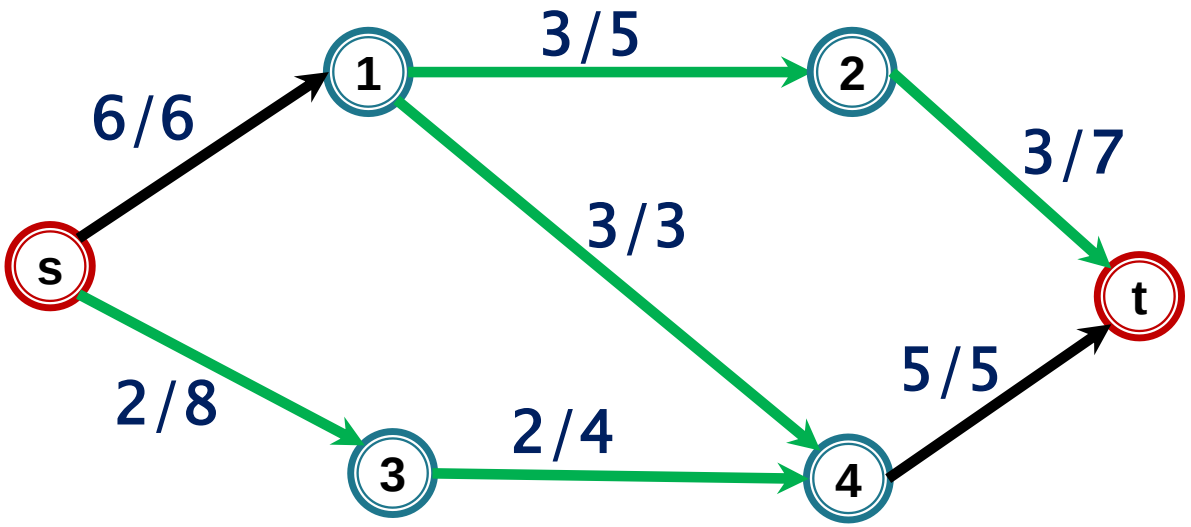
Algoritmul FORD-FULKERSON

de determinare a unui flux maxim

+ a unei tăieturi minime

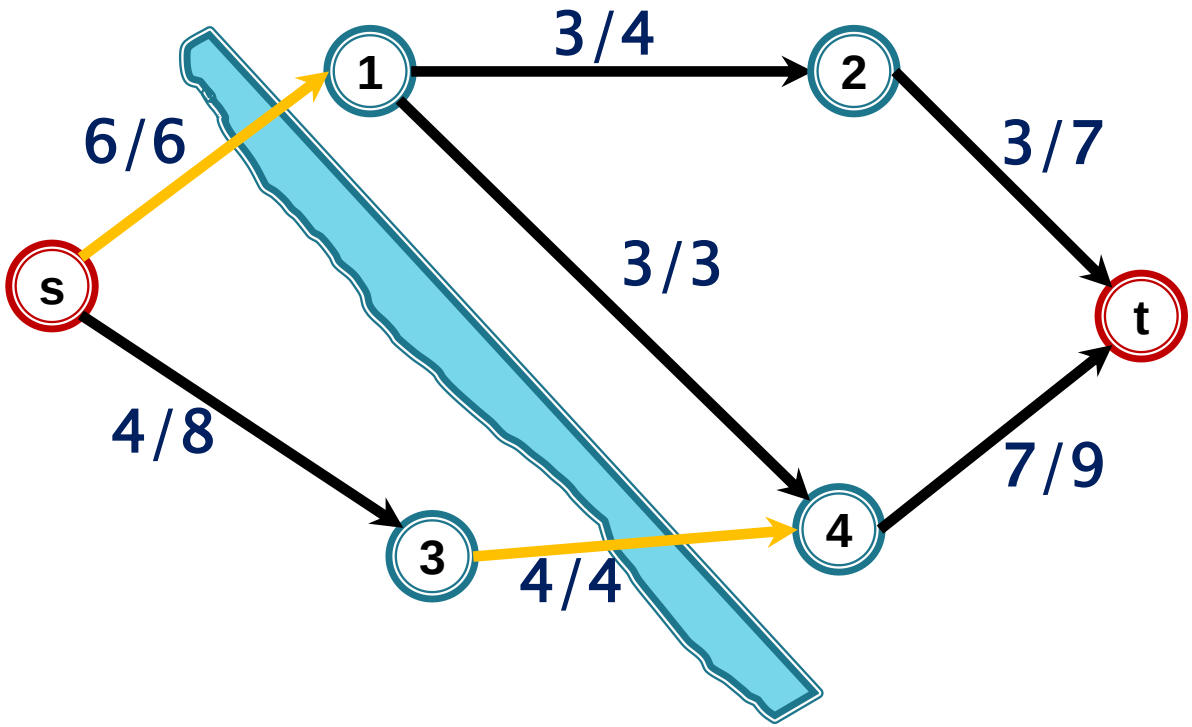
Algoritmul Ford-Fulkerson

Amintim din exemplele anterioare:



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

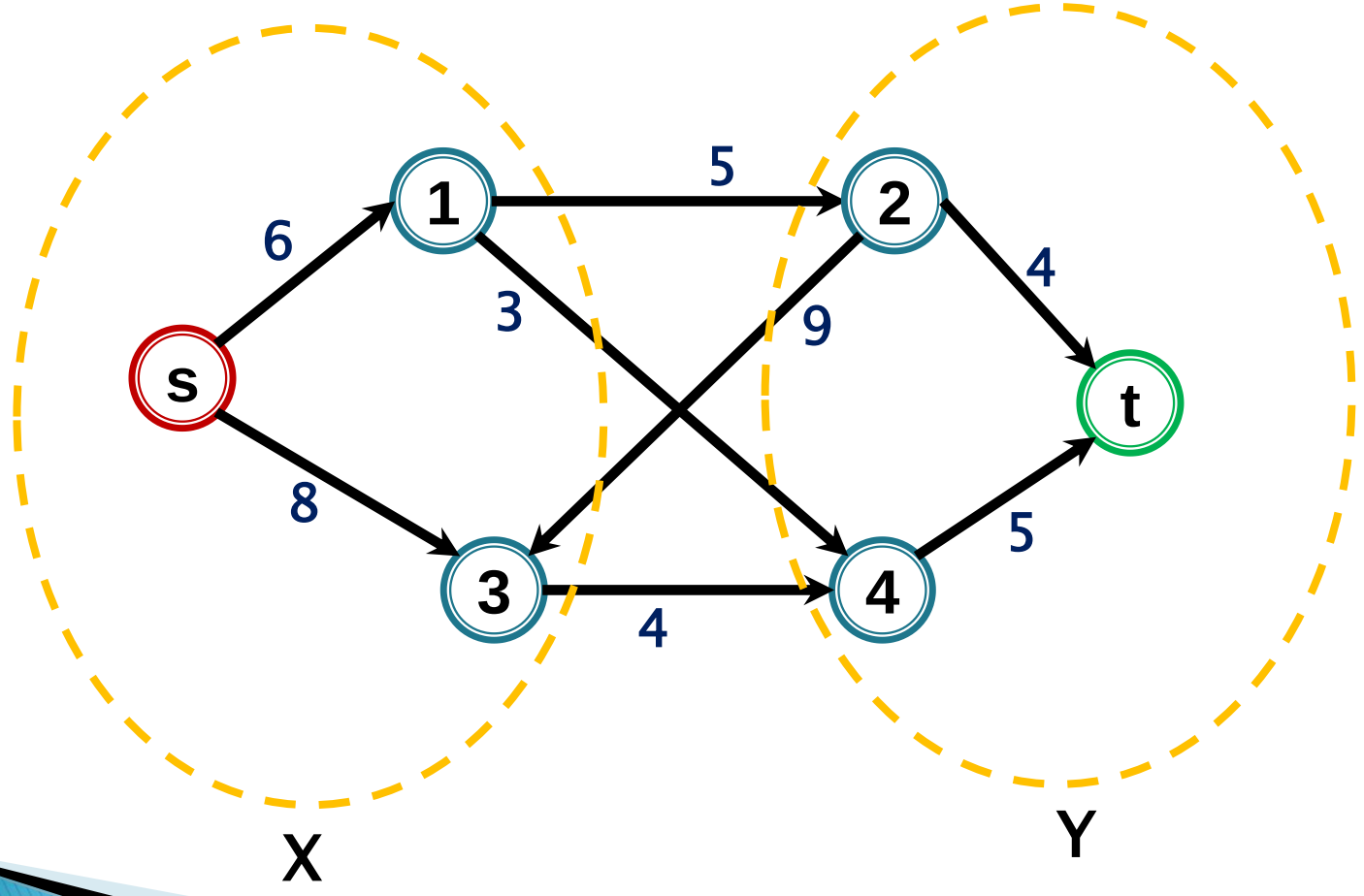
arc în sens direct,
mai putem trimite 5-3=2 unități



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)

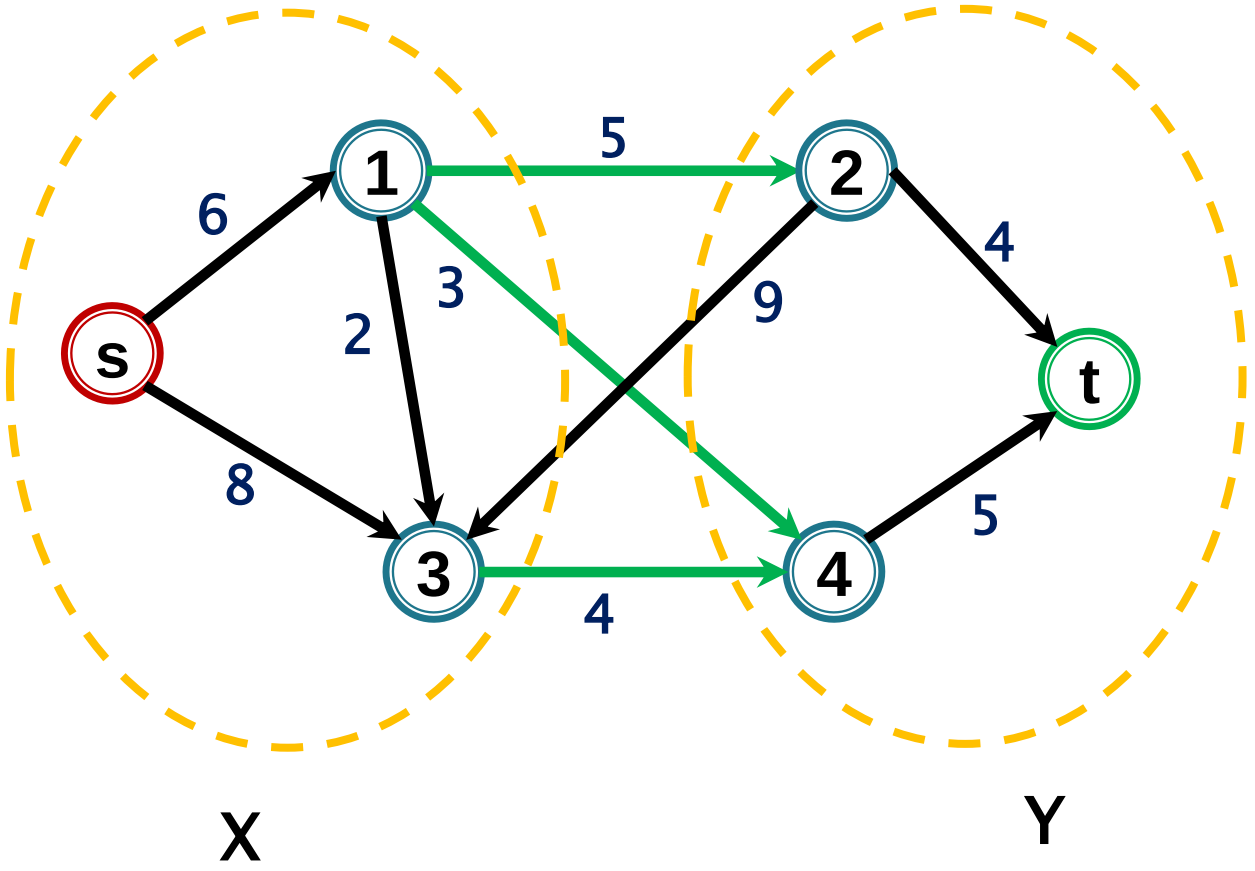
Fie N = (G, {s}, {t}, l, c) o reţea

- O **tăietură** K= (X, Y) în reţea este o (bi)partiţie (X, Y) a mulţimii vârfurilor V astfel încât s ∈ X şi t ∈ Y



Fie K= (X, Y) o tăietură

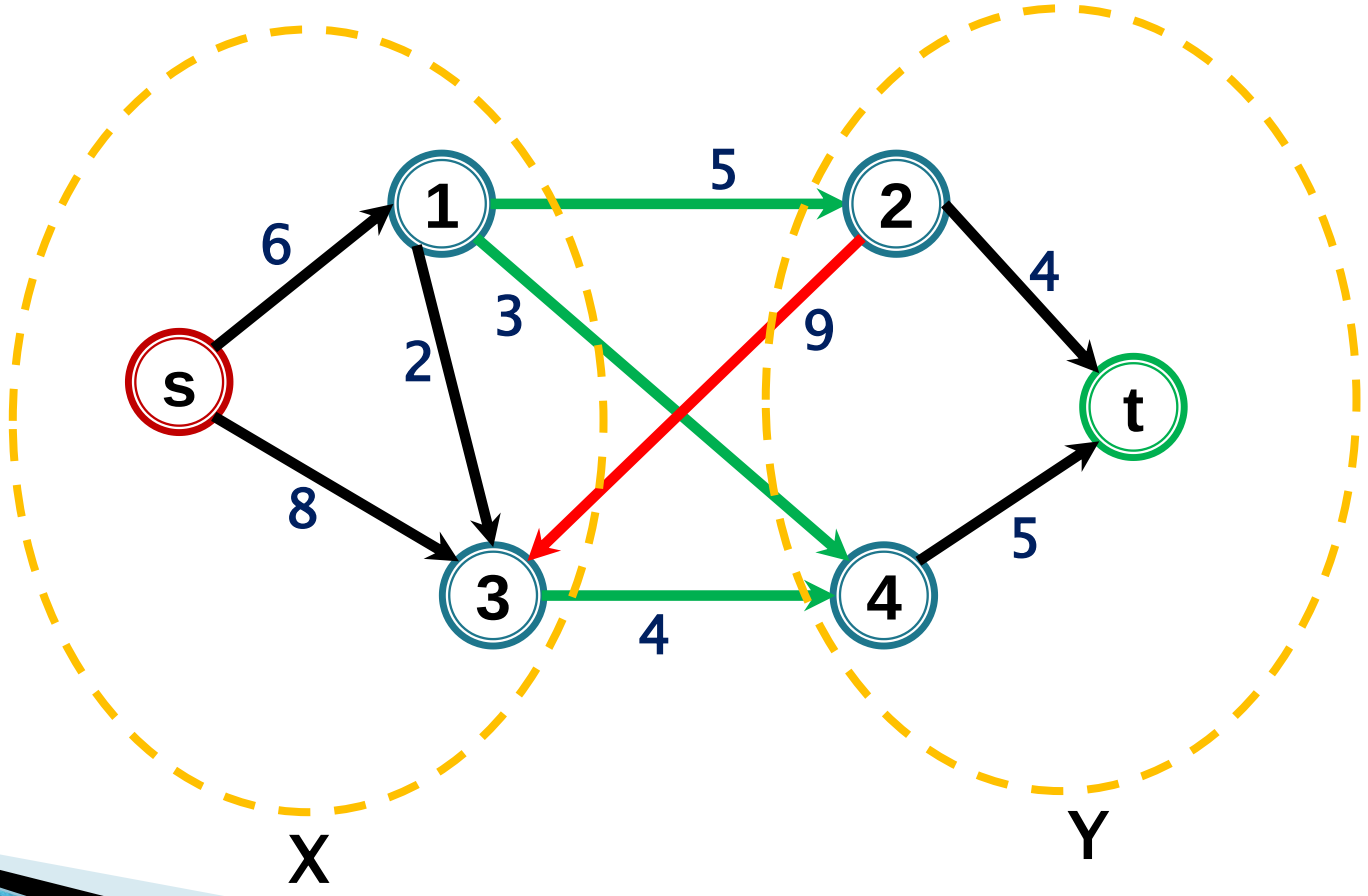
- **Capacitatea tăieturii** K = (X, Y)



$$c(K) = c(X, Y) = 5 + 3 + 4 = 12$$

Fie K= (X, Y) o tăietură

- $xy \in E$ cu $x \in X, y \in Y$ = **arc direct** al lui K
- $yx \in E$ cu $x \in X, y \in Y$ = **arc invers** al lui K



- Vom demonstra că pentru orice flux f și orice tăietură K

$$val(f) \leq c(K)$$

- Dacă avem **egalitate** $\Rightarrow$ f flux maxim, K tăietură minimă

(MAX-FLOW, MIN-CUT)

Algoritmul Ford-Fulkerson

Definim noțiunile necesare descrierii și studiului algoritmului:

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

Fie N = (G, {s}, {t}, l, c) o reţea

- Un **lanţ** este o succesiune de vârfuri **distincte** și arce din G

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

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}, l, c) o reţea

- Un **lanţ** este o succesiune de vârfuri **distincte** și arce din G

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

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)

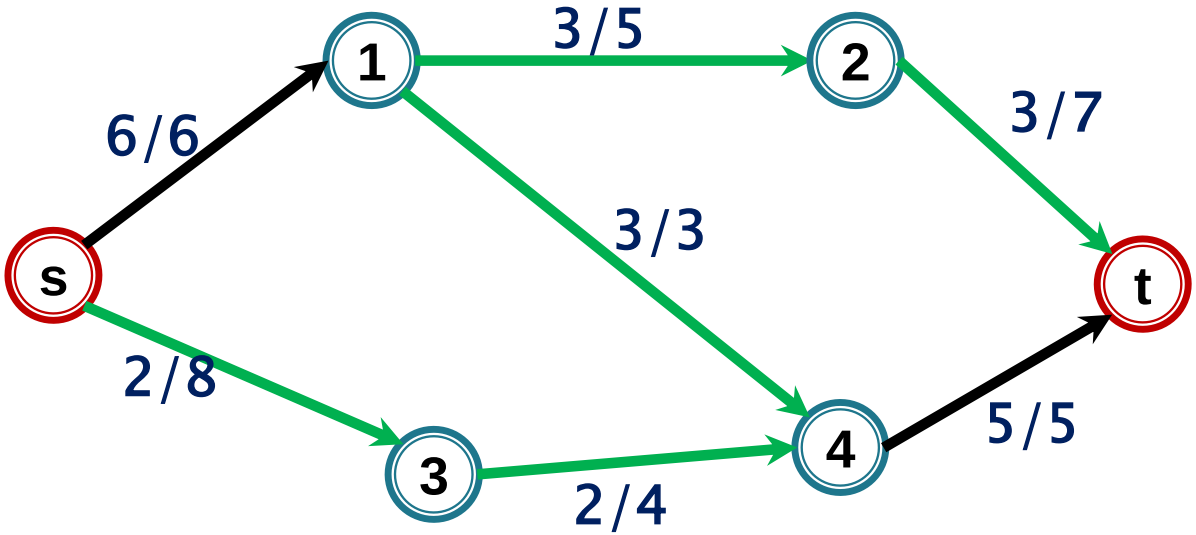
Dacă

- $e_i = v_{i-1}v_i \in E(G)$, 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 lanţului P**

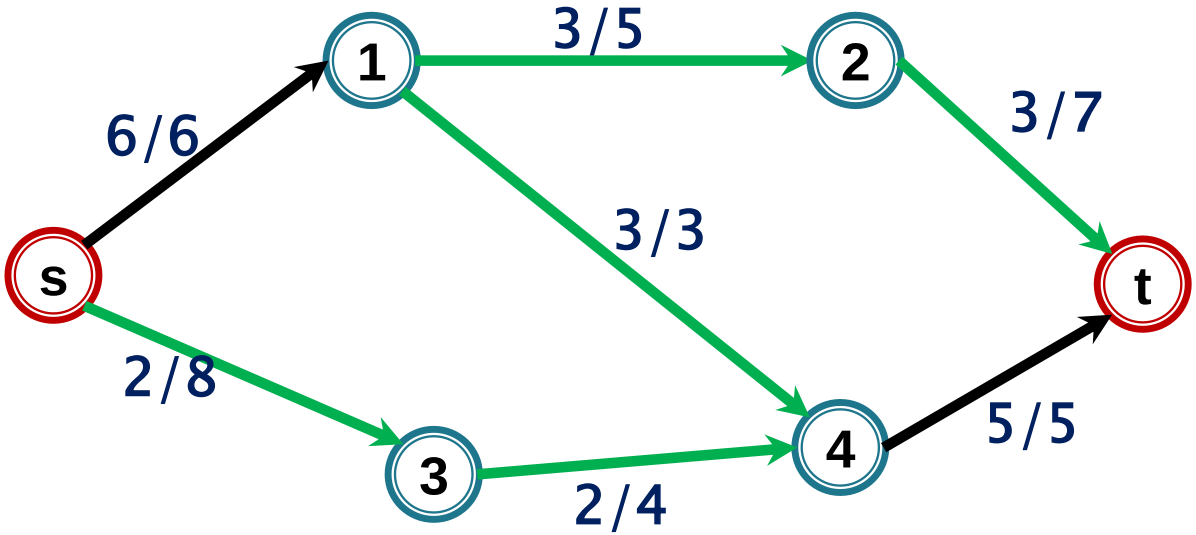
$$P = [v_0, v_1, \ldots, v_{k-1}, v_k]$$

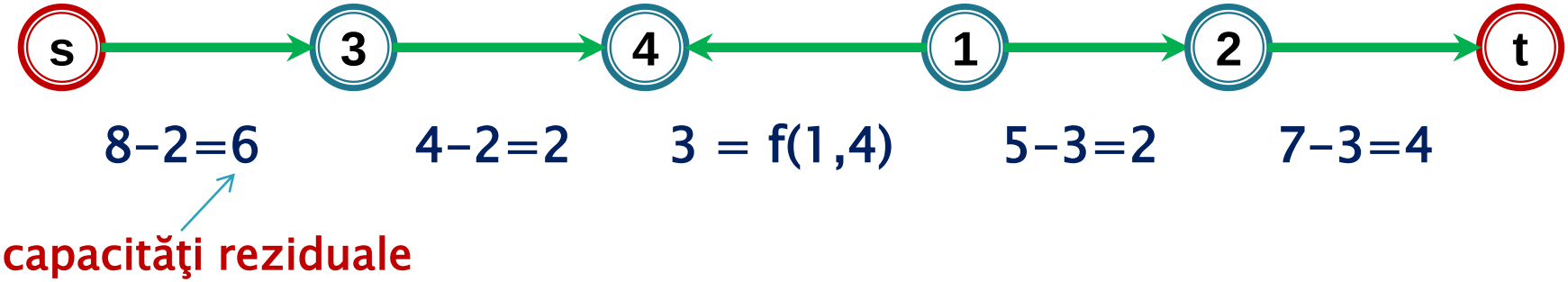
- Fie N reţea, f flux în N, P un lanţ
- Asociem fiecărui arc *e* din P o pondere, numită **capacitate reziduală** în P



capacităţi reziduale?

- Fie N reţea, f flux în N, P un lanţ
- Asociem fiecărui arc *e* din P o pondere, numită **capacitate reziduală** în P





- Fie N reţea, f flux în N, P un lanţ
- 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 lanţului P

- **Capacitatea reziduală** a unui lanţ P este

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

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

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

- **Capacitatea reziduală** a lanţului P

i(P) = ?

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

- **Capacitatea reziduală** a lanţului P

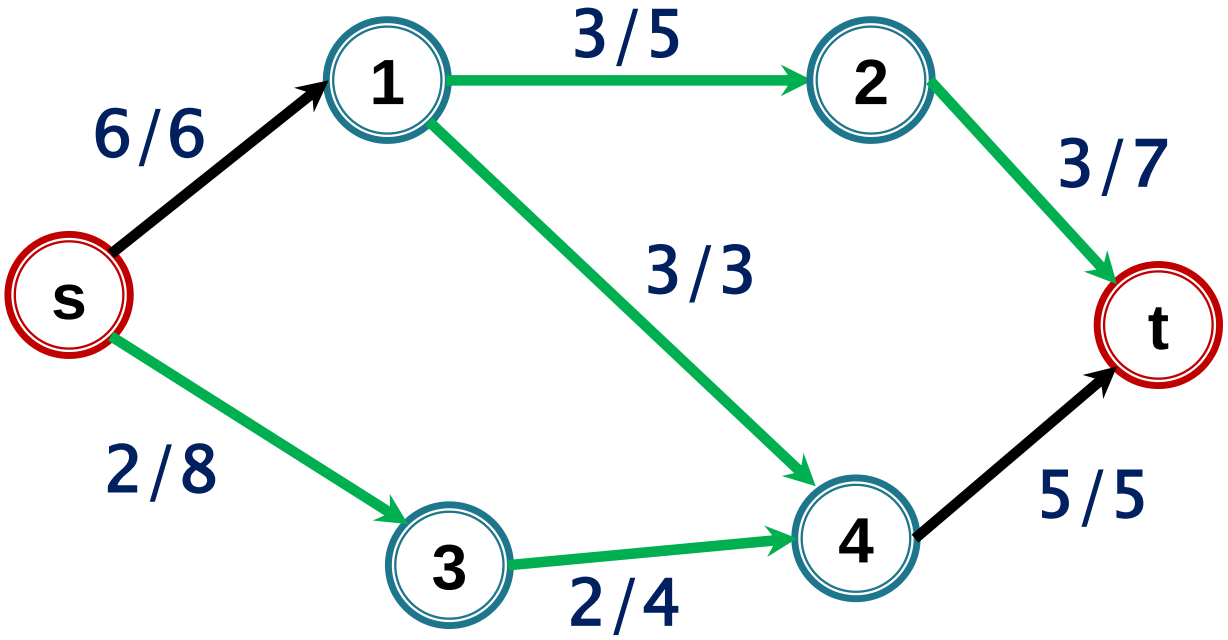
$$i(P) = \min\{6, 2, 3, 2, 4\} = 2$$

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

Fluxuri în rețele de transport

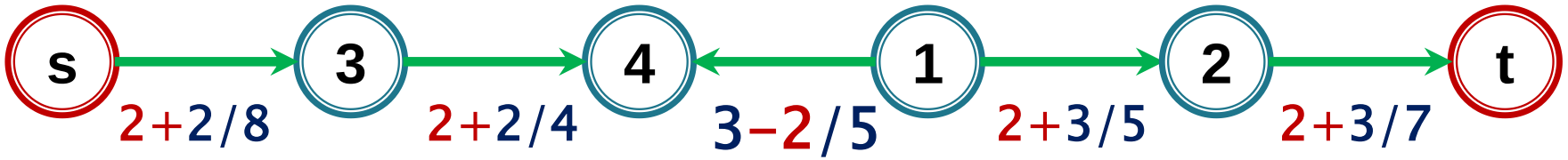
- Fie N– rețea, f flux în N, P un s–t lanţ **f–nesaturat**.
- **Fluxul revizuit de–a lungul lanţului P** se defineşte ca fiind $f_P : E \to \mathbb{N}$,

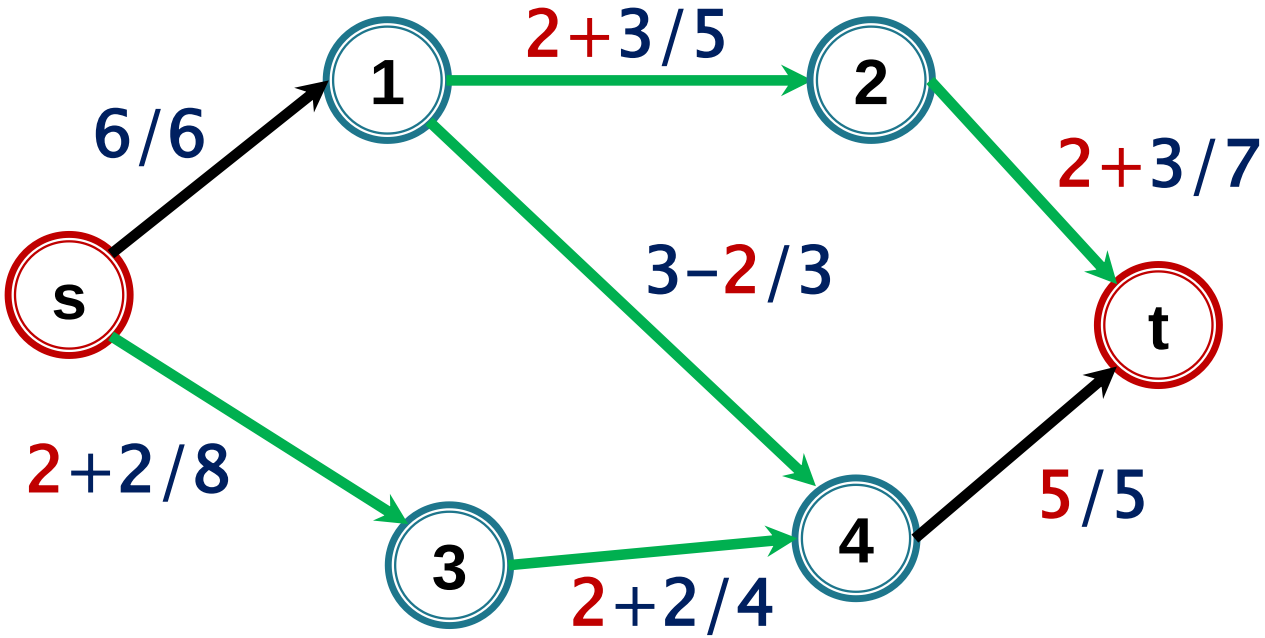
$$f_P(e) = \begin{cases} f(e) + i(P), & \text{dacă } e \text{ este arc direct în } P \\ f(e) - i(P), & \text{dacă } e \text{ este arc invers în } P \\ f(e), & \text{altfel} \end{cases}$$

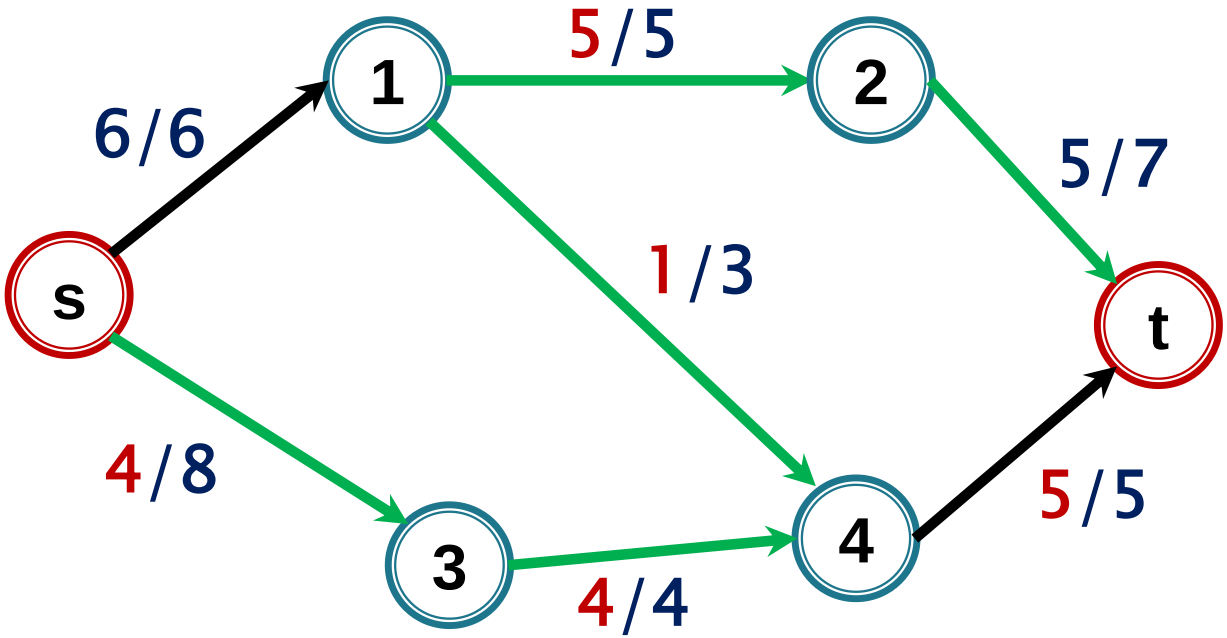


Considerăm s–t lanţul P evidenţiat şi revizuim fluxul

i(P) = 2




1
2
s
t
3
4
6/6
2+3/5
2+3/7
3−2/3
2+2/8
5/5
2+2/4



Fluxul după revizuirea de-a lungul lanţului P

- Proprietăţi ale fluxului revizuit

Proprietăţi ale fluxului revizuit

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

Fie P un s-t lanţ f-nesaturat în G și f_p fluxul revizuit de-a lungul lanţului P. Atunci

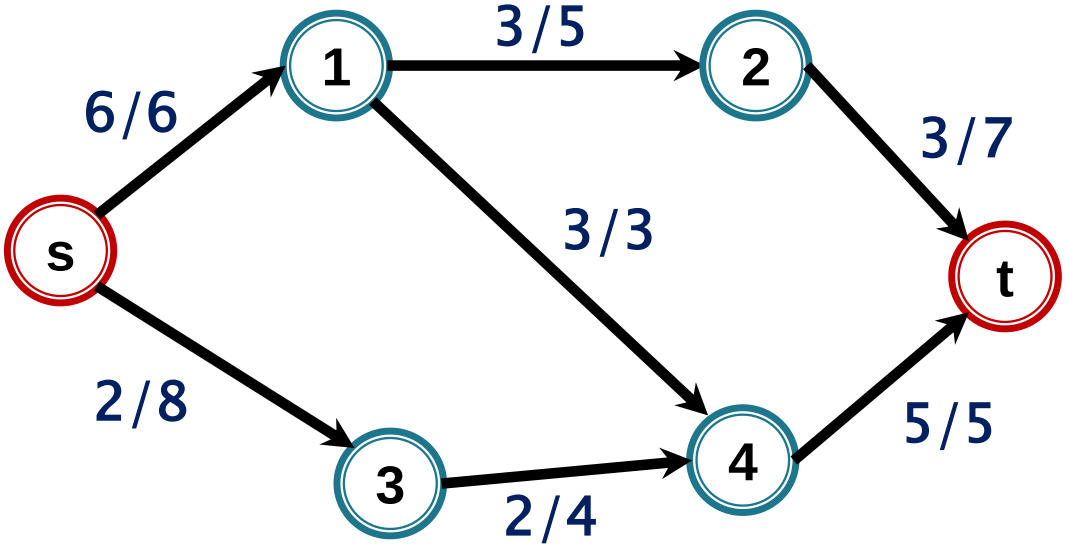
- f_p este flux în G

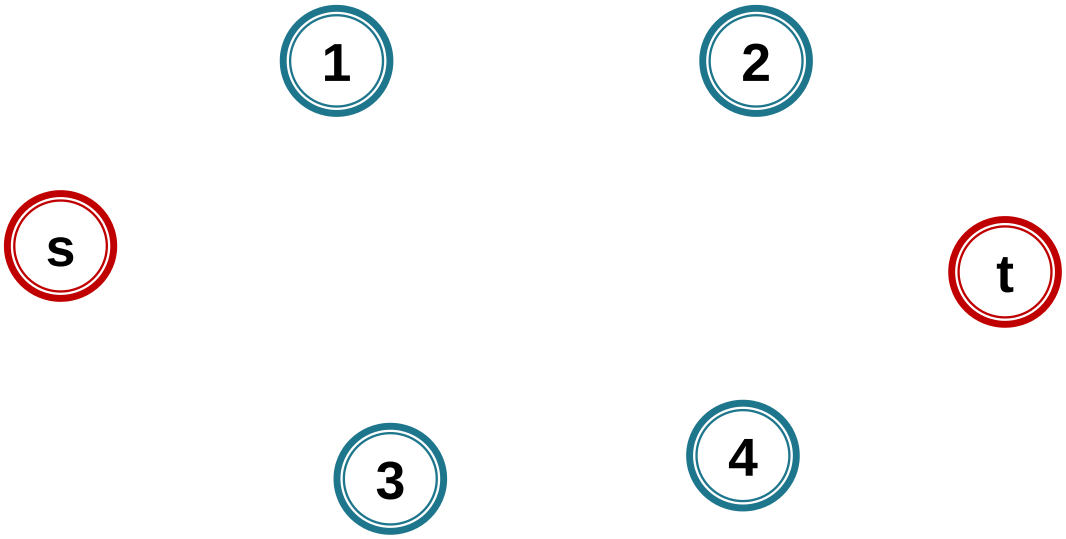
și

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

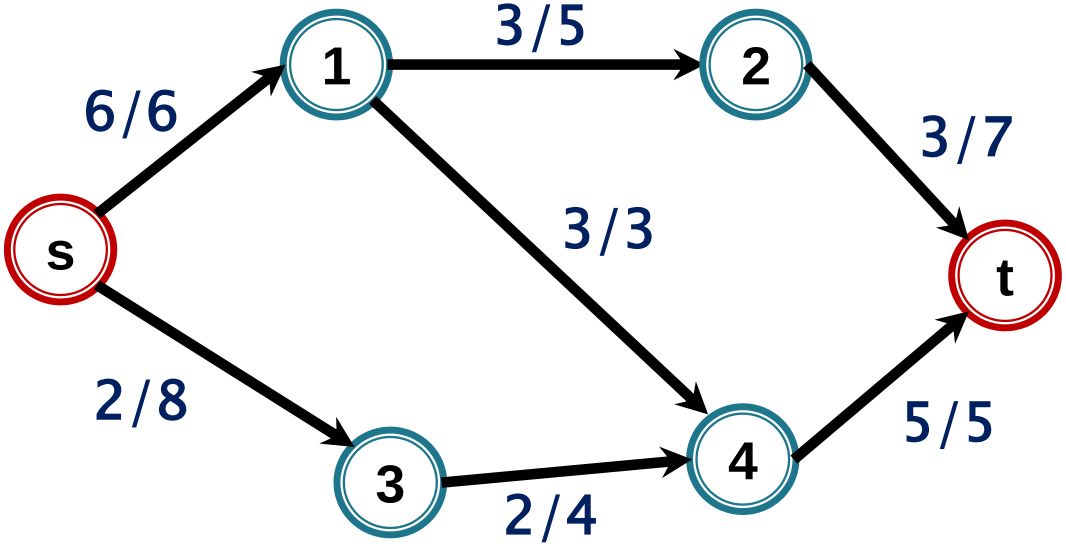
(Temă)

