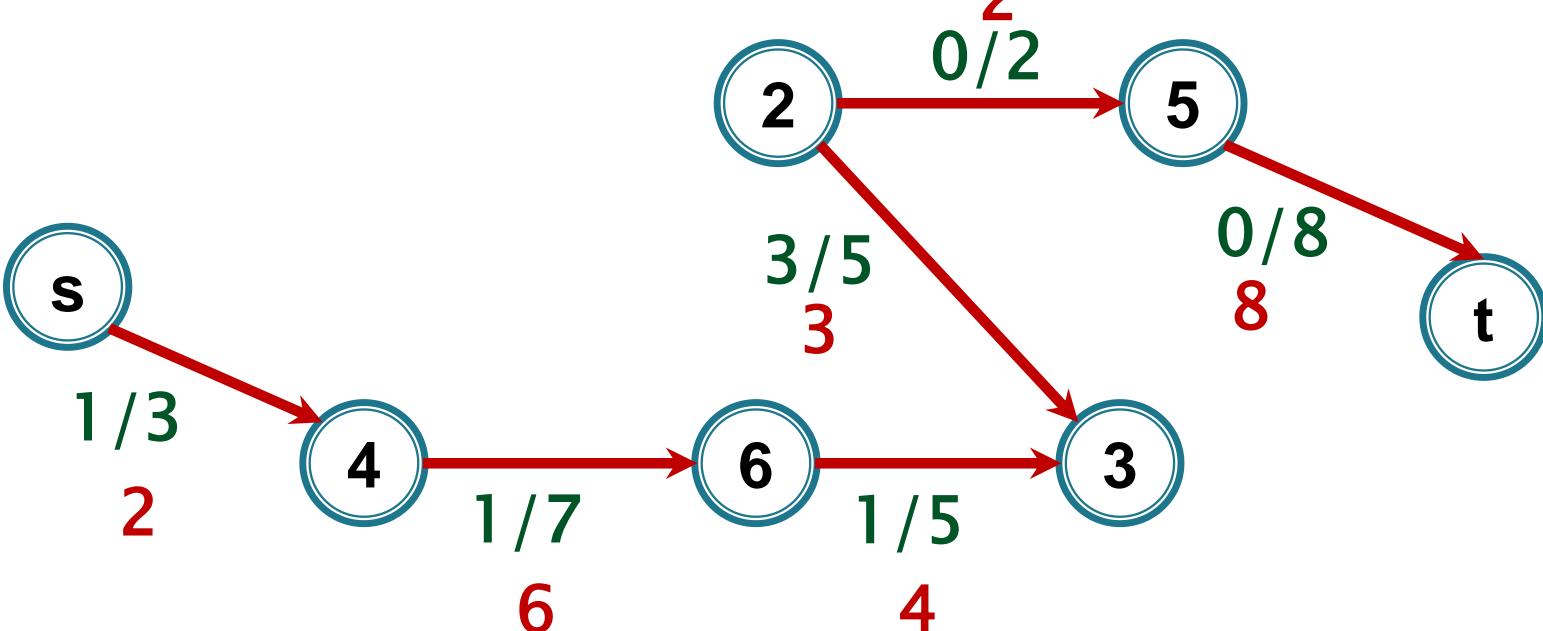
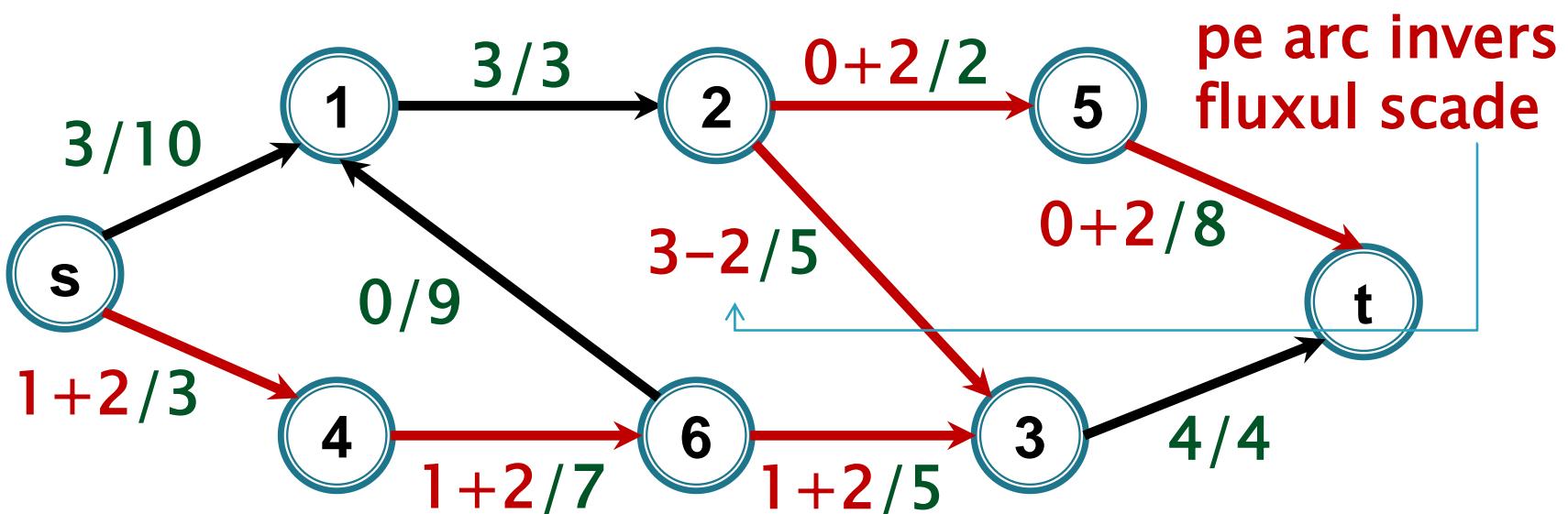


pentru arc invers  
capac. reziduală=fluxul



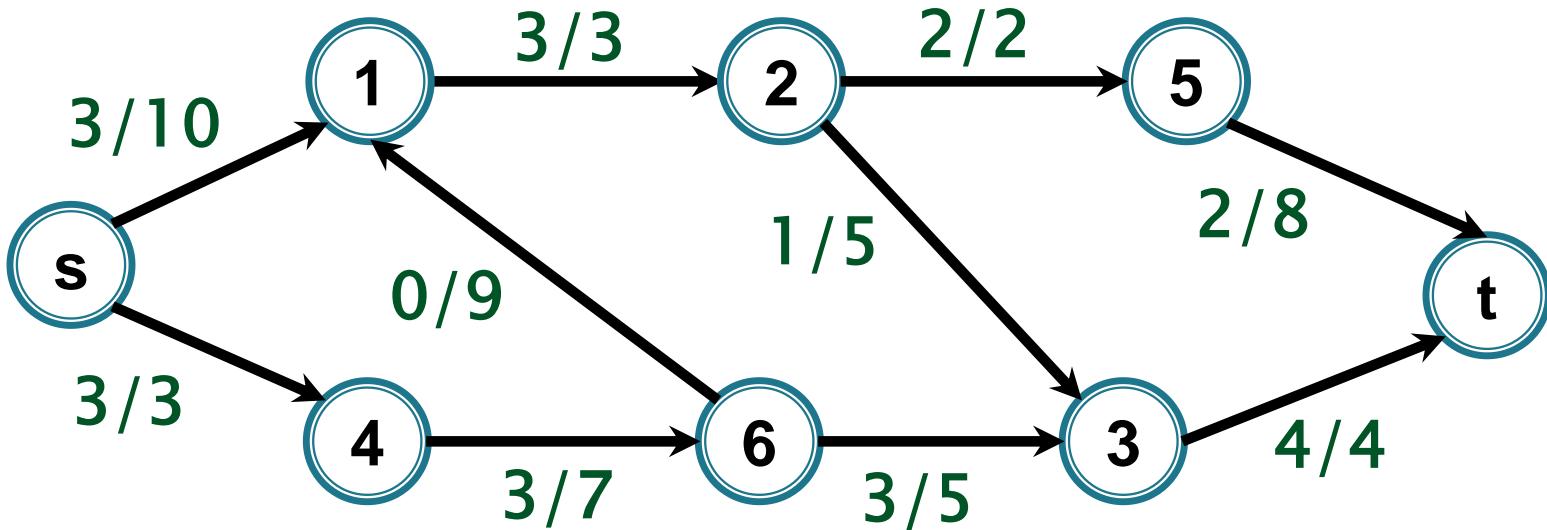


$$i(P) = \min\{ 2, 6, 4, 3, 2, 8\} = 2$$

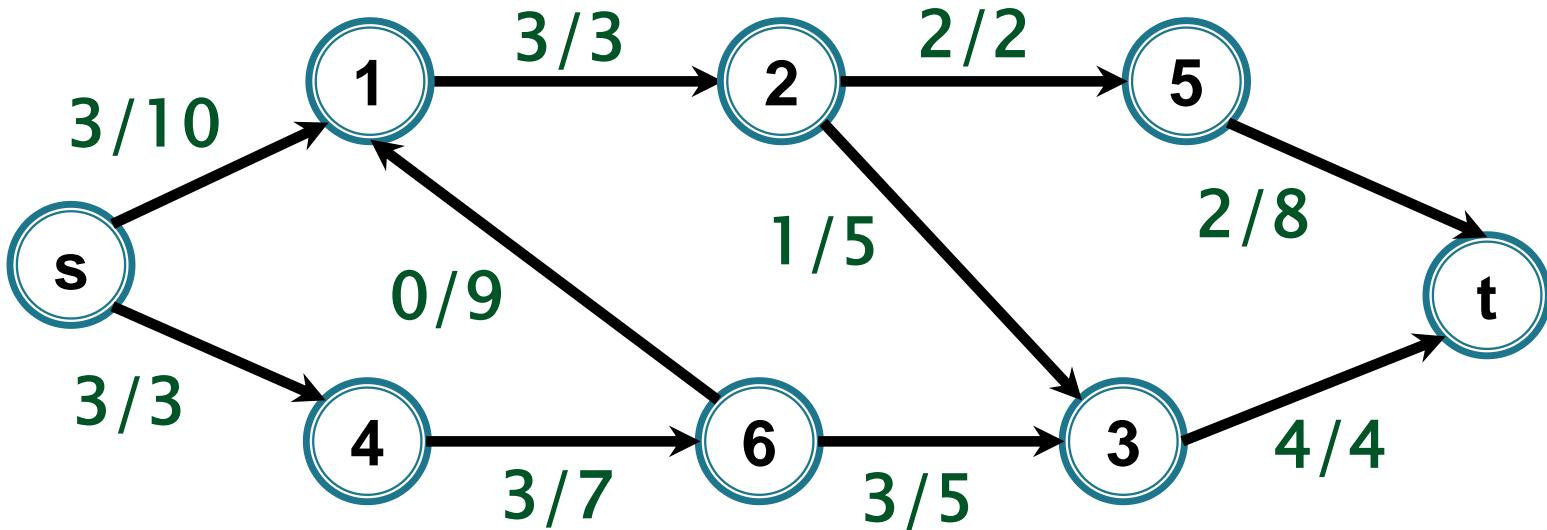


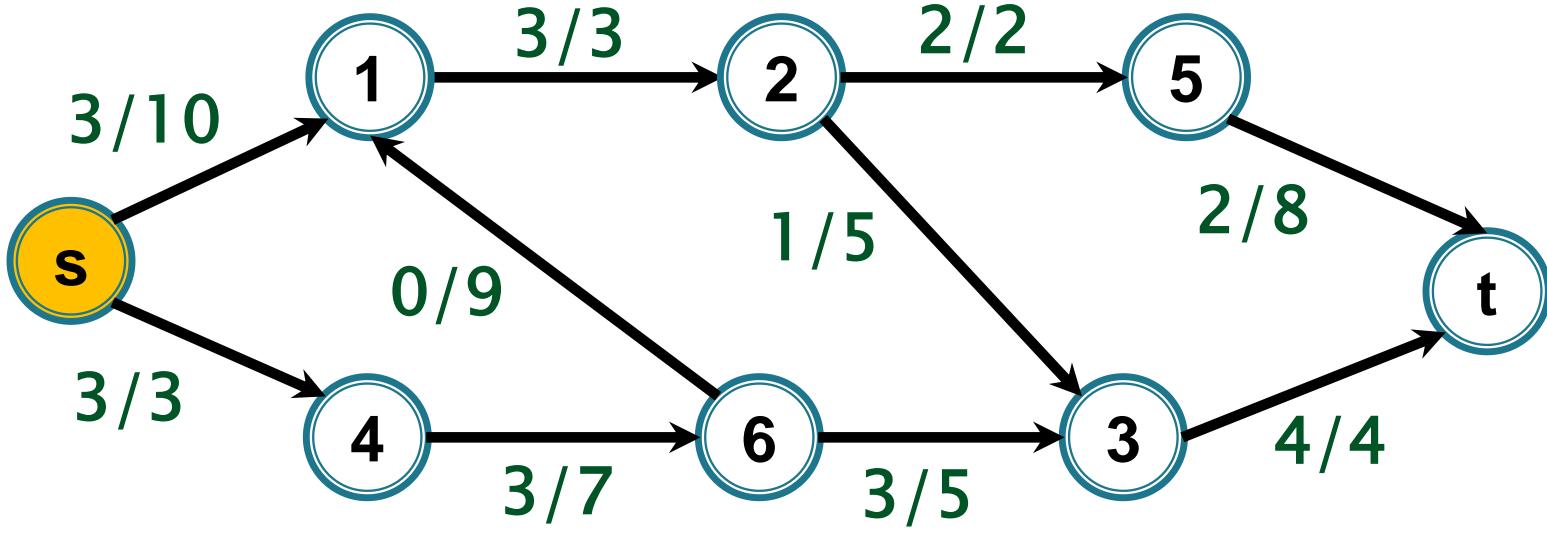
$$i(P) = \min\{ 2, 6, 4, 3, 2, 8 \} = 2$$

pe arc invers  
fluxul scade

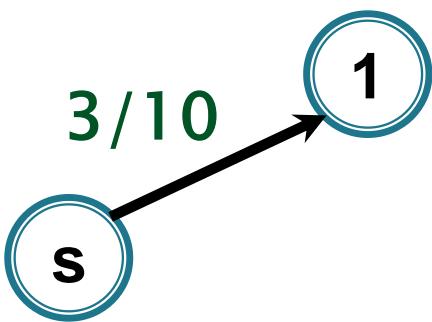
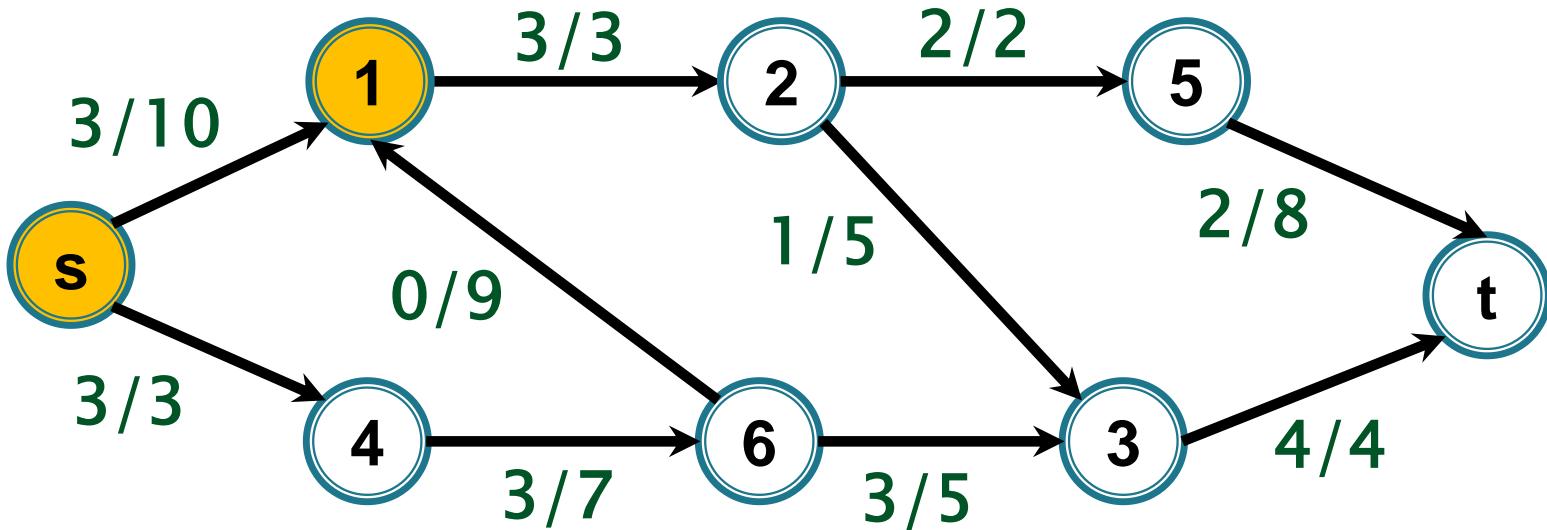


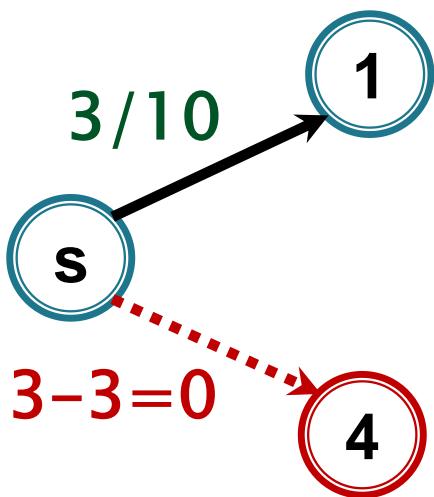
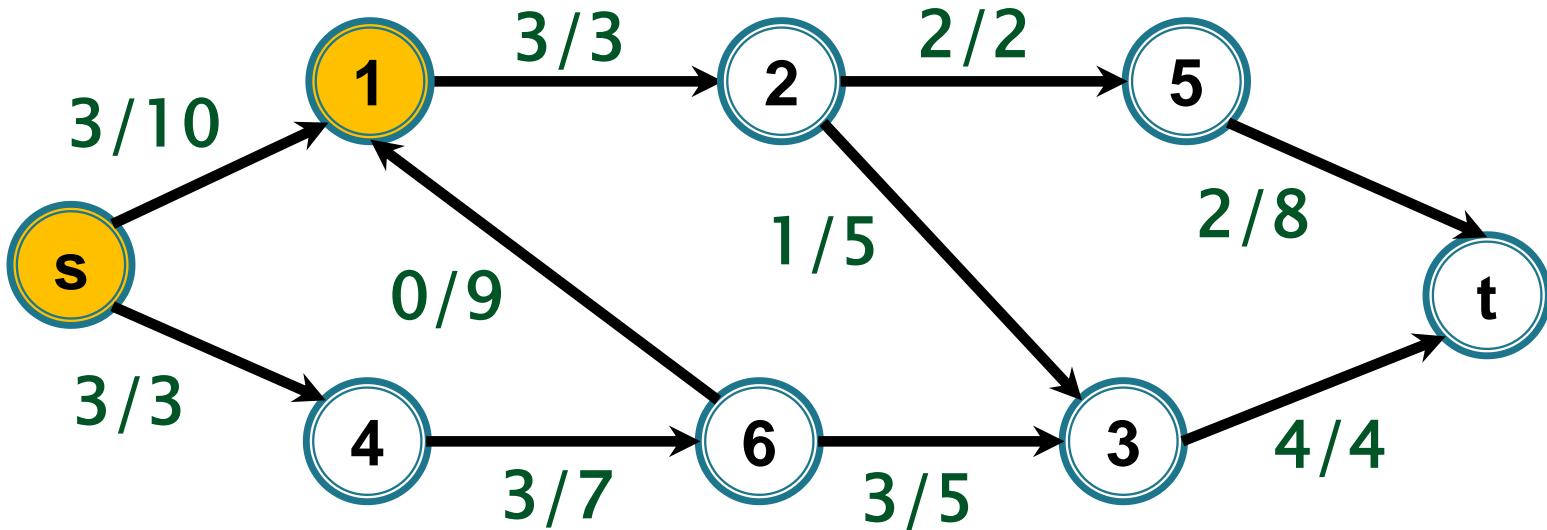
**construieste\_s-t\_lant\_nesat\_BF**

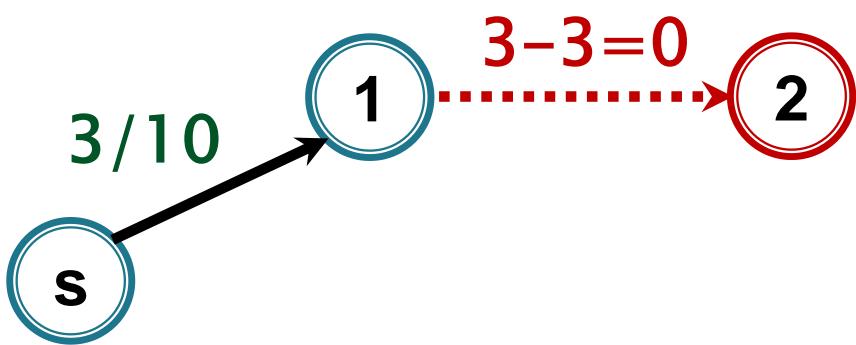
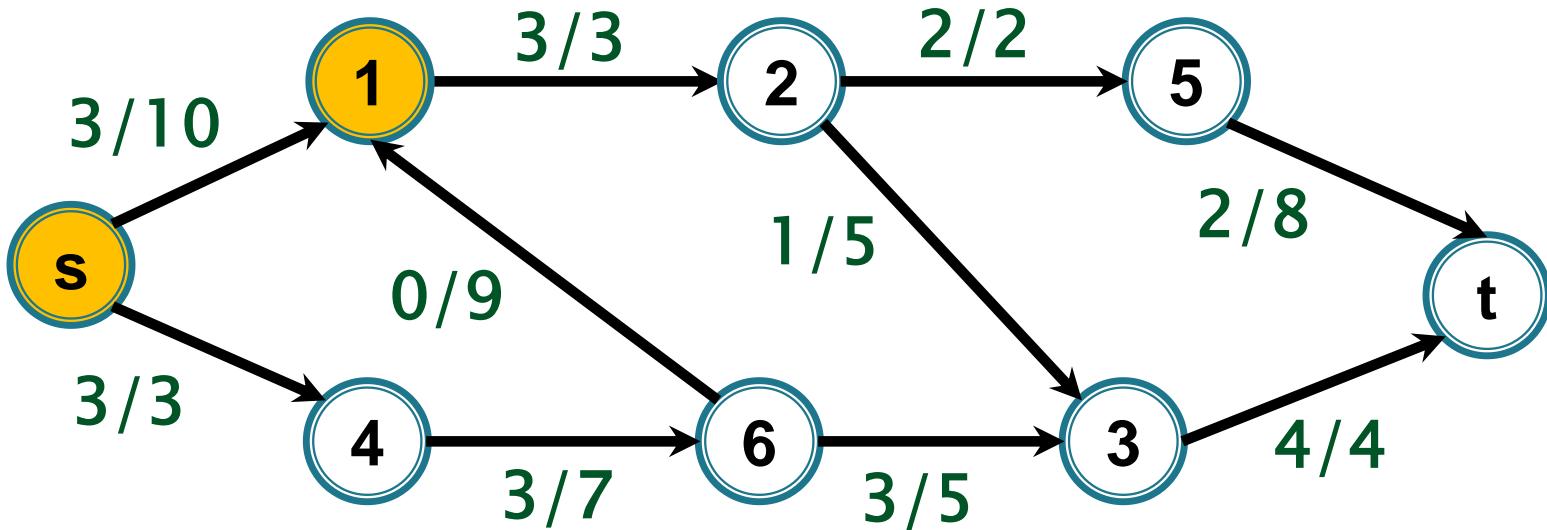


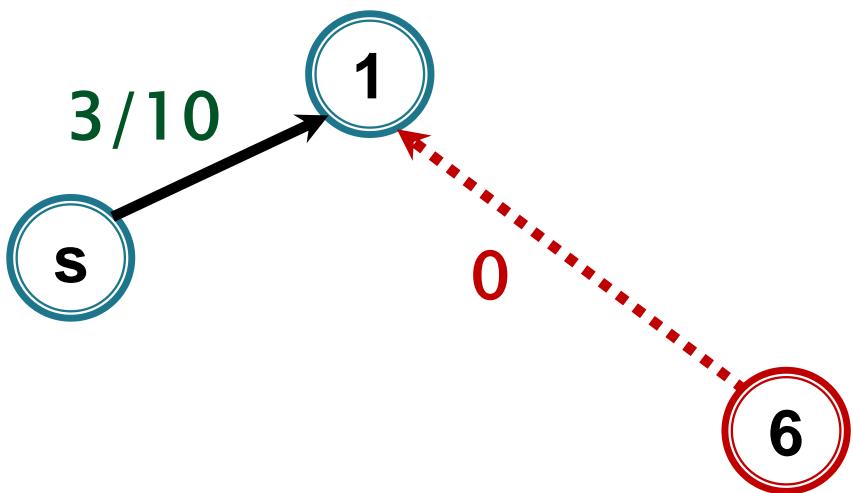
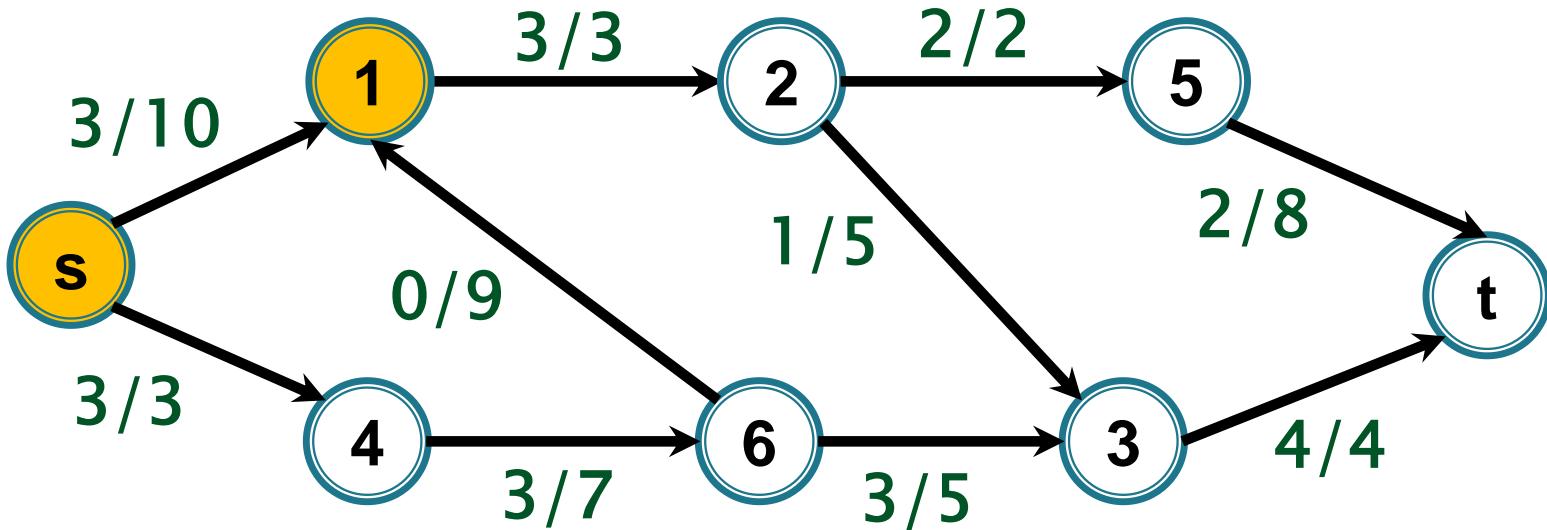


$s$









t nu este accesibil din s  $\Rightarrow$  STOP

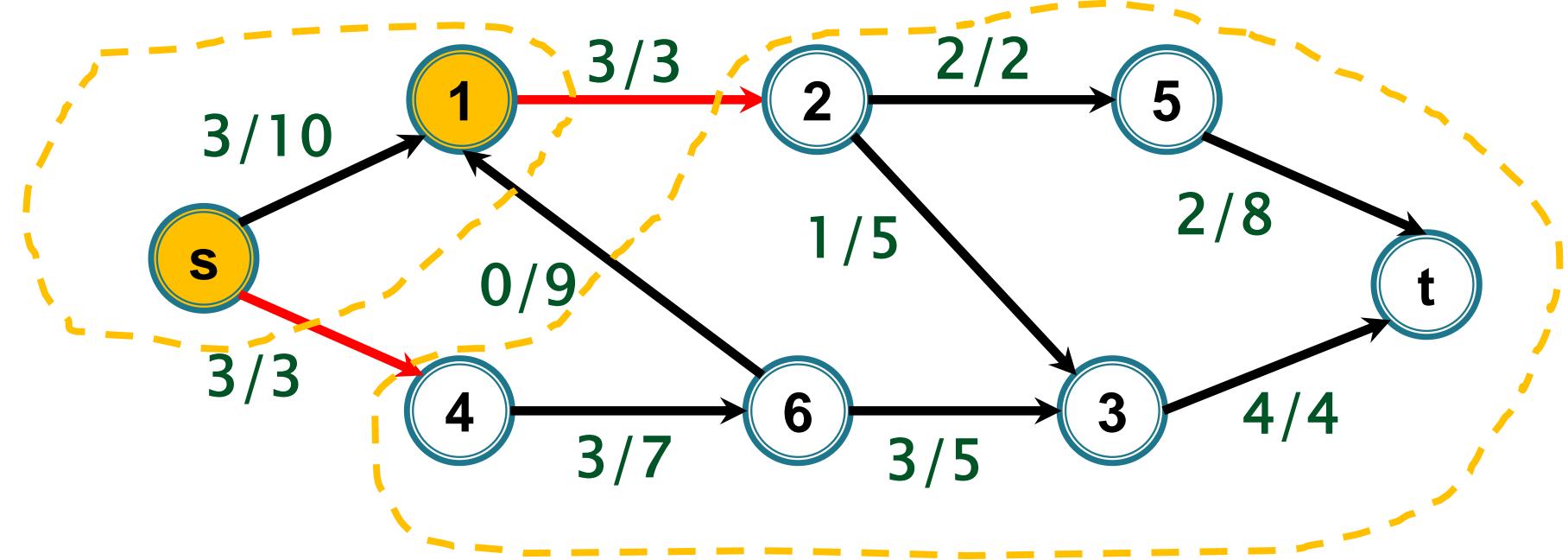
- f este flux maxim

$t$  nu este accesibil din  $s \Rightarrow \text{STOP}$

- $f$  este flux maxim
- tăietura determinată de vârfurile accesibile din  $s$  la ultimul pas prin lanțuri  $f$ -nesaturate este tăietură minimă (= din **vârfurile vizitate la ultimul pas**)

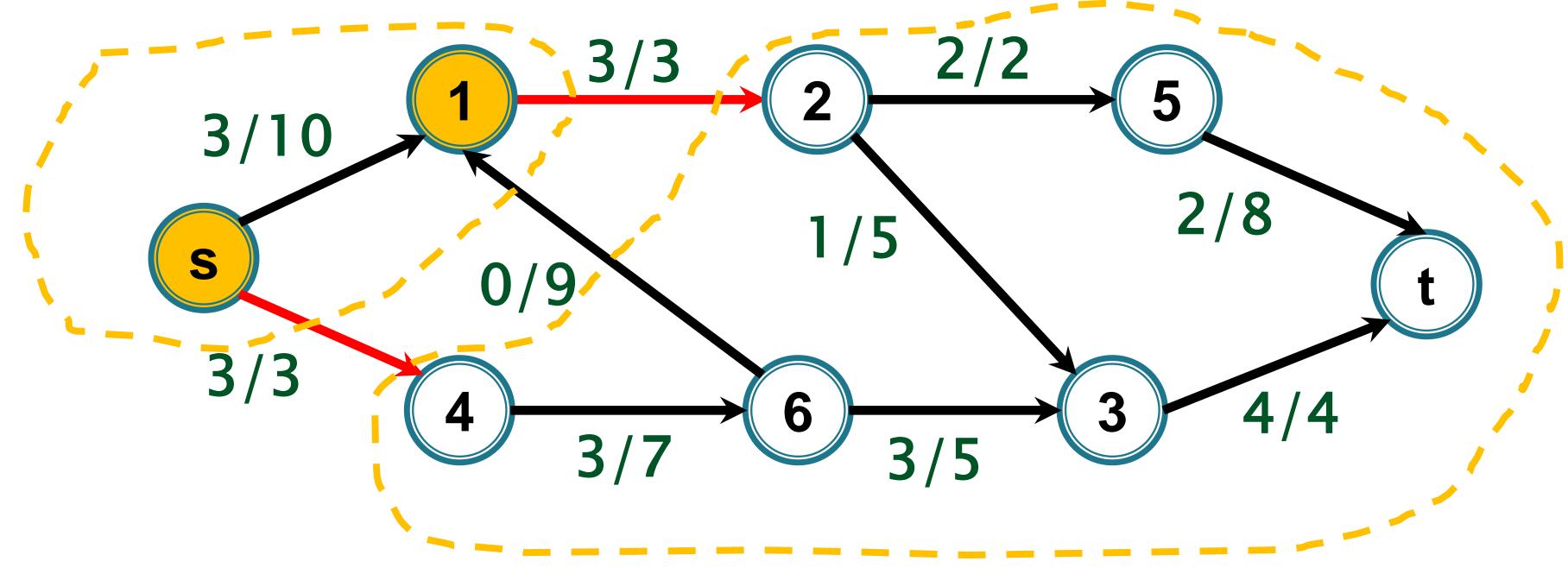
(vom demonstra !!!





Tăietură minima  $K = (\{s, 1\}, \{2, 3, 4, 5, 6, t\})$

$$c(K) = ?$$



Tăietură minima  $K = (\{s, 1\}, \{2, 3, 4, 5, 6, t\})$

$$c(K) = c(1, 2) + c(s, 4) = 3 + 3 = 6$$

= valoare flux maxim

# **Sugestii de implementare**

## **Algoritmul EDMONDS-KARP**

# Implementare

- ▶ Memorăm lanțurile (arborele BF) folosind vector **tata**
- ▶ Convenție – pentru arcele inverse  $(i,j)$  ținem minte tatăl cu semnul minus  
**tata[j] = -i**

`construye_s-t_lant_nesat_BF()`

```
construieste_s-t_lant_nesat_BF()
pentru(v∈V) executa tata[v] ←0; viz[v] ←0
```

```
construieste_s-t_lant_nesat_BF()
    pentru(v∈V) executa tata[v] ←0; viz[v] ←0
    coada C ← ∅
    adauga(s, C)
    viz[s]← 1
```

```
construieste_s-t_lant_nesat_BF()
    pentru ( $v \in V$ ) executa  $tata[v] \leftarrow 0$ ;  $viz[v] \leftarrow 0$ 
    coada C  $\leftarrow \emptyset$ 
    adauga (s, C)
    viz[s]  $\leftarrow 1$ 
    cat timp  $C \neq \emptyset$  executa
        i  $\leftarrow$  extrage (C)
```

```
construieste_s-t_lant_nesat_BF()
    pentru (v ∈ V) executa tata[v] ← 0; viz[v] ← 0
    coada C ← ∅
    adauga(s, C)
    viz[s] ← 1
    cat timp C ≠ ∅ executa
        i ← extrage(C)
        pentru (ij ∈ E) executa arc direct
            dacă (viz[j]=0 și c(ij)-f(ij)>0) atunci
```

```
construieste_s-t_lant_nesat_BF()
    pentru (v ∈ V) executa tata[v] ← 0; viz[v] ← 0
    coada C ← ∅
    adauga(s, C)
    viz[s] ← 1
    cat timp C ≠ ∅ executa
        i ← extrage(C)
        pentru (ij ∈ E) executa arc direct
            dacă (viz[j]=0 și c(ij)-f(ij)>0) atunci
                adauga(j, C)
                viz[j] ← 1; tata[j] ← i
```

```

construieste_s-t_lant_nesat_BF()
    pentru (v ∈ V) executa tata[v] ← 0; viz[v] ← 0
    coada C ← ∅
    adauga(s, C)
    viz[s] ← 1
    cat timp C ≠ ∅ executa
        i ← extrage(C)
        pentru (ij ∈ E) executa arc direct
            dacă (viz[j]=0 și c(ij)-f(ij)>0) atunci
                adauga(j, C)
                viz[j] ← 1; tata[j] ← i
            daca (j=t) atunci STOP și returnează true(1)