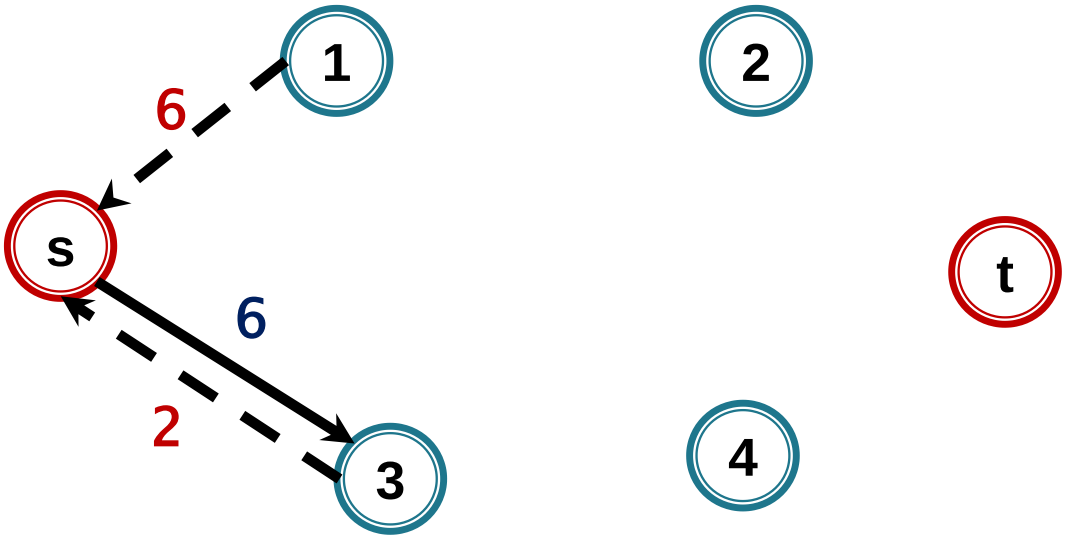
Graf rezidual



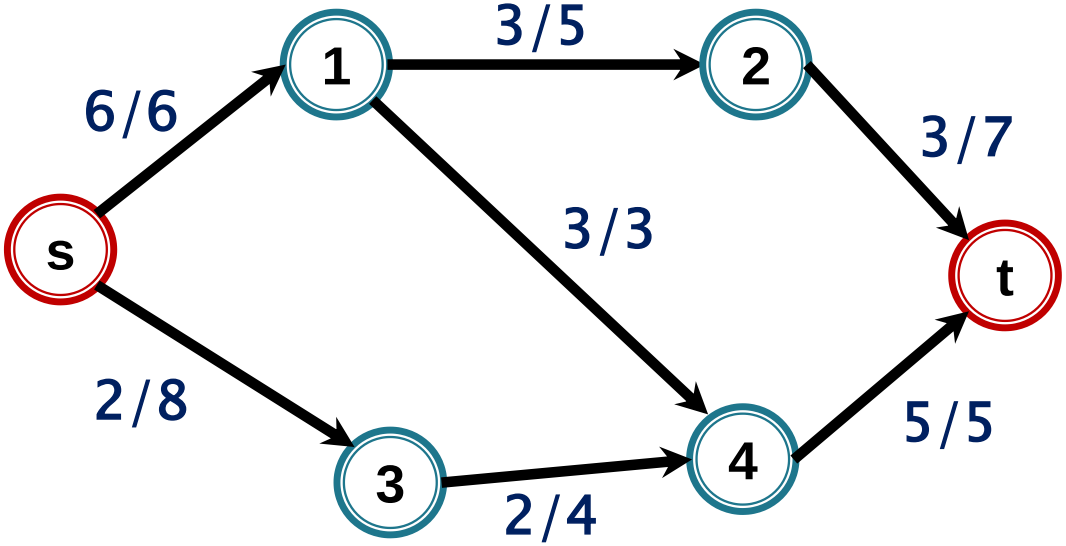


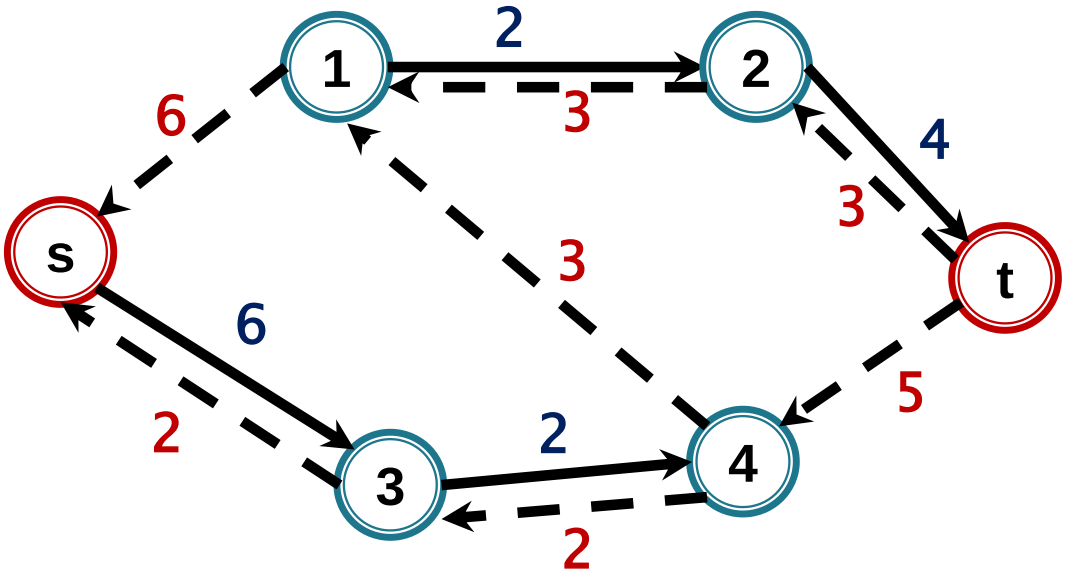
Graf rezidual



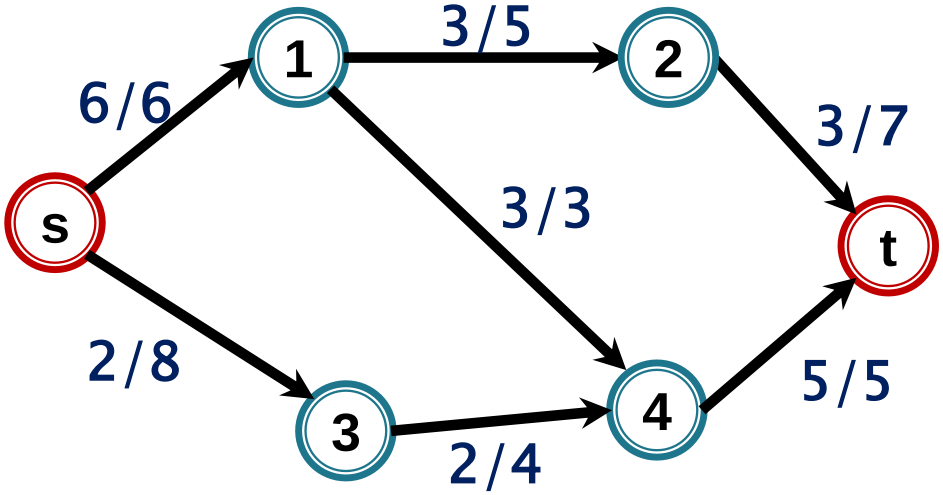


Graf rezidual

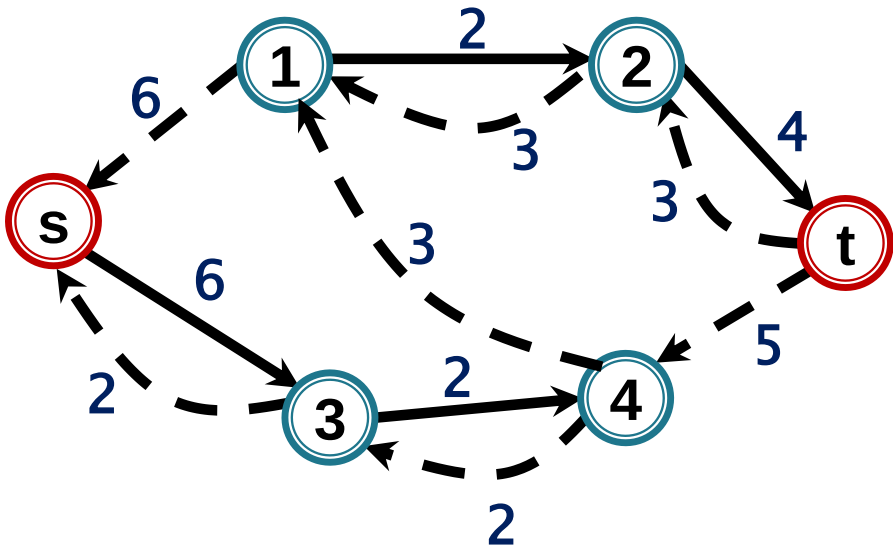




Rețeaua de transport



Graful rezidual

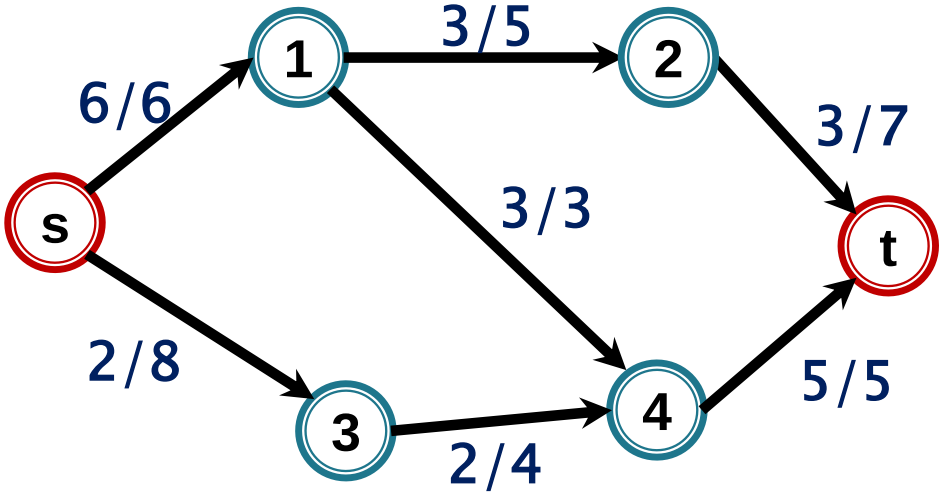


(Multi)graful rezidual G_f asociat fluxului f = graf orientat ponderat, cu funcția de pondere (de capacitate) c_f construit astfel:

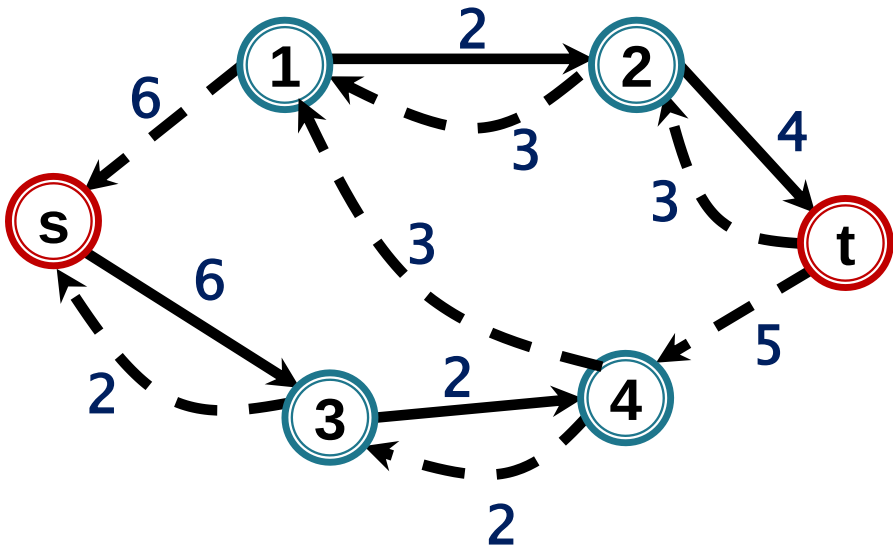
- $V(G_f) = V(G)$
- Pentru fiecare arc $e = uv$ cu $c(e) - f(e) > 0$,

- Pentru fiecare arc $e = uv$ cu $f(e) > 0$,

Rețeaua de transport



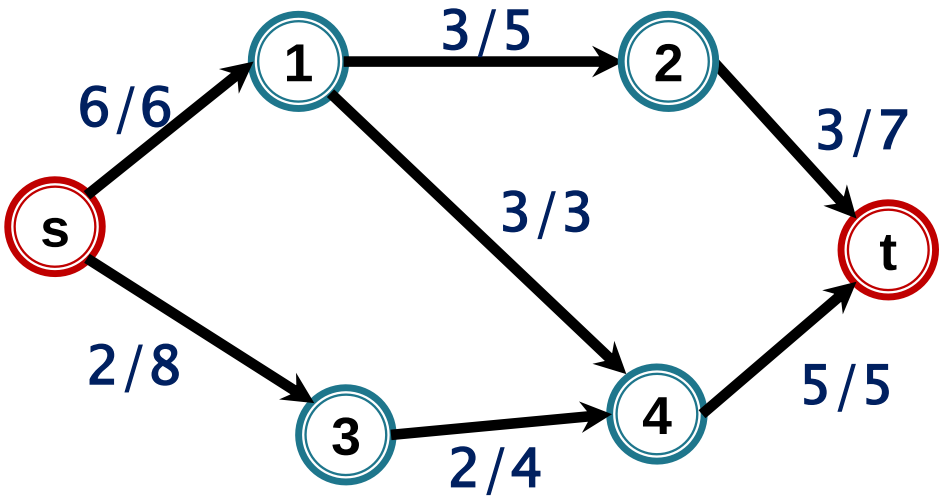
Graful rezidual



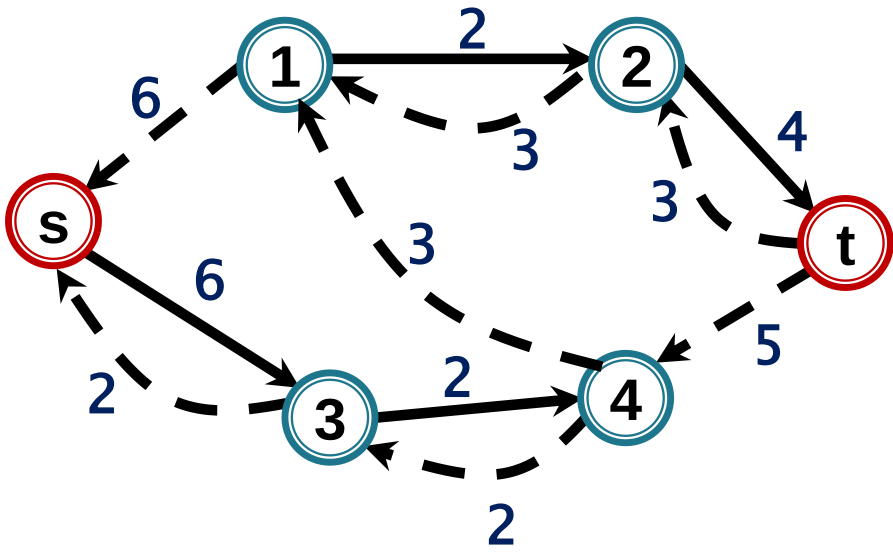
(Multi)graful rezidual G_f asociat fluxului f = graf orientat ponderat, cu funcția de pondere (de capacitate) c_f construit astfel:

- $V(G_f) = V(G)$
- 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



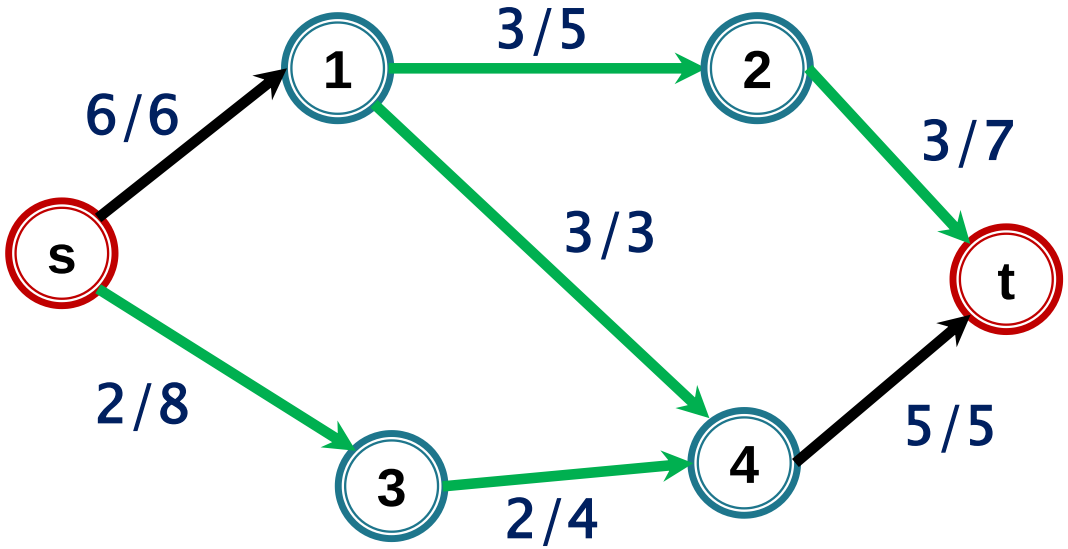
Graful rezidual



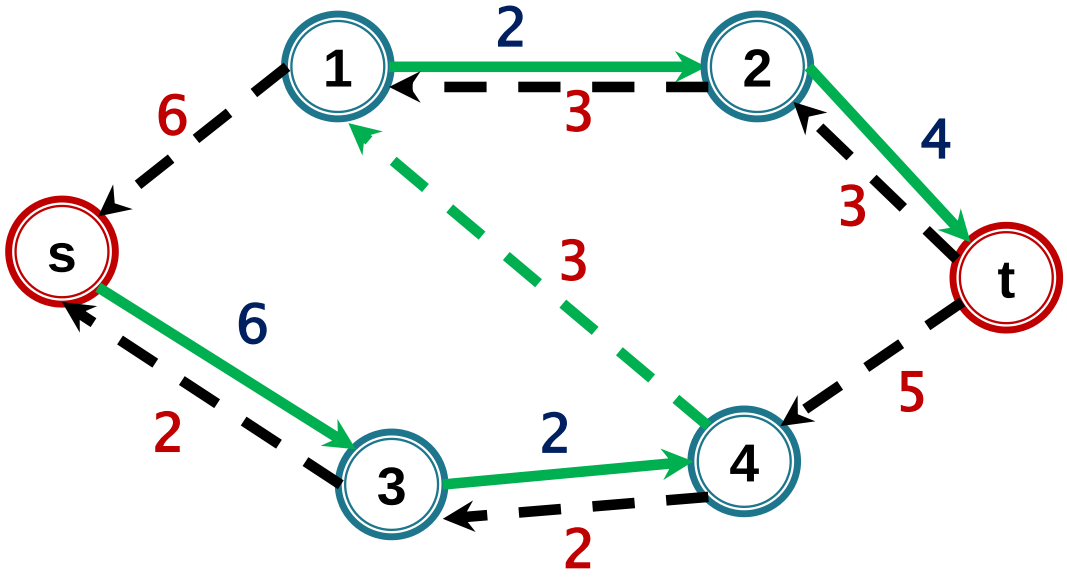
(Multi)graful rezidual G_f asociat fluxului f = graf orientat ponderat, cu funcția de pondere (de capacitate) c_f construit astfel:

- $V(G_f) = V(G)$
- 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

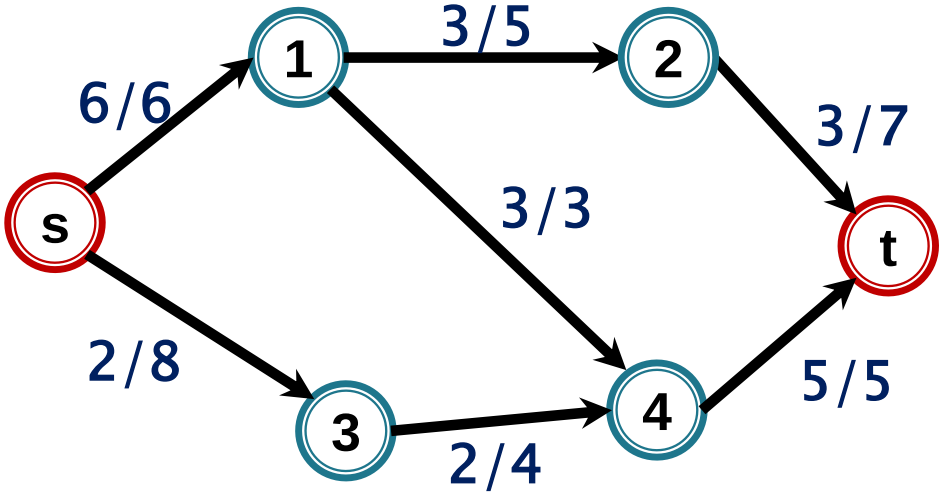


s–t lanț f–nesaturat

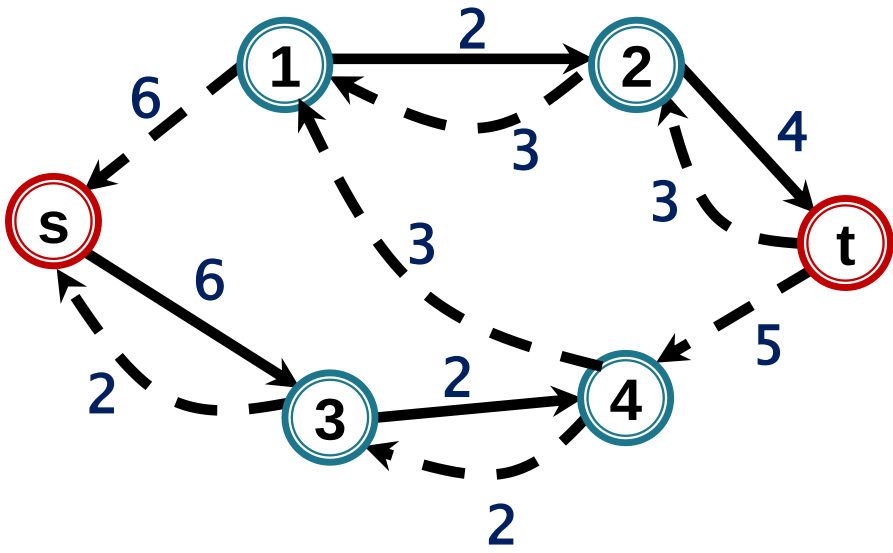


s–t drum în graful rezidual

Rețeaua de transport



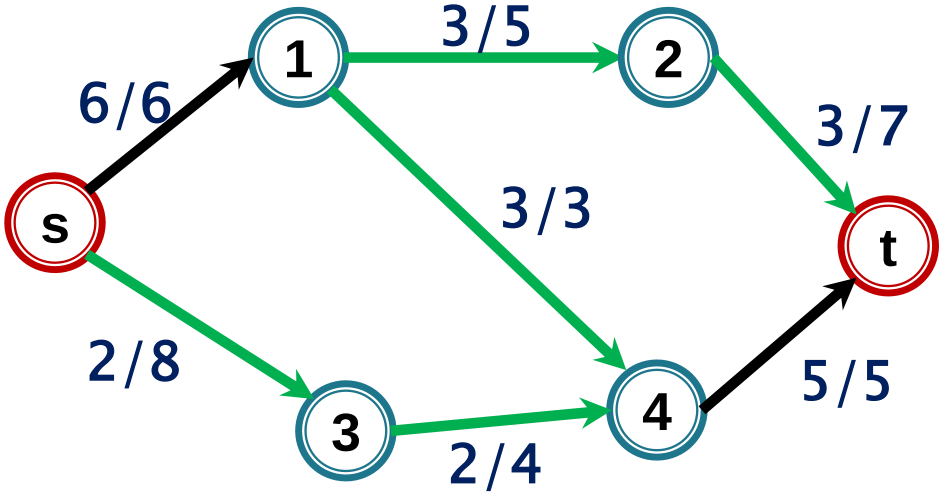
Graful rezidual



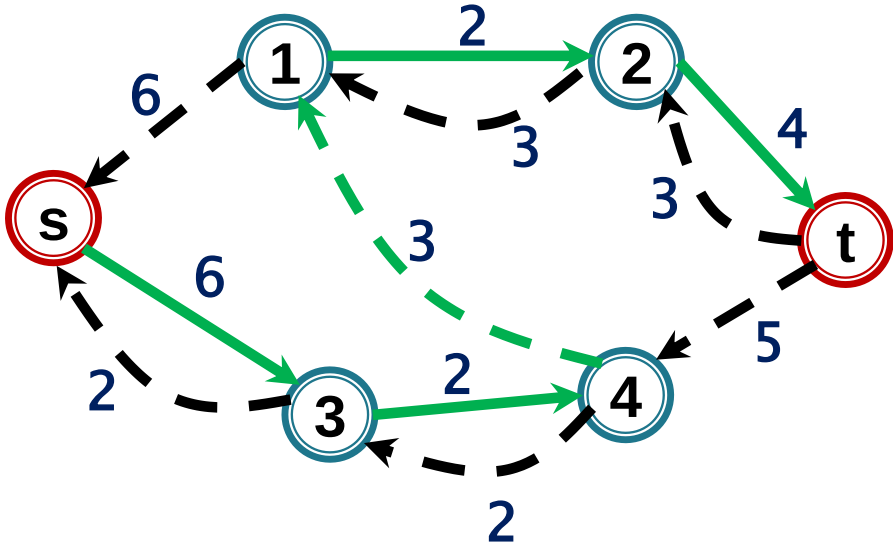
Capacitatea unui **drum** P în G_f:

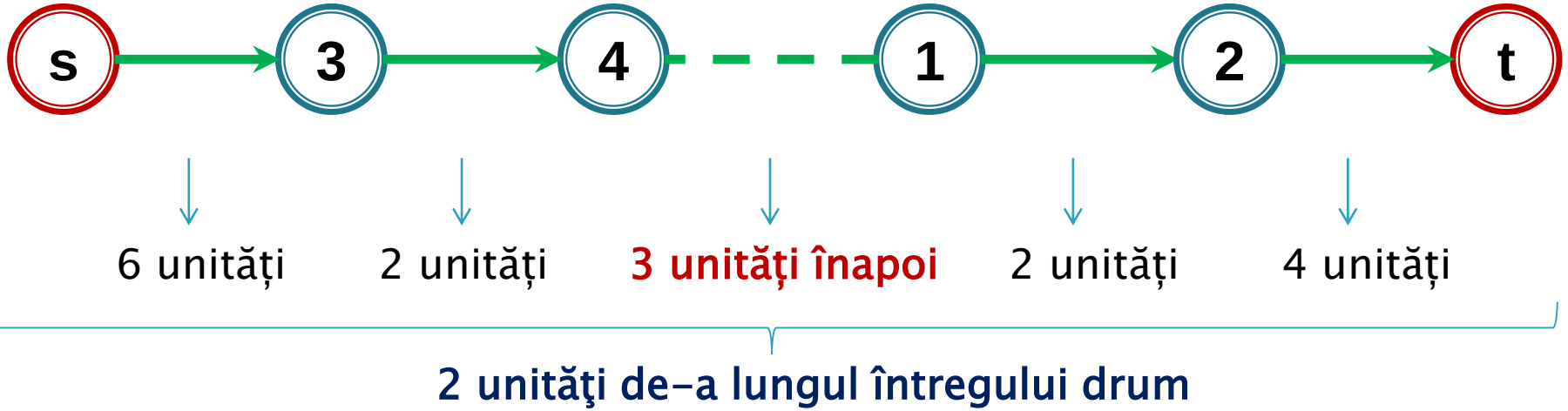
$$c_f(P) = \min\{c_f(e) \mid e \in E(P)\}$$

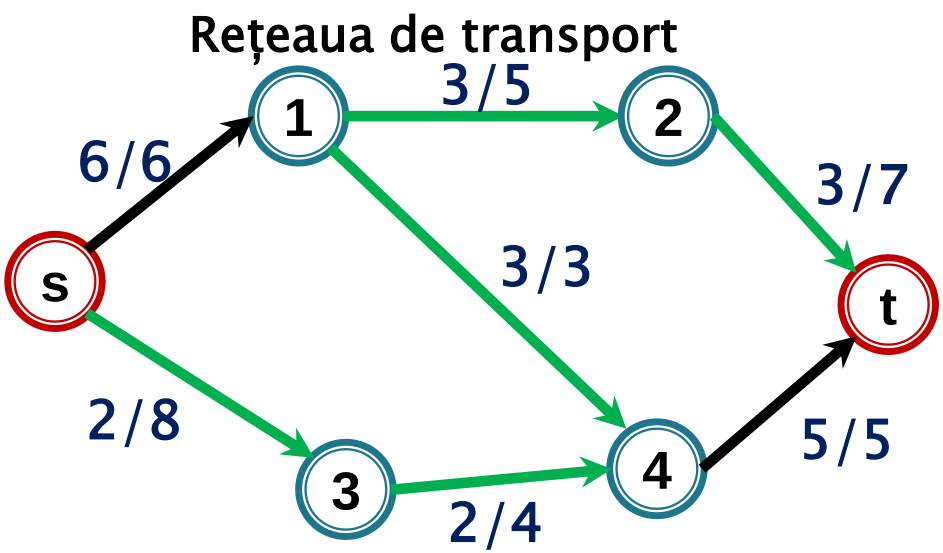
Rețeaua de transport

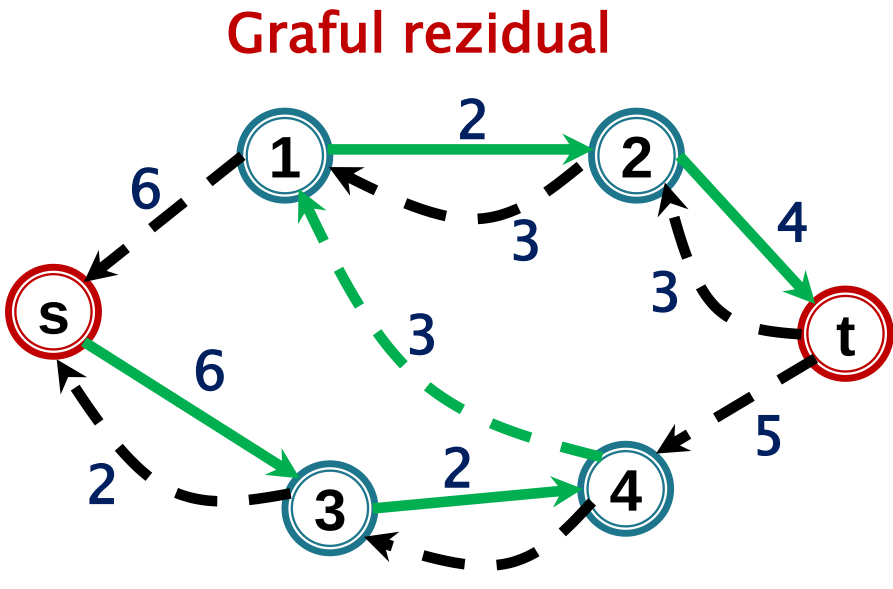


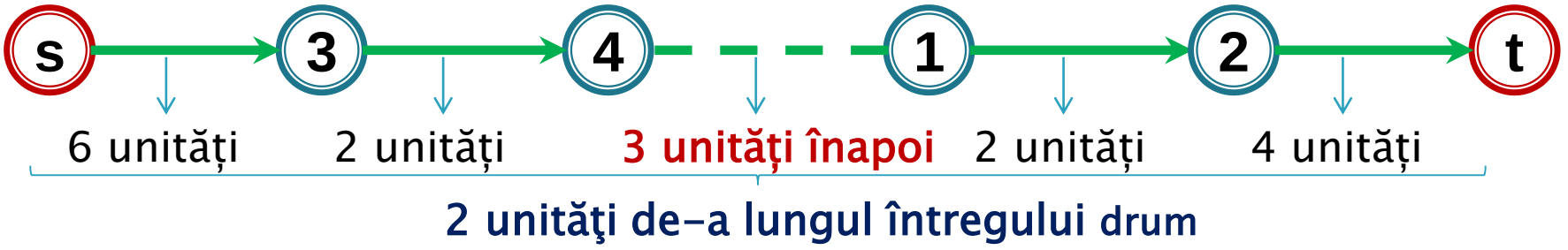
Graful rezidual







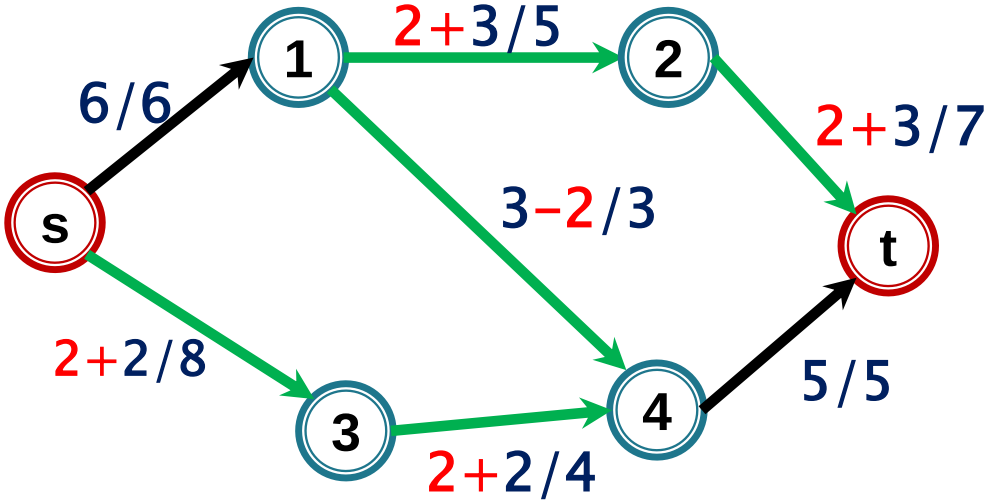




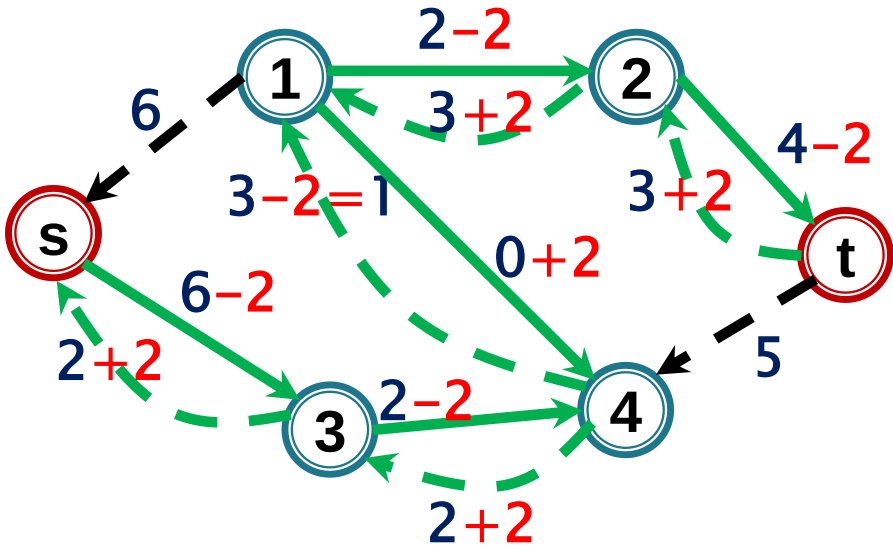
- P este un s -t **lanț f-nesaturat** în G ⇔ P este s- t **drum** în G_f
- În plus i(P) (în G) = c_f (P) (în G_f)
- Un arc e este **arc invers** în lanțul P în G ⇔