```

```

construieste_s-t_lant_nesat_BF()
    pentru (v ∈ V) executa tata[v] ← 0; viz[v] ← 0
    coada C ← ∅
    adauga(s, C)
    viz[s] ← 1
    cat timp C ≠ ∅ executa
        i ← extrage(C)
        pentru (ij ∈ E) executa arc direct
            dacă (viz[j]=0 și c(ij)-f(ij)>0) atunci
                adauga(j, C)
                viz[j] ← 1; tata[j] ← i
            daca (j=t) atunci STOP și returnează true(1)
        pentru (ji ∈ E) executa arc invers

```

```

construieste_s-t_lant_nesat_BF()
    pentru (v ∈ V) executa tata[v] ← 0; viz[v] ← 0
    coada C ← ∅
    adauga(s, C)
    viz[s] ← 1
    cat timp C ≠ ∅ executa
        i ← extrage(C)
        pentru (ij ∈ E) executa arc direct
            dacă (viz[j]=0 și c(ij)-f(ij)>0) atunci
                adauga(j, C)
                viz[j] ← 1; tata[j] ← i
            daca (j=t) atunci STOP și returnează true(1)
        pentru (ji ∈ E) executa arc invers
            daca (viz[j]=0 și f(ji)>0) atunci

```

```

construieste_s-t_lant_nesat_BF()
    pentru (v ∈ V) executa tata[v] ← 0; viz[v] ← 0
    coada C ← ∅
    adauga(s, C)
    viz[s] ← 1
    cat timp C ≠ ∅ executa
        i ← extrage(C)
        pentru (ij ∈ E) executa arc direct
            dacă (viz[j]=0 și c(ij)-f(ij)>0) atunci
                adauga(j, C)
                viz[j] ← 1; tata[j] ← i
            daca (j=t) atunci STOP și returnează true(1)
        pentru (ji ∈ E) executa arc invers
            daca (viz[j]=0 și f(ji)>0) atunci
                adauga(j, C)
                viz[j] ← 1; tata[j] ← -i

```

```

construieste_s-t_lant_nesat_BF()
    pentru (v ∈ V) executa tata[v] ← 0; viz[v] ← 0
    coada C ← ∅
    adauga(s, C)
    viz[s] ← 1
    cat timp C ≠ ∅ executa
        i ← extrage(C)
        pentru (ij ∈ E) executa arc direct
            dacă (viz[j]=0 și c(ij)-f(ij)>0) atunci
                adauga(j, C)
                viz[j] ← 1; tata[j] ← i
            daca (j=t) atunci STOP și returnează true(1)
        pentru (ji ∈ E) executa arc invers
            daca (viz[j]=0 și f(ji)>0) atunci
                adauga(j, C)
                viz[j] ← 1; tata[j] ← -i
            daca (j=t) atunci STOP și returnează true(1)
    returnează false(0)

```

# Algoritmul Edmonds–Karp

## ► Complexitate

- Algoritm generic Ford Fulkerson  $O(mL)/O(nmC)$
- Implementare Edmonds Karp  $O(nm^2)$

# Implementare. Algoritmul Edmonds–Karp

Amintim:

a determina un s–t lanț nesaturat folosind BF în G

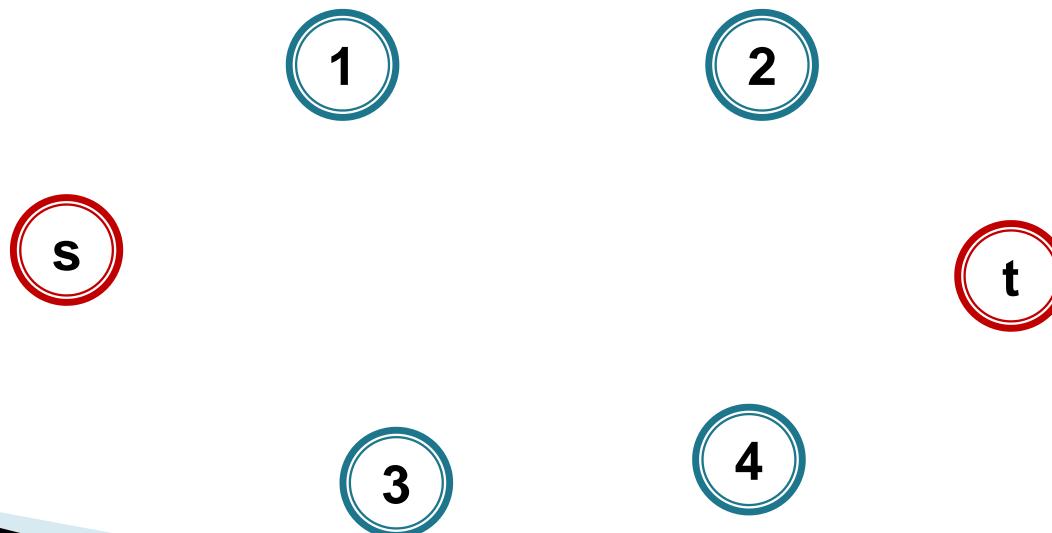
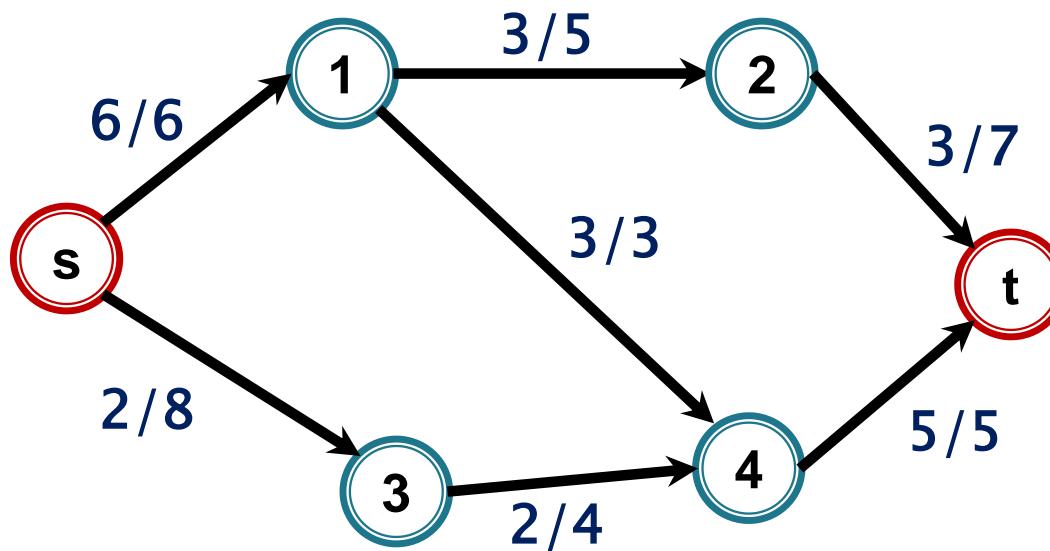
↔

a determina un s–t drum folosind BF în graful rezidual  $G_f$

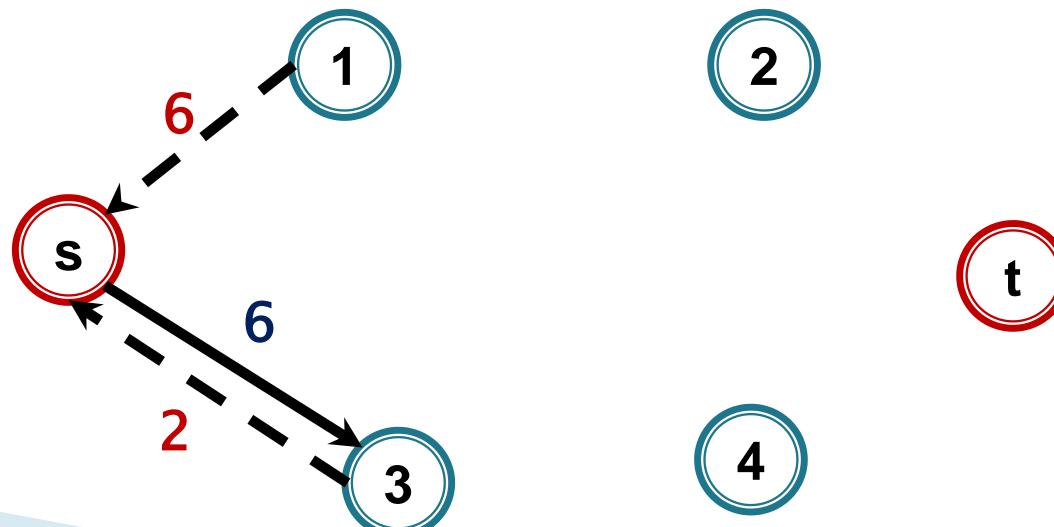
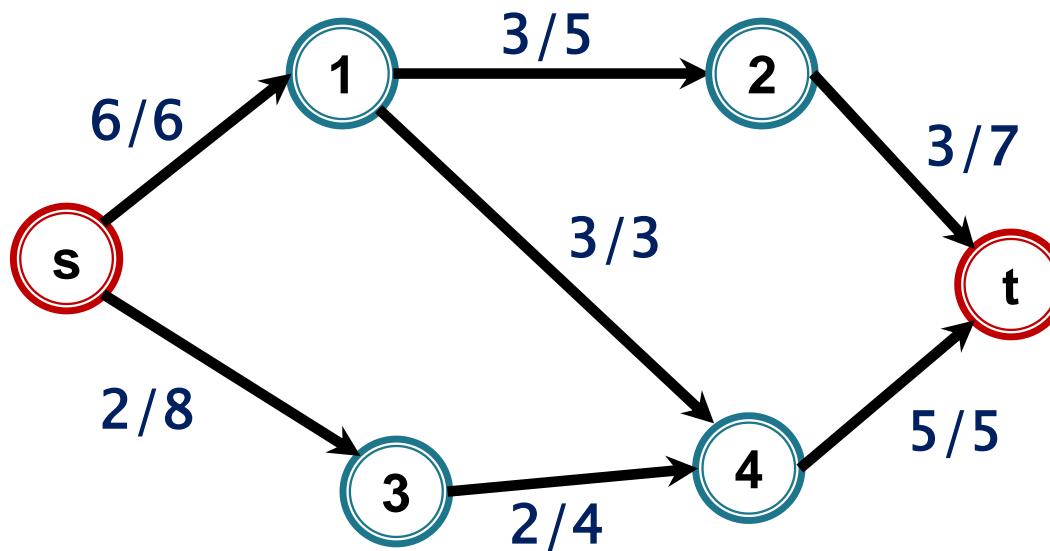
## **Varianta 2 de implementare**

**revizuirea fluxului folosind  
s-t drumuri în graful rezidual**

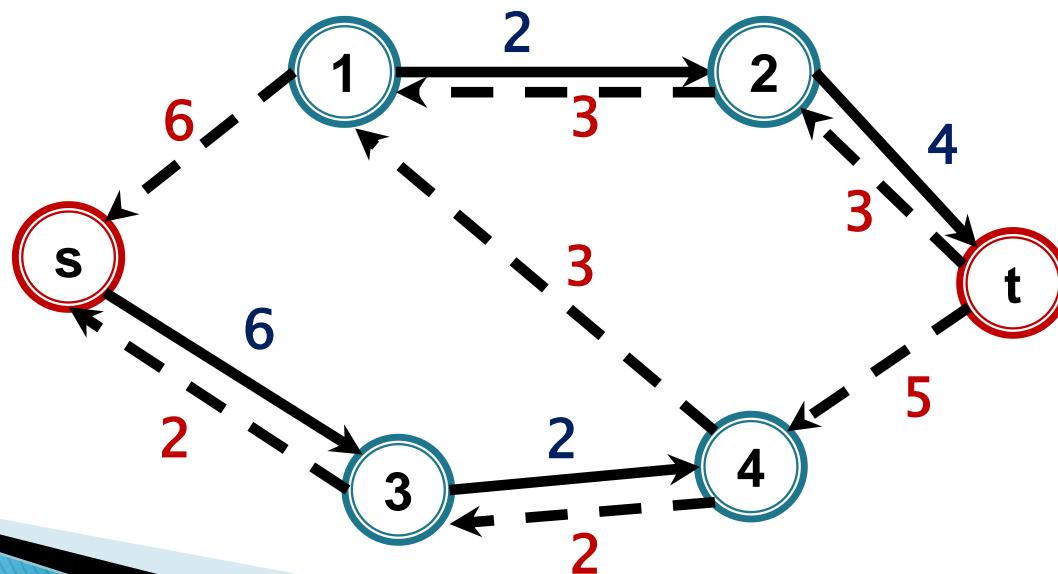
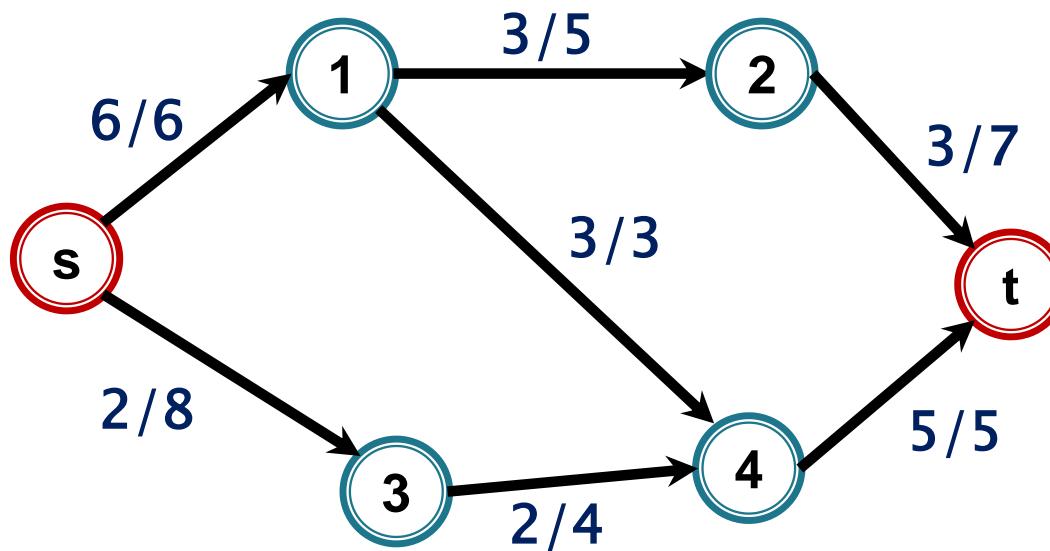
# Graf rezidual



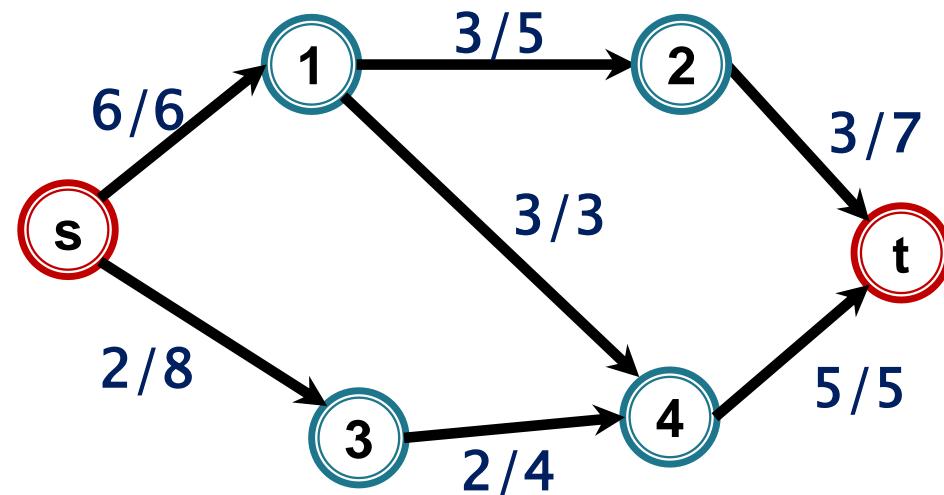
# Graf rezidual



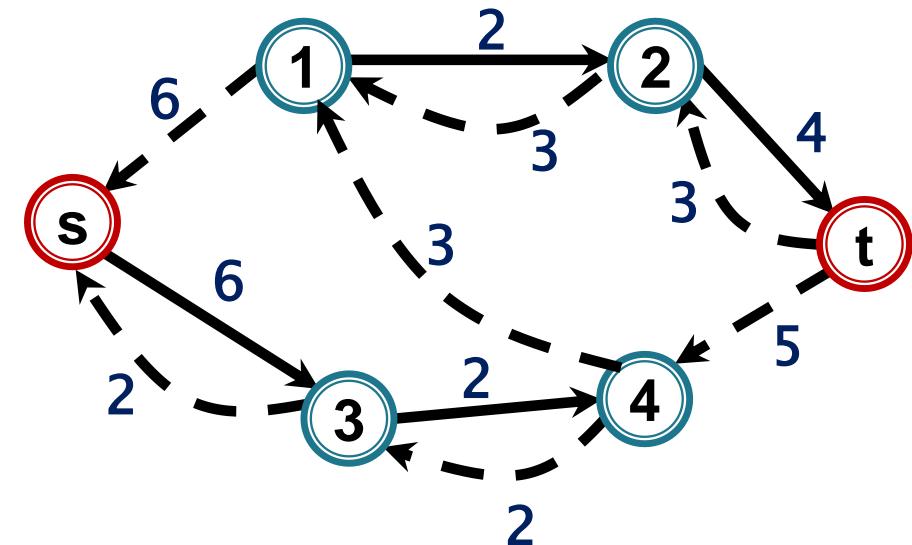
# Graf rezidual



Rețeaua de transport



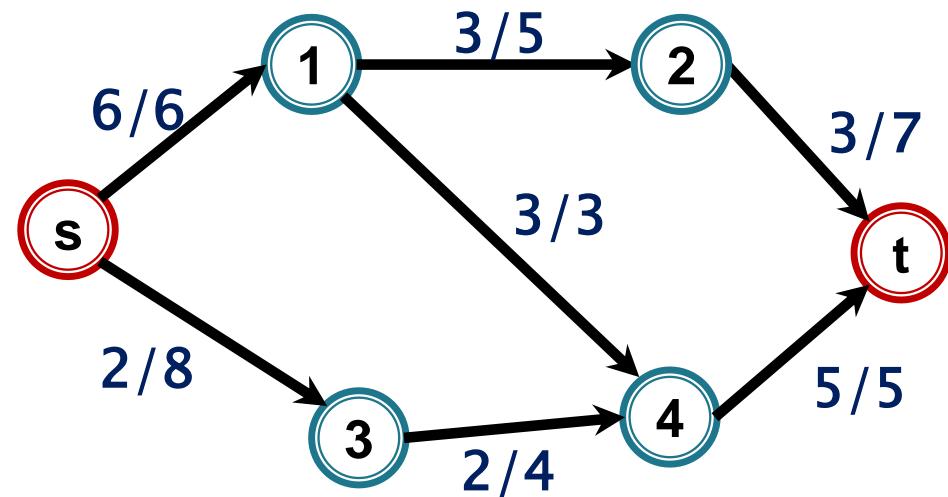
Graful rezidual



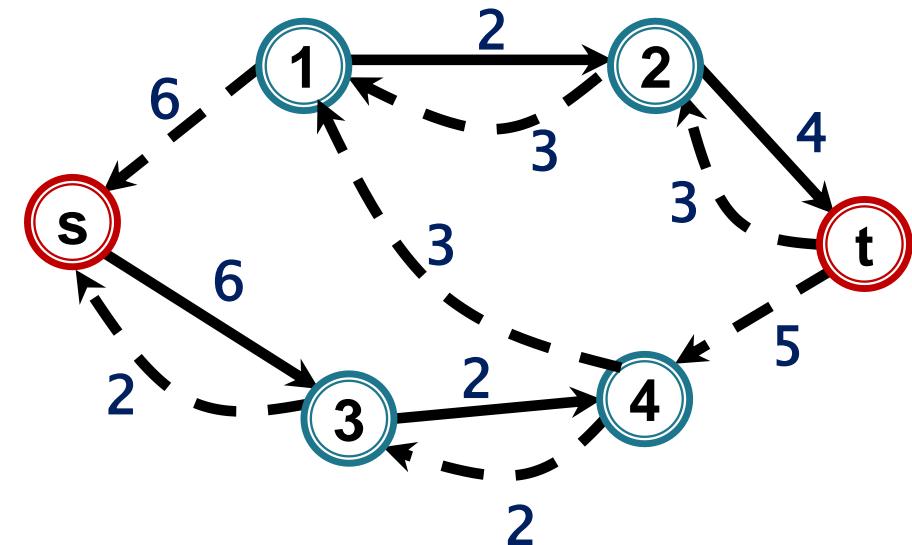
(Multi)graful rezidual  $G_f$  asociat fluxului  $f =$  graf orientat ponderat, cu funcția de pondere (de capacitate)  $c_f$  construit astfel:

- $V(G_f) = V(G)$
- Pentru fiecare arc  $e = uv$  cu  $c(e) - f(e) > 0$ ,
- Pentru fiecare arc  $e = uv$  cu  $f(e) > 0$ ,

Rețeaua de transport



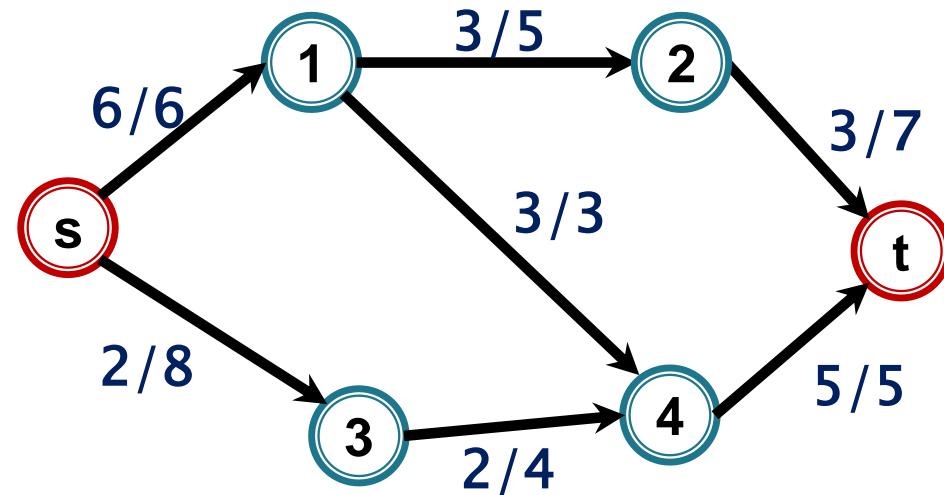
Graful rezidual



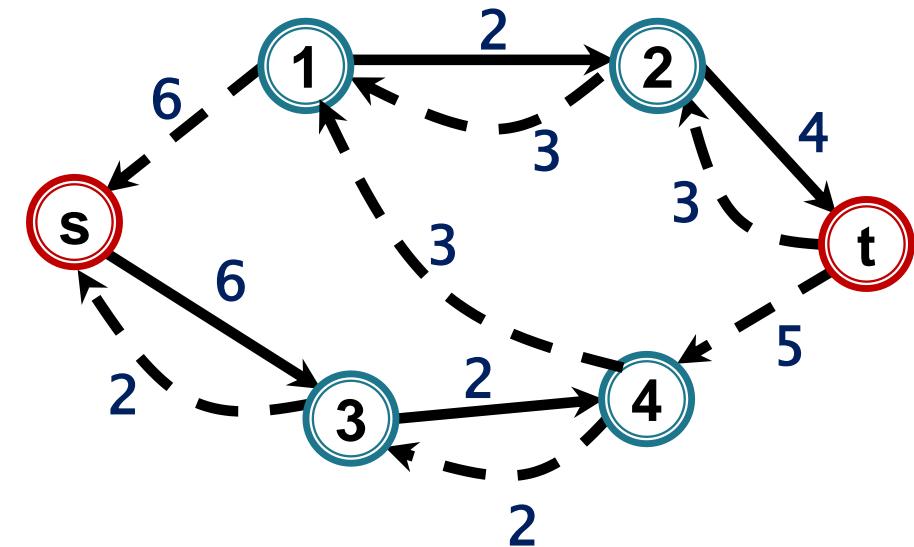
(Multi)graful rezidual  $G_f$  asociat fluxului  $f$  = graf orientat ponderat, cu funcția de pondere (de capacitate)  $c_f$  construit astfel:

- $V(G_f) = V(G)$
- Pentru fiecare arc  $e = uv$  cu  $c(e) - f(e) > 0$ , adăugăm arcul  $e$  la  $G_f$  cu ponderea  $c_f(e) = c(e) - f(e)$
- Pentru fiecare arc  $e = uv$  cu  $f(e) > 0$ ,

Rețeaua de transport



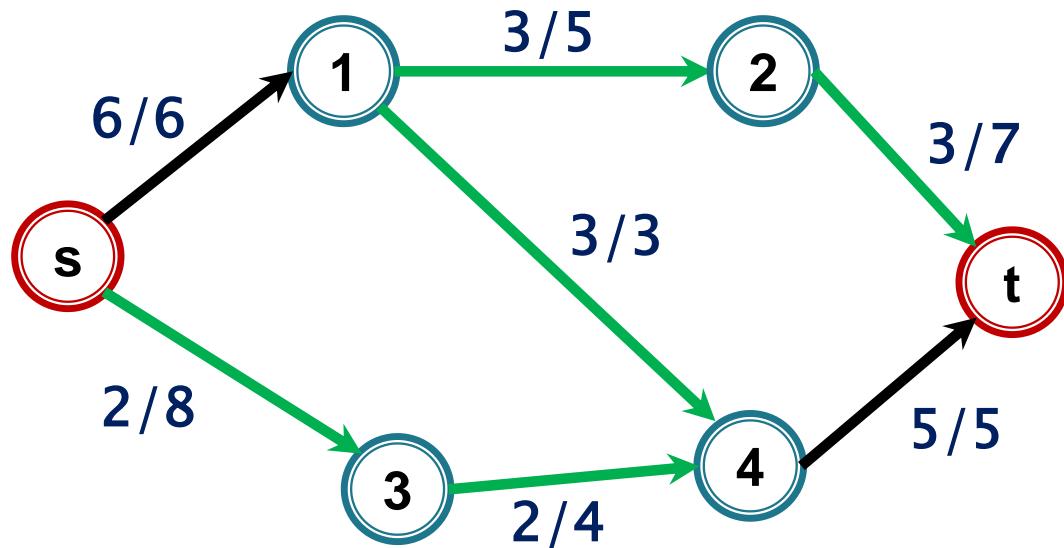
Graful rezidual



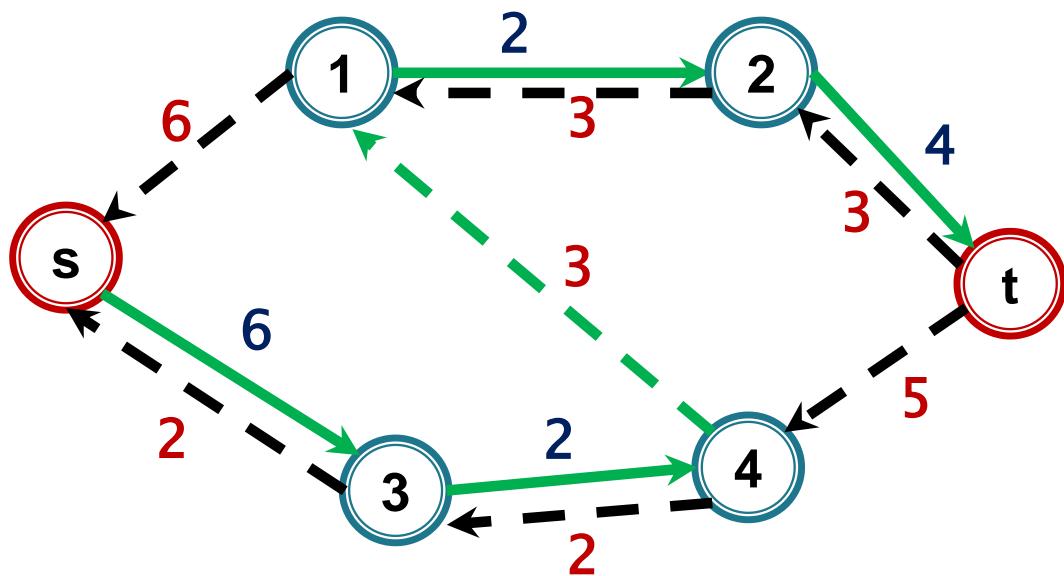
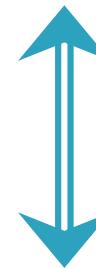
(Multi)graful rezidual  $G_f$  asociat fluxului  $f$  = graf orientat ponderat, cu funcția de pondere (de capacitate)  $c_f$  construit astfel:

- $V(G_f) = V(G)$
- Pentru fiecare arc  $e = uv$  cu  $c(e) - f(e) > 0$ , adăugăm arcul  $e$  la  $G_f$  cu ponderea  $c_f(e) = c(e) - f(e)$
- Pentru fiecare arc  $e = uv$  cu  $f(e) > 0$ , adăugăm la  $G_f$  arcul  $e^{-1} = vu$  (inversul arcului  $e$ ) cu ponderea asociată  $c_f(e^{-1}) = f(e)$ .

## Graf rezidual

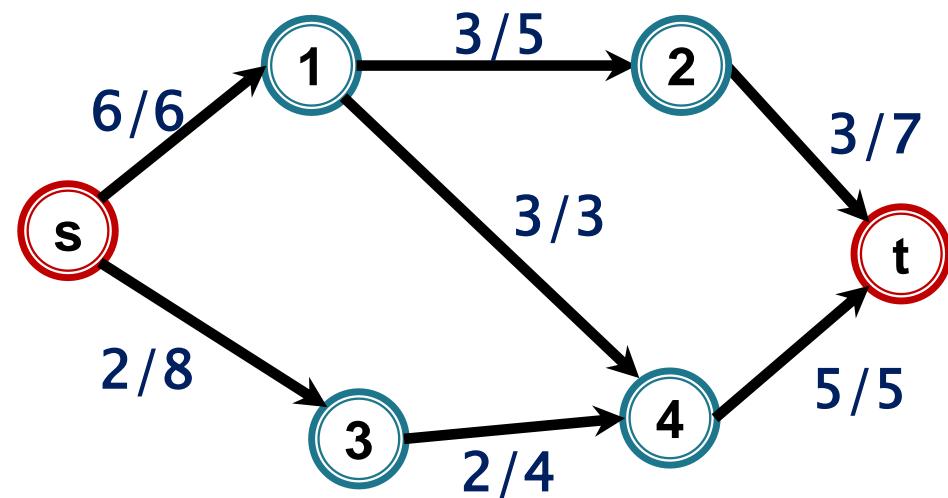


$s-t$  lanț f-nesaturat

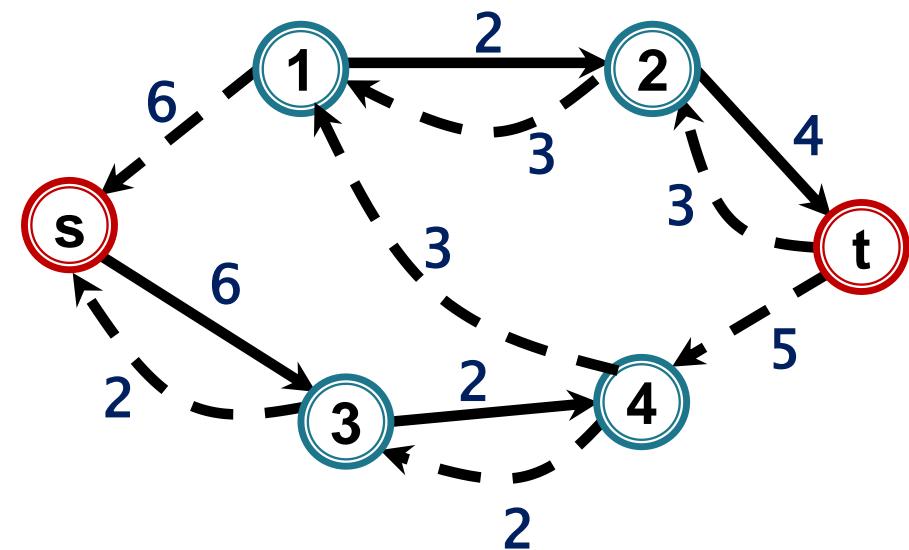


$s-t$  drum în graful rezidual

Rețeaua de transport



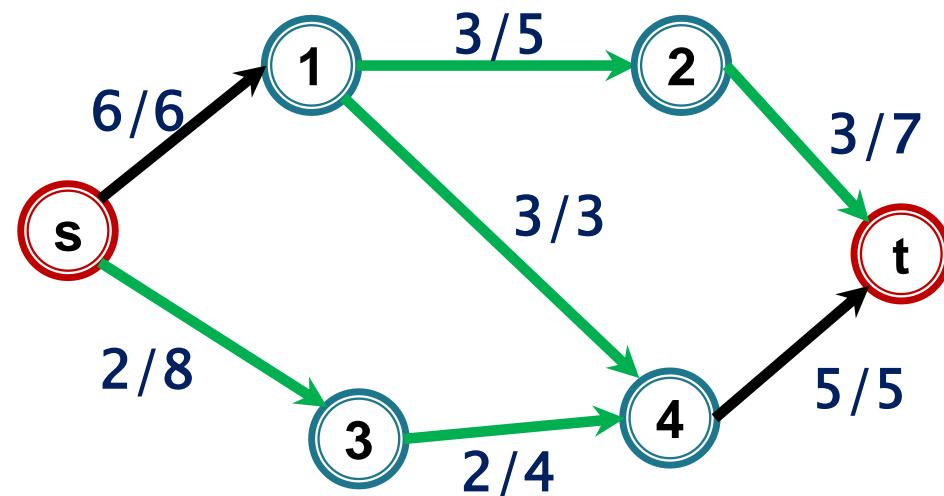
Graful rezidual



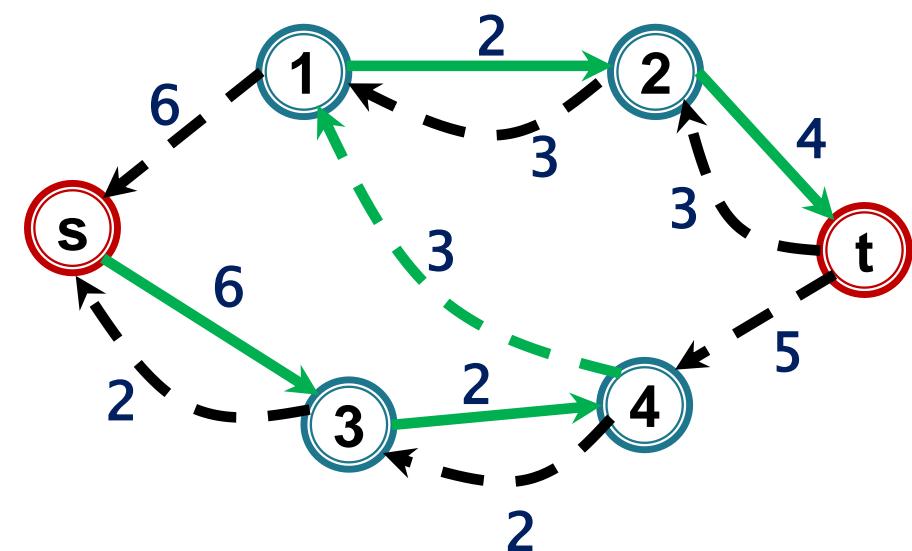
Capacitatea unui **drum** P în  $G_f$ :