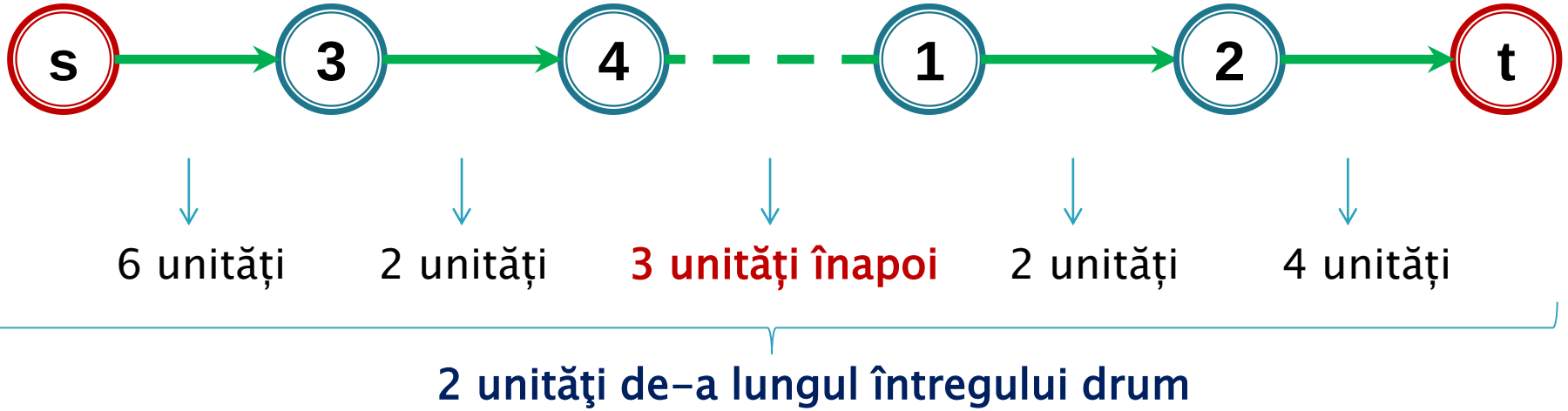
  $\mathbf{e^{-1}}$ este arc în drumul P din G_f

Rețeaua de transport

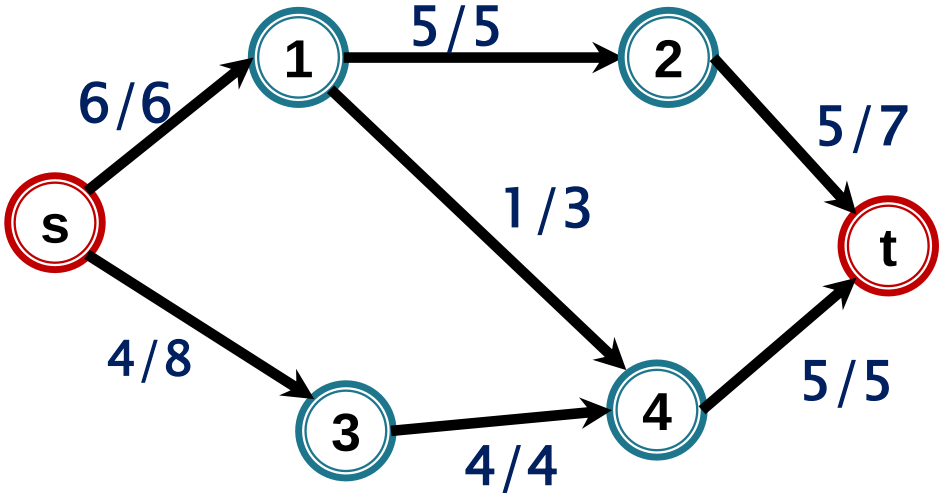


Graful rezidual

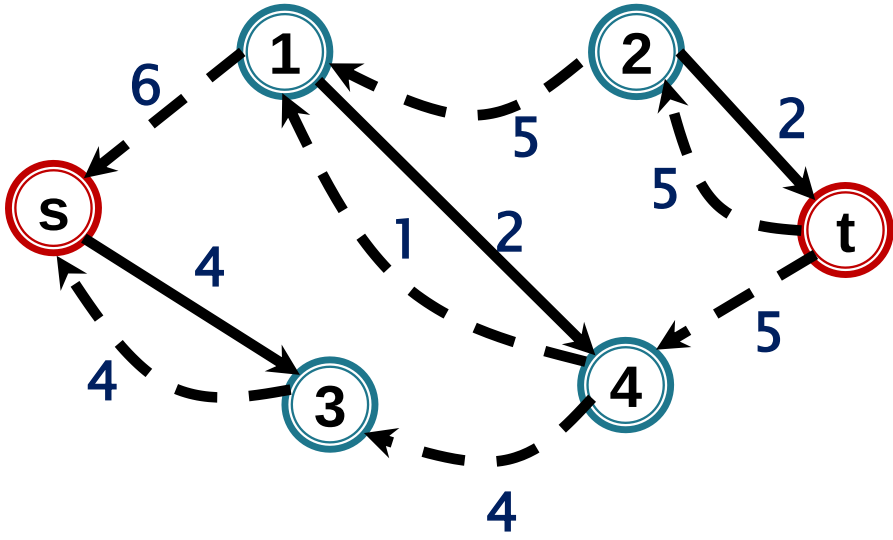




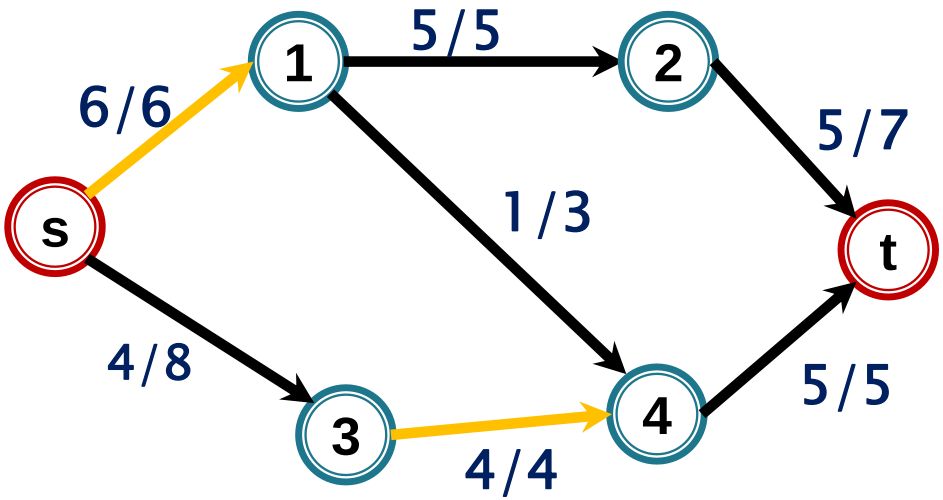
Rețeaua de transport



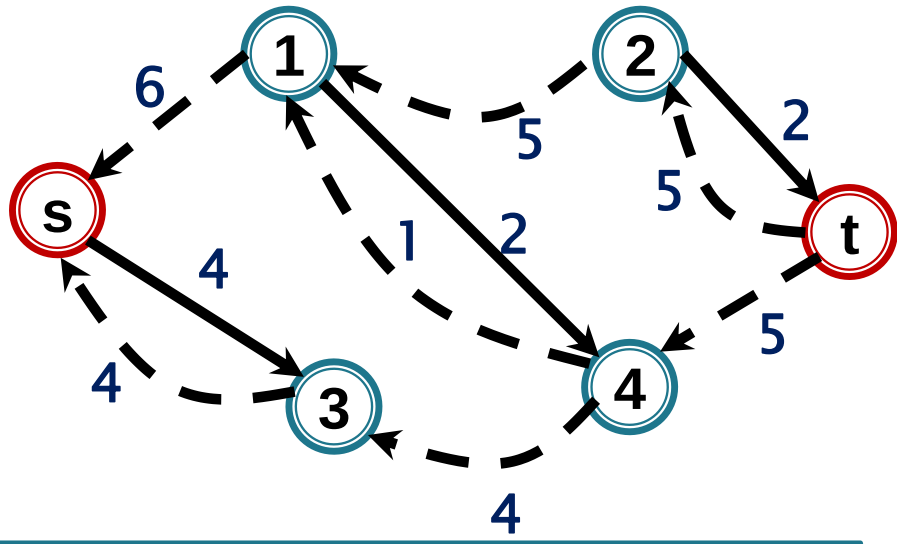
Graful rezidual



Rețeaua de transport



Graful rezidual



tăietură saturată ⇔ nu mai există s-t drum în graful rezidual

⇔ s-t flux maxim

este tăietură minimă

Algoritmul FORD-FULKERSON

Pseudocod

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

- Fie f un flux în N (de exemplu f ≡ 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 ≡ 0 fluxul vid:

  $f(e) = 0, \forall\ e \in E$))

- Cât timp **există un s-t lanţ f-nesaturat P în G**
  - determină un astfel de lanţ P
  - revizuieşte fluxul f de-a lungul lanţului 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 = mulțimea 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 lanţ f-nesaturat?
  - Criteriul după care construim lanţul f-nesaturat influenţează numărul de etape (iteraţii 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 $\Rightarrow$

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

Algoritm FORD – FULKERSON

Complexitate

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

Algoritm FORD – FULKERSON

Complexitate

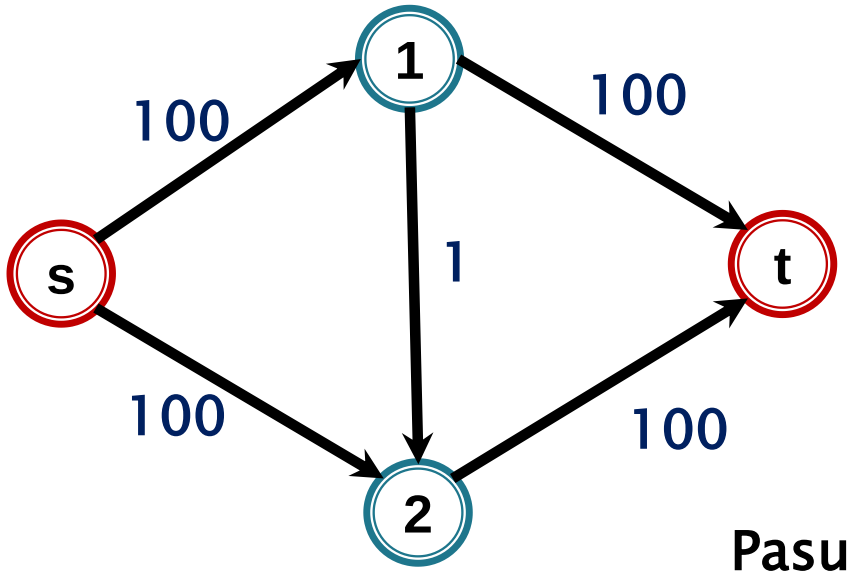
- **O(mL), unde L este capacitatea minimă a unei tăieturi**

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

- **O(nmC) unde**

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

Criteriul după care construim lanţul f-nesaturat influenţează 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 lanţ f-nesaturat?

Algoritmul Ford-Fulkerson

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

= **s-t drum în graful reziudal**

Algoritmul Ford-Fulkerson

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

- **Parcurgerea 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 şi considerând doar arce cu capacitatea reziduală pozitivă (în raport cu lanţurile construite prin parcurgere, memorate cu vectorul tata)

- **Alte criterii de construcţie lanţ $\Rightarrow$ alţi algoritmi**

Implementarea algoritmului FORD-FULKERSON

Varianta cu drumuri minime

$\Rightarrow$ 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 ⇔ 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 lanţ 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 lanţ nu există

  (şi **true** dacă l-a putut construi)

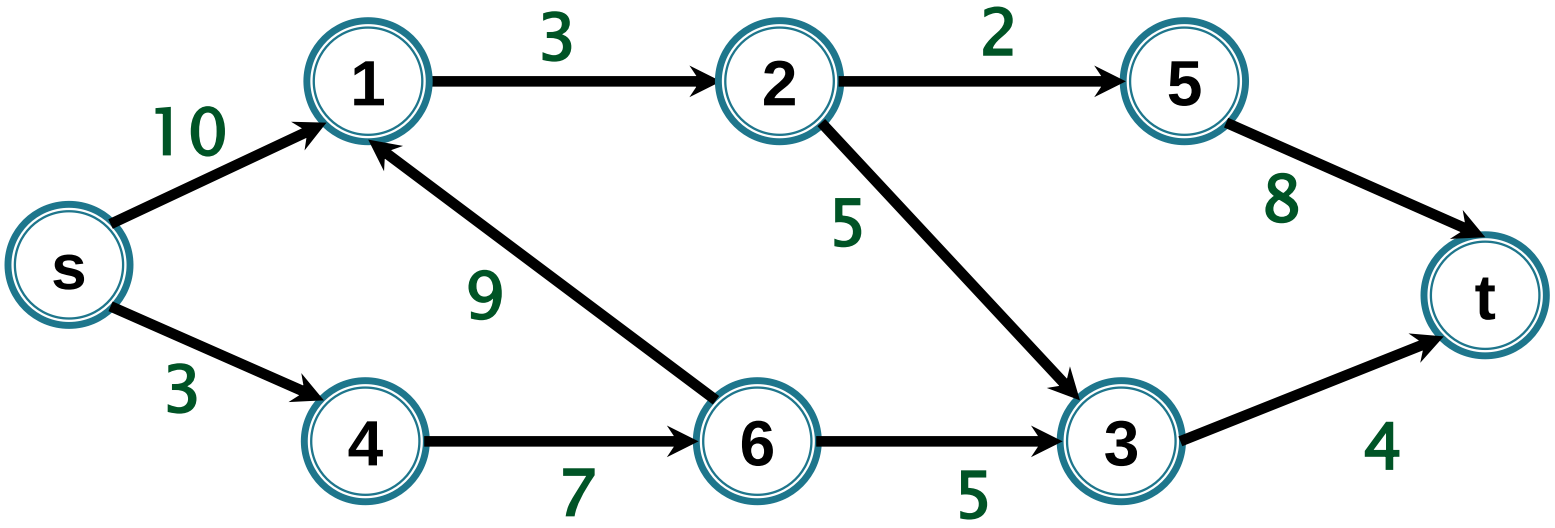
Implementare. Algoritmul Edmonds-Karp

revizuieste_flux_lant()

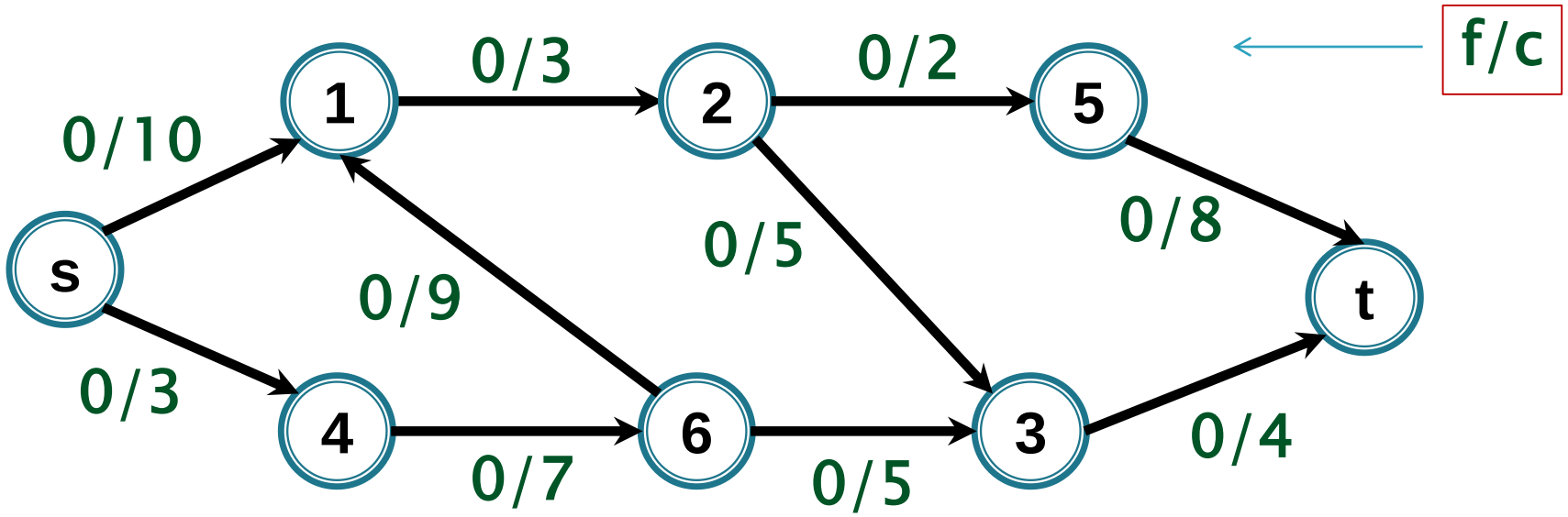
- **fie P s-t lanțul găsit în construieste_s-t_lant_nesat_BF()**
- **calculăm i(P)**
- **pentru fiecare arc *e* al lanțului P**
  - creștem cu i(P) fluxul pe *e* dacă este arc direct
  - scădem cu i(P) fluxul pe *e* dacă este arc invers