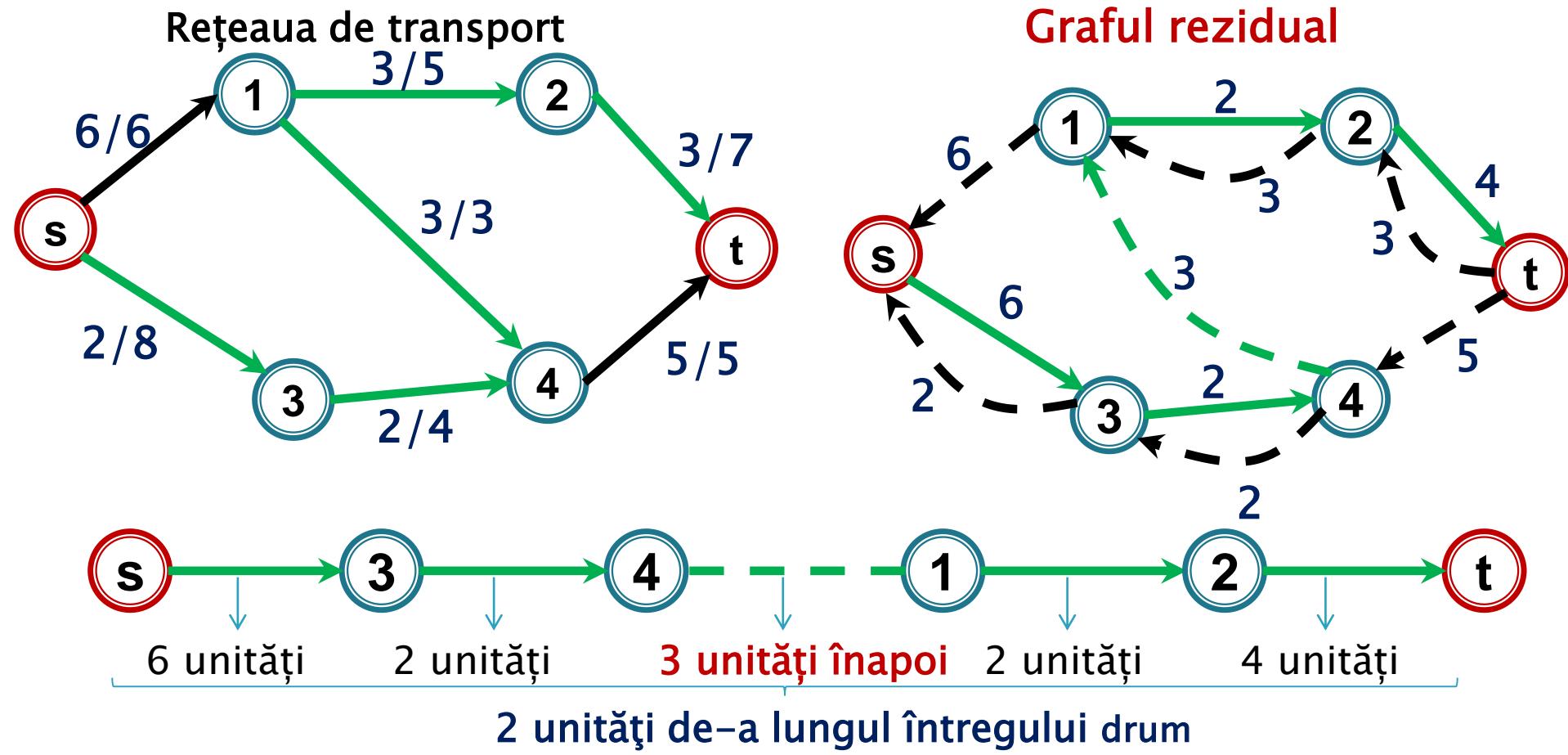
$$c_f(P) = \min\{c_f(e) \mid e \in E(P)\}$$

Rețeaua de transport



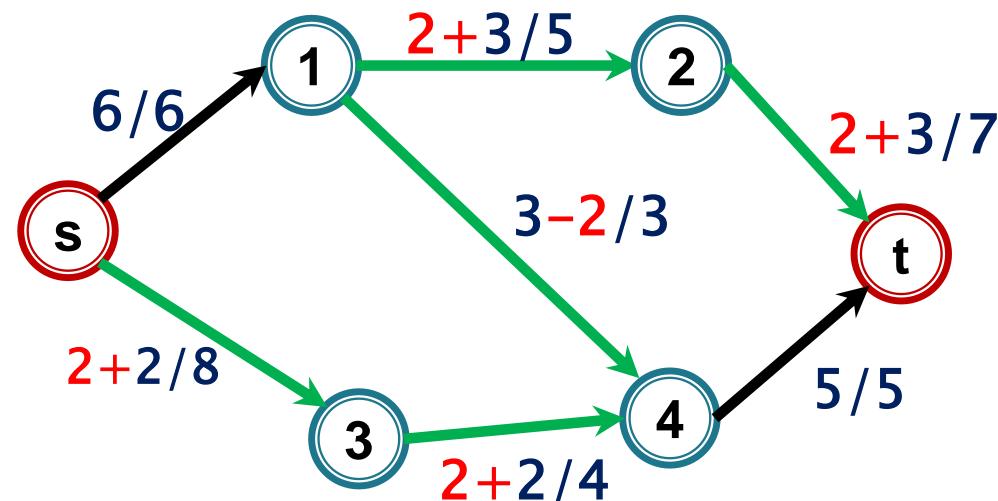
Graful rezidual



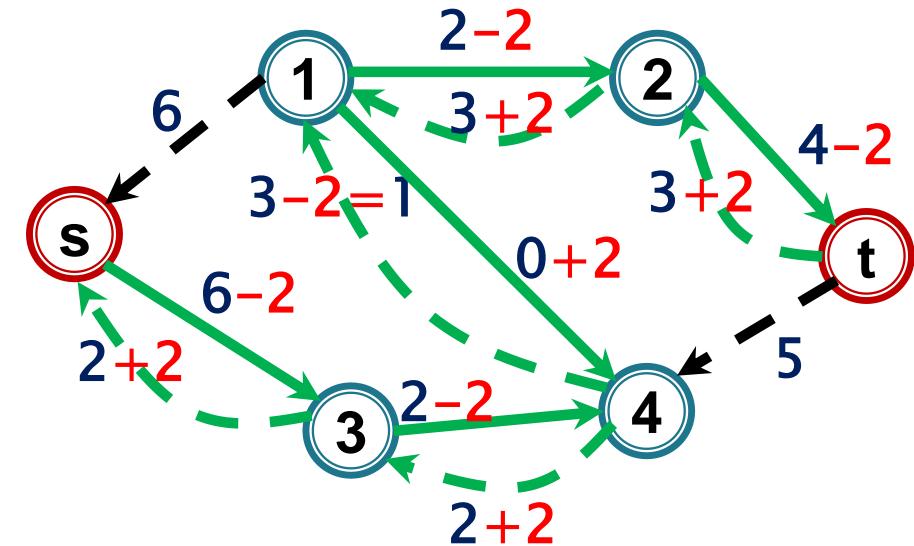


- P este un s - t **lanț f-nesaturat** în G  $\Leftrightarrow$  P este s - t **drum** în  $G_f$
- În plus  $i(P)$  (în G) =  $c_f(P)$  (în  $G_f$ )
- Un arc e este **arc invers** în lanțul P în G  $\Leftrightarrow$   
 $e^{-1}$  este arc în drumul P din  $G_f$

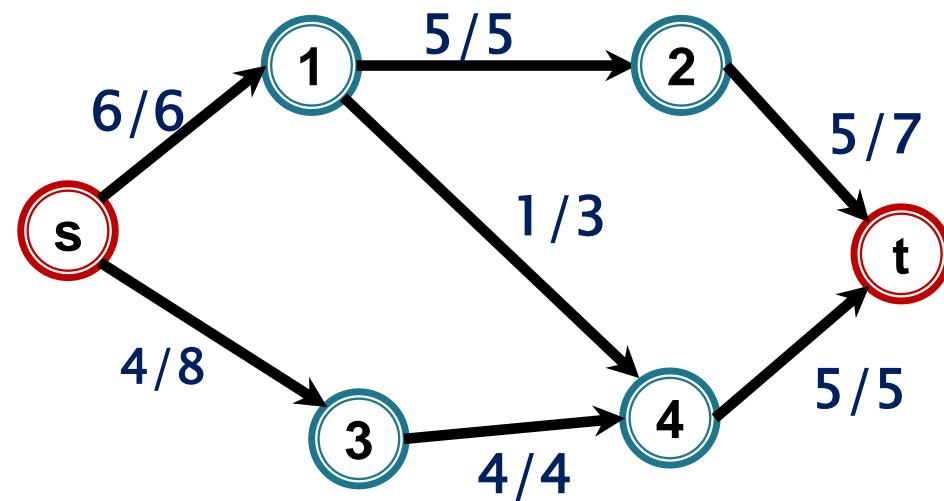
Rețeaua de transport



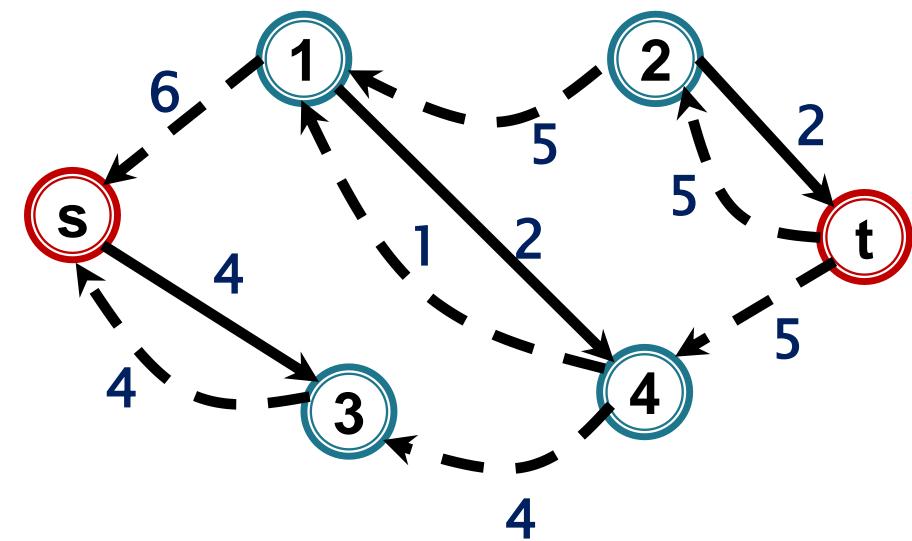
Graful rezidual



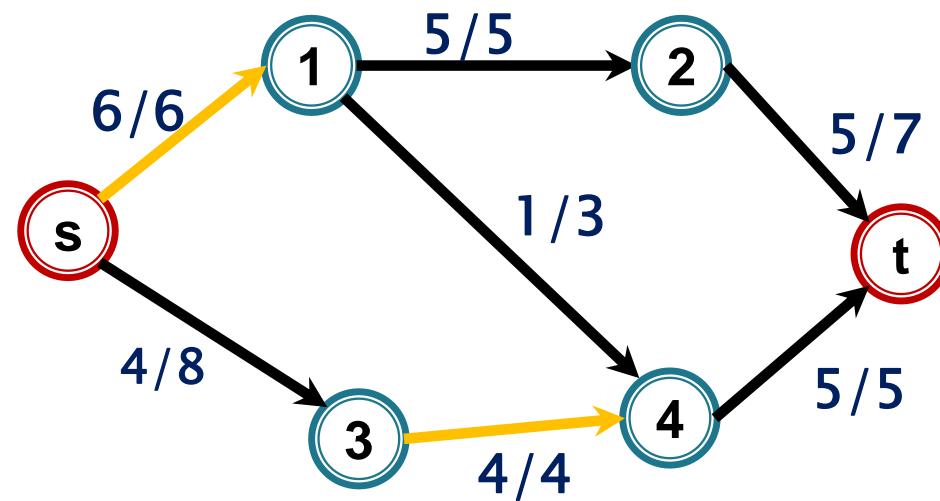
Rețeaua de transport



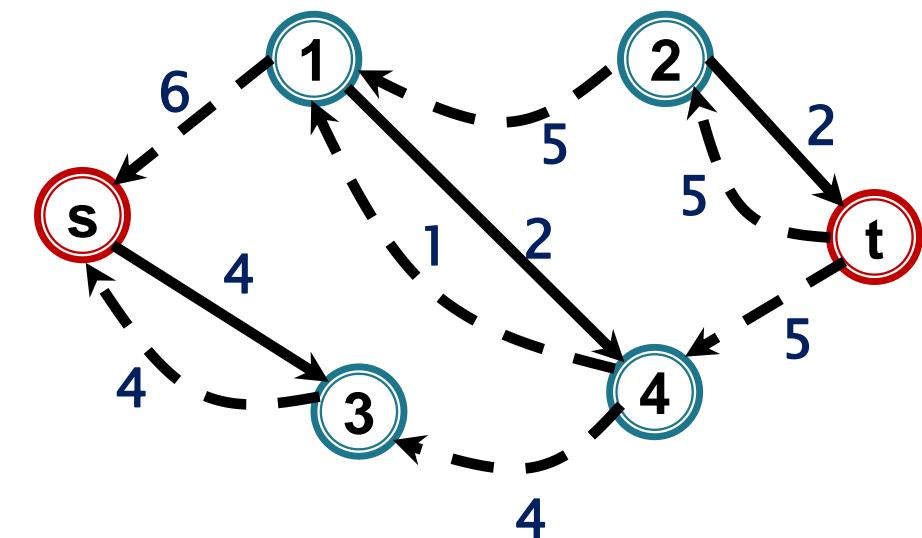
Graful rezidual



Rețeaua de transport



Graful rezidual



tăietură saturată

$\Leftrightarrow$  nu mai există  $s-t$  drum în graful rezidual

$\Leftrightarrow$   $s-t$  flux maxim

este tăietură minimă

# Algoritmul Edmonds-Karp - implementare folosind graf rezidual

Schema devine:

**initializeaza\_flux\_nul()**

cat timp (**construieste\_s-t\_drum\_in G\_f\_BF()=true**) executa

**revizuieste\_flux\_lant()**

actualizeaza  $G_f$

afiseaza\_flux()

Detaliem această schemă

# Algoritmul Edmonds-Karp – implementare folosind graf rezidual

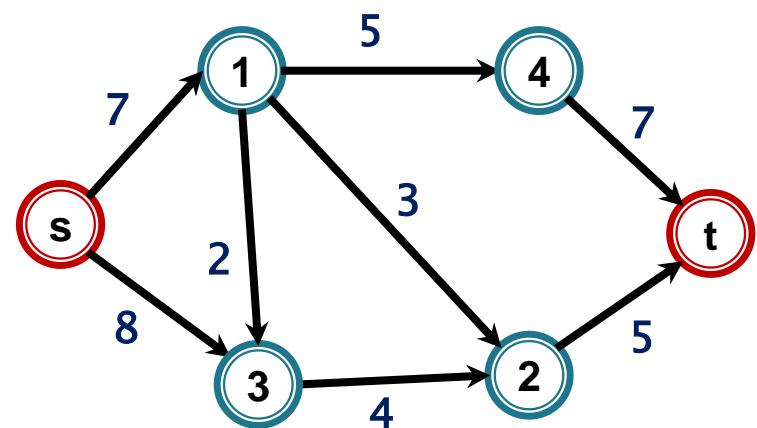
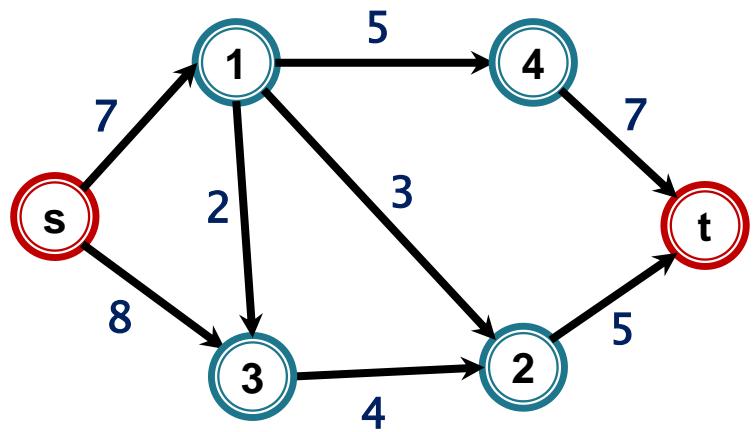
## Detalii (implementare – laborator)

1. Setăm  $f := 0$  fluxul vid ( $f(e) = 0, \forall e \in E$ )
2. Construim  $G_f :=$  graful rezidual pentru  $f$
3. Cât timp există un s-t drum în  $G_f$ 
  - determină  $P$  un s-t drum minim în  $G_f$  folosind BF (pentru arcele cu  $c_f(e) > 0$ )
  - actualizează  $G_f$ 

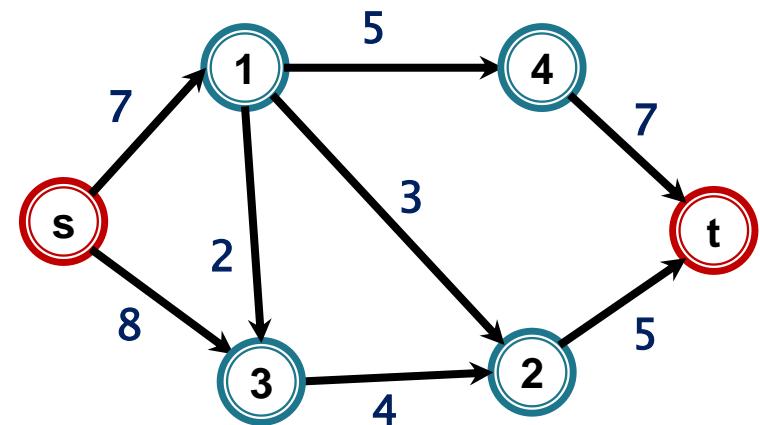
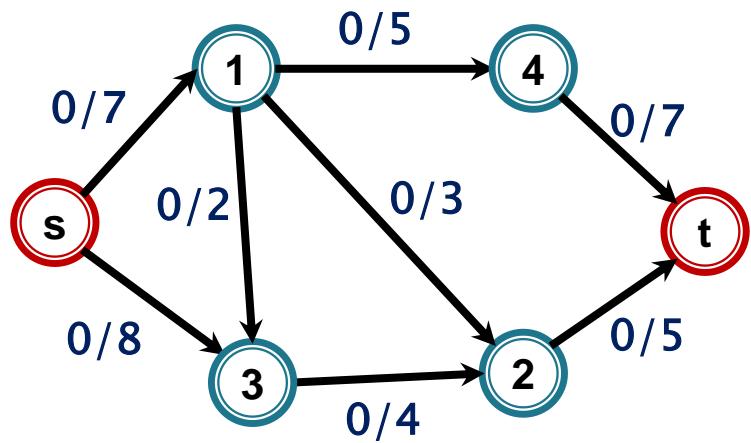
pentru  $e \in E(P) \subseteq E(G_f)$

$$c_f(e) \leftarrow c_f(e) - cfP;$$
$$c_f(e^{-1}) \leftarrow c_f(e^{-1}) + cfP$$
4. returnează  $f$

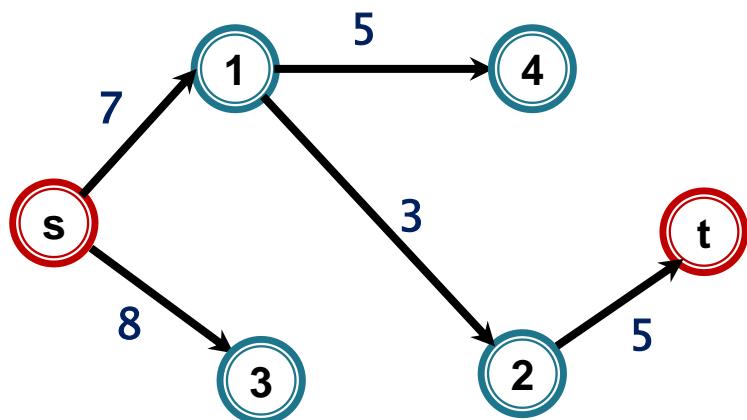
## Graful rezidual $G_f$



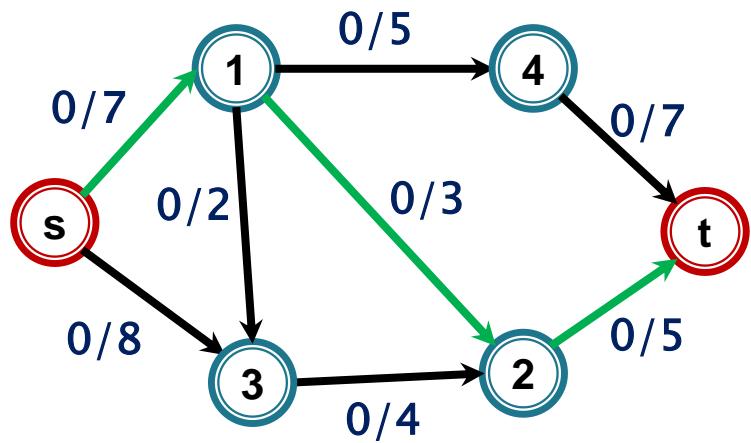
## Graful rezidual $G_f$



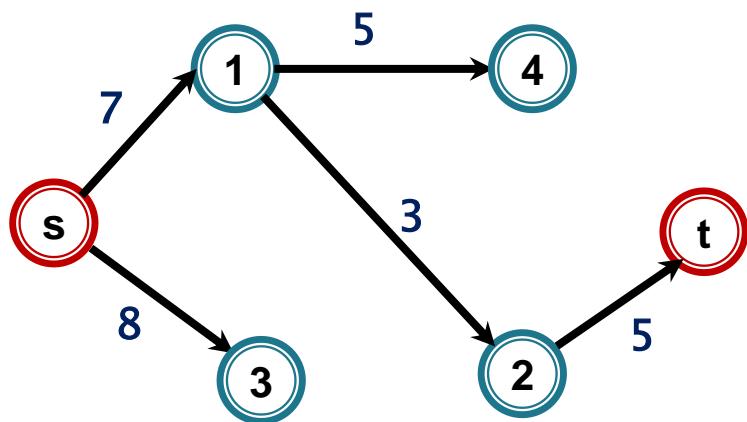
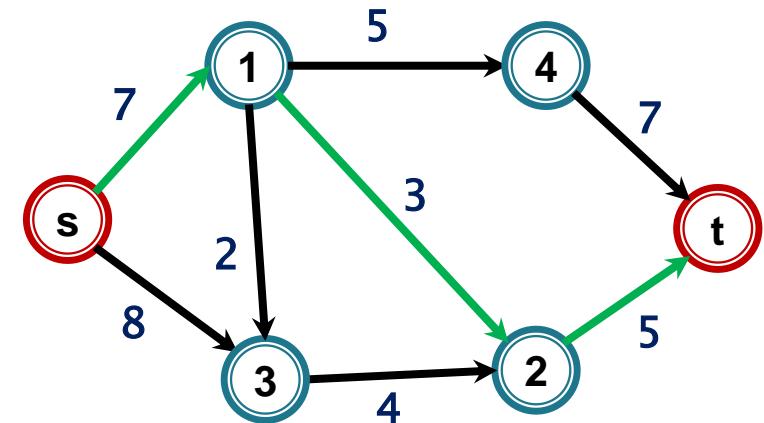
$BF(s)$  - în graful rezidual



## Graful rezidual $G_f$



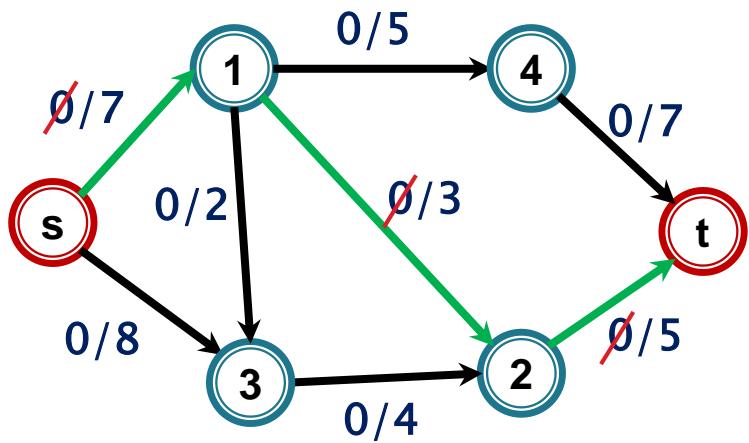
$BF(s)$  - în graful rezidual



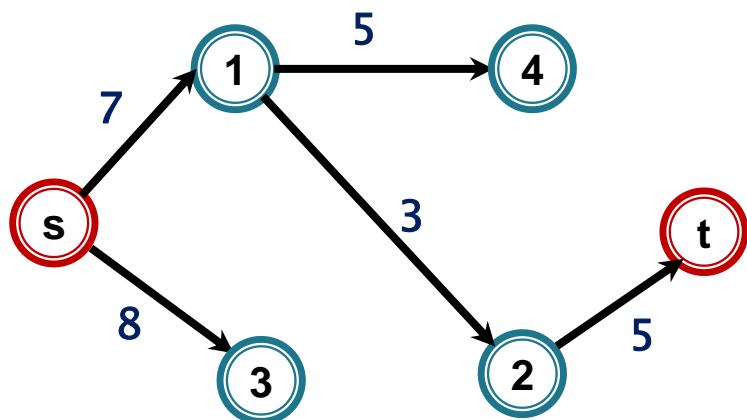
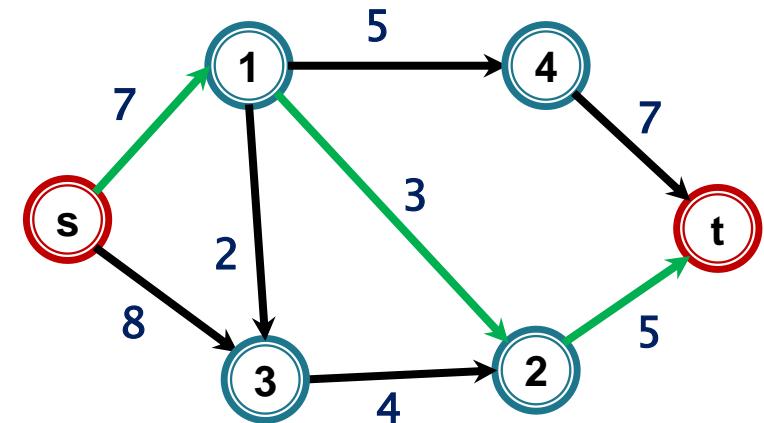
Drumul de creștere  $[s, 1, 2, t]$  – capacitate reziduală 3

Revizuim fluxul

## Graful rezidual $G_f$



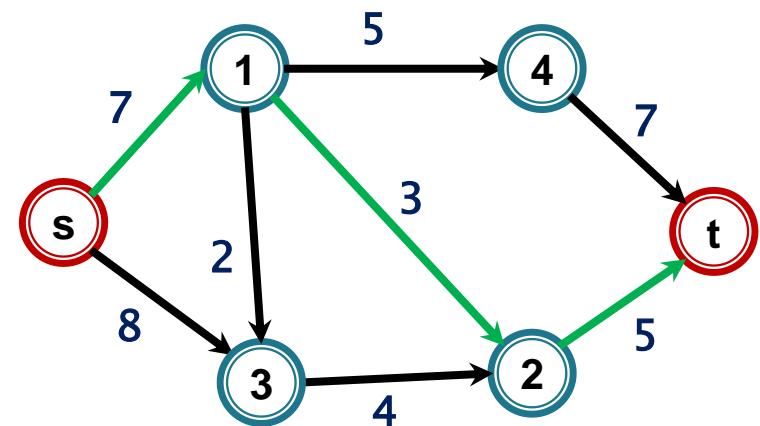
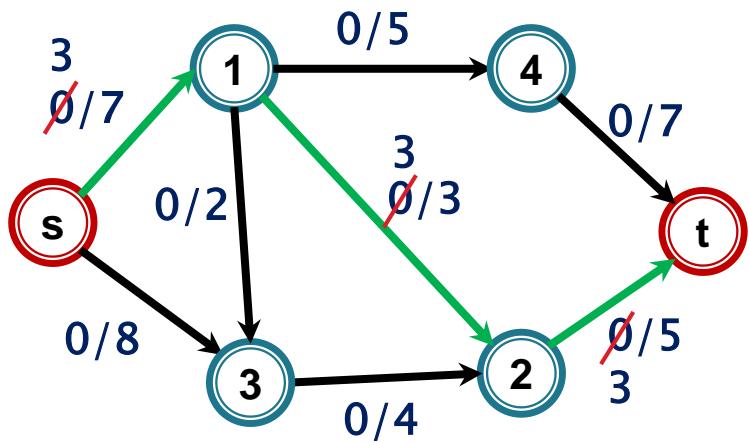
$BF(s)$  - în graful rezidual



Drumul de creștere  $[s, 1, 2, t]$  – capacitate reziduală 3

Revizuim fluxul

## Graful rezidual $G_f$



## Actualizăm rețeaua reziduală

pentru  $e \in E(P) \subseteq E(G_f)$

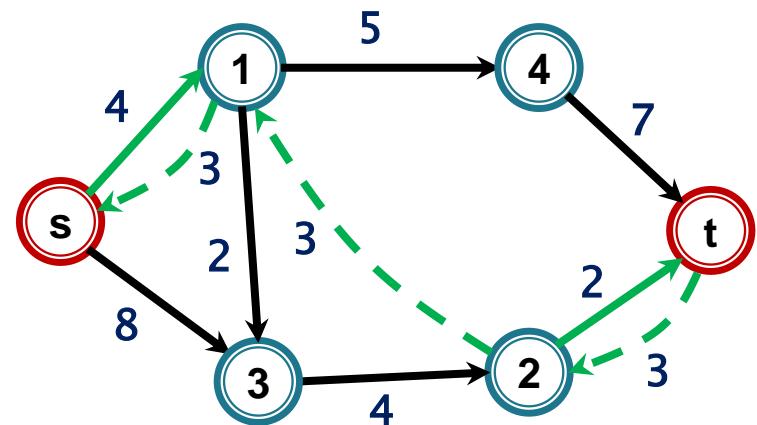
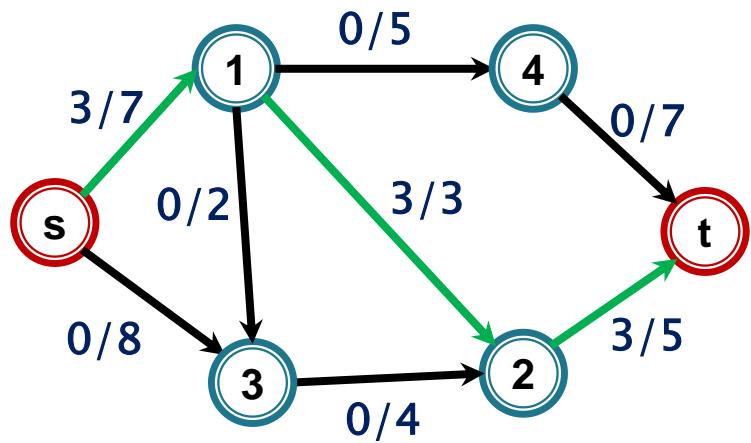
$$c_f(e) \leftarrow c_f(e) - cf_P;$$

dacă  $c_f(e)=0$  elimină  $e$  din  $G_f$  (se ignora în parcursare)

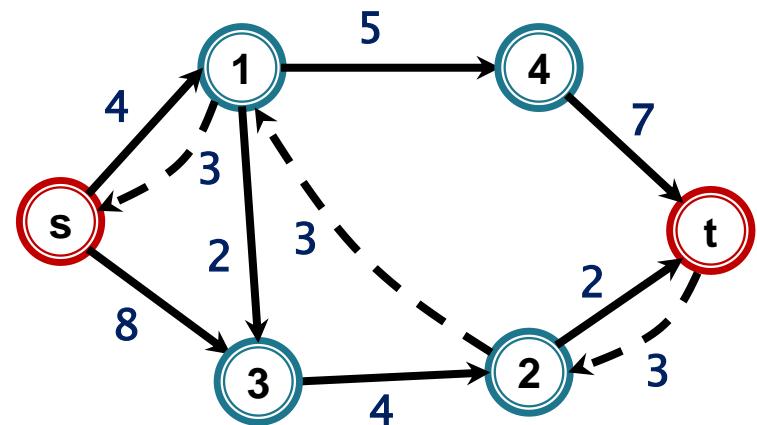
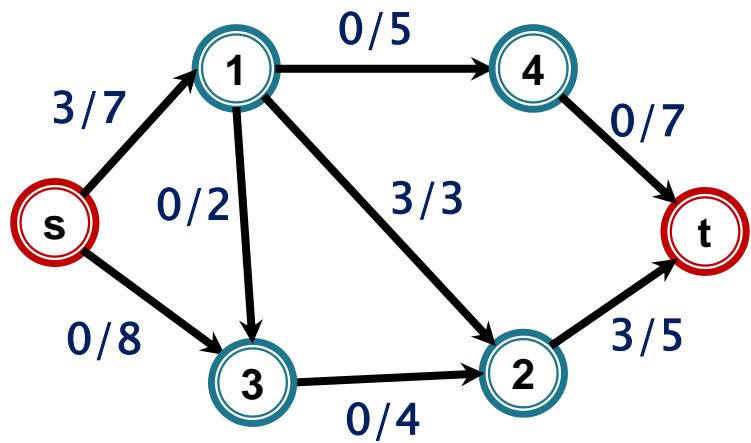
$$c_f(e^{-1}) \leftarrow c_f(e^{-1}) + cf_P$$

dacă  $c_f(e^{-1}) > 0$  și  $e^{-1} \notin G_f$  atunci adaugă  $e^{-1}$  la  $G_f$

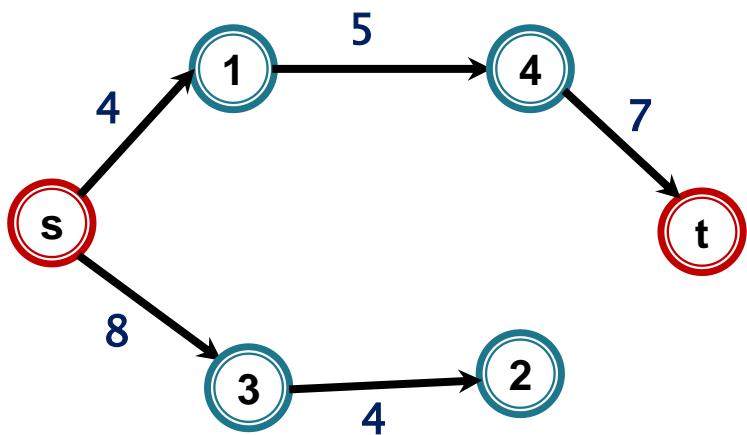
## Graful rezidual $G_f$



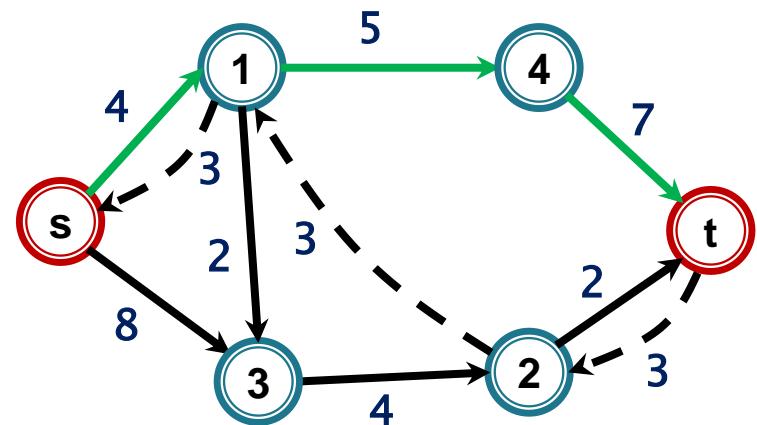
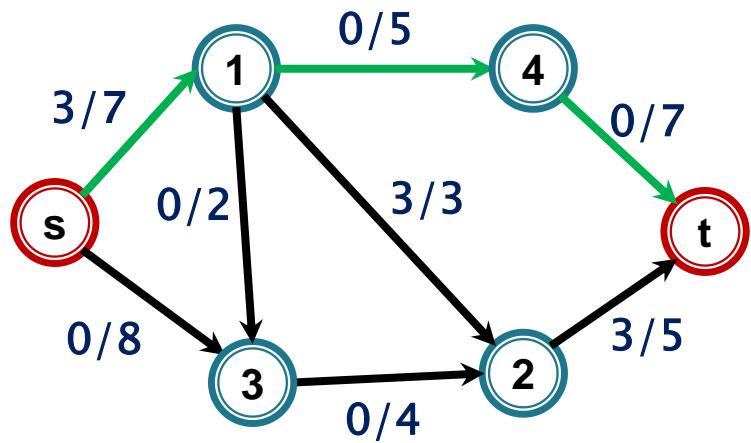
## Graful rezidual $G_f$