Exemplu

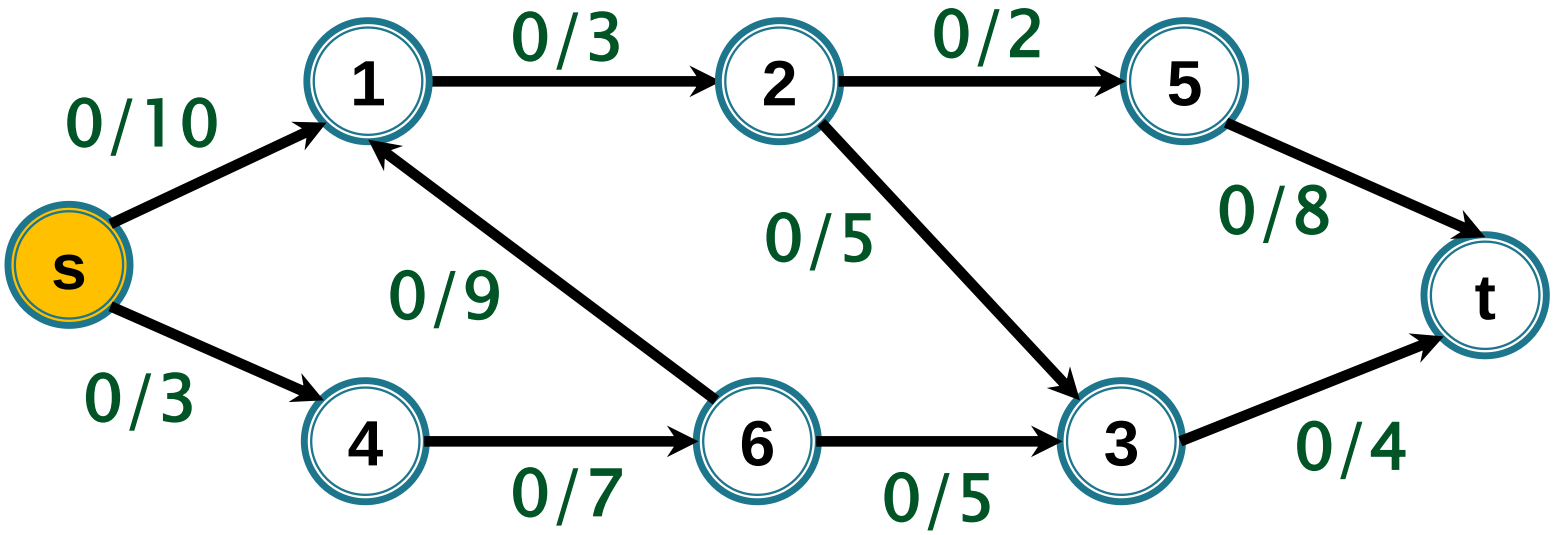
Algoritmul EDMONDS-KARP


1
2
5
s
t
4
6
3
10
3
2
9
5
8
3
7
5
4

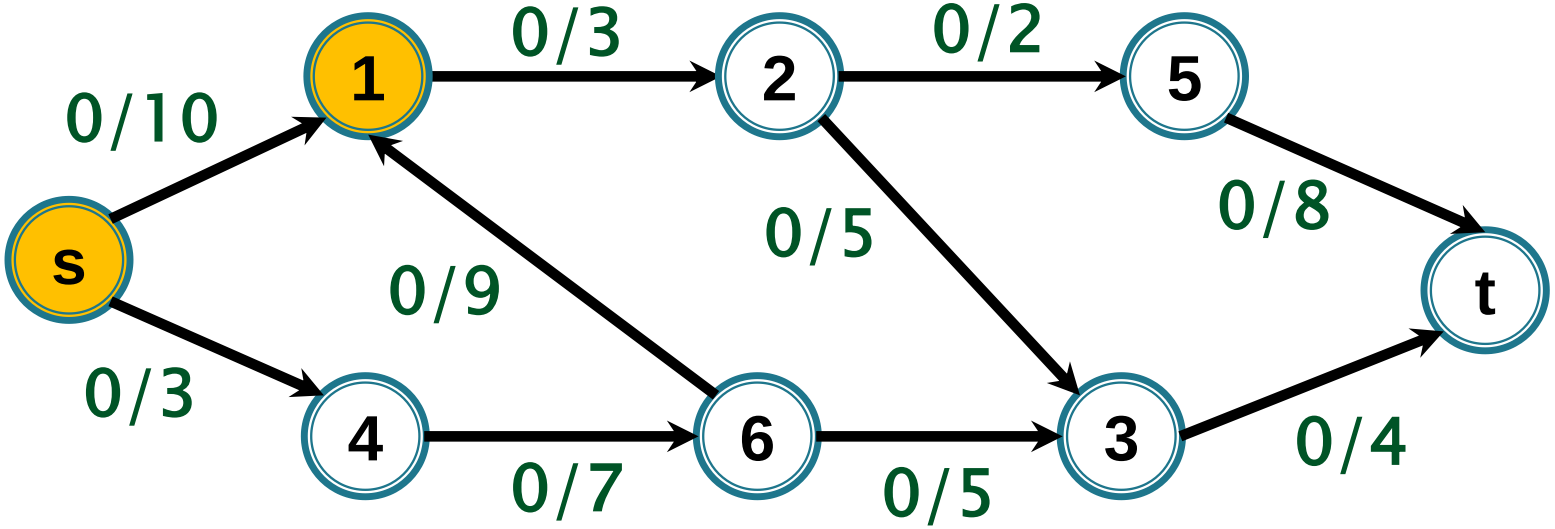
initializeaza_flux_nul


f/c
1
2
5
s
t
4
6
3
0/10
0/3
0/2
0/5
0/8
0/9
0/3
0/4
0/7
0/5

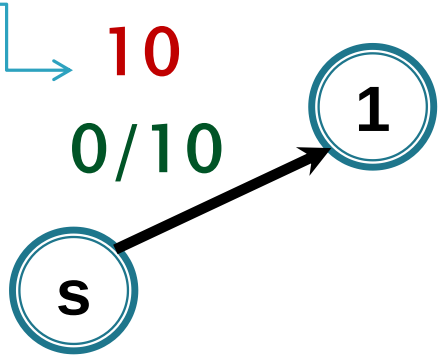
construieste_s-t_lant_nesat_BF

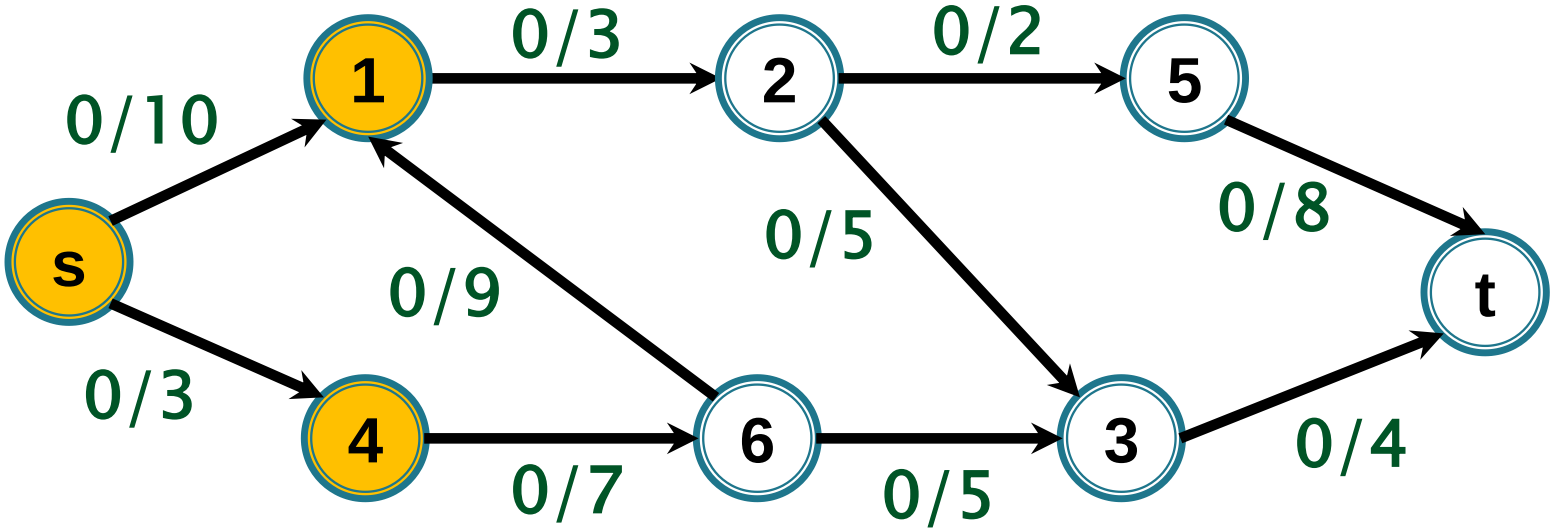

s
1
2
5
t
4
6
3
0/10
0/3
0/2
0/8
0/5
0/9
0/3
0/7
0/5
0/4

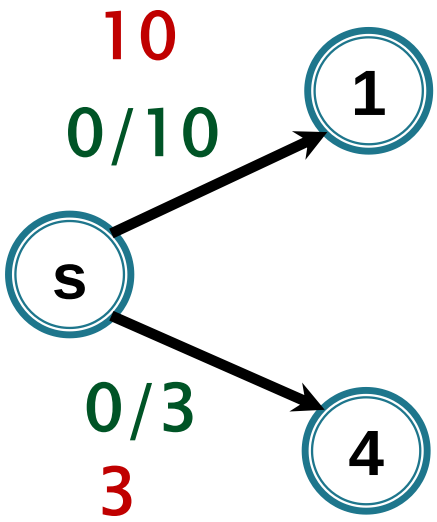

s

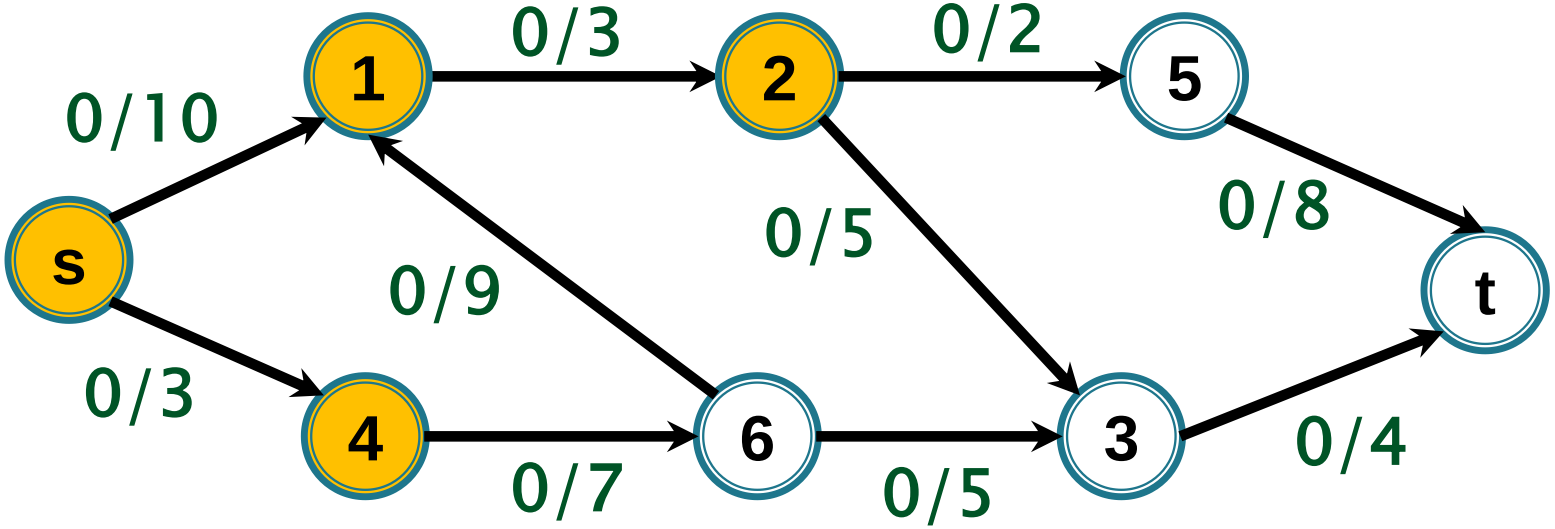


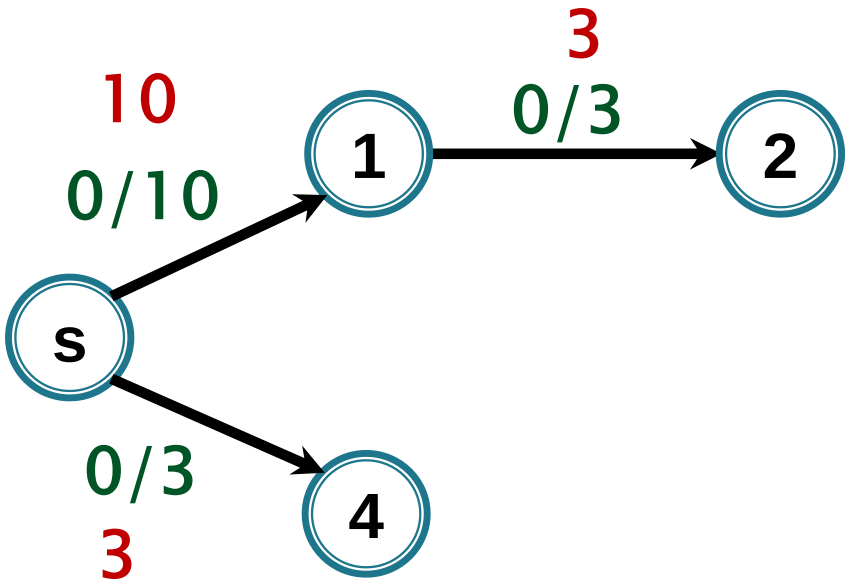
Capacitatea reziduală

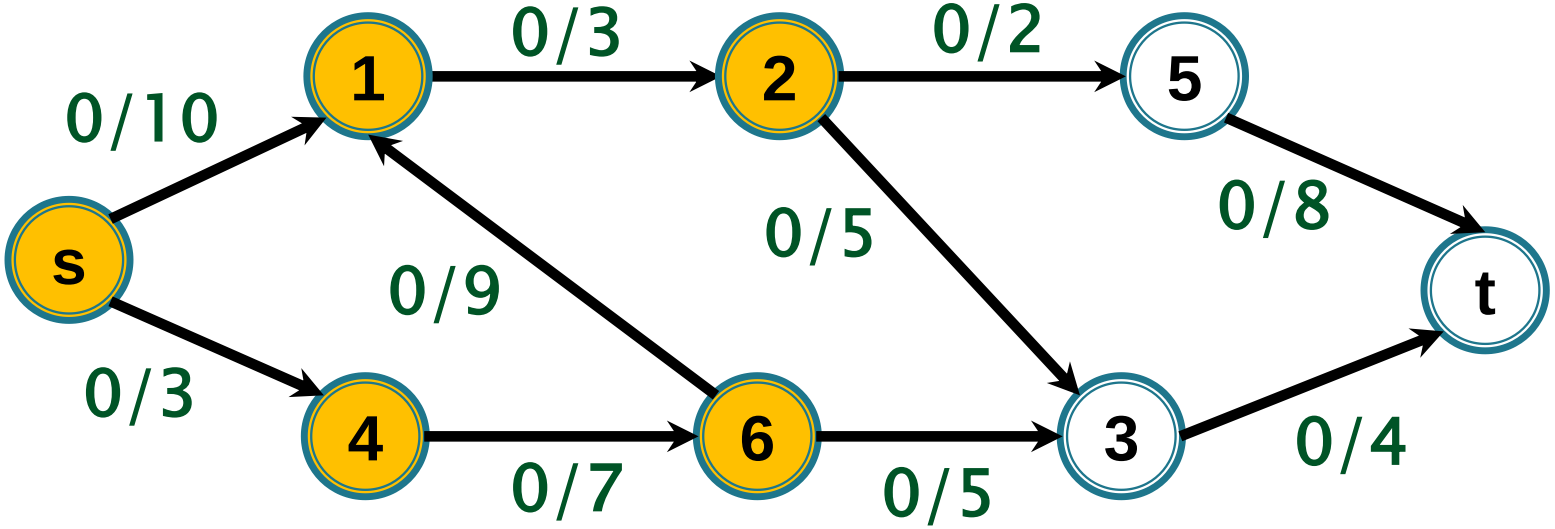


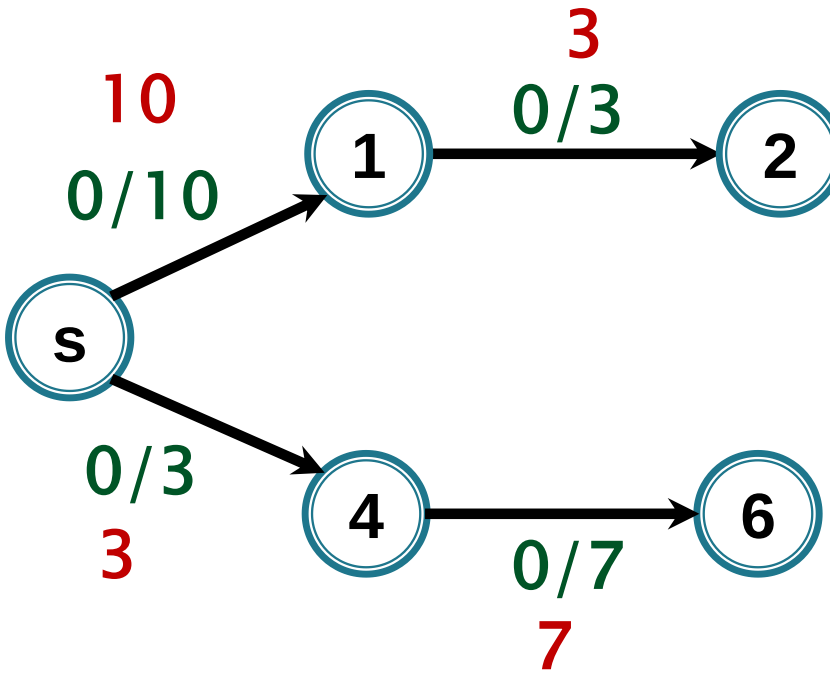

1
2
5
s
t
4
6
3
0/3
0/2
0/10
0/5
0/8
0/9
0/3
0/4
0/7
0/5

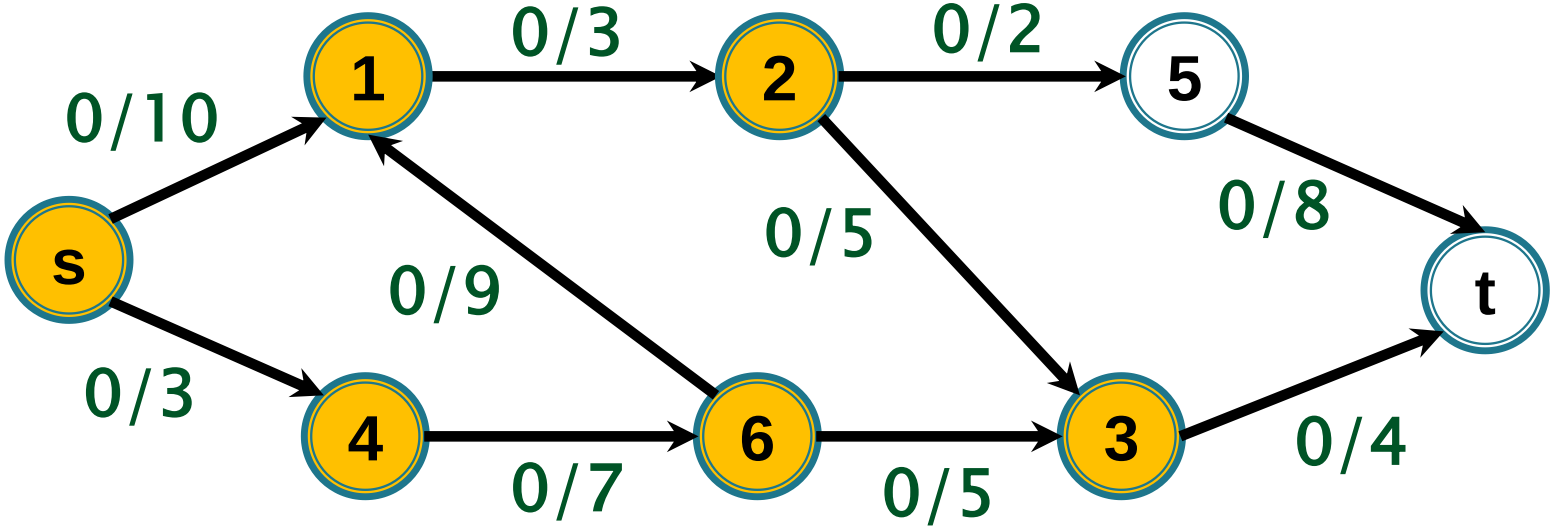

10
1
0/10
s
0/3
4
3

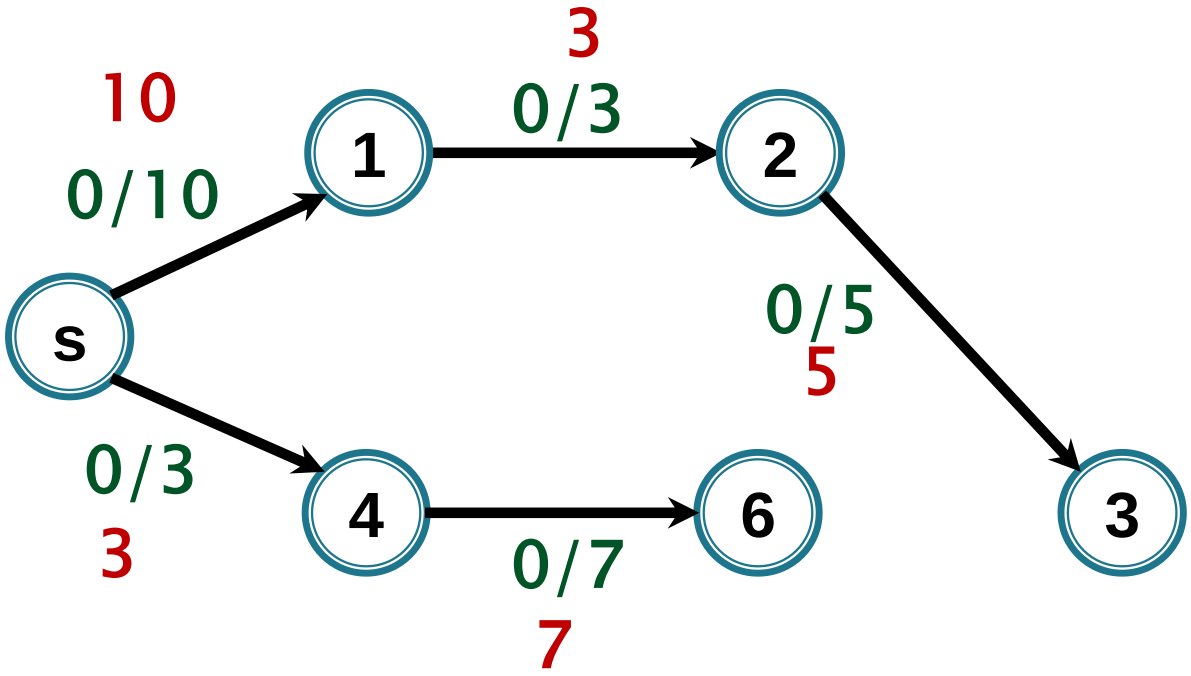

0/3
0/2
1
2
5
0/10
0/5
0/8
s
0/9
t
0/3
4
6
3
0/4
0/7
0/5

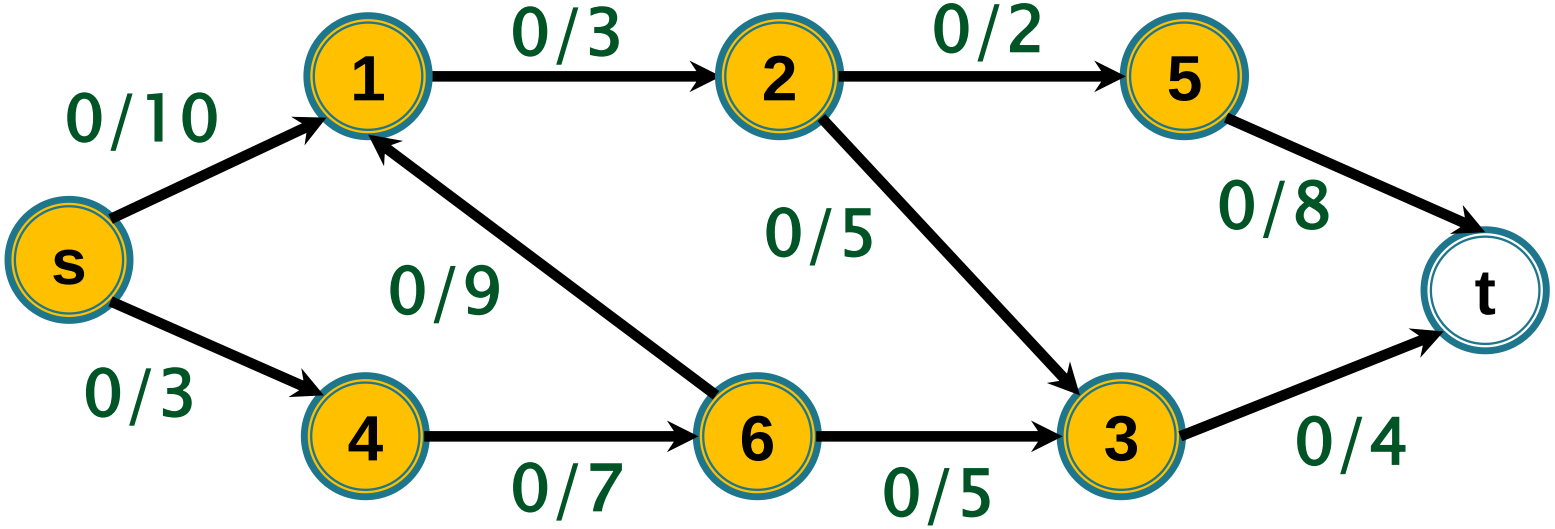

3
10
0/3
1
2
0/10
s
0/3
4
3

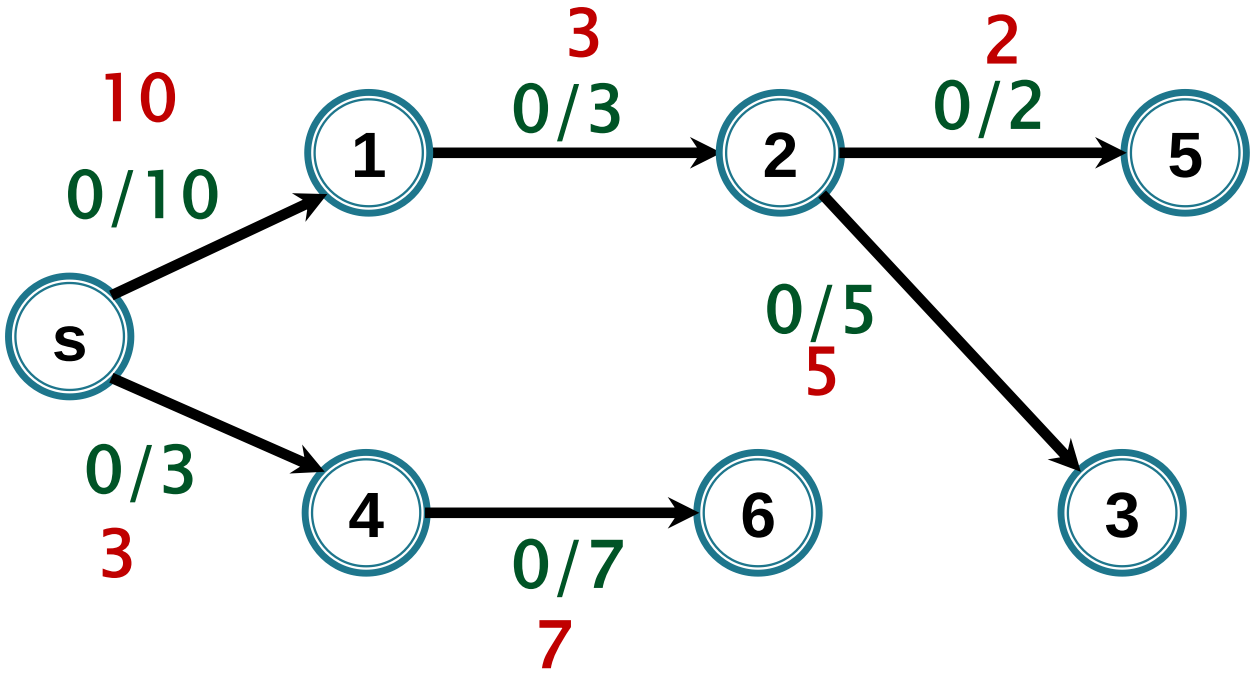

0/3
0/2
0/10
0/5
0/8
0/9
0/3
0/7
0/5
0/4
s
1
2
5
t
4
6
3

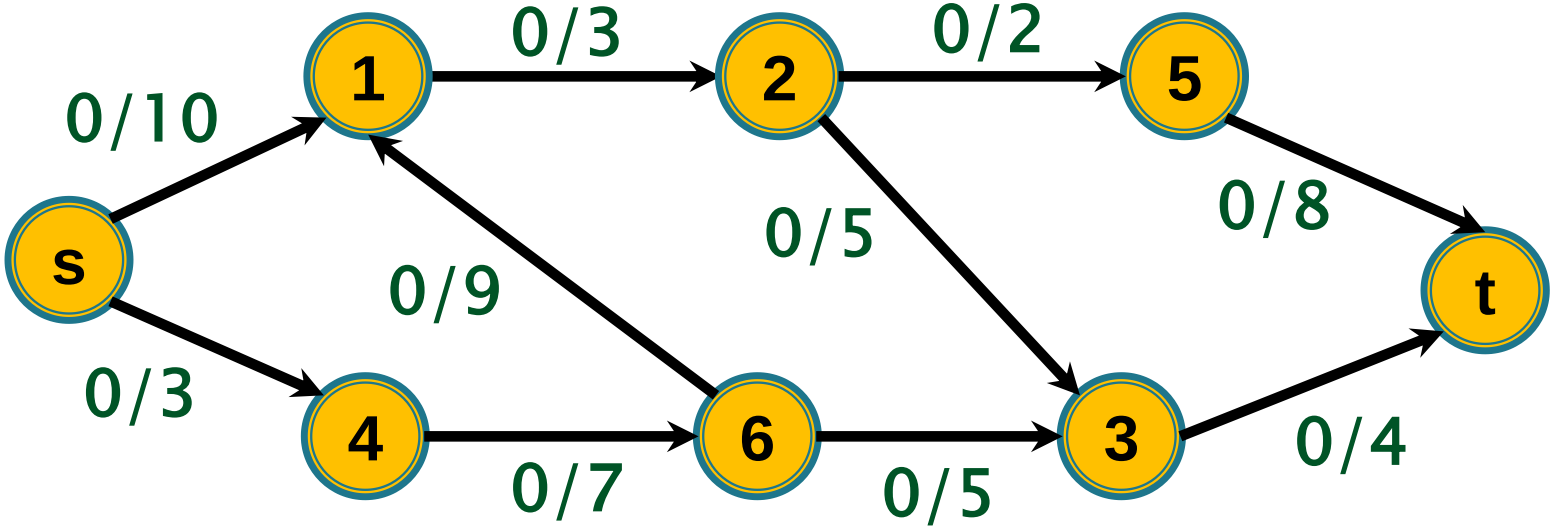

3
10
0/3
0/10
0/3
3
0/7
7
s
1
2
4
6

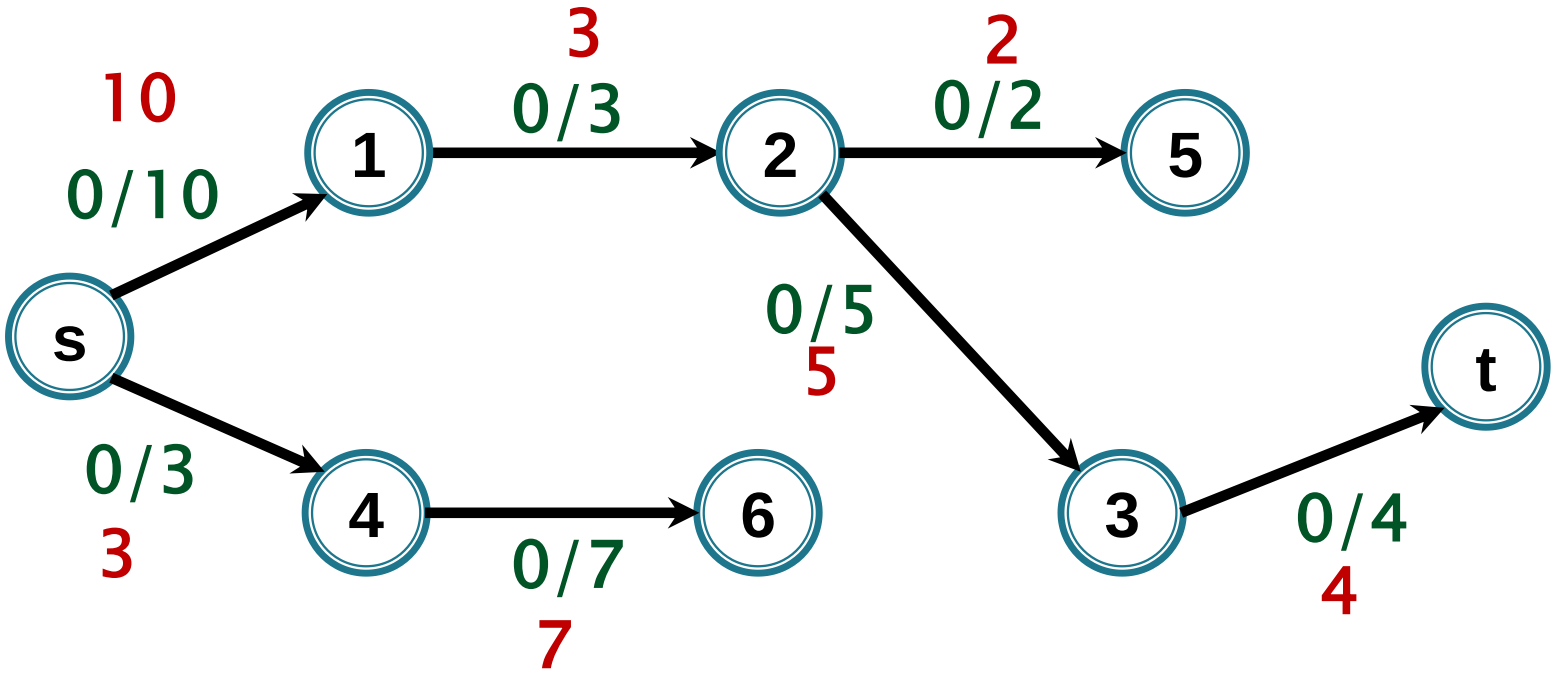

0/3
0/2
0/10
0/5
0/8
0/9
0/3
0/7
0/5
0/4
s
1
2
5
t
4
6
3

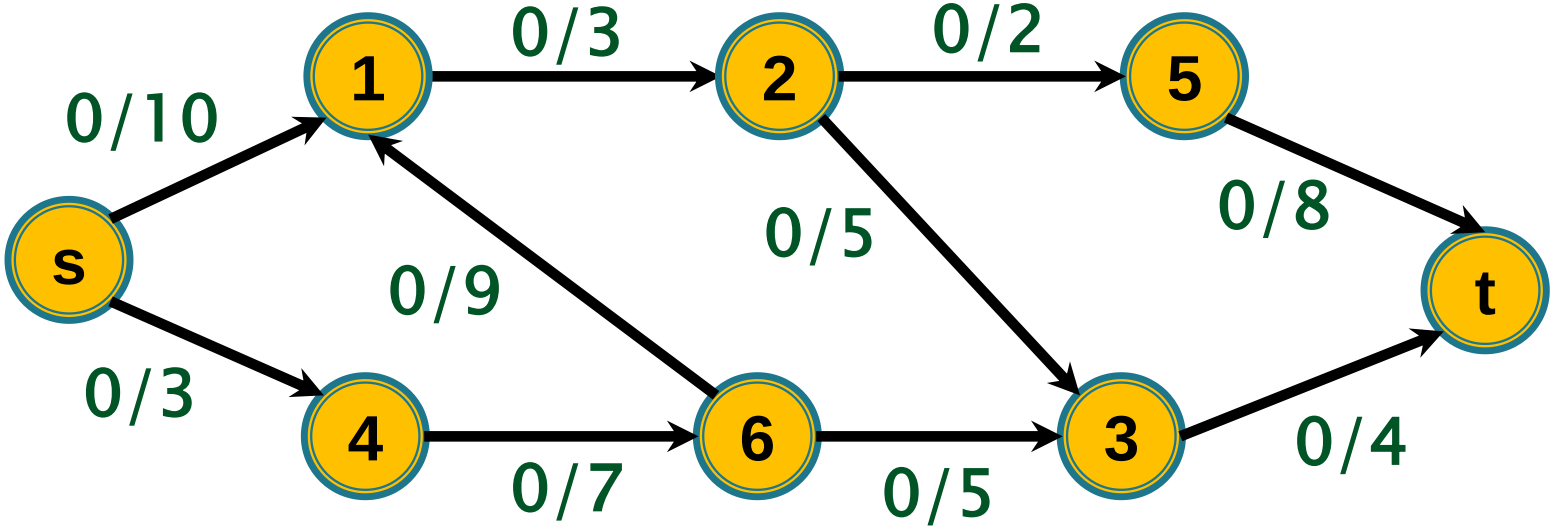

3
10
0/3
0/10
0/5
5
0/3
3
0/7
7
s
1
2
4
6
3

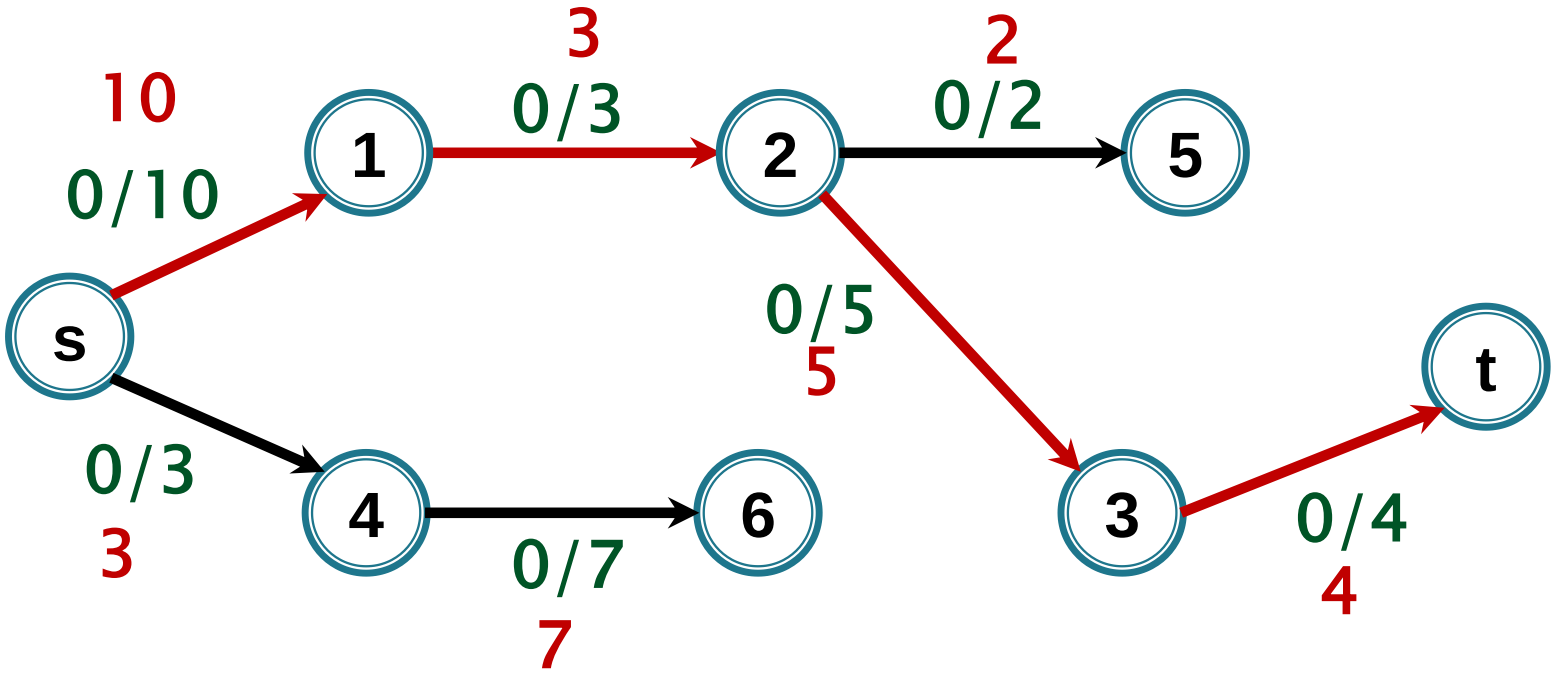

1
2
5
s
t
4
6
3
0/10
0/3
0/2
0/5
0/8
0/9
0/3
0/4
0/7
0/5


3
2
10
0/3
0/2
1
2
5
0/10
s
0/5
5
0/3
3
4
6
3
0/7
7

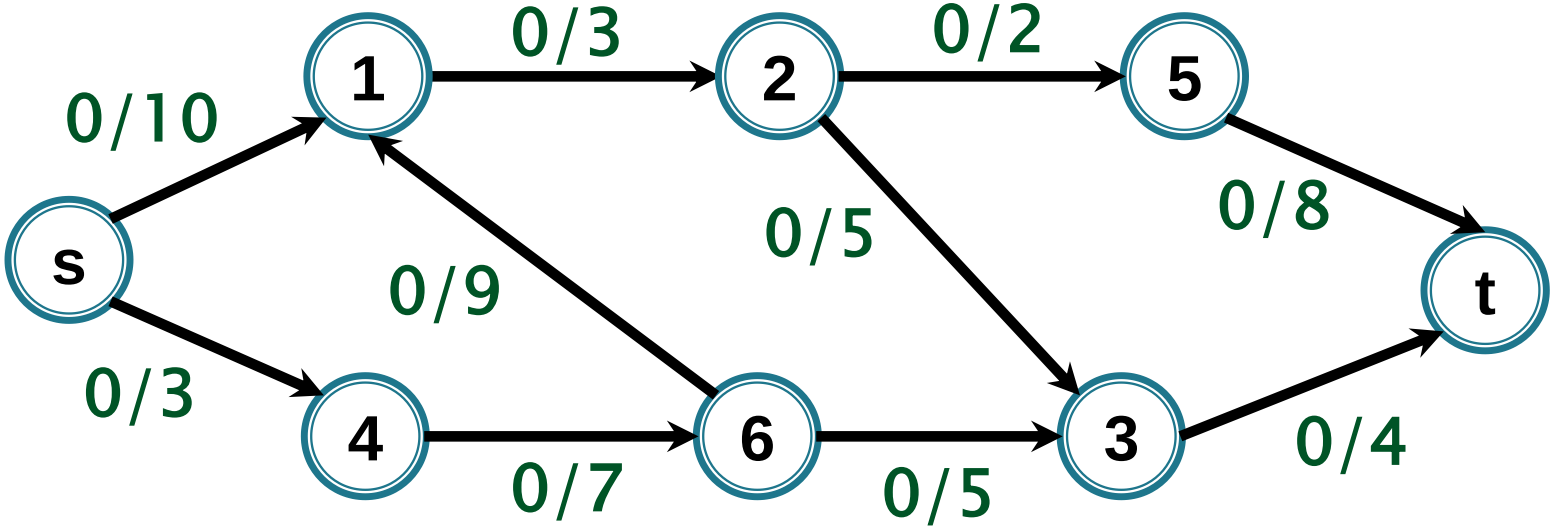

0/3
0/2
0/10
0/5
0/8
0/9
0/3
0/4
0/7
0/5
s
1
2
5
t
4
6
3

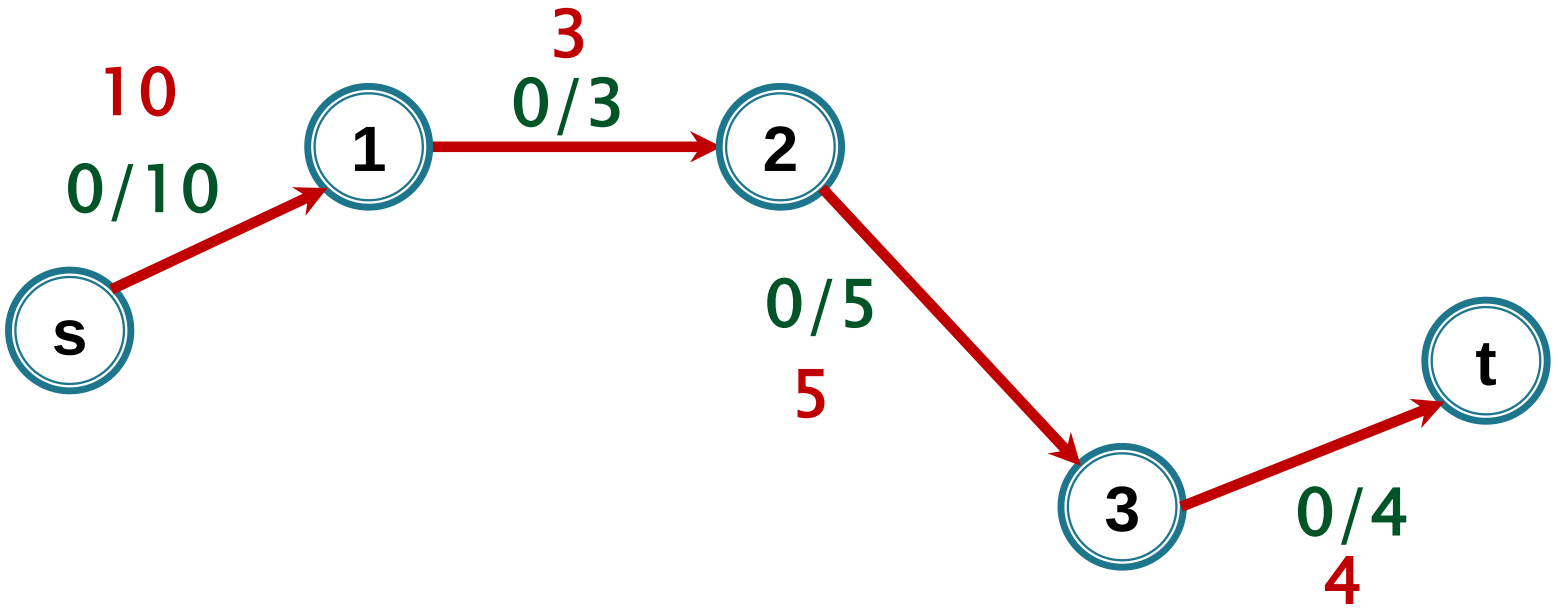

3
2
10
0/3
0/2
0/10
0/5
5
0/3
3
0/7
7
0/4
4
s
1
2
5
t
4
6
3


0/3
0/2
0/10
0/5
0/8
0/9
0/3
0/4
0/7
0/5
s
1
2
5
t
4
6
3

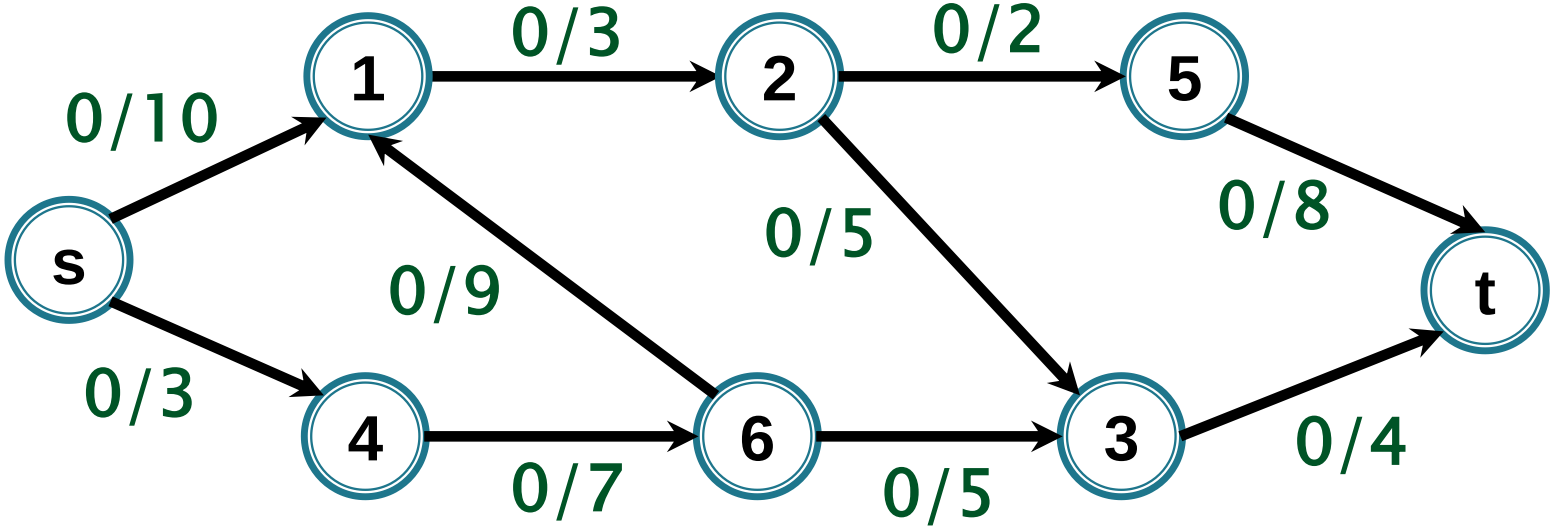

3
2
10
0/3
0/2
0/10
0/5
5
0/3
3
0/7
7
0/4
4
s
1
2
5
t
4
6
3

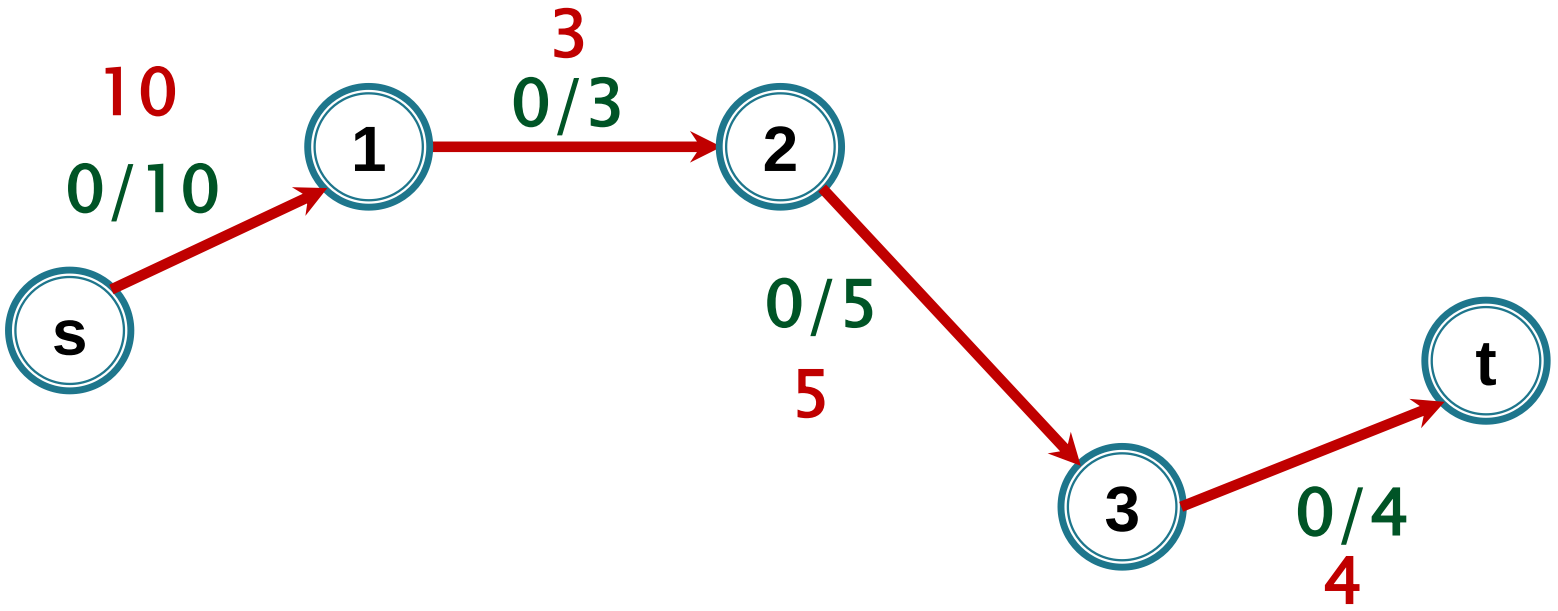
revizuieste_flux_lant



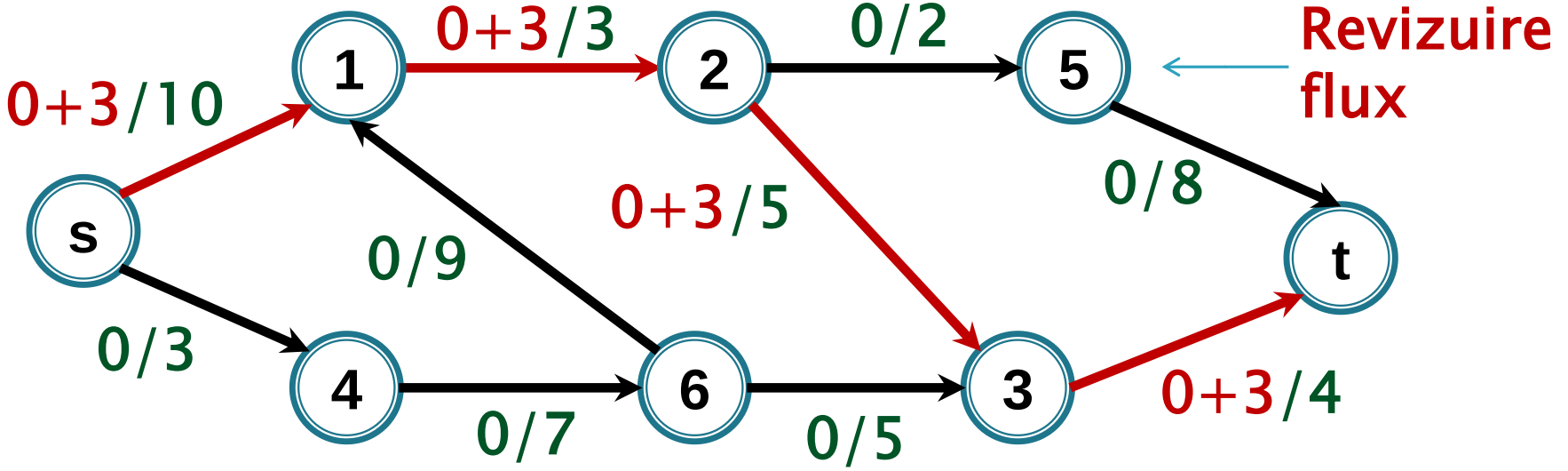


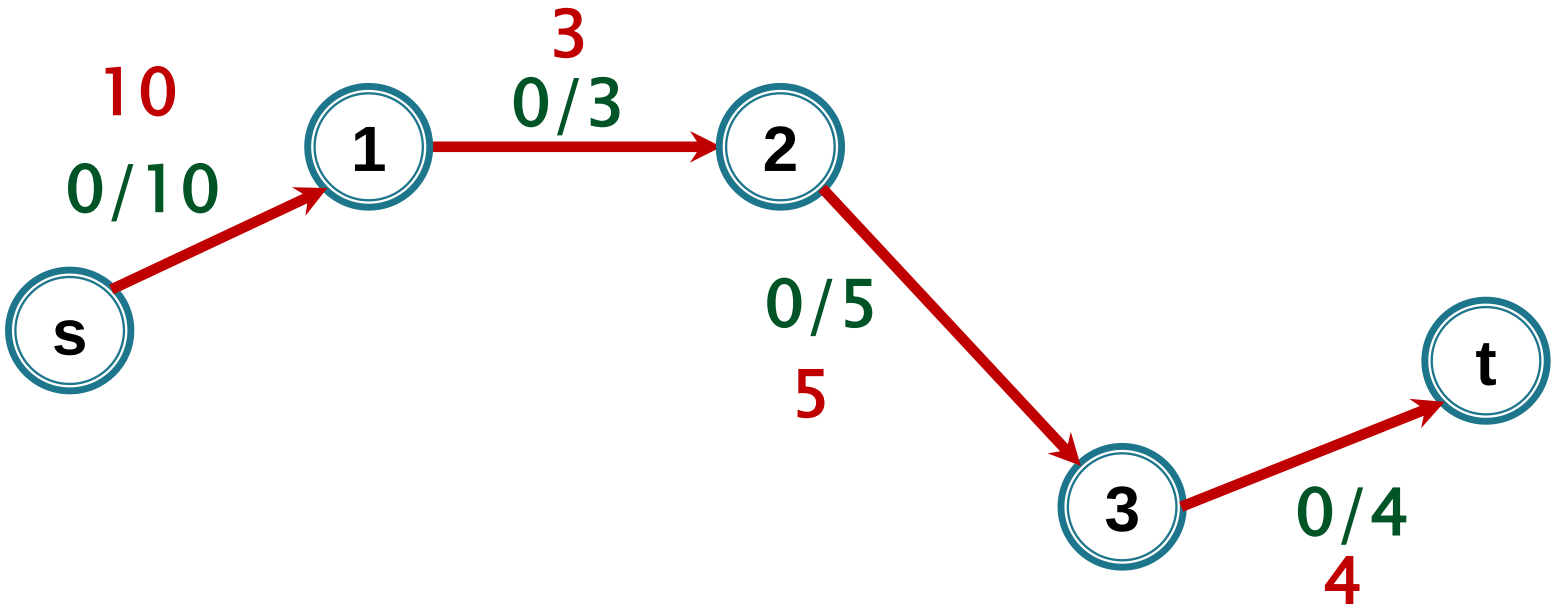
i(P) = ?