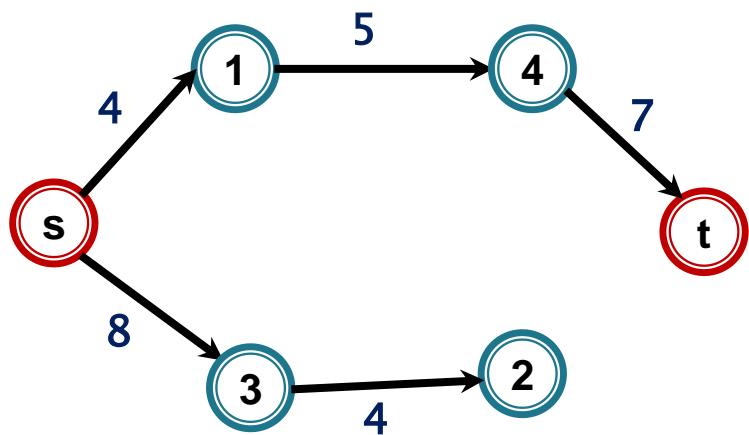
$BF(s)$  - în graful rezidual



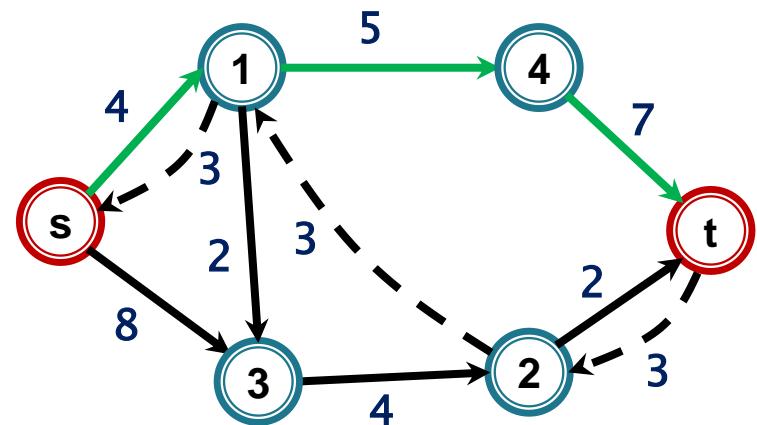
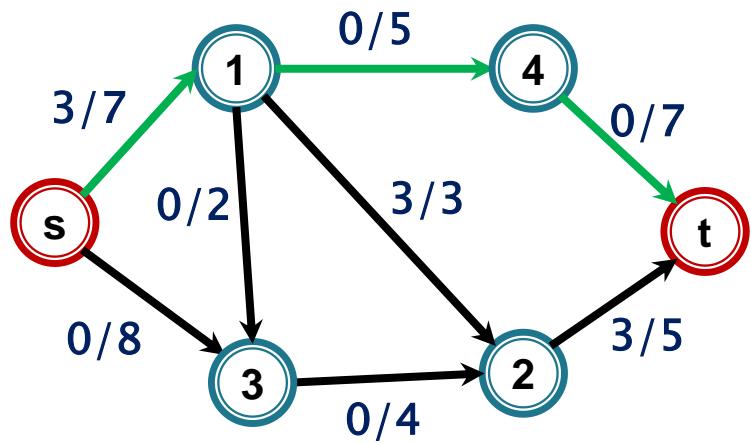
## Graful rezidual $G_f$



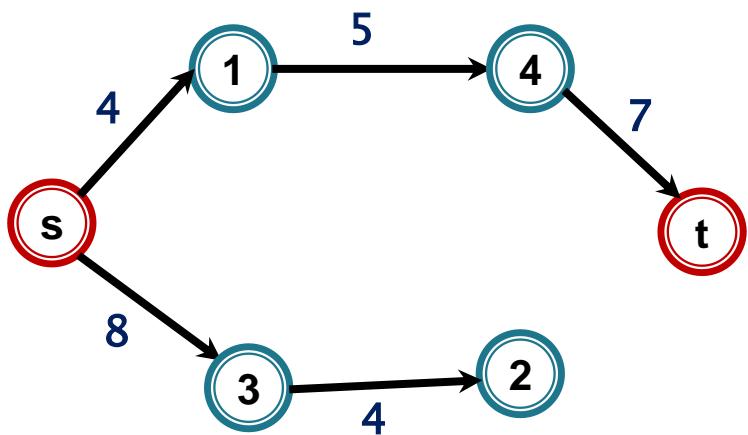
$BF(s)$  - în graful rezidual



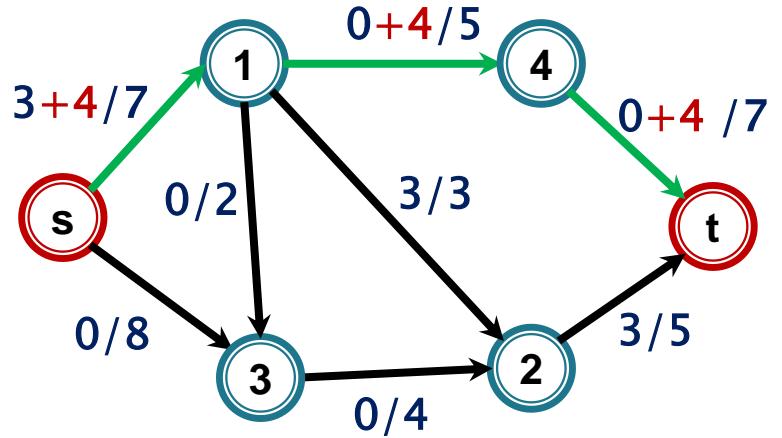
## Graful rezidual $G_f$



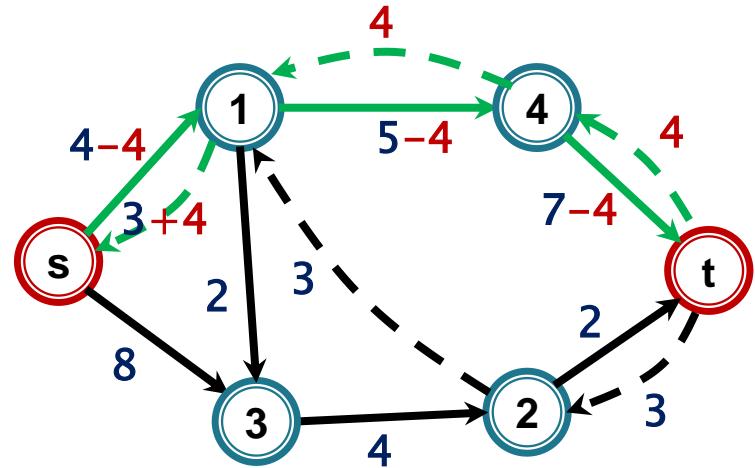
$BF(s)$  - în graful rezidual



Drumul de creștere  $[s, 1, 4, t]$  – capacitate reziduală 4

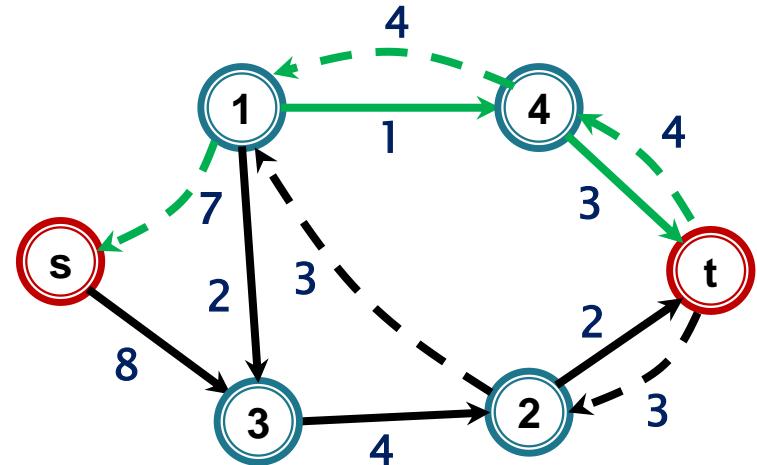
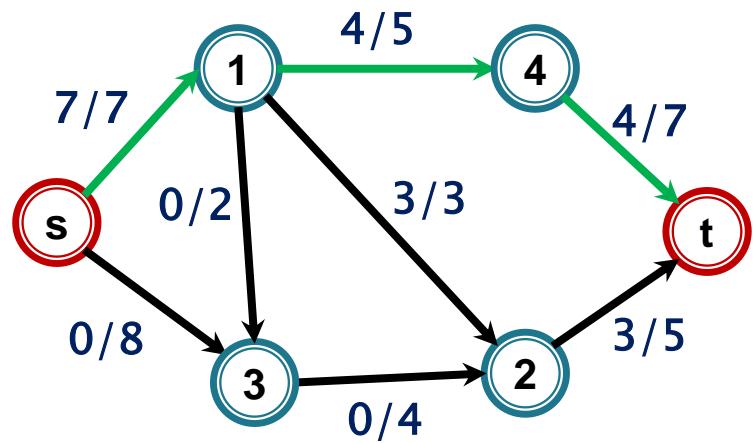


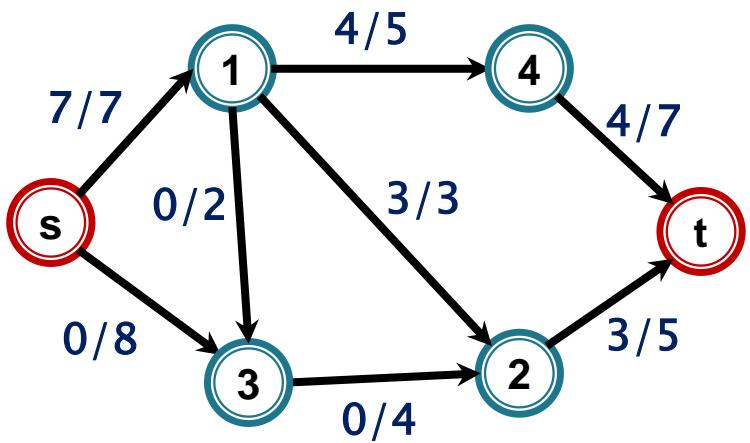
Graful rezidual  $G_f$



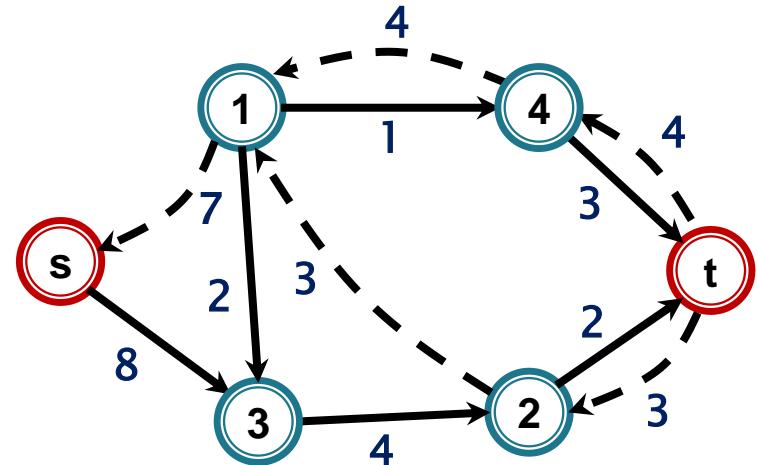
actualizare  $G_f$ :

## Graful rezidual $G_f$

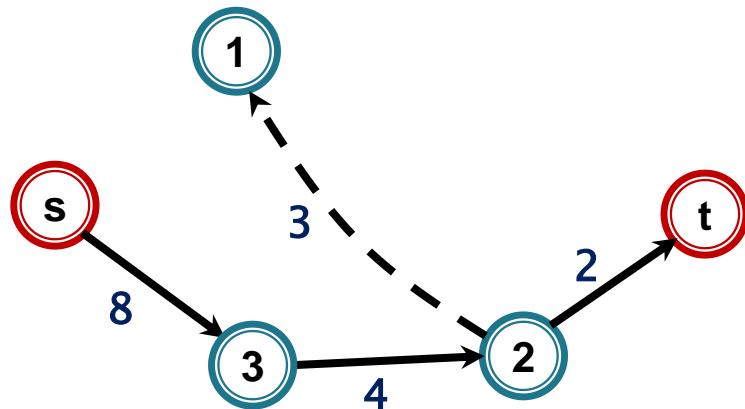


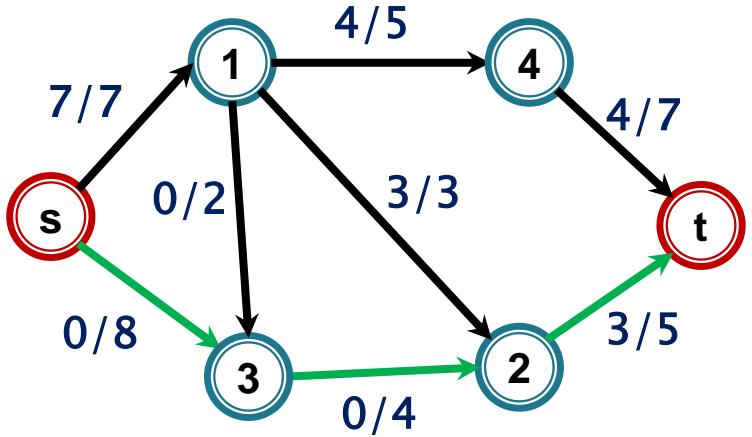


Graful rezidual  $G_f$

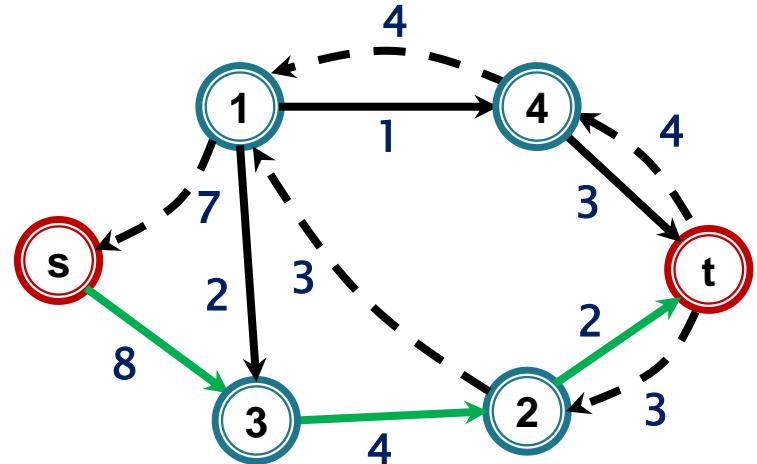


$BF(s)$  - în graful rezidual

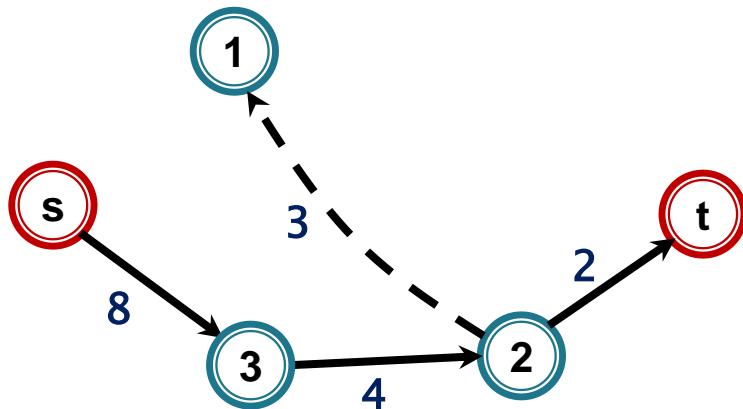




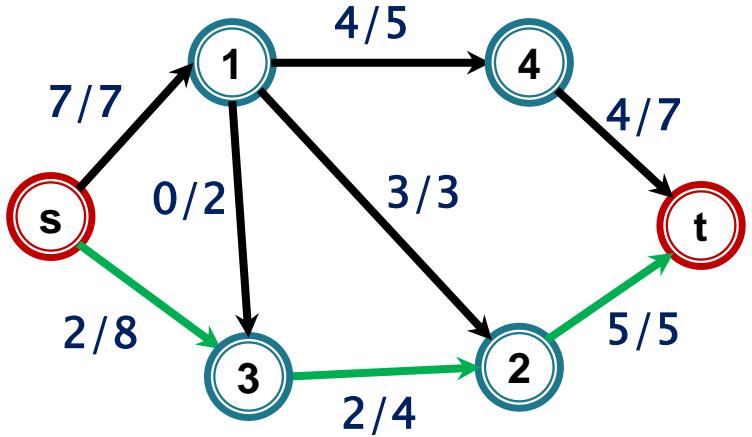
Graful rezidual  $G_f$



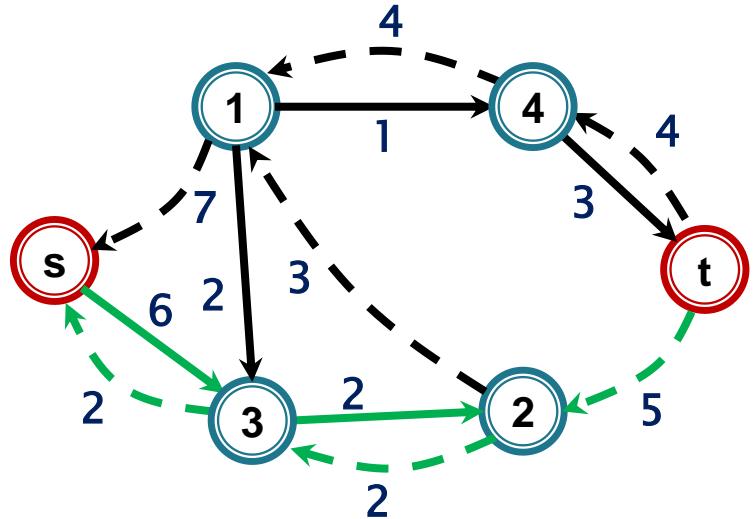
$BF(s)$  - în graful rezidual

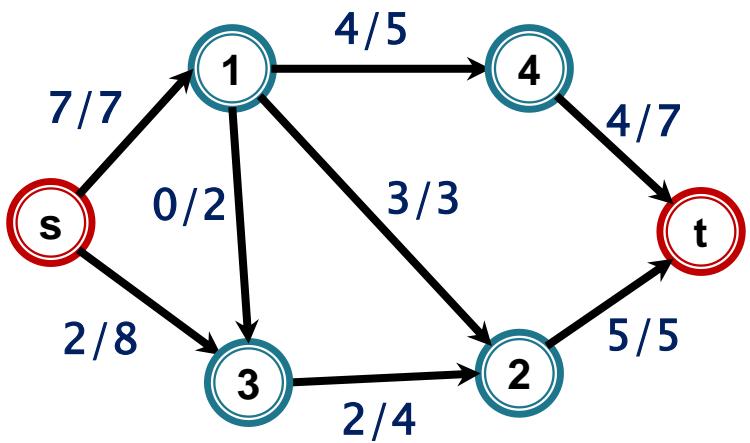


Drumul de creștere  $[s, 3, 2, t]$  - capacitate reziduală  $2$

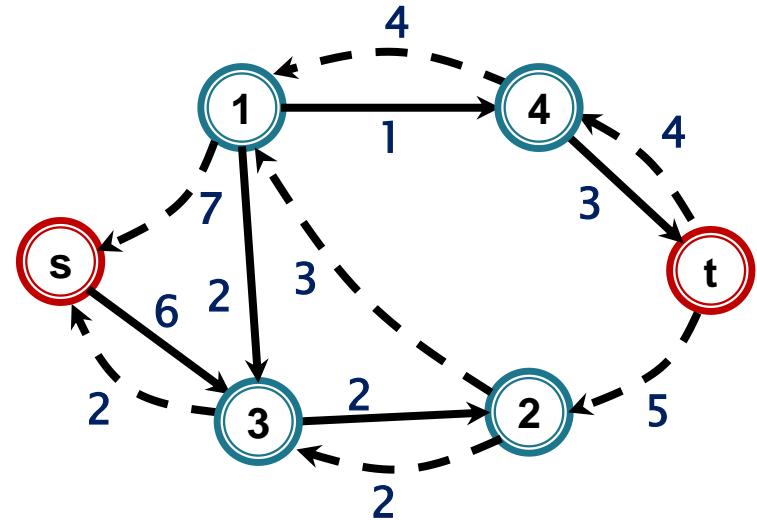


Graful rezidual  $G_f$

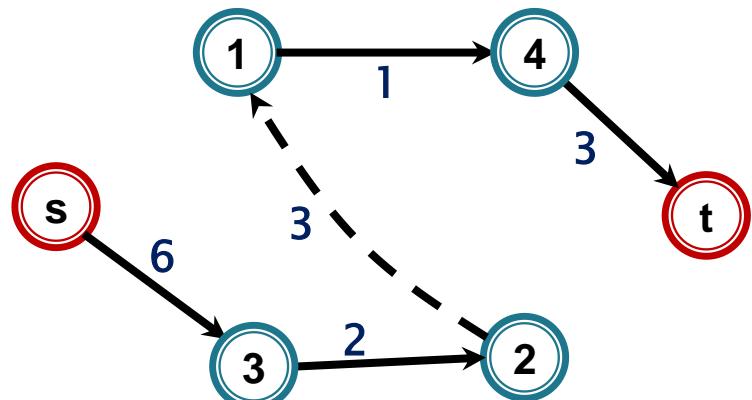


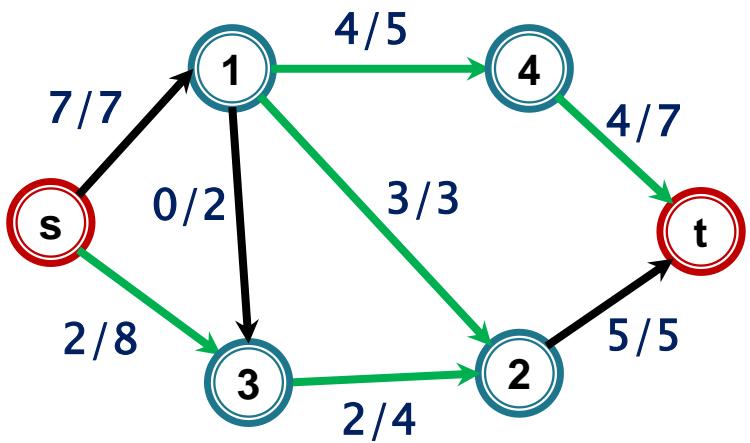


Graful rezidual  $G_f$

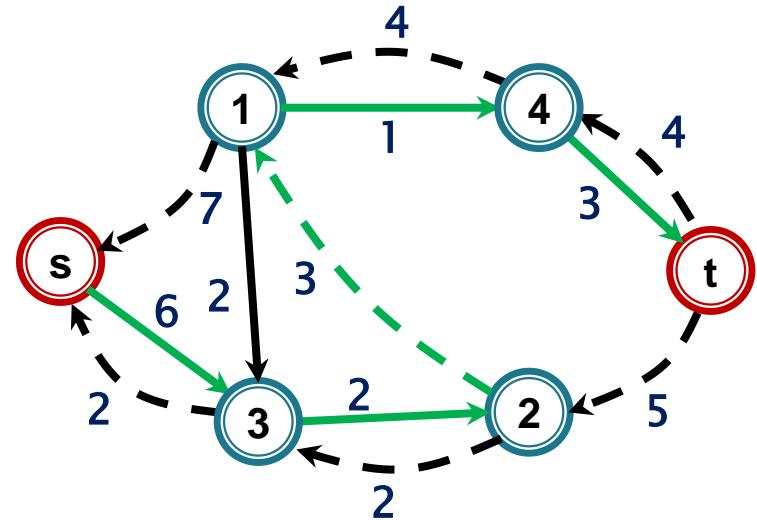


$BF(s)$  - pe graful rezidual

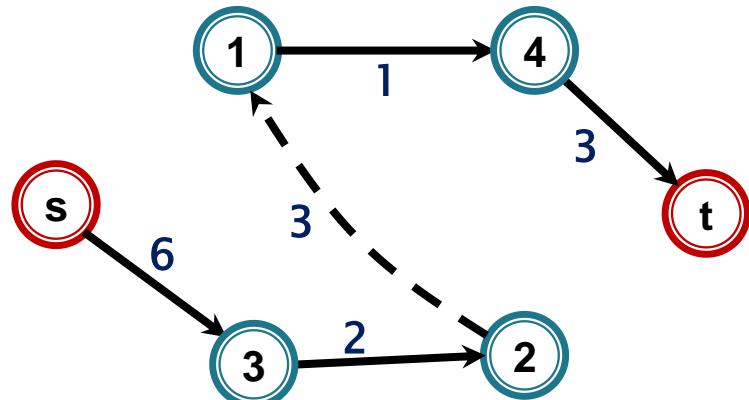




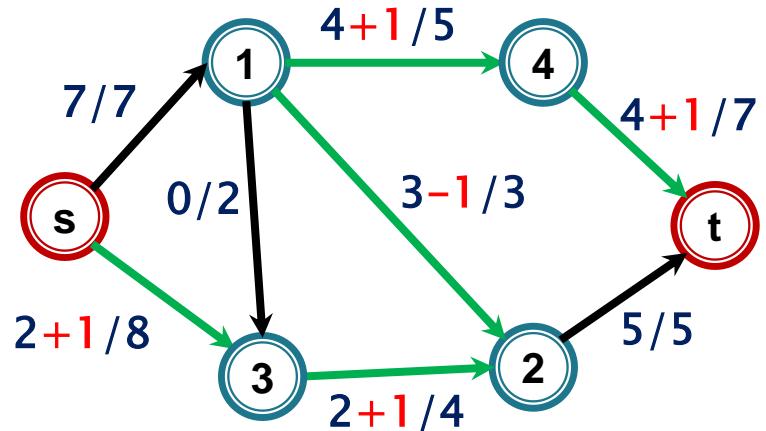
Graful rezidual  $G_f$



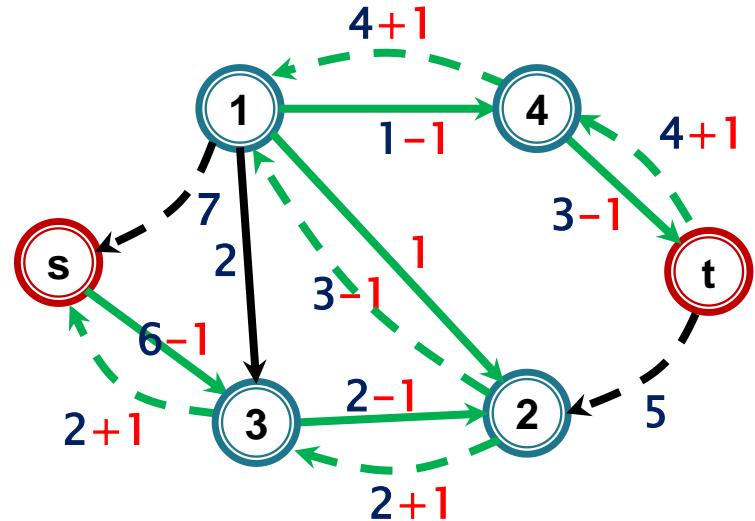
$BF(s)$  - în graful rezidual



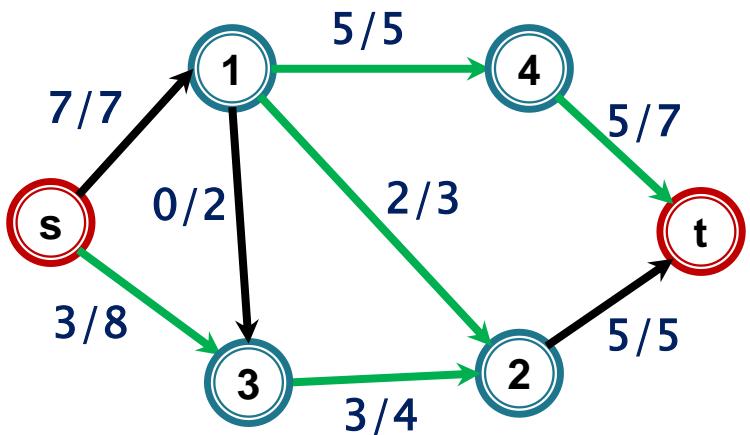
Drumul de creștere  $[s, 3, 2, 1, 4, t]$  - capacitate reziduală 1



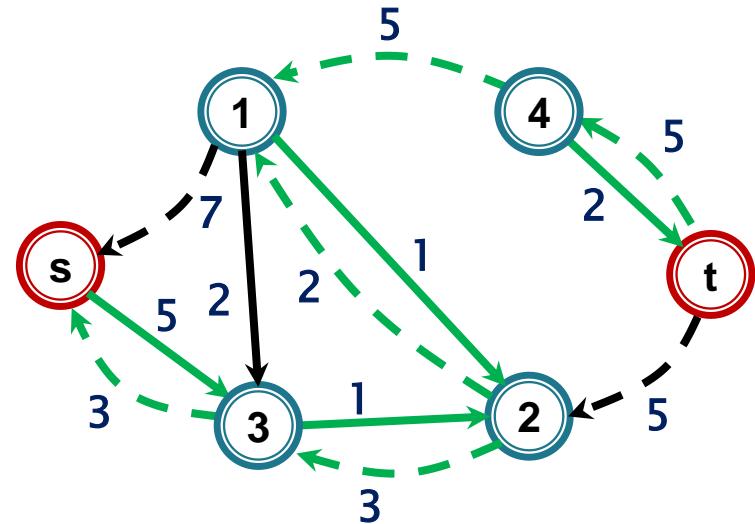
Graful rezidual  $G_f$

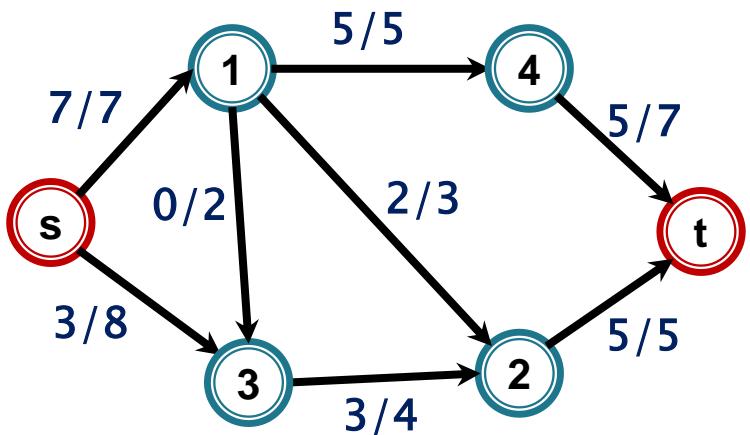


actualizare  $G_f$ :

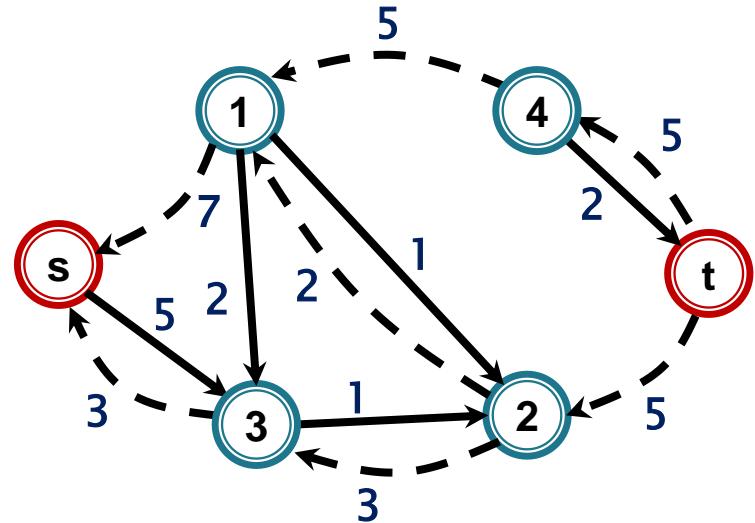


Graful rezidual  $G_f$

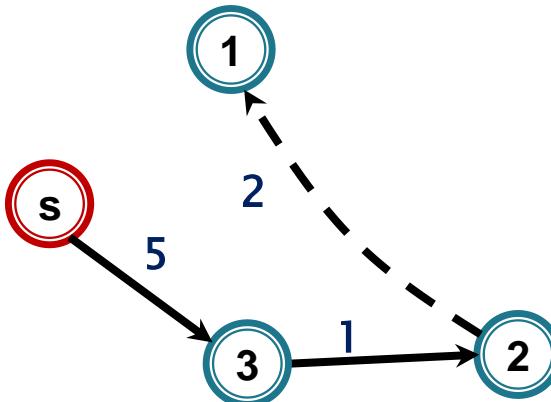


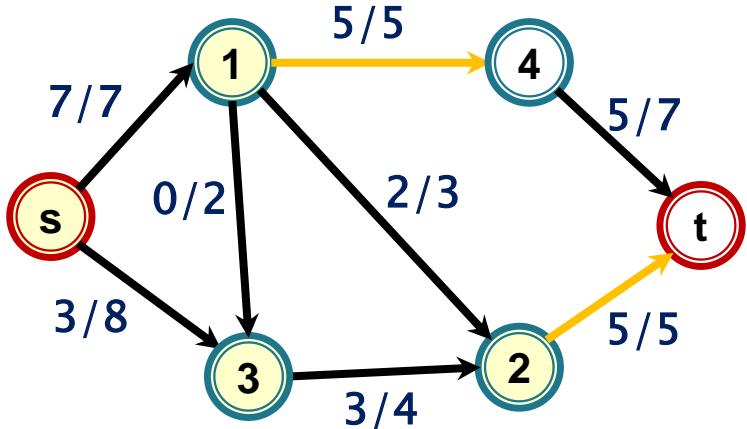


Graful rezidual  $G_f$

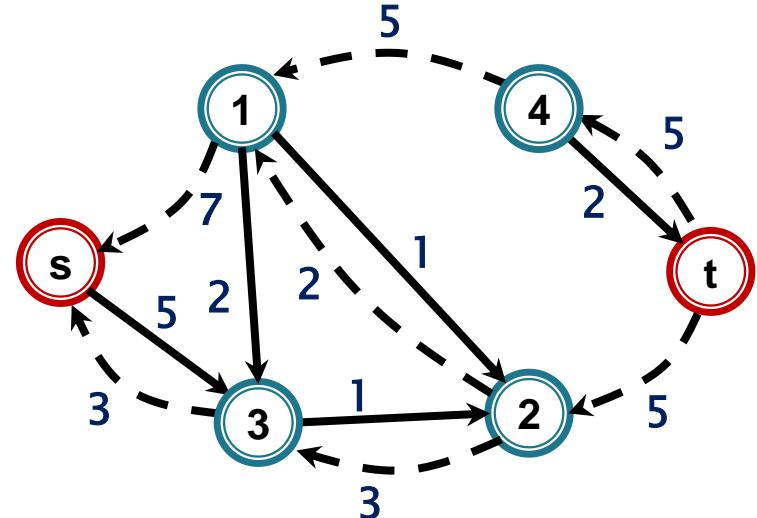


$BF(s)$  - în graful rezidual

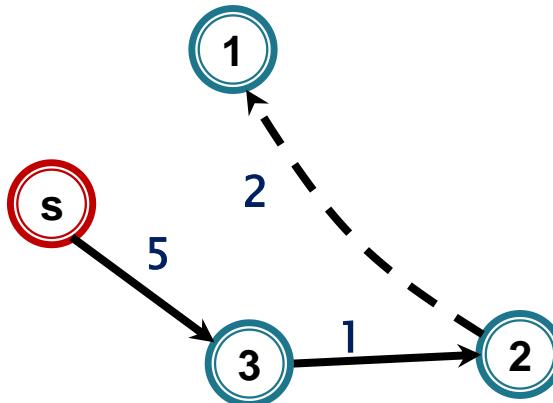




Graful rezidual  $G_f$



$BF(s)$  - în graful rezidual



Nu mai există drumul de creștere  $\Rightarrow s-t$  flux maxim (valoare 10) +  $s-t$  tăietură minimă (de capacitate tot 10, determinată de vârfurile accesibile din  $s$  în  $G_f$ :  $S = \{s, 1, 3, 2\}$ )

# Algoritmul Edmonds-Karp – implementare folosind graf rezidual

## Detalii (implementare – laborator)

1. Setăm  $f := 0$  fluxul vid ( $f(e) = 0, \forall e \in E$ )
2. Construim  $G_f :=$  graful rezidual pentru  $f$
3. Cât timp există un s-t drum în  $G_f$ 
  - determină  $P$  un s-t drum minim în  $G_f$  folosind BF (pentru arcele cu  $c_f(e) > 0$ )
  - actualizează  $G_f$ 

pentru  $e \in E(P) \subseteq E(G_f)$

$$c_f(e) \leftarrow c_f(e) - cfP;$$
$$c_f(e^{-1}) \leftarrow c_f(e^{-1}) + cfP$$
4. returnează  $f$