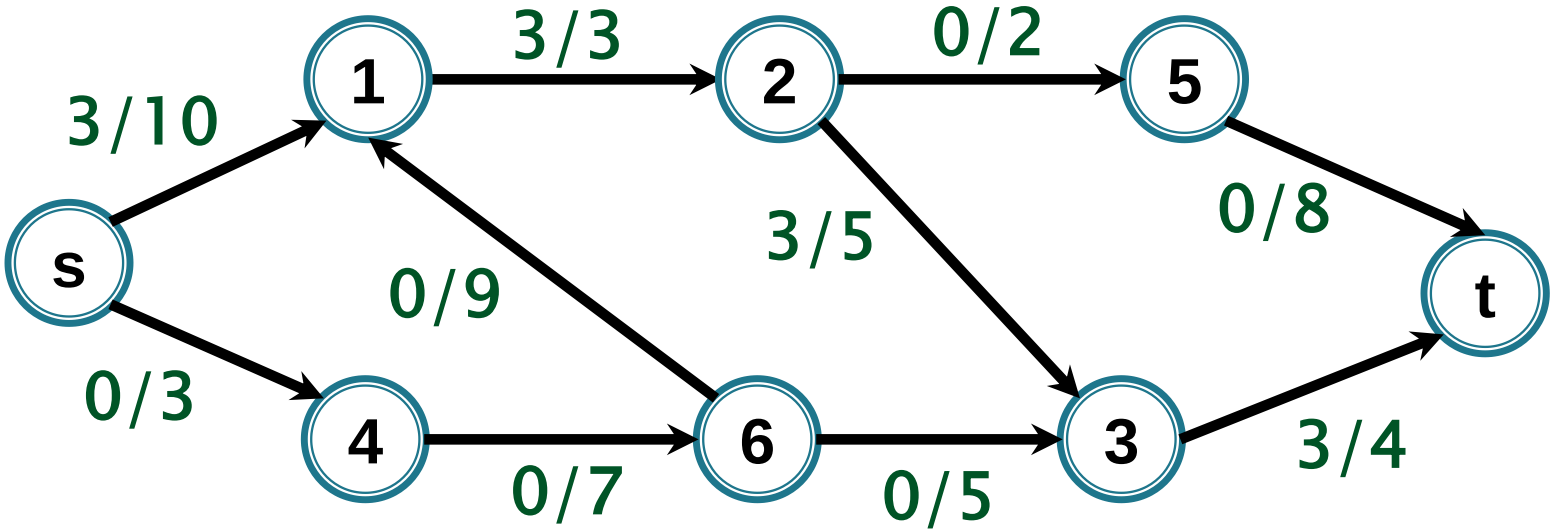


$i(P) = \min\{10, 3, 5, 4\} = 3$

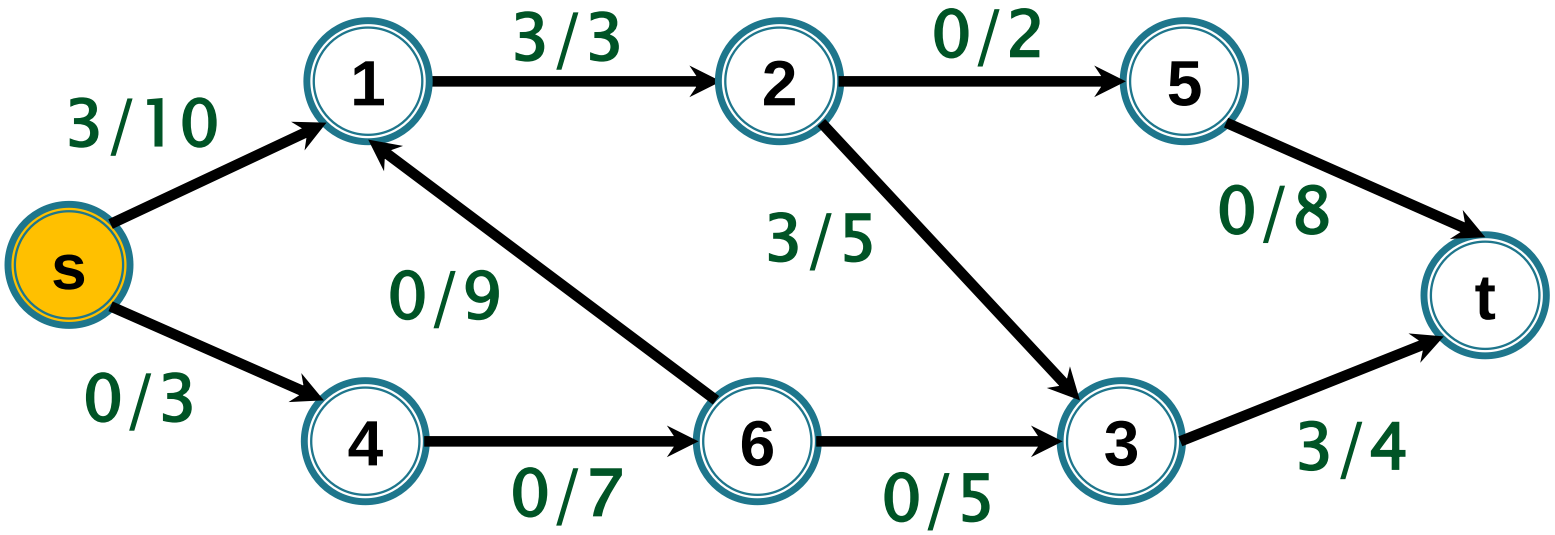




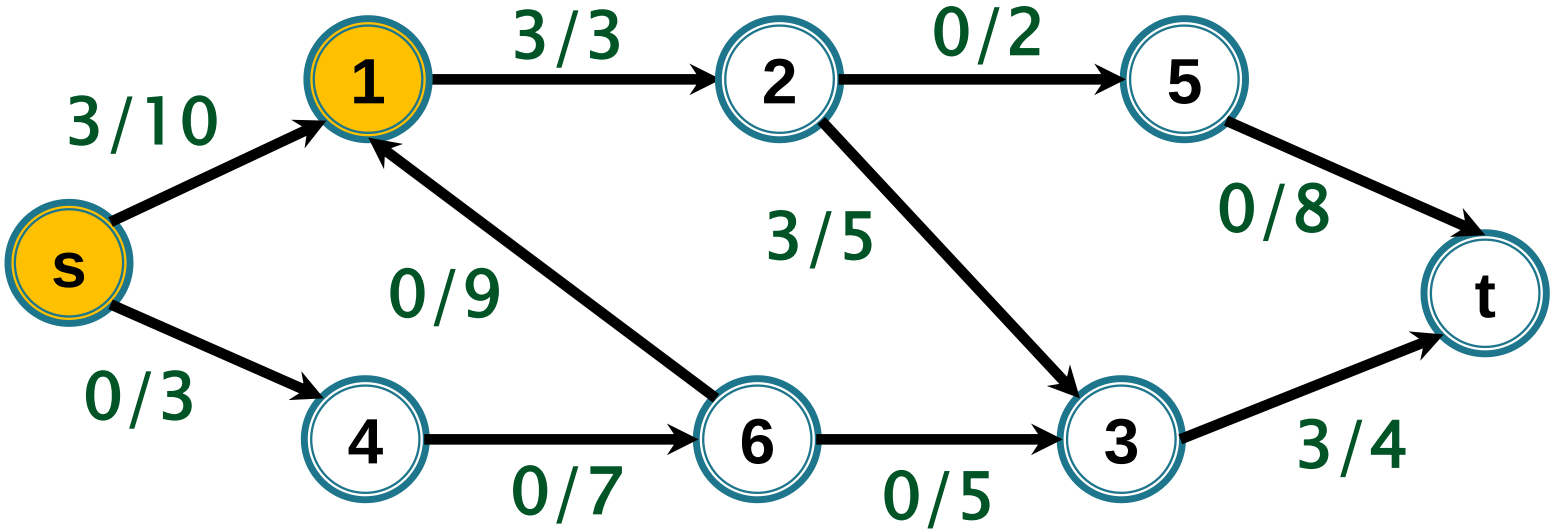
$i(P) = \min\{10, 3, 5, 4\} = 3$

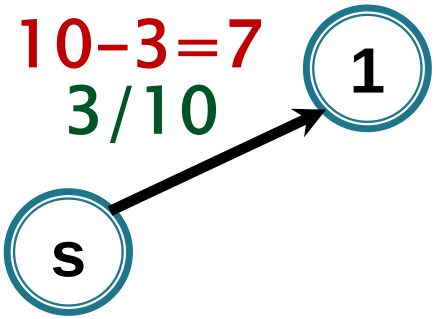

s
1
2
5
t
4
6
3
3/10
3/3
0/2
0/8
3/5
0/9
0/3
0/7
0/5
3/4

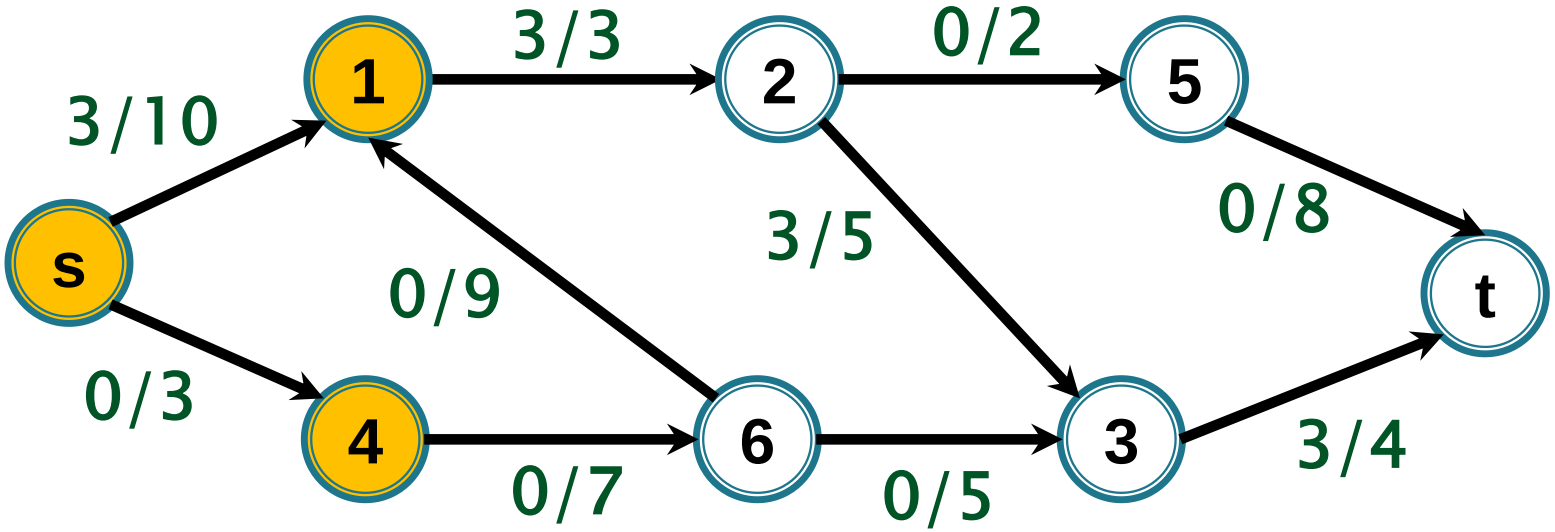
construieste_s-t_lant_nesat_BF

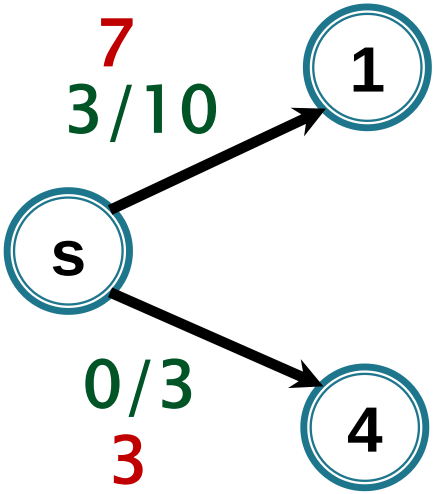

s
1
2
5
t
4
6
3
3/10
3/3
0/2
0/8
3/5
0/9
0/3
0/7
0/5
3/4

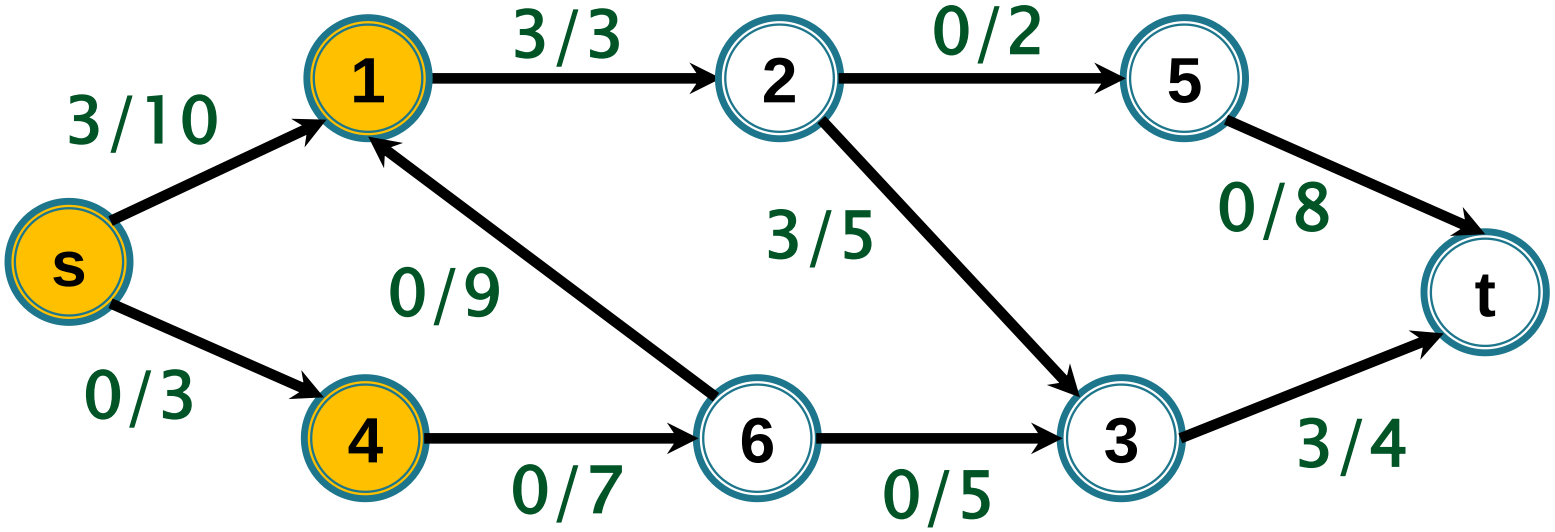

s

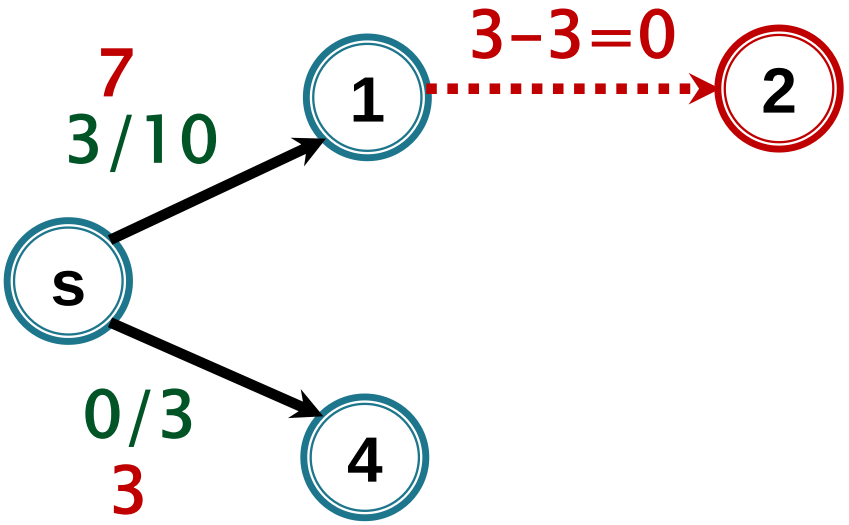

1
2
5
s
t
4
6
3
3/3
0/2
3/10
3/5
0/8
0/9
0/3
0/7
0/5
3/4

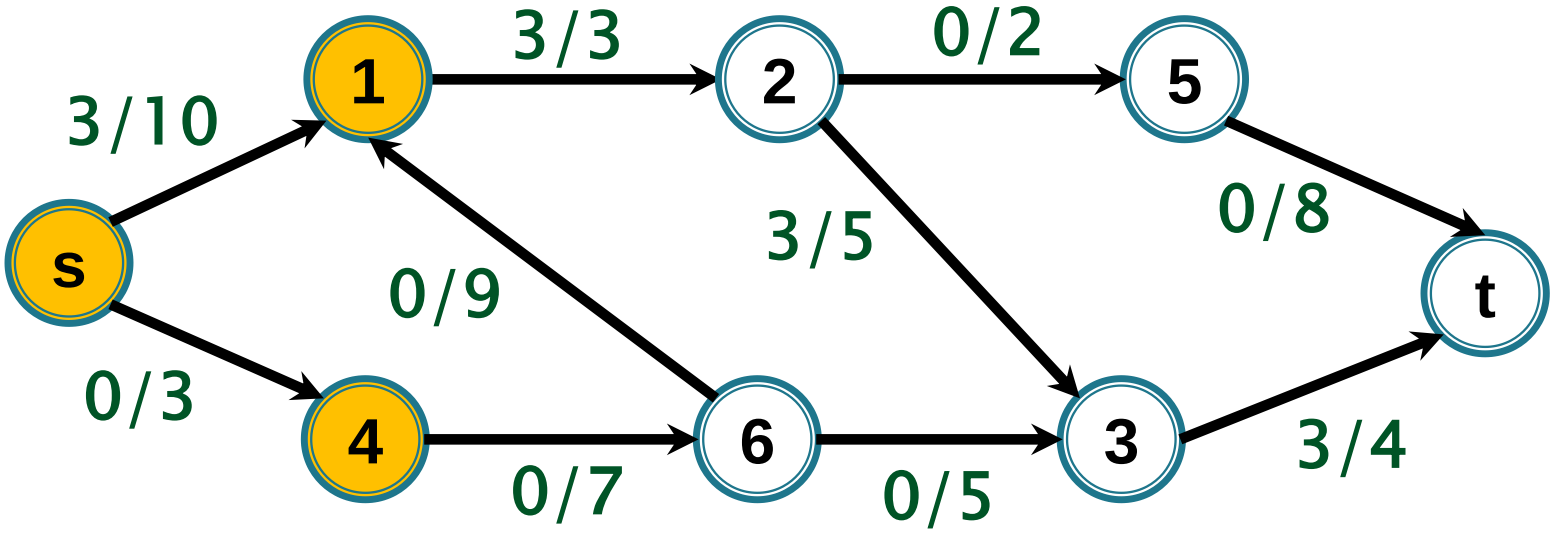

10-3=7
3/10
1
s

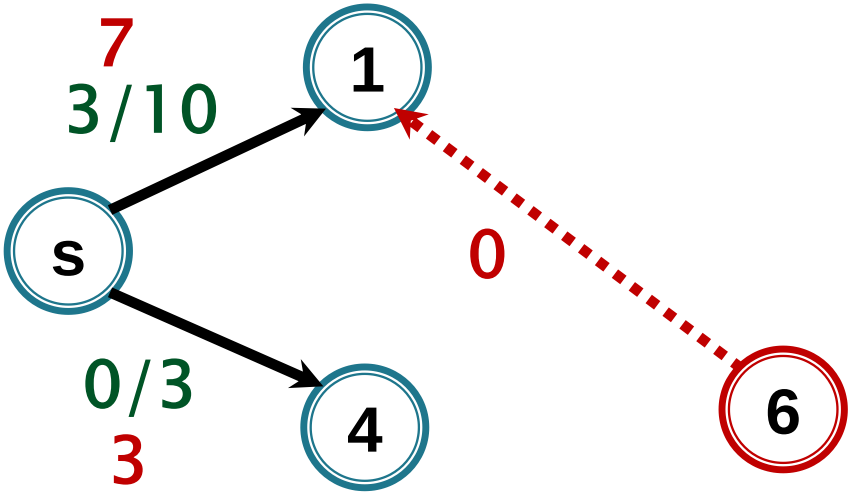

1
2
5
s
t
4
6
3
3/3
0/2
3/10
3/5
0/8
0/9
0/3
0/7
0/5
3/4

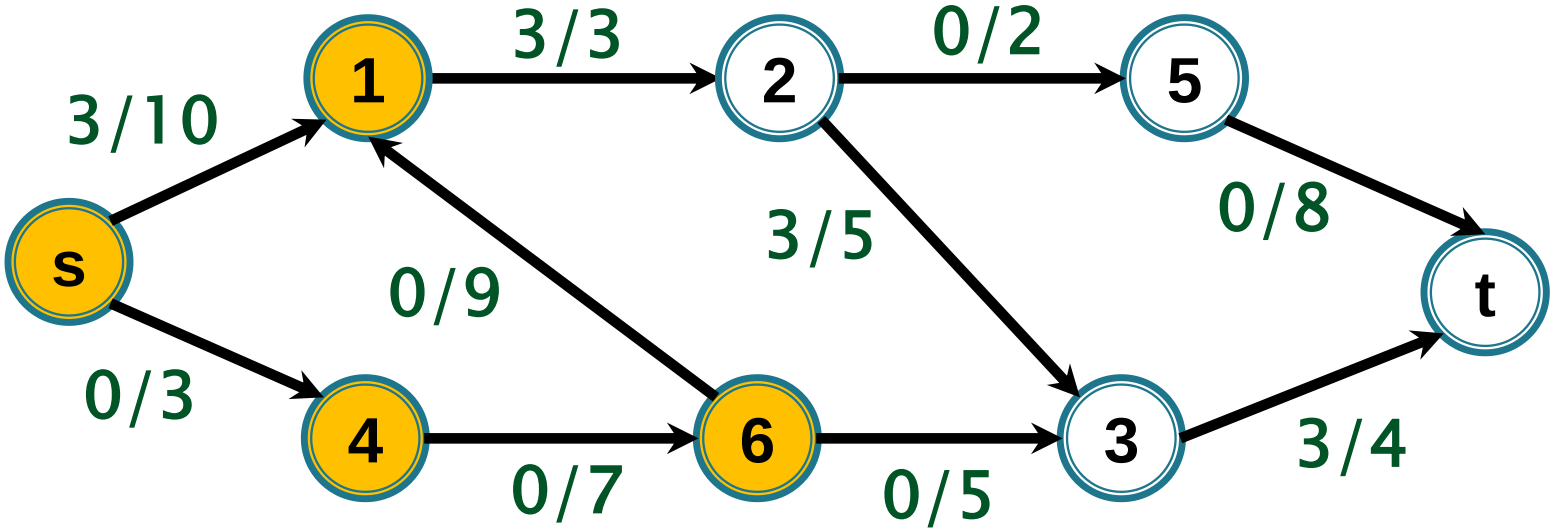

7
1
3/10
s
0/3
4
3

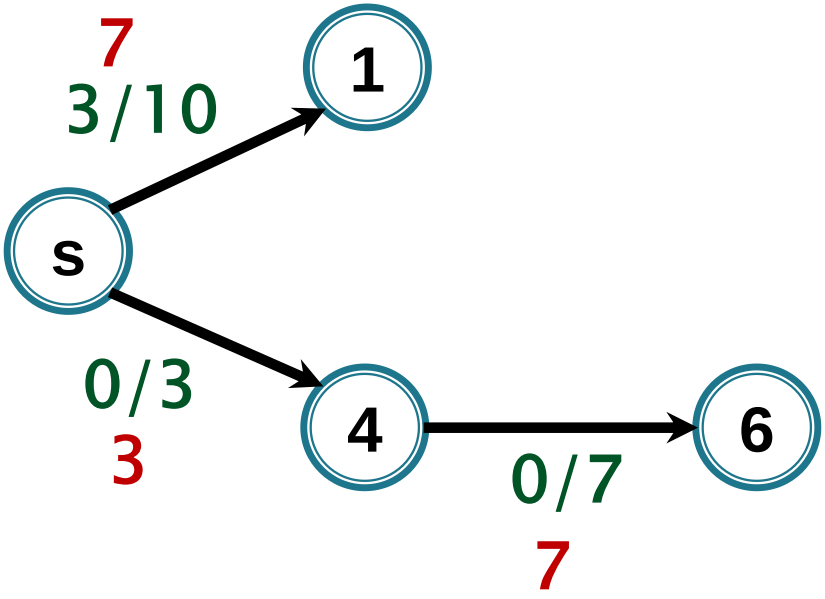

1
2
5
s
t
4
6
3
3/3
0/2
3/10
3/5
0/8
0/9
0/3
0/7
0/5
3/4

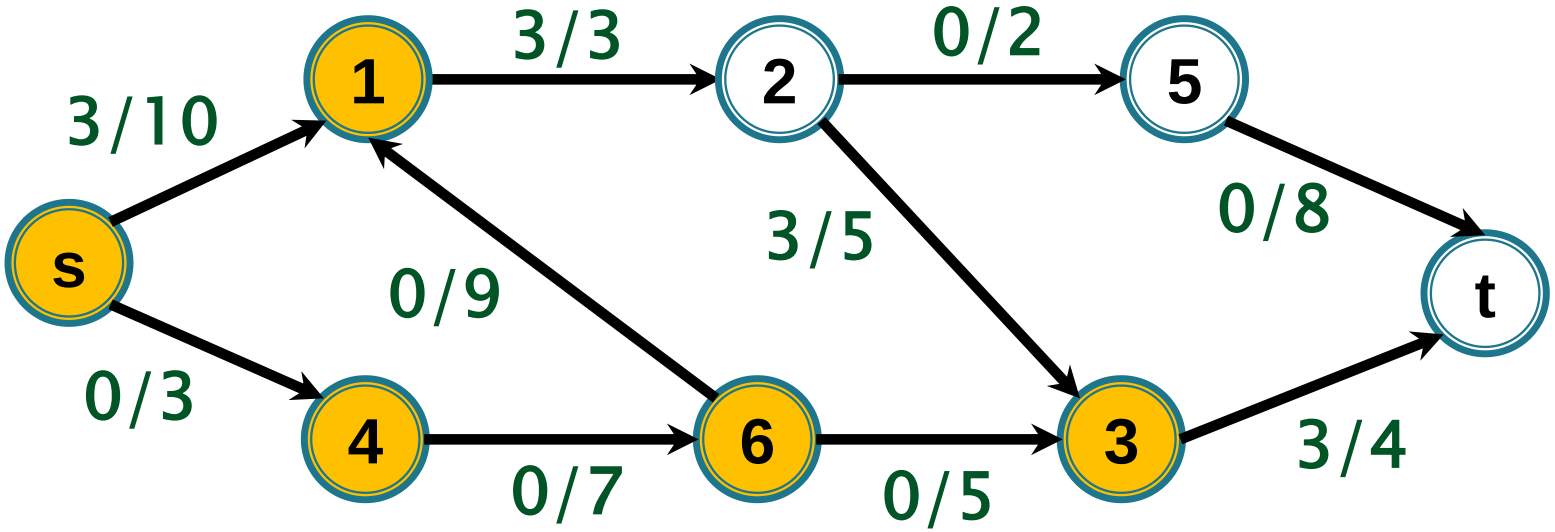

3-3=0
7
1
2
3/10
s
0/3
4
3

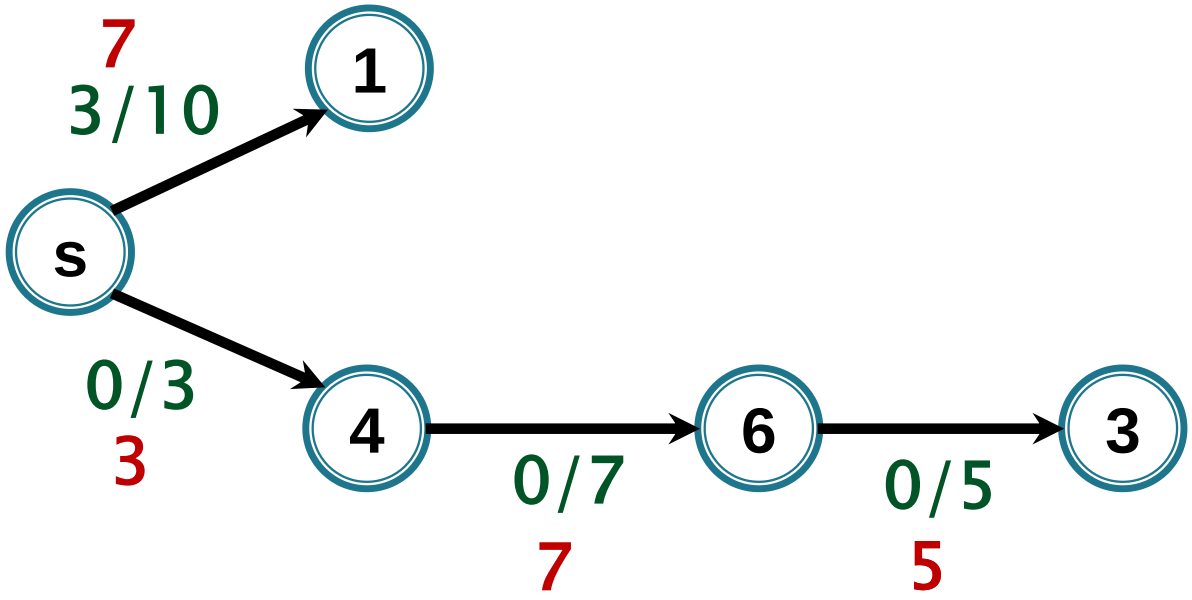

1
2
5
s
t
4
6
3
3/3
0/2
3/10
3/5
0/8
0/9
0/3
0/7
0/5
3/4

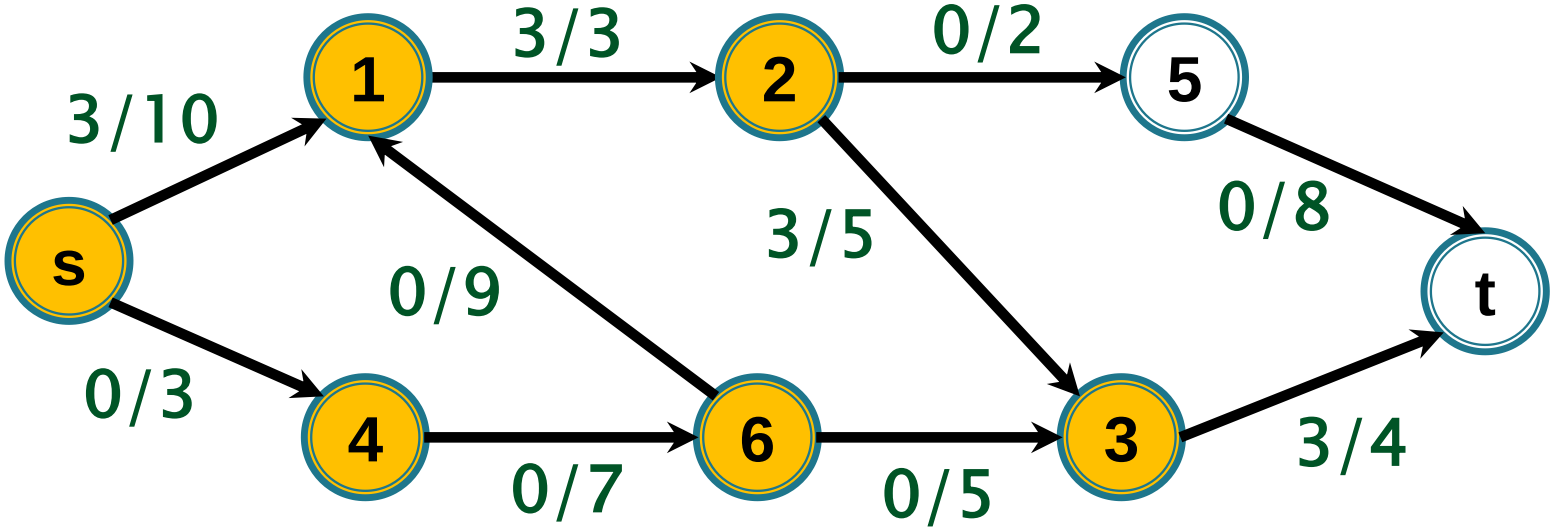

7
3/10
1
s
0
0/3
3
4
6

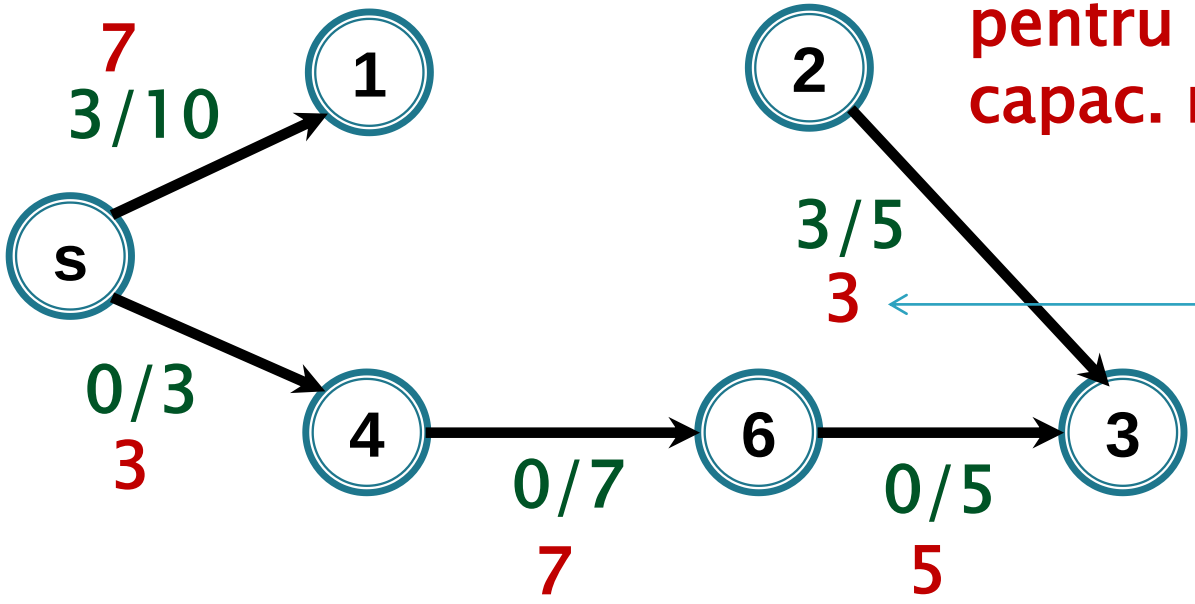

1
2
5
s
t
4
6
3
3/3
0/2
3/10
3/5
0/8
0/9
0/3
0/7
0/5
3/4


7
1
3/10
s
0/3
3
4
6
0/7
7

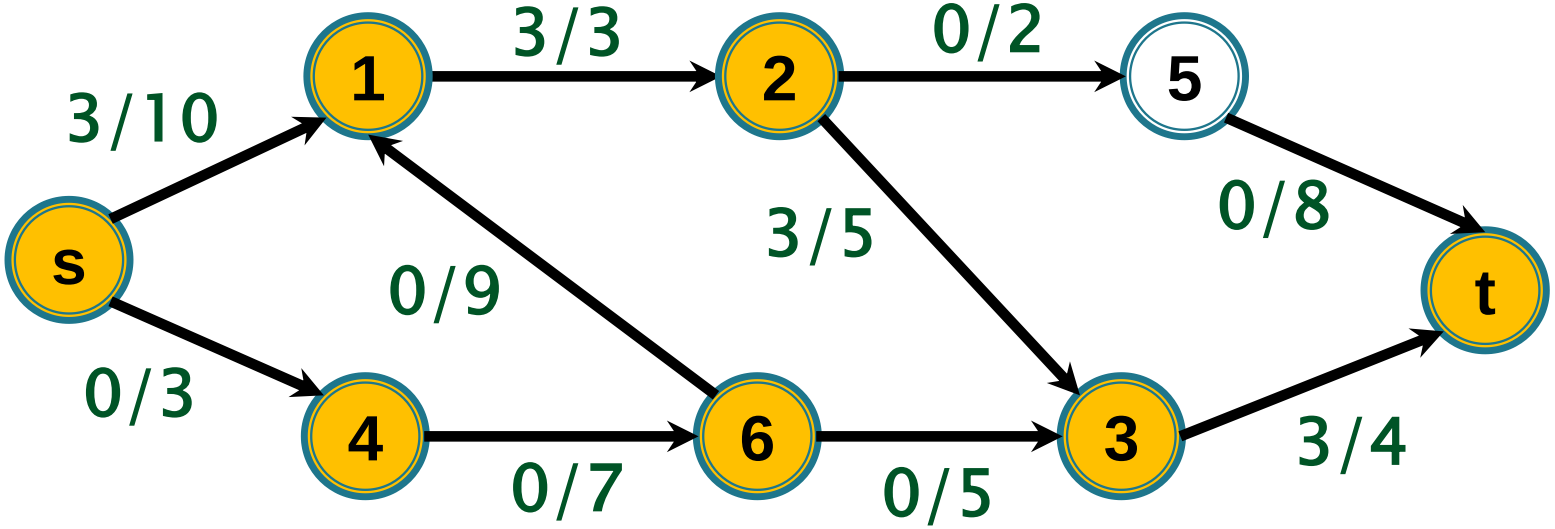

1
2
5
s
t
4
6
3
3/3
0/2
3/10
3/5
0/8
0/9
0/3
0/7
0/5
3/4

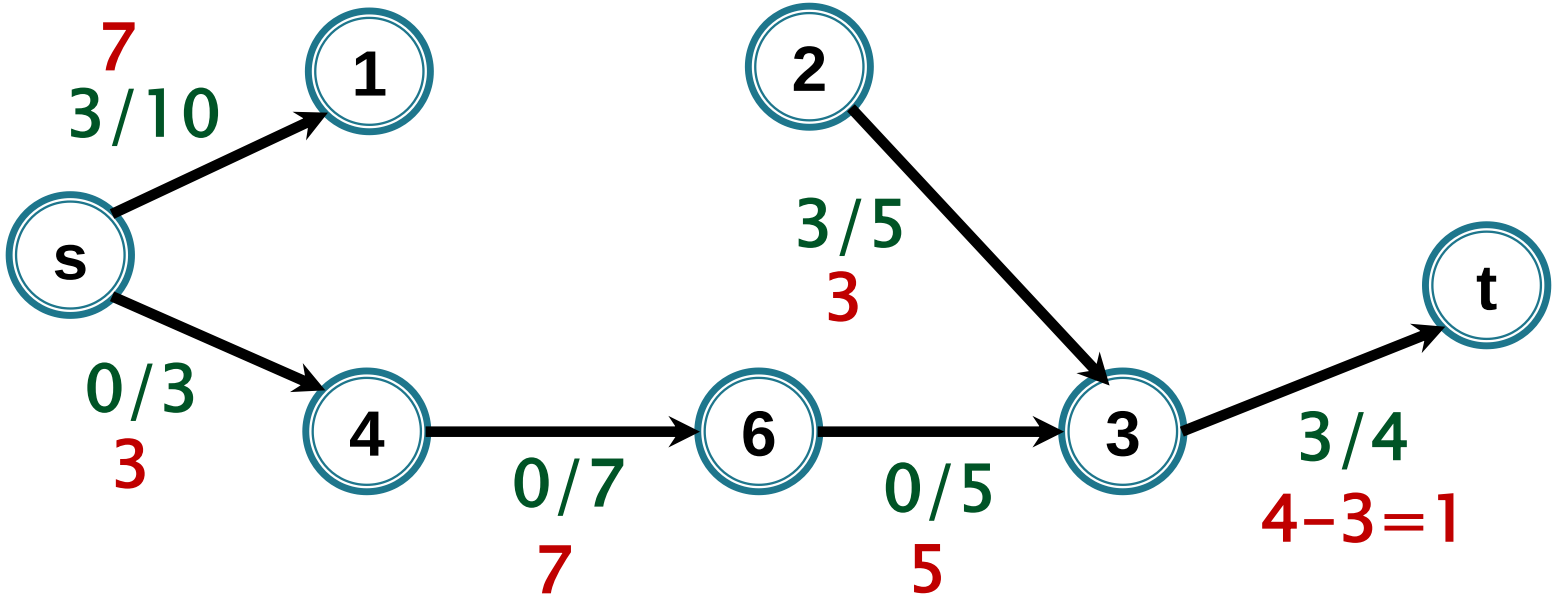

7
1
3/10
s
0/3
3
4
6
3
0/7
0/5
7
5



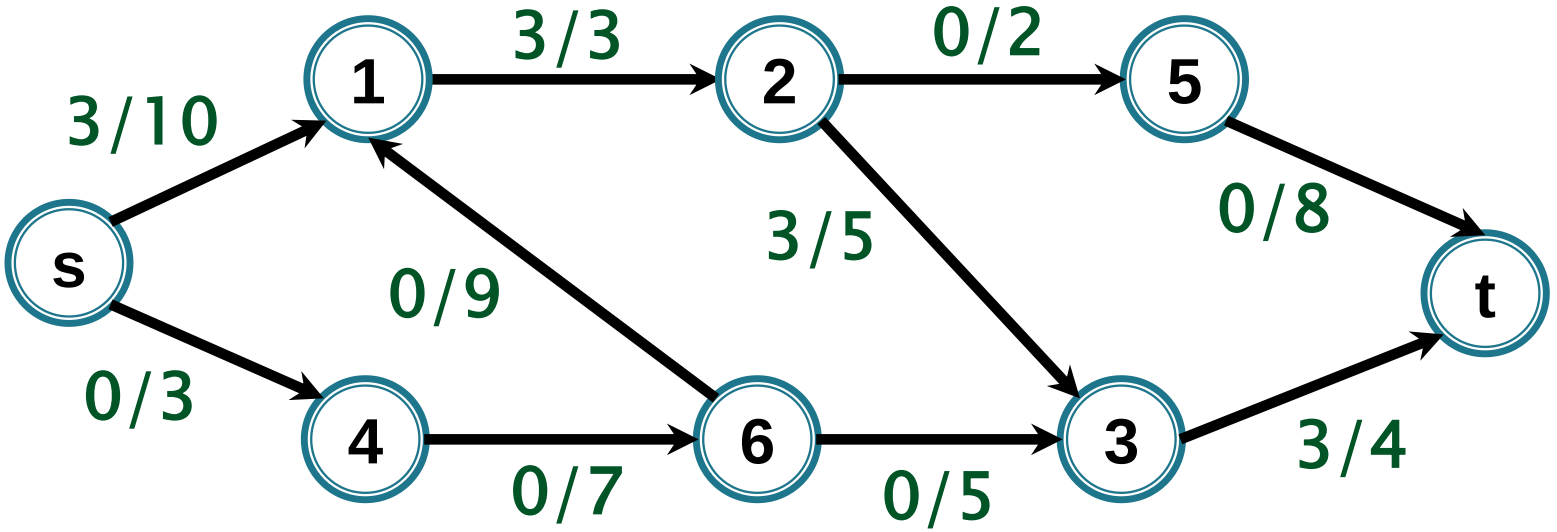


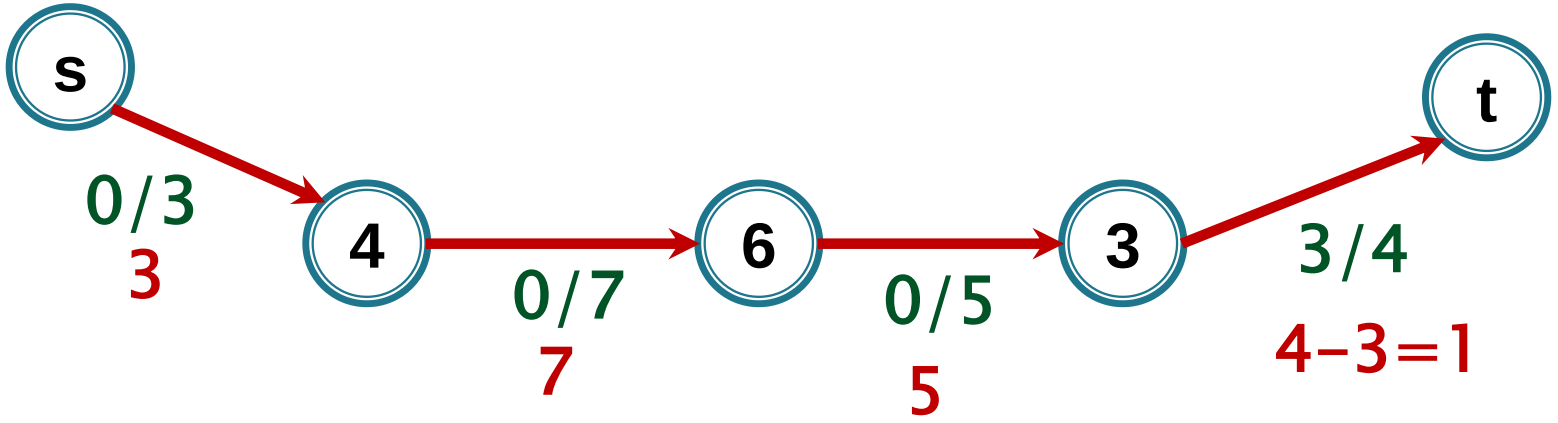
pentru arc invers
capac. reziduală=fluxul

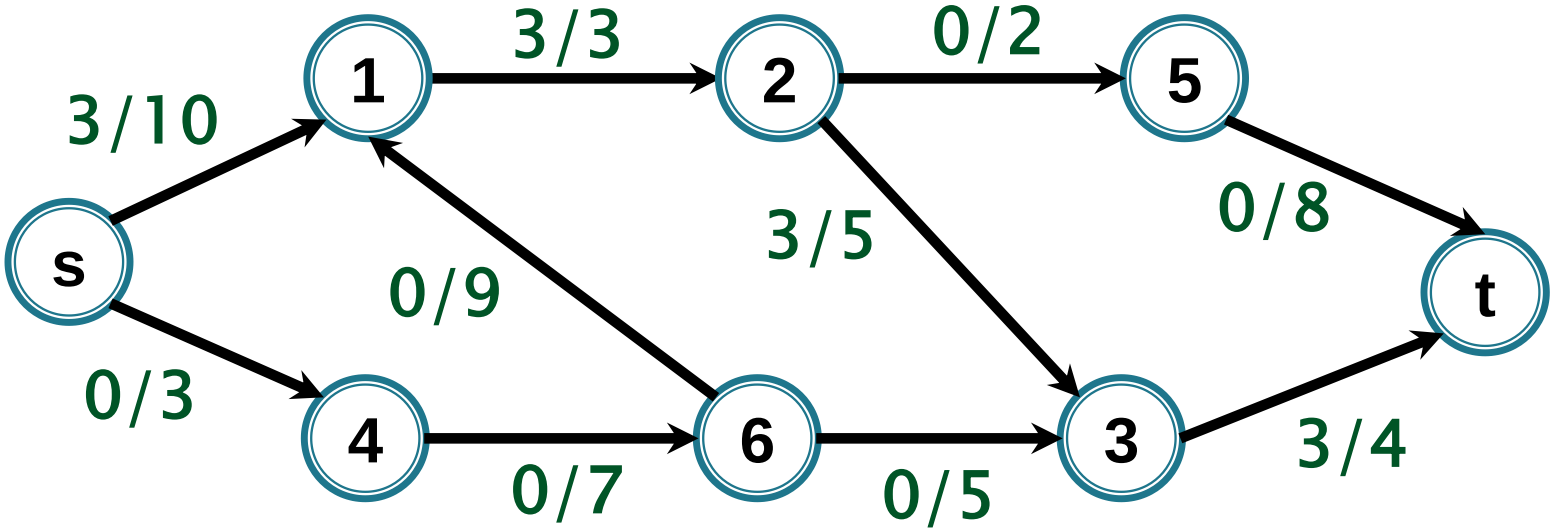

1
2
5
s
t
4
6
3
3/3
0/2
3/10
3/5
0/8
0/9
0/3
3/4
0/7
0/5

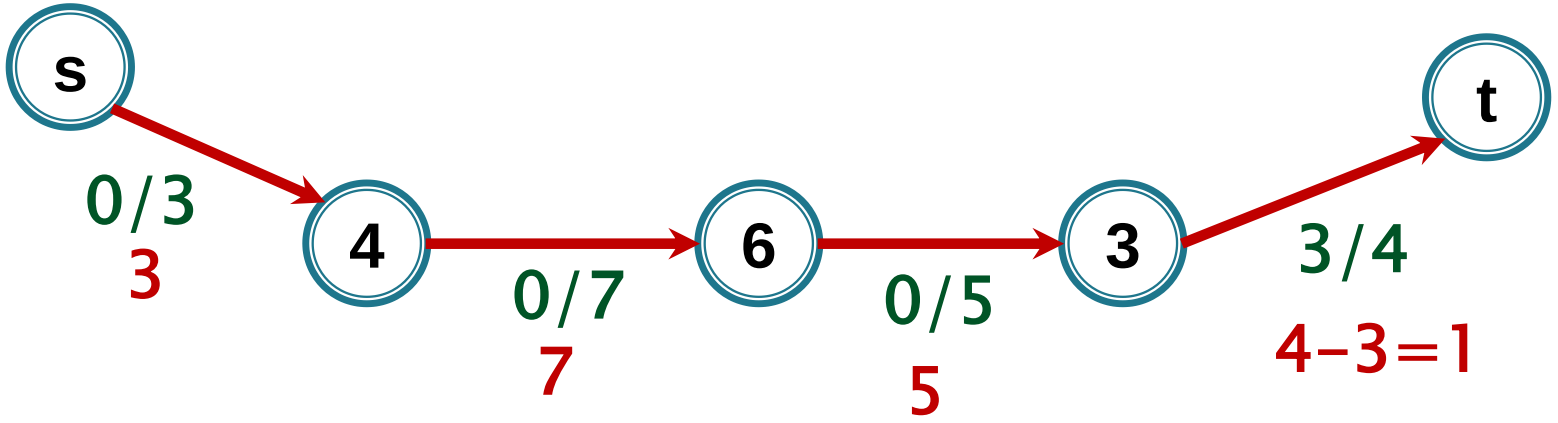

7
1
2
3/10
3/5
3
s
t
0/3
3
4
6
3
3/4
4−3=1
0/7
7
0/5
5

revizuieste_flux_lant

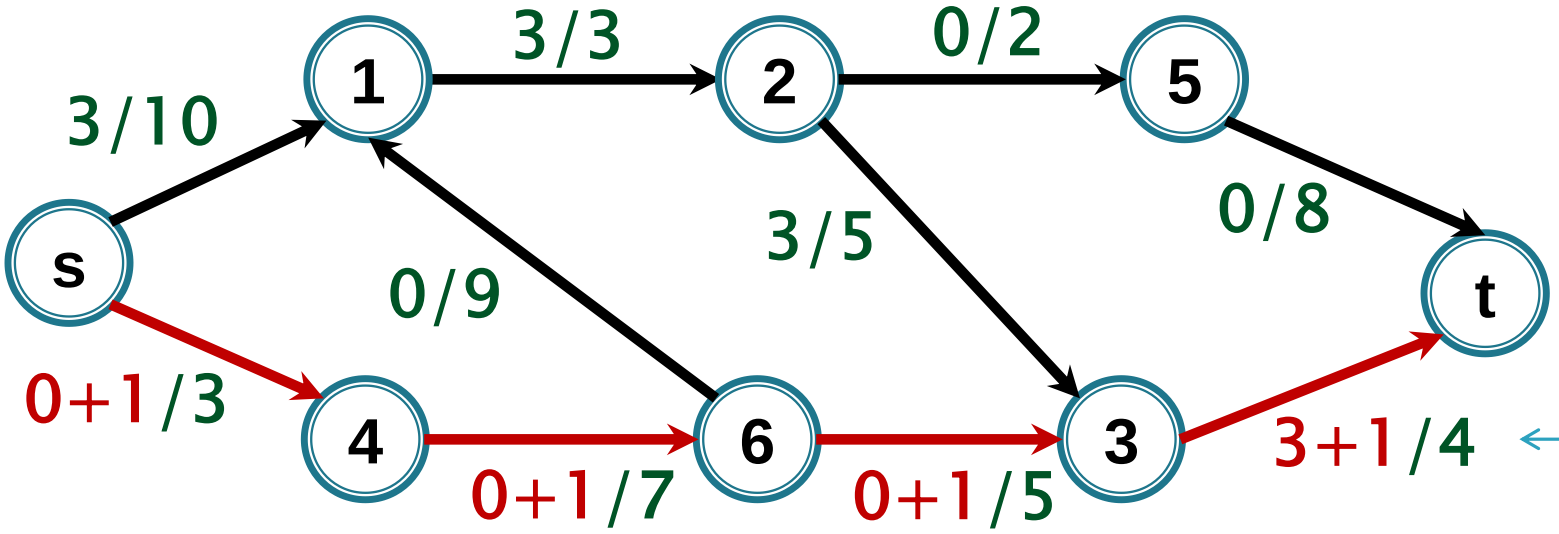

1
2
5
s
t
4
6
3
3/3
0/2
3/10
3/5
0/8
0/9
0/3
3/4
0/7
0/5


s
t
4
6
3
0/3
3
0/7
7
0/5
5
3/4
4-3=1

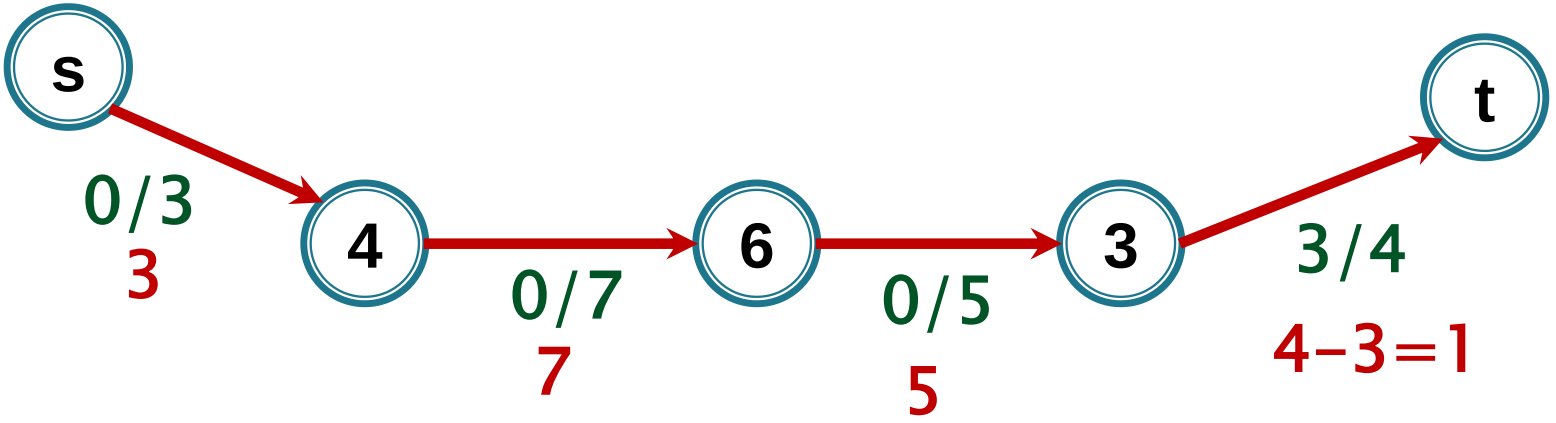




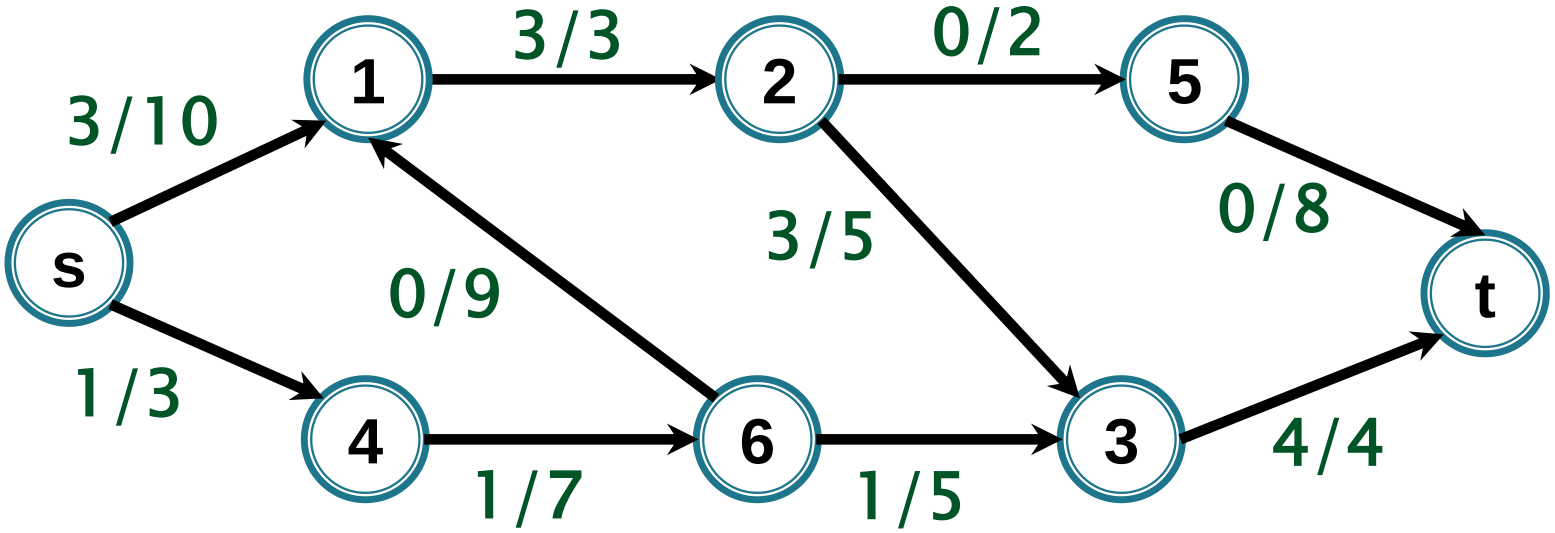
i(P) = min{ 3, 7, 5, 1} =1



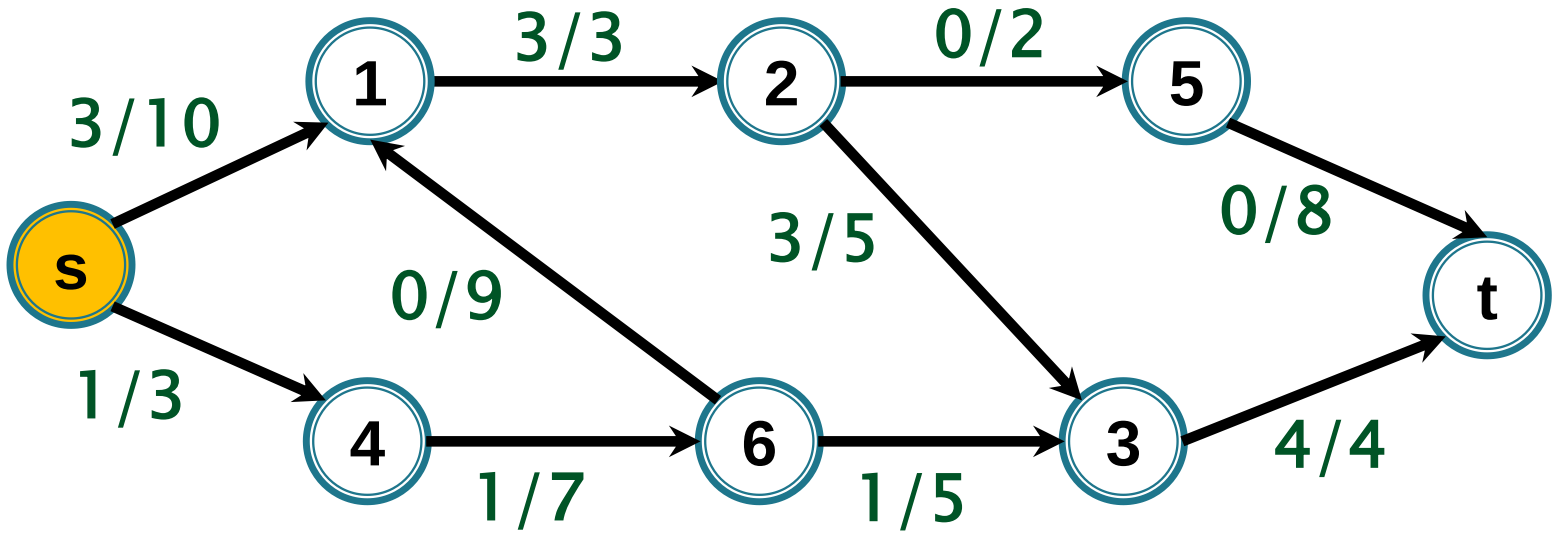
Revizuire flux



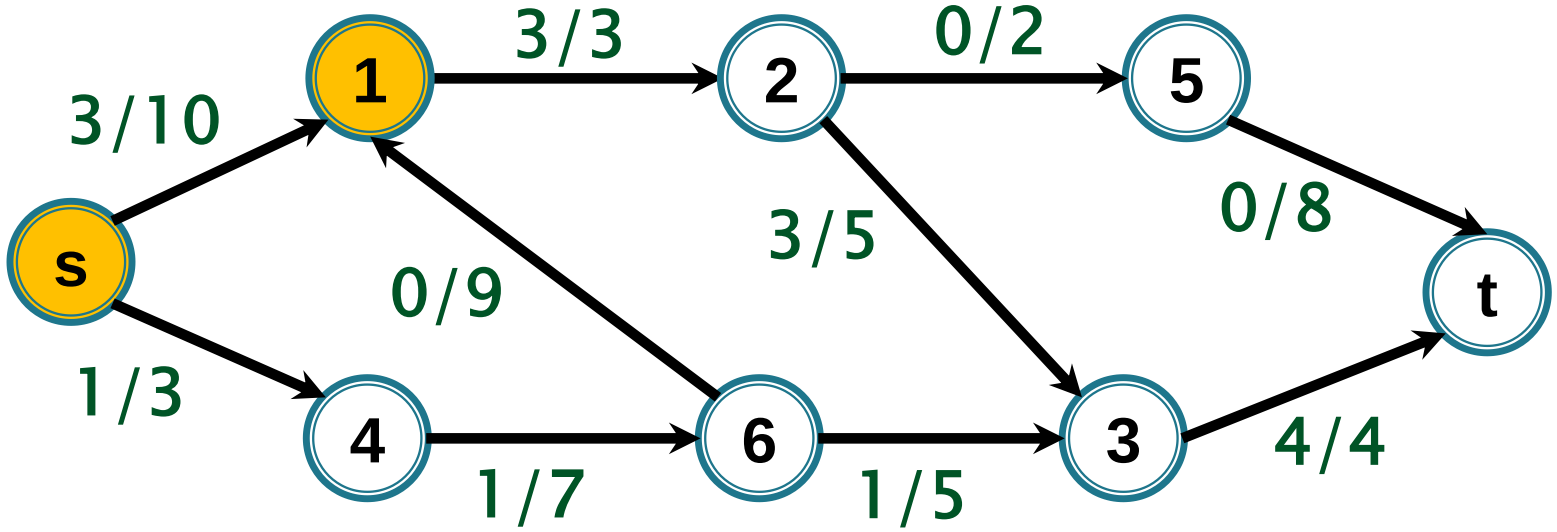
$i(P) = \min\{ 3, 7, 5, 1\} = 1$

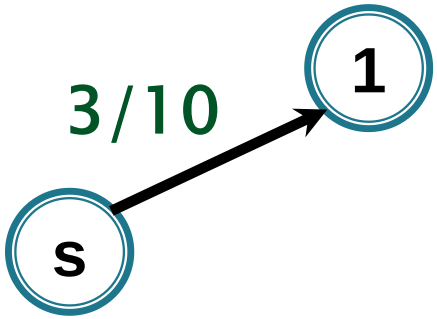

s
1
2
5
t
4
6
3
3/10
3/3
0/2
0/8
3/5
0/9
1/3
1/7
1/5
4/4

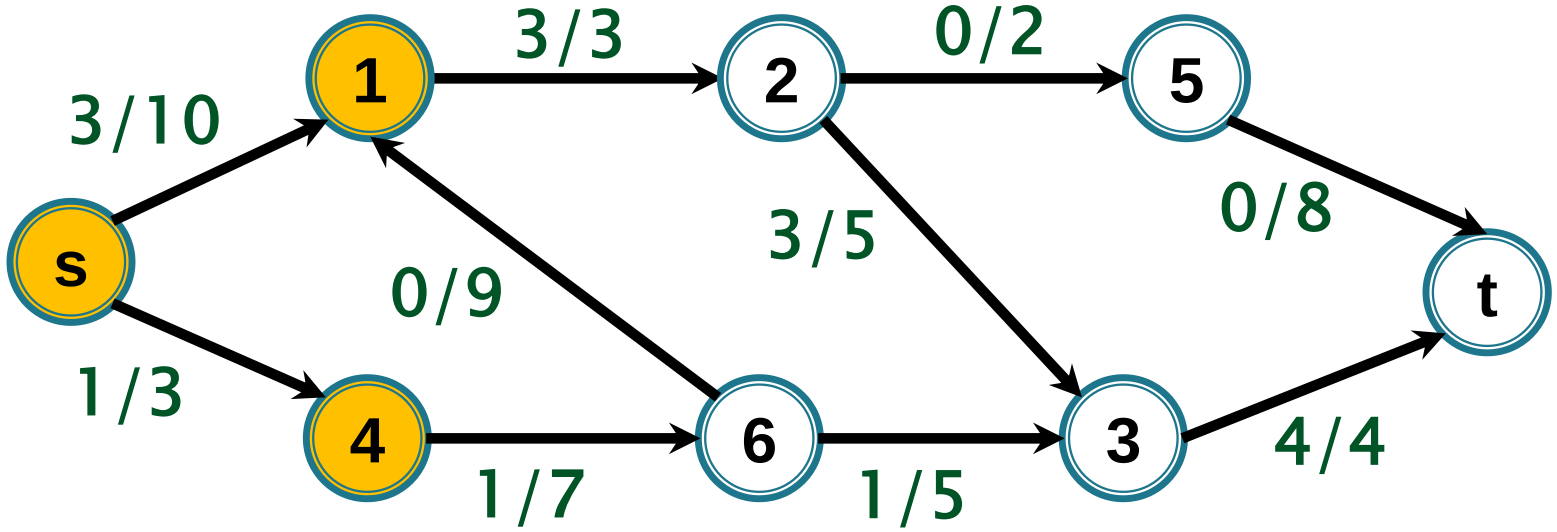
construieste_s-t_lant_nesat_BF

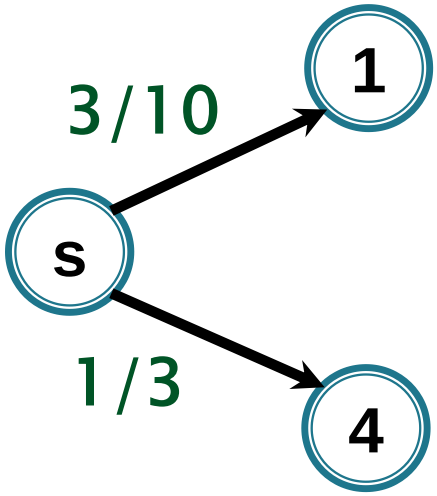

s
1
2
5
t
4
6
3
3/10
3/3
0/2
0/8
3/5
0/9
1/3
1/7
1/5
4/4

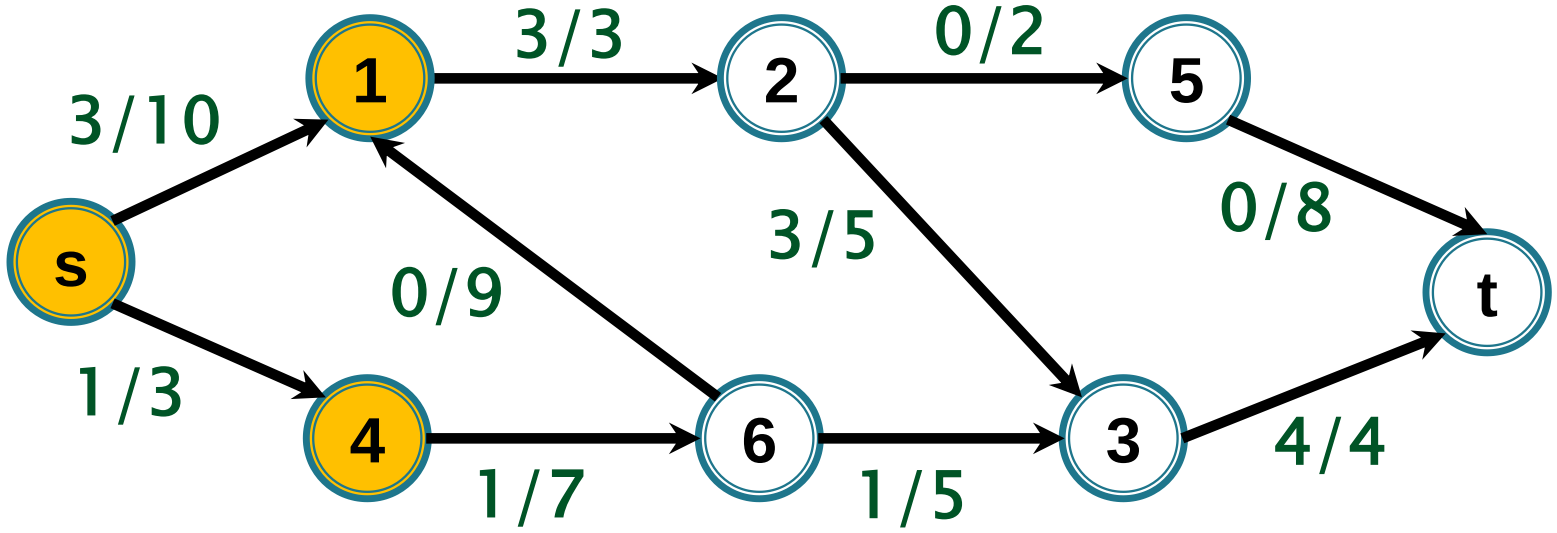

s

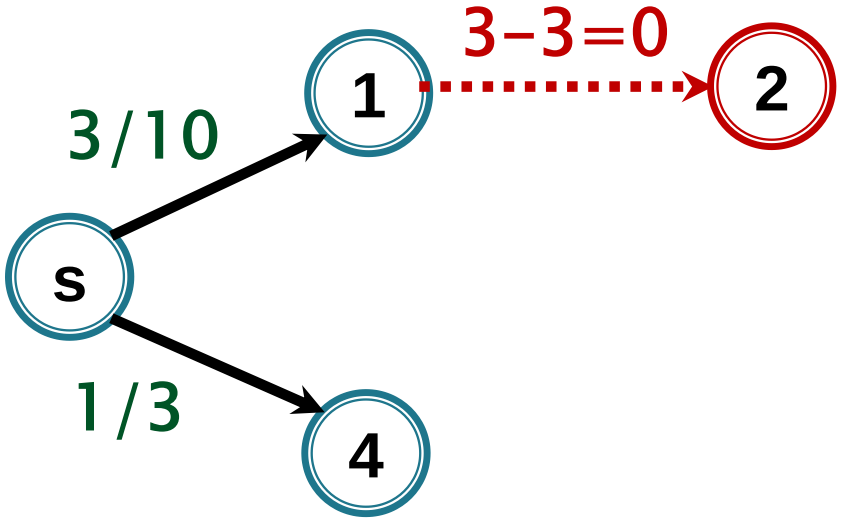

s
1
2
3
4
5
6
t
3/10
1/3
3/3
0/2
0/9
3/5
0/8
1/7
1/5
4/4

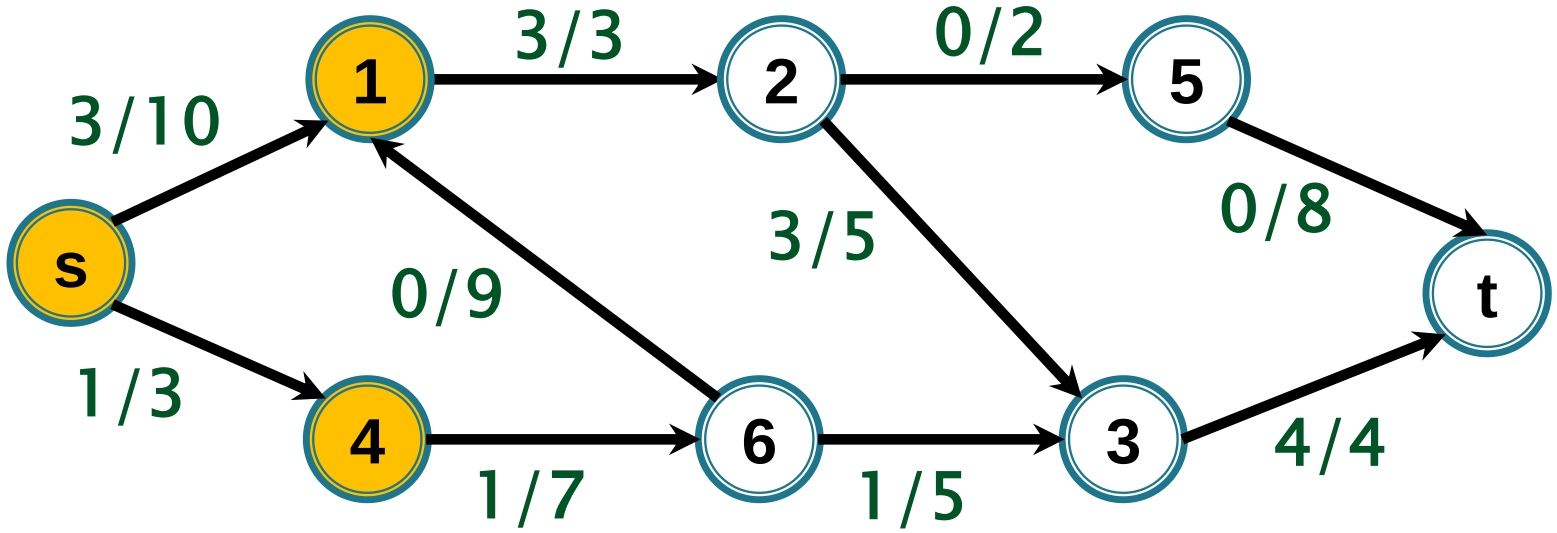

s
1
3/10

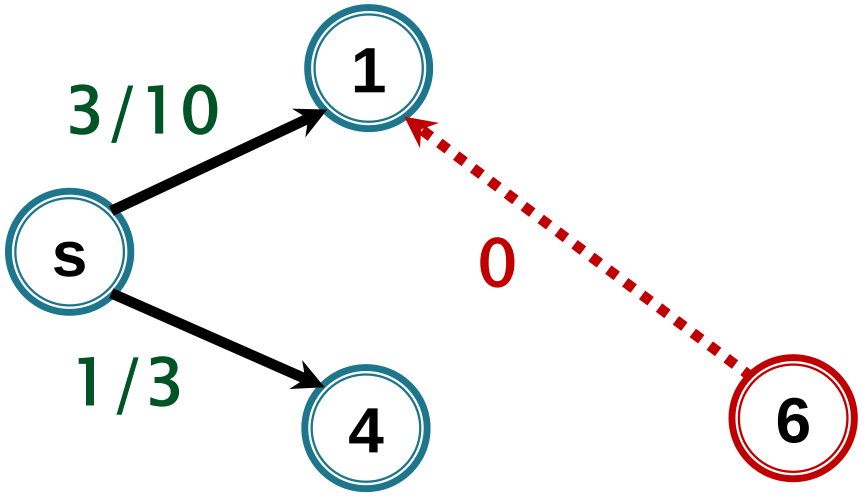

1
2
5
s
t
4
6
3
3/3
0/2
3/10
3/5
0/8
0/9
1/3
4/4
1/7
1/5

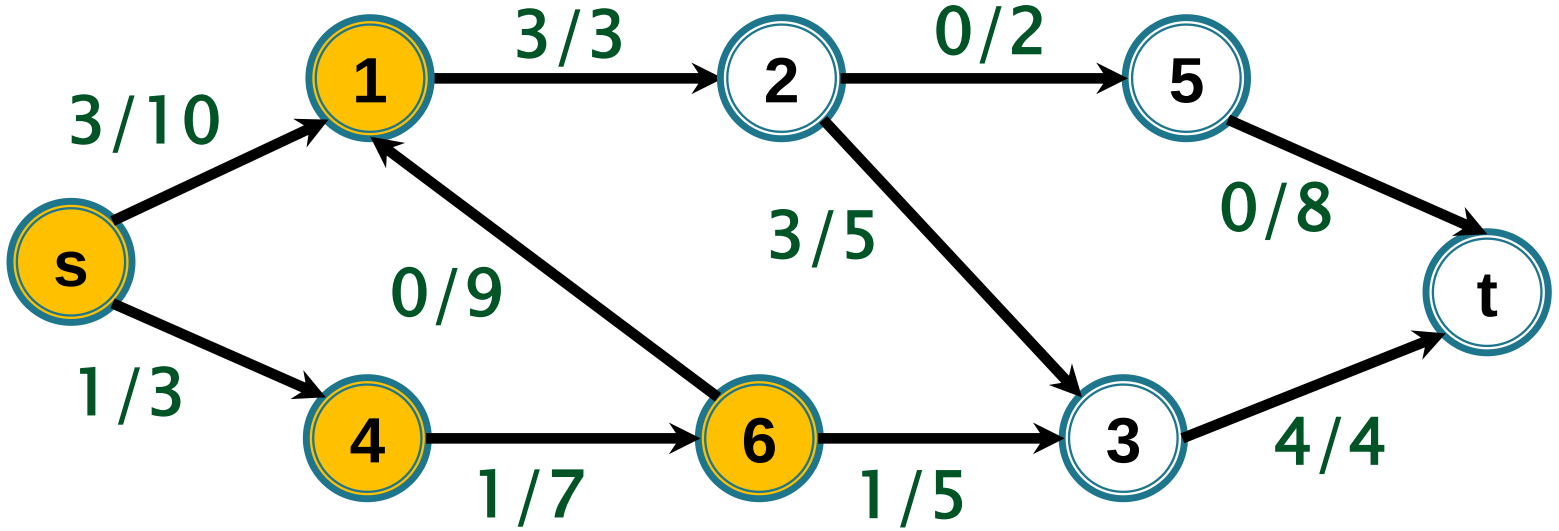

1
3/10
s
1/3
4

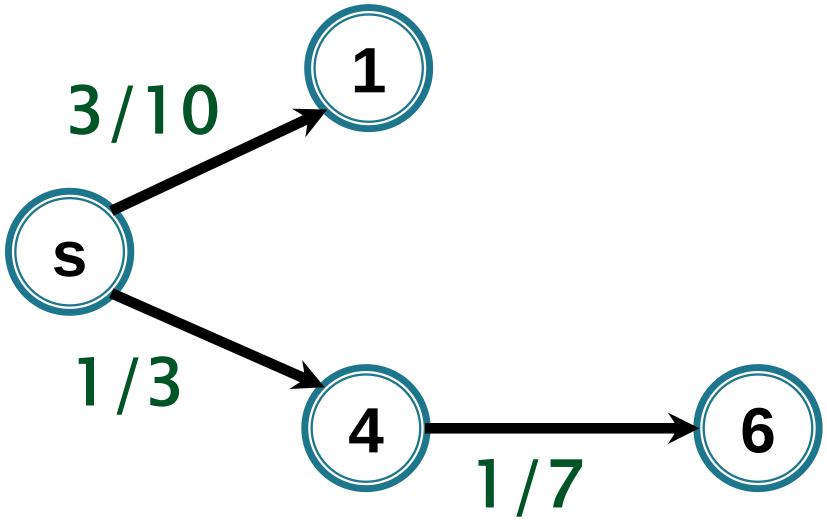

1
2
5
s
t
4
6
3
3/3
0/2
3/10
3/5
0/8
0/9
1/3
4/4
1/7
1/5

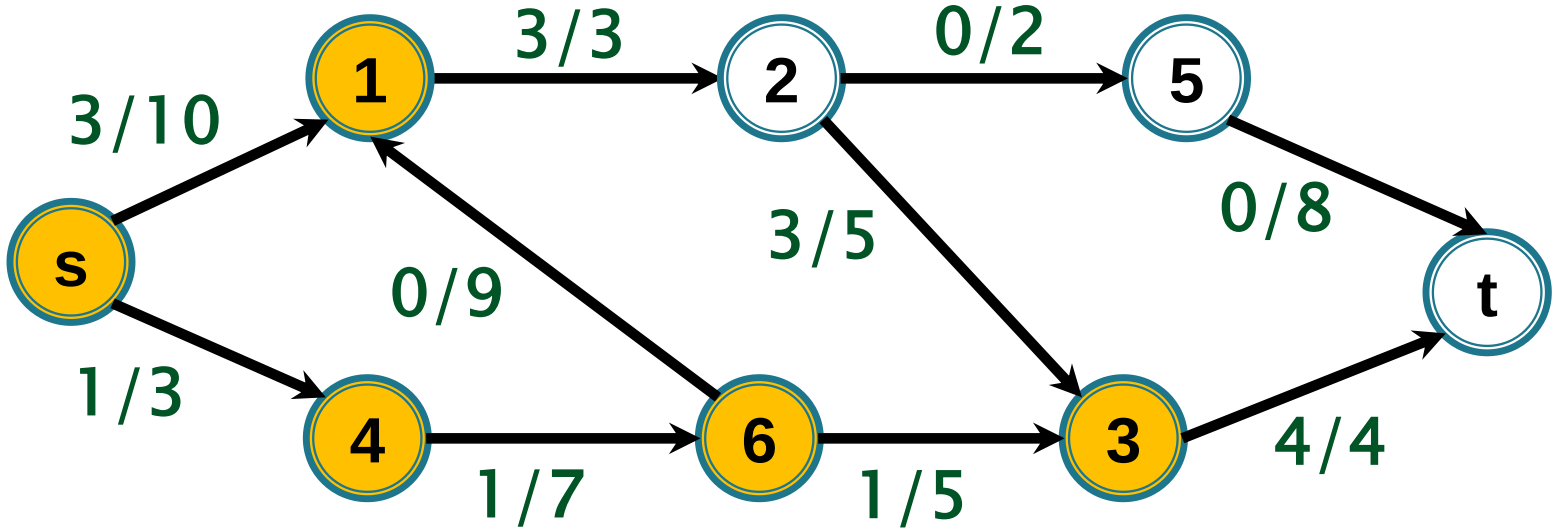

3-3=0
1
2
3/10
s
1/3
4

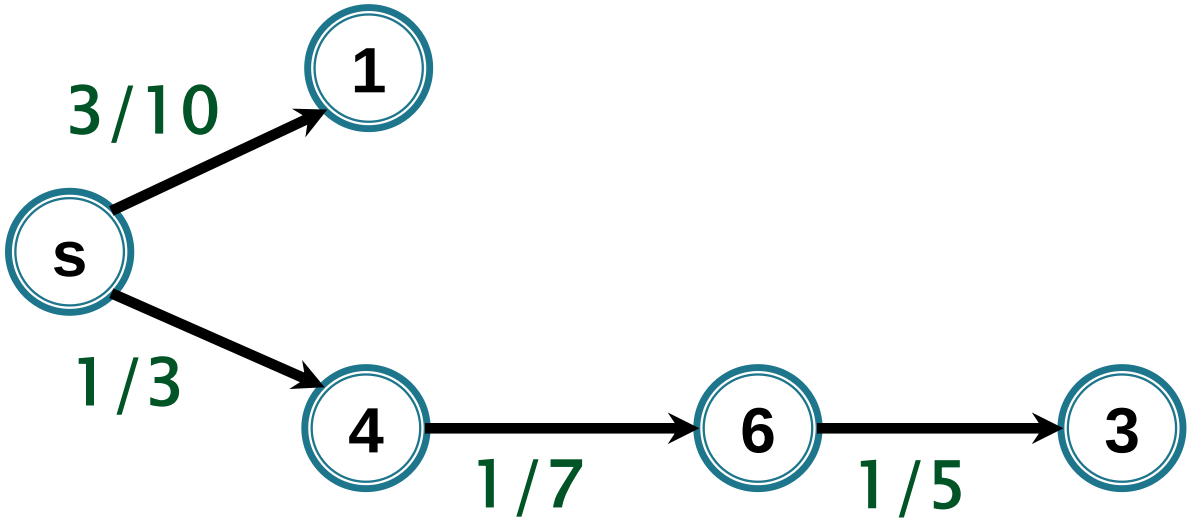

1
2
5
s
t
4
6
3
3/3
0/2
3/10
3/5
0/8
0/9
1/3
4/4
1/7
1/5

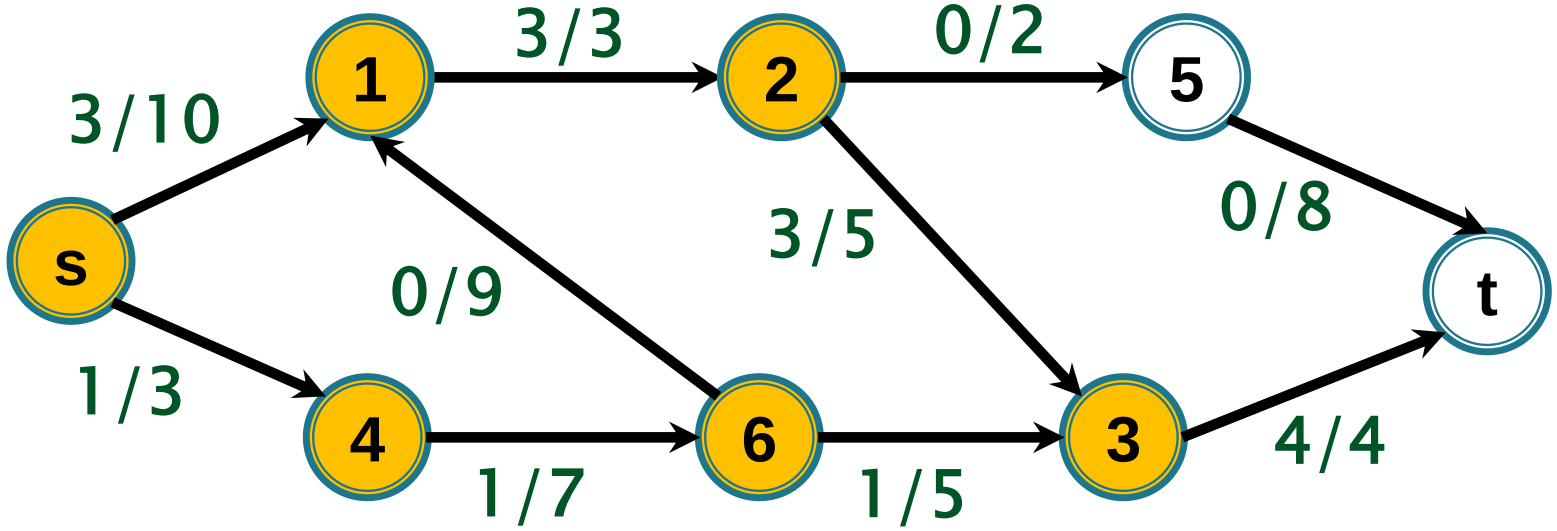

1
3/10
s
0
1/3
4
6

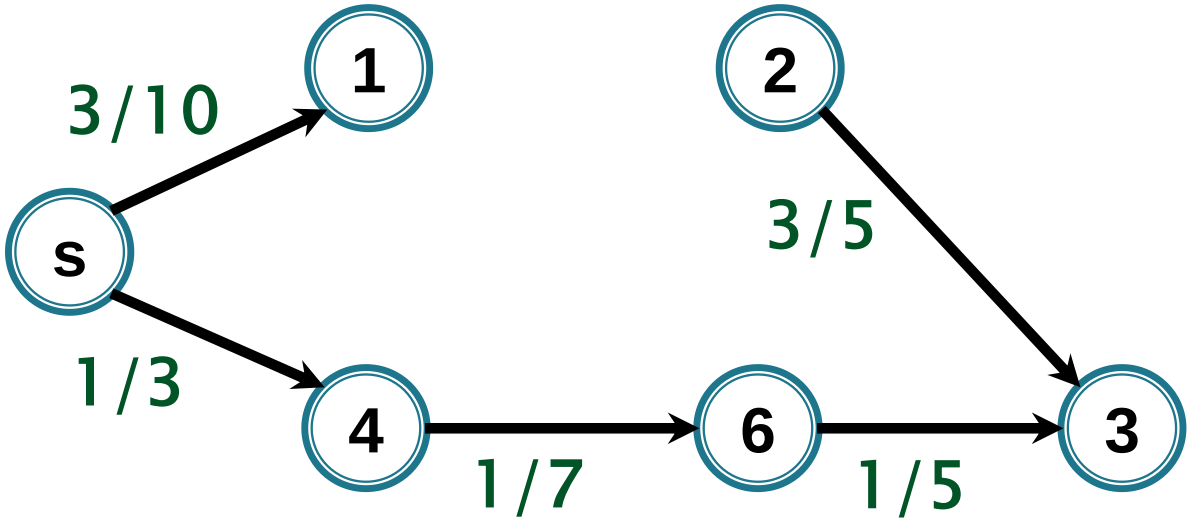

1
2
5
s
t
4
6
3
3/3
0/2
3/10
3/5
0/8
0/9
1/3
4/4
1/7
1/5

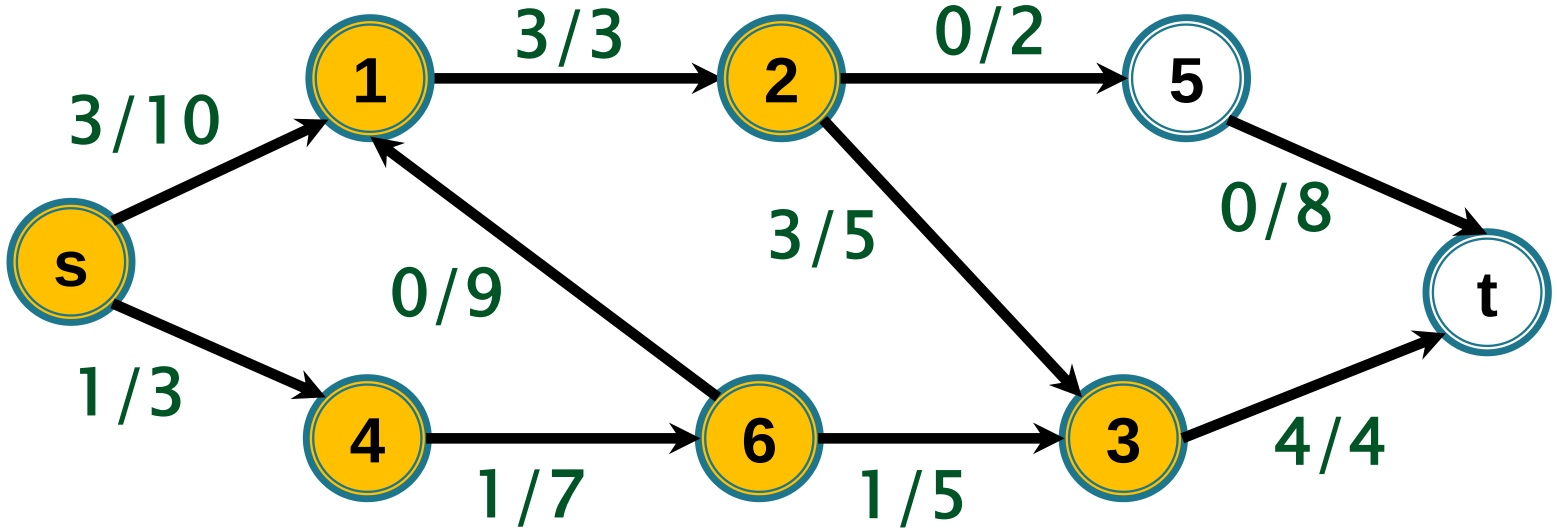

1
3/10
s
1/3
4
6
1/7

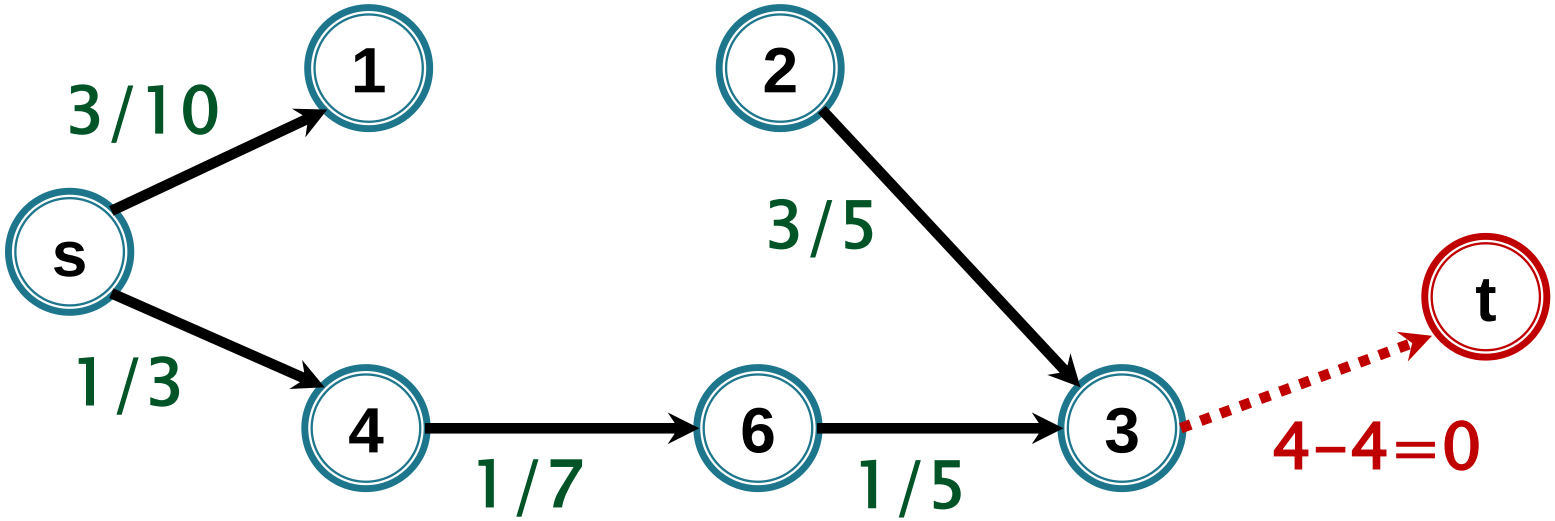

1
2
5
s
t
4
6
3
3/3
0/2
3/10
3/5
0/8
0/9
1/3
4/4
1/7
1/5

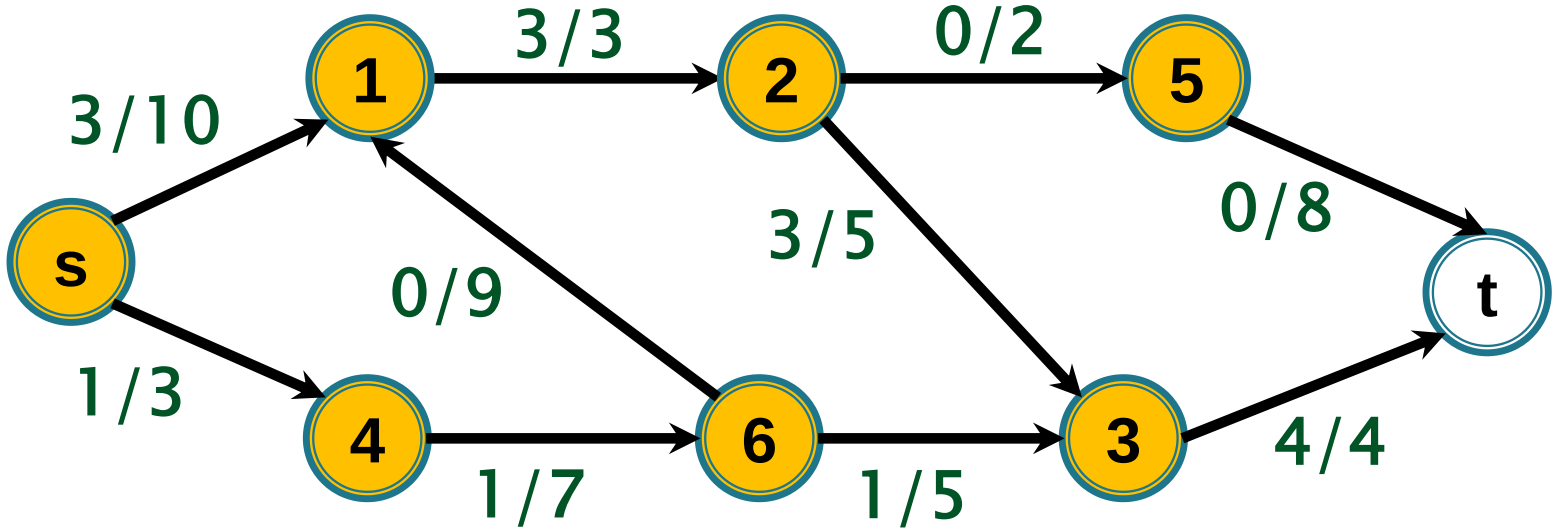

1
3/10
s
1/3
4
6
3
1/7
1/5

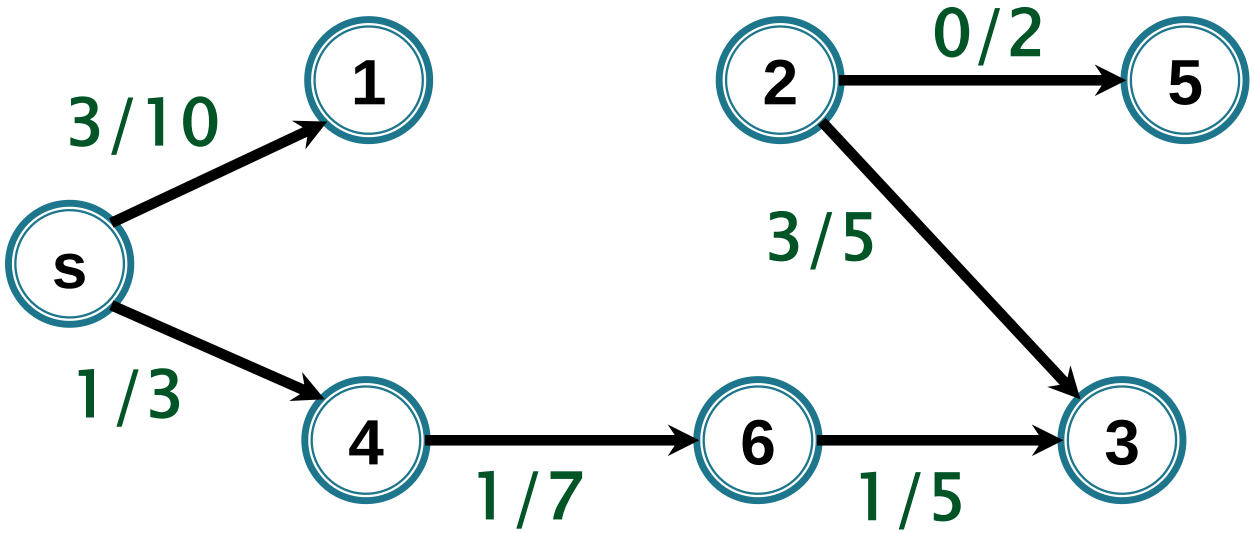

1
2
5
s
t
4
6
3
3/3
0/2
3/10
3/5
0/8
0/9
1/3
4/4
1/7
1/5

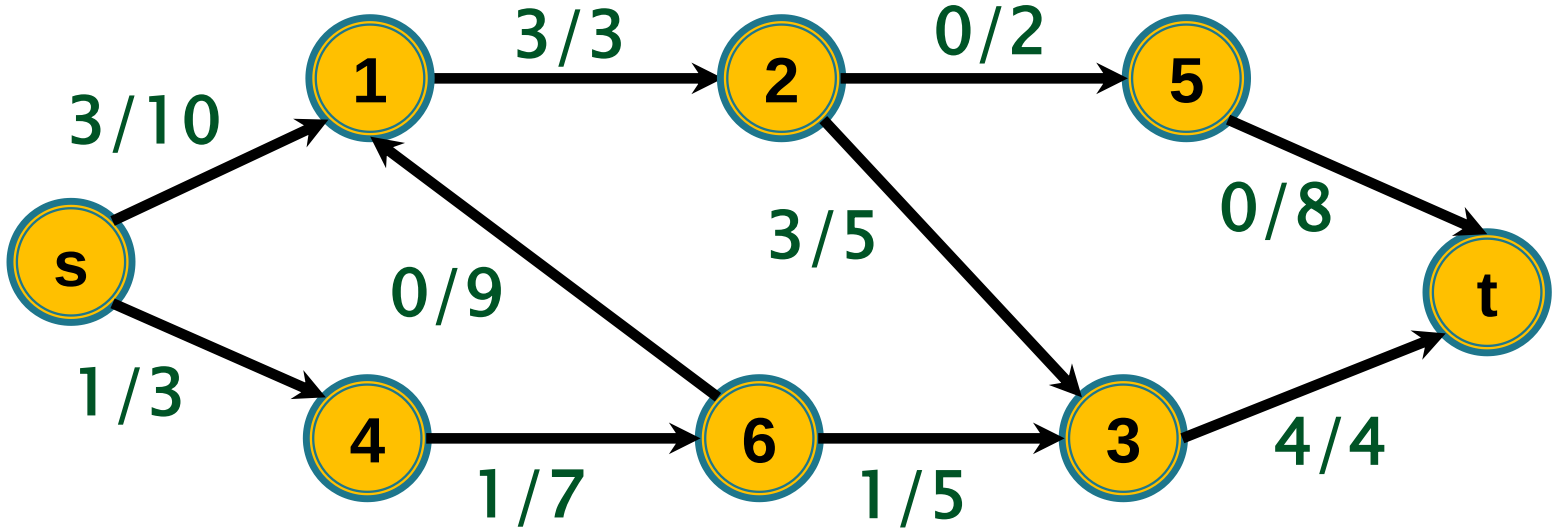

1
2
3/10
s
3/5
1/3
4
6
3
1/7
1/5

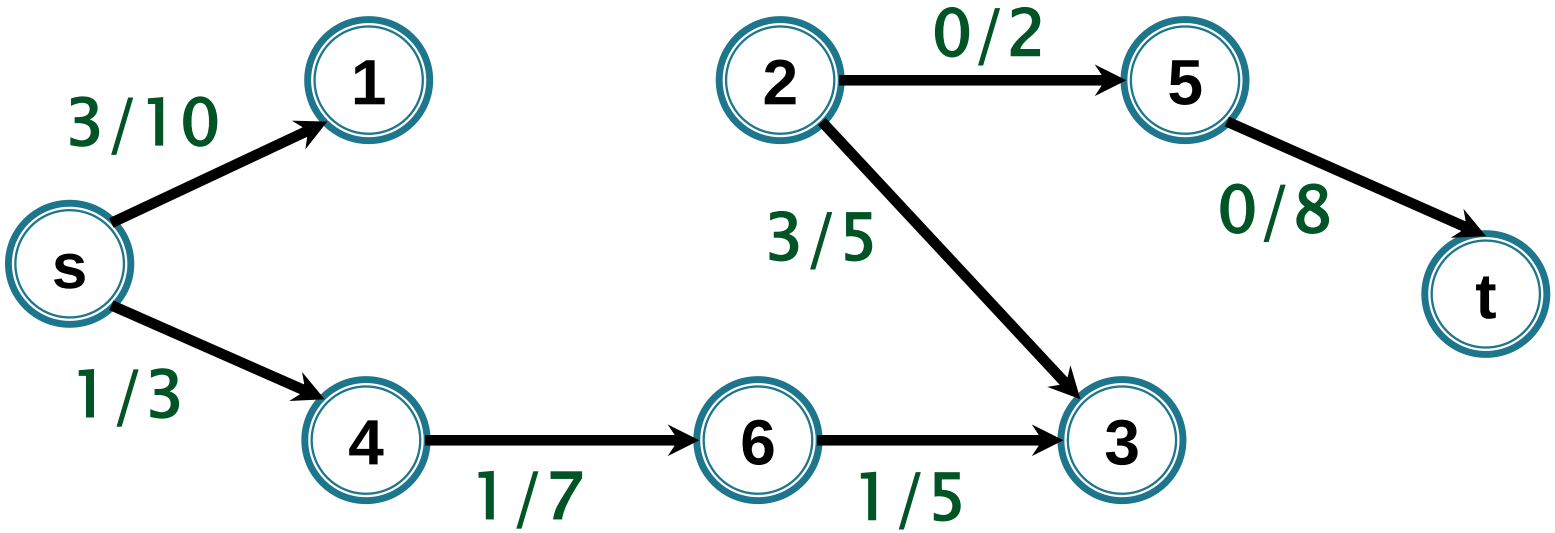

s
1
2
3
4
5
6
t
3/10
1/3
3/3
0/2
0/9
3/5
0/8
1/7
1/5
4/4


s
1
2
3
4
6
t
3/10
1/3
3/5
1/7
1/5
4−4=0

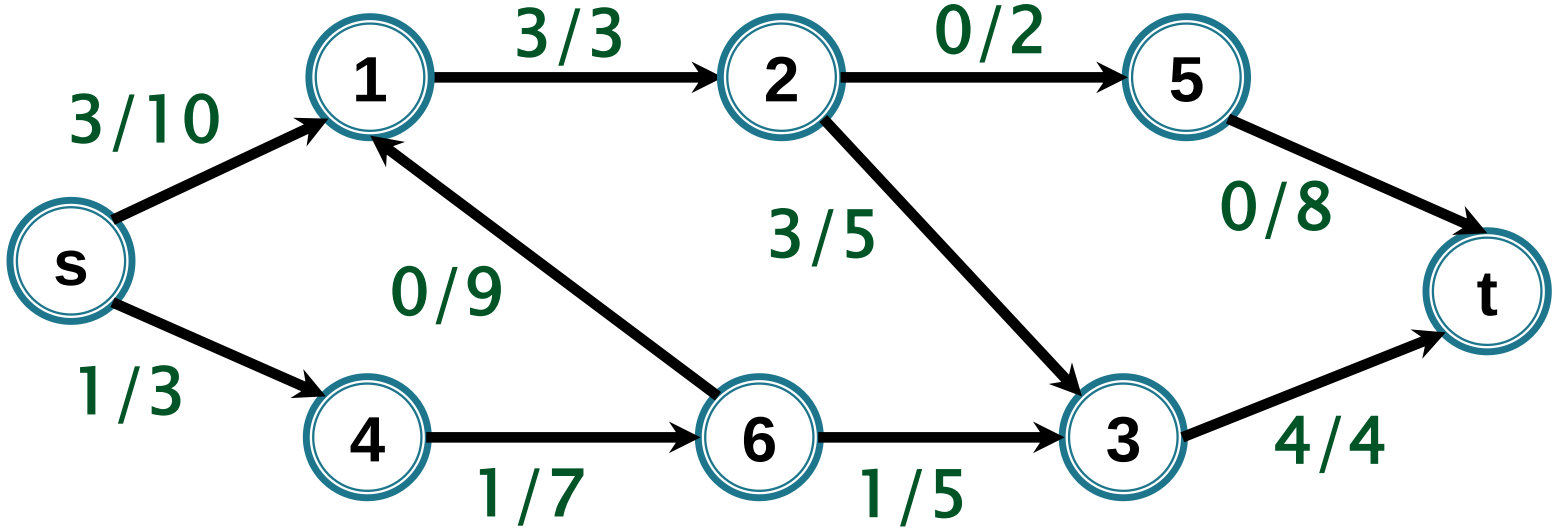

s
1
2
3
4
5
6
t
3/10
1/3
3/3
0/2
0/9
3/5
0/8
1/7
1/5
4/4

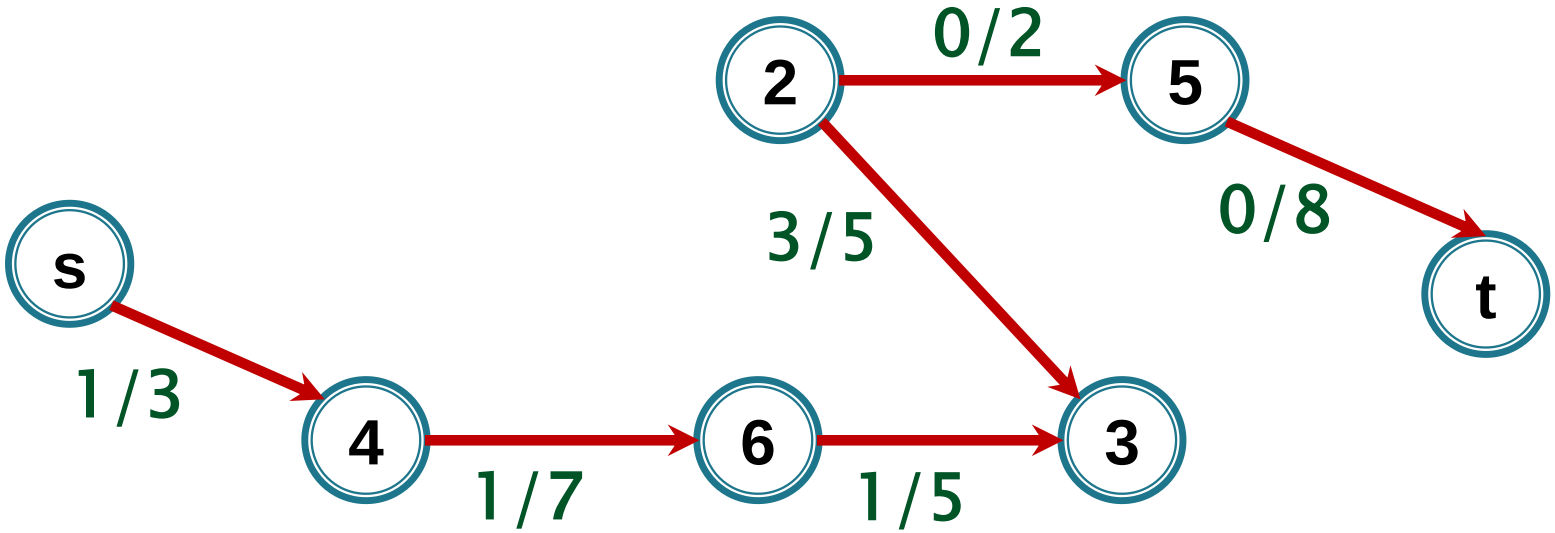

s
1
2
3
4
5
6
3/10
1/3
0/2
3/5
1/7
1/5

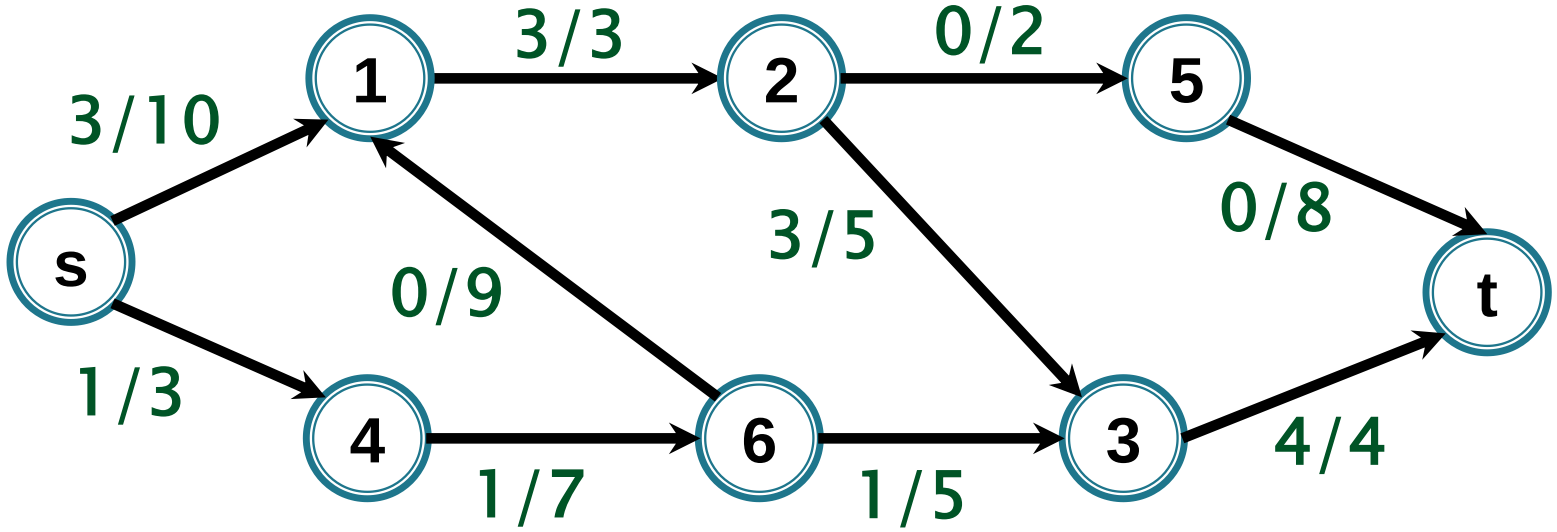

s
1
2
3
4
5
6
t
3/10
1/3
3/3
0/9
0/2
3/5
0/8
1/7
1/5
4/4


s
1
2
3
4
5
6
t
3/10
1/3
0/2
3/5
0/8
1/7
1/5

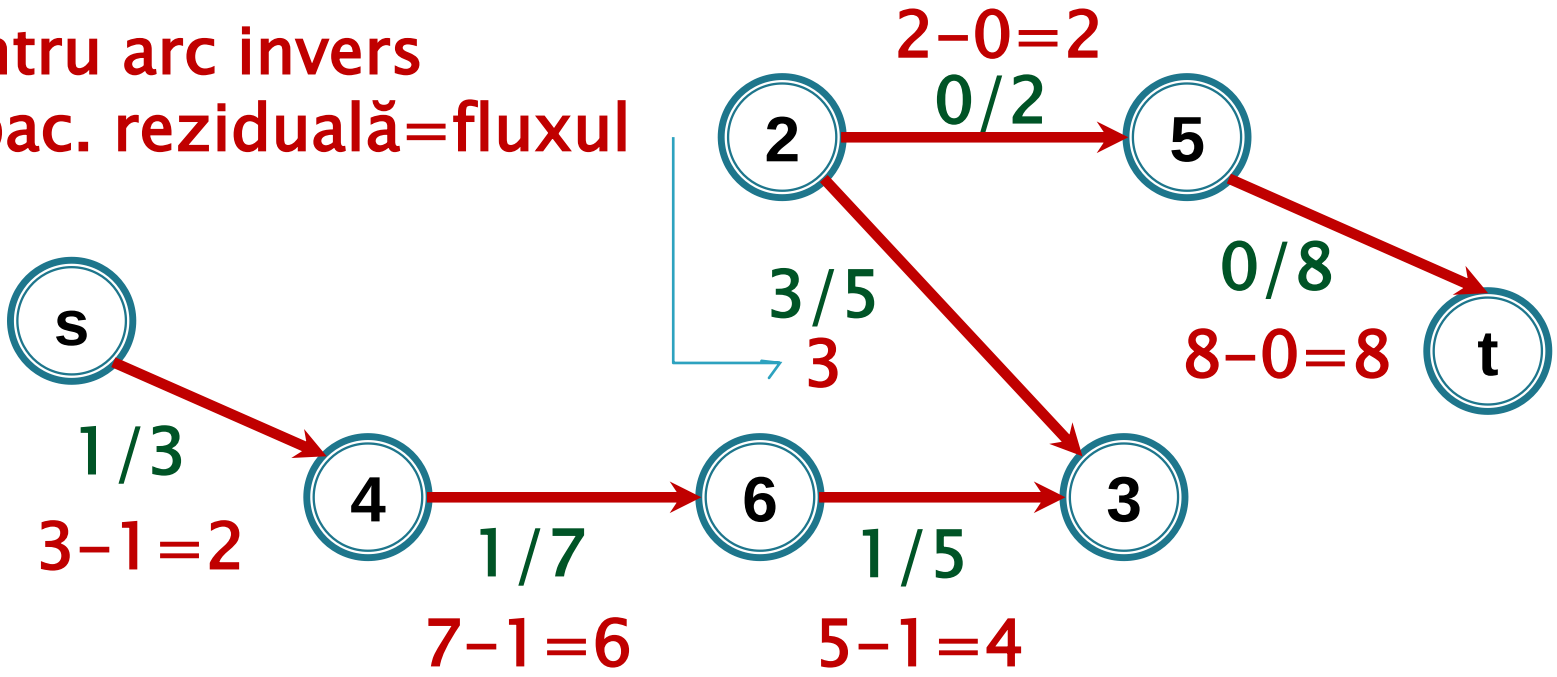
revizuieste_flux_lant

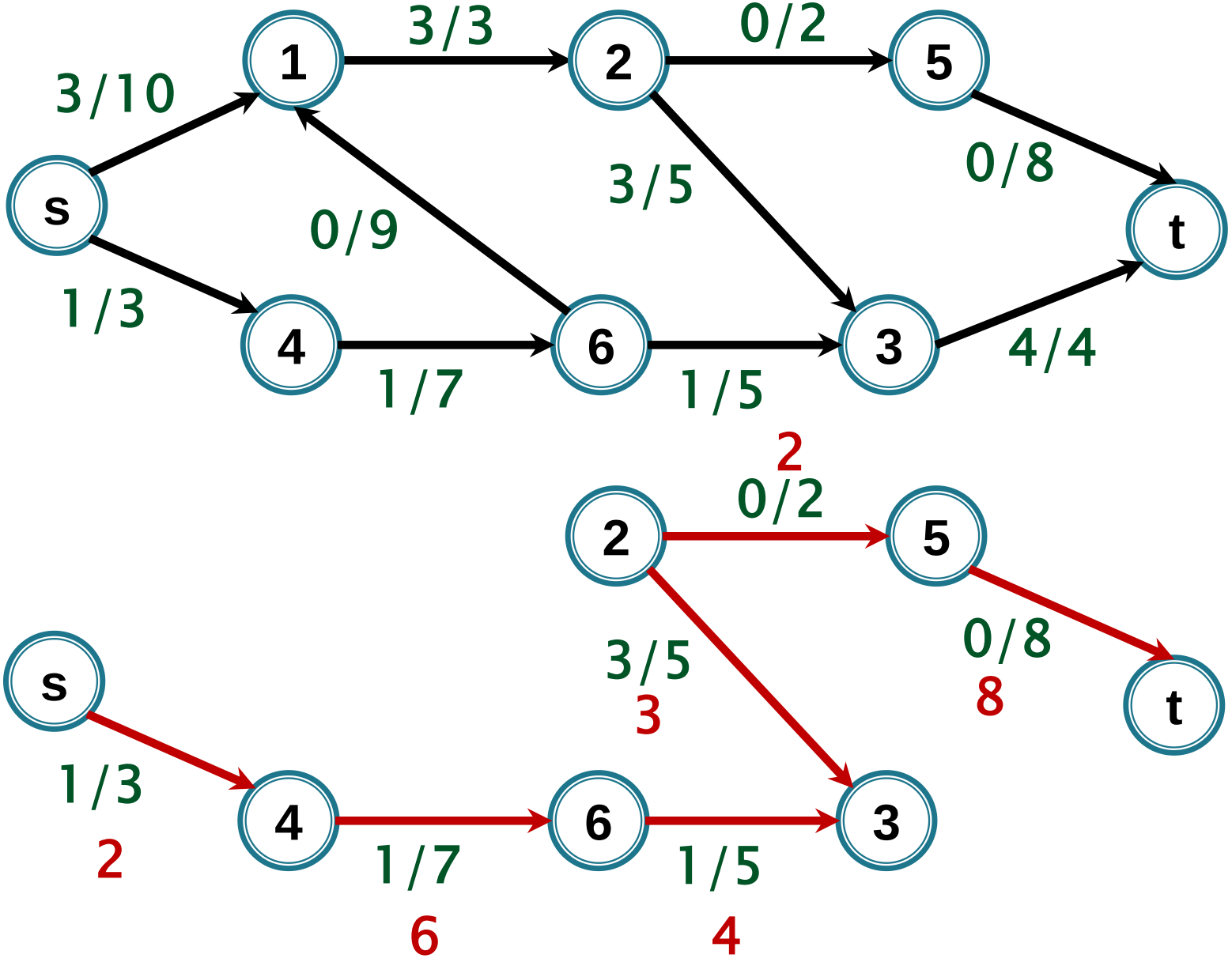

1
2
5
s
t
4
6
3
3/3
0/2
3/10
3/5
0/8
0/9
1/3
4/4
1/7
1/5


2
5
s
t
4
6
3
0/2
3/5
0/8
1/3
1/7
1/5

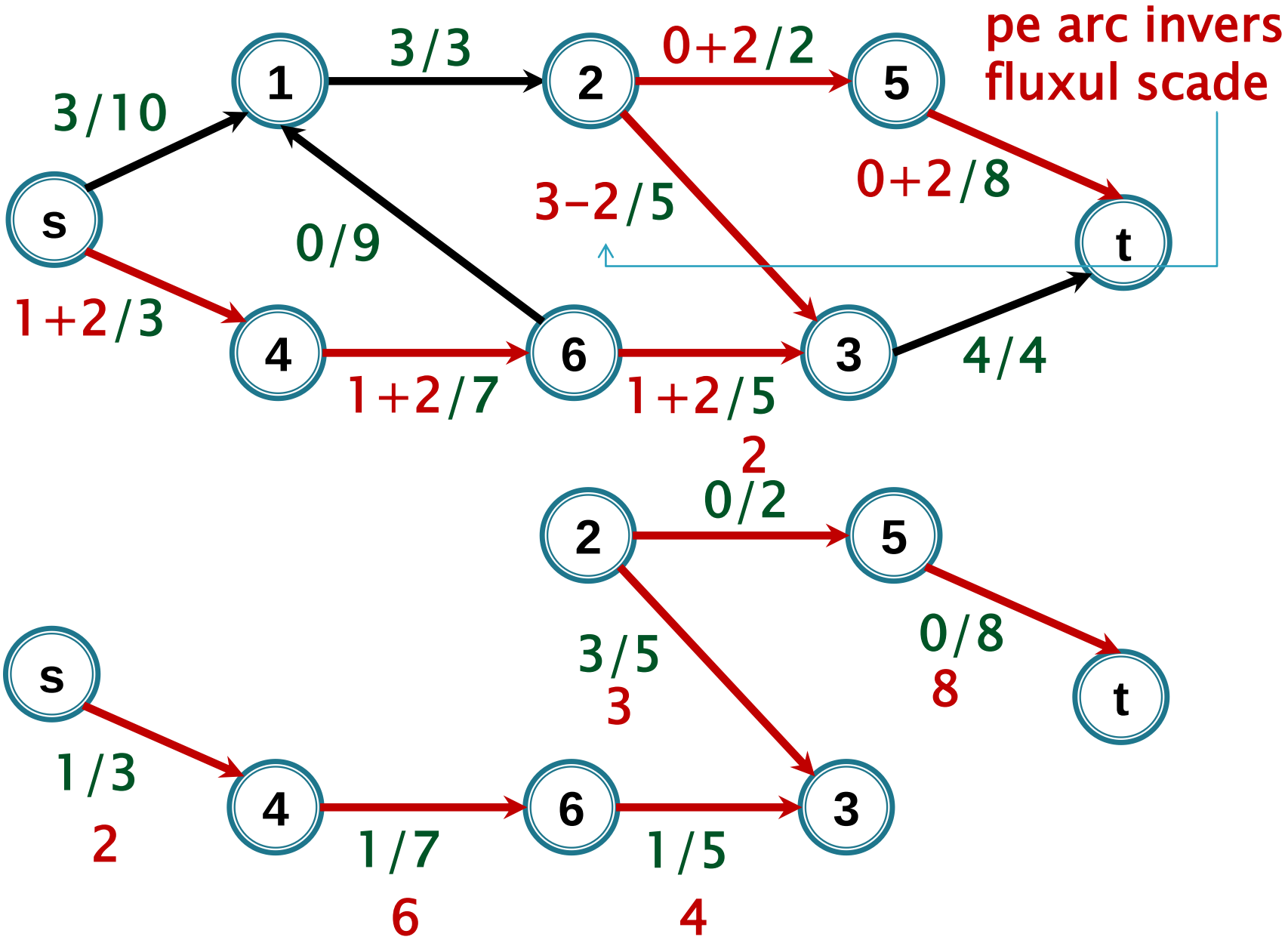


pentru arc invers
capac. reziduală=fluxul

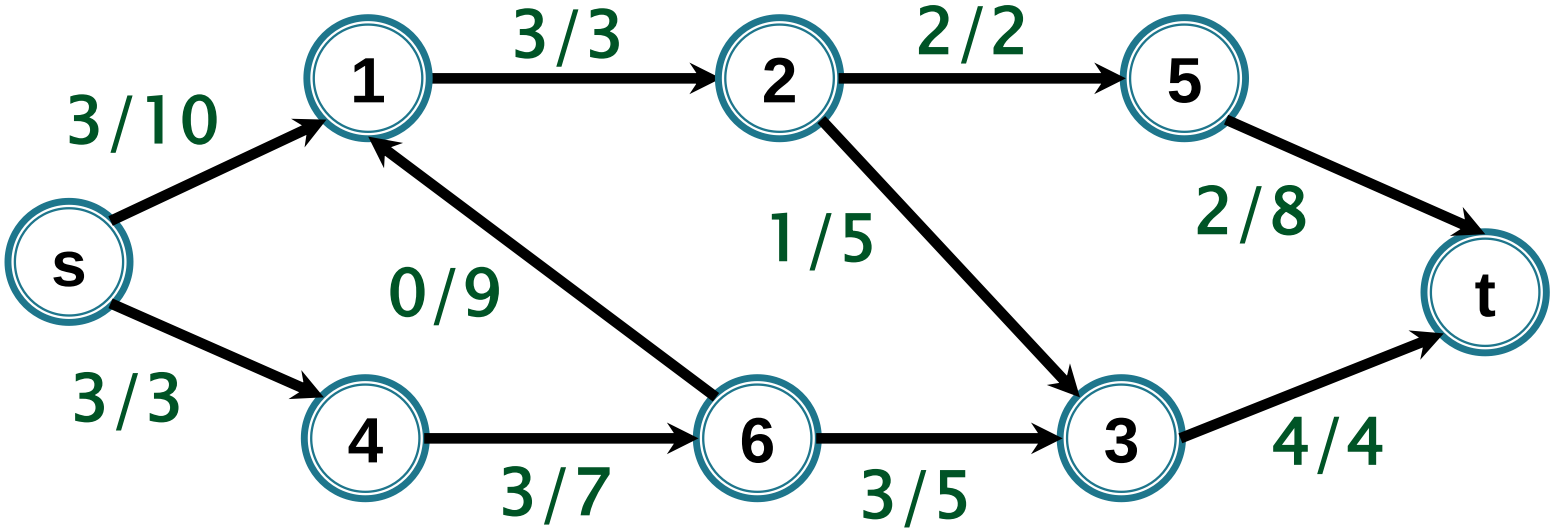




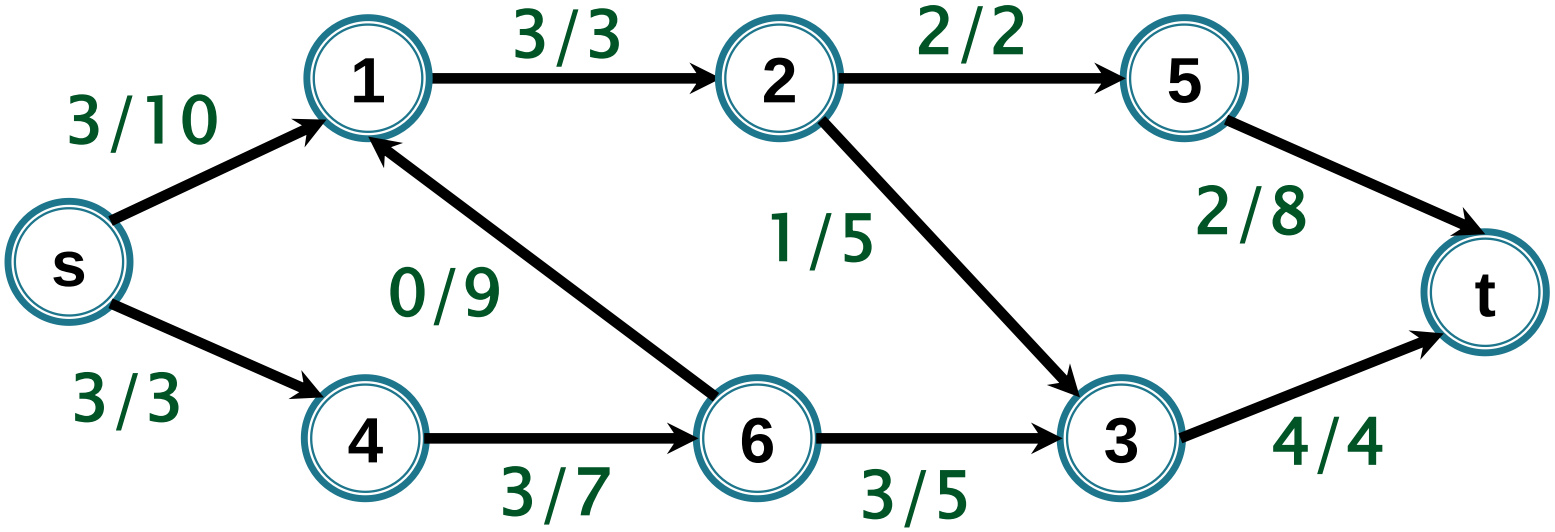
$$i(P) = \min\{ 2, 6, 4, 3, 2, 8\} = 2$$

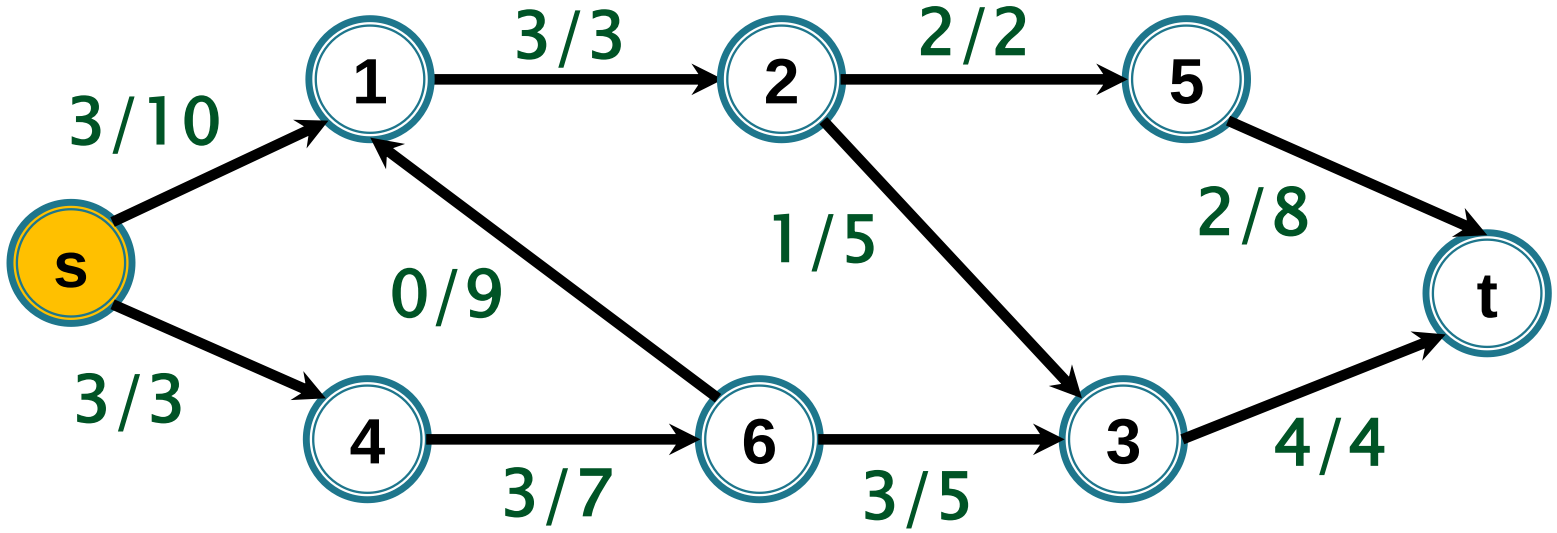


$$i(P) = \min\{ 2, 6, 4, 3, 2, 8\} = 2$$

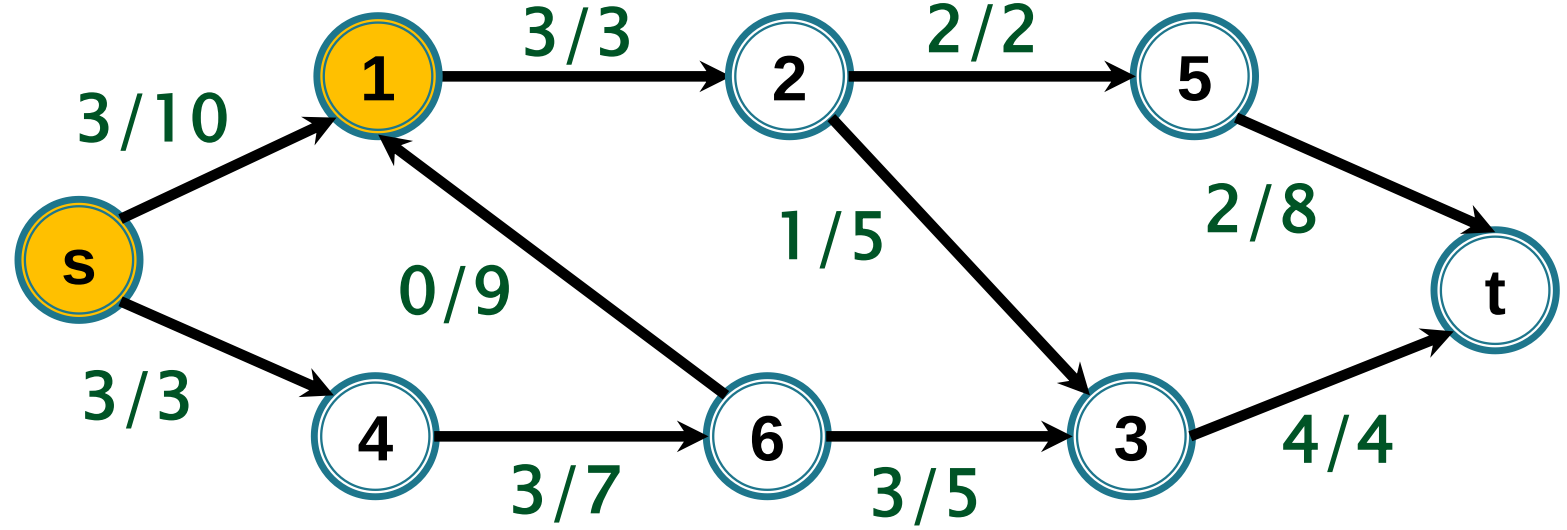

s
1
2
3
4
5
6
t
3/10
3/3
2/2
1/5
2/8
0/9
3/3
4/4
3/7
3/5

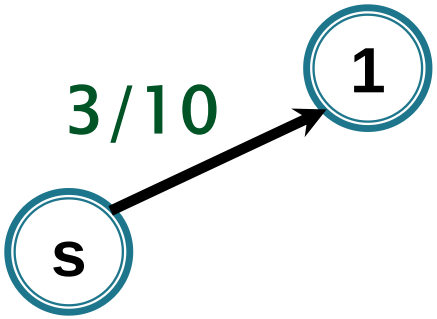
construieste_s-t_lant_nesat_BF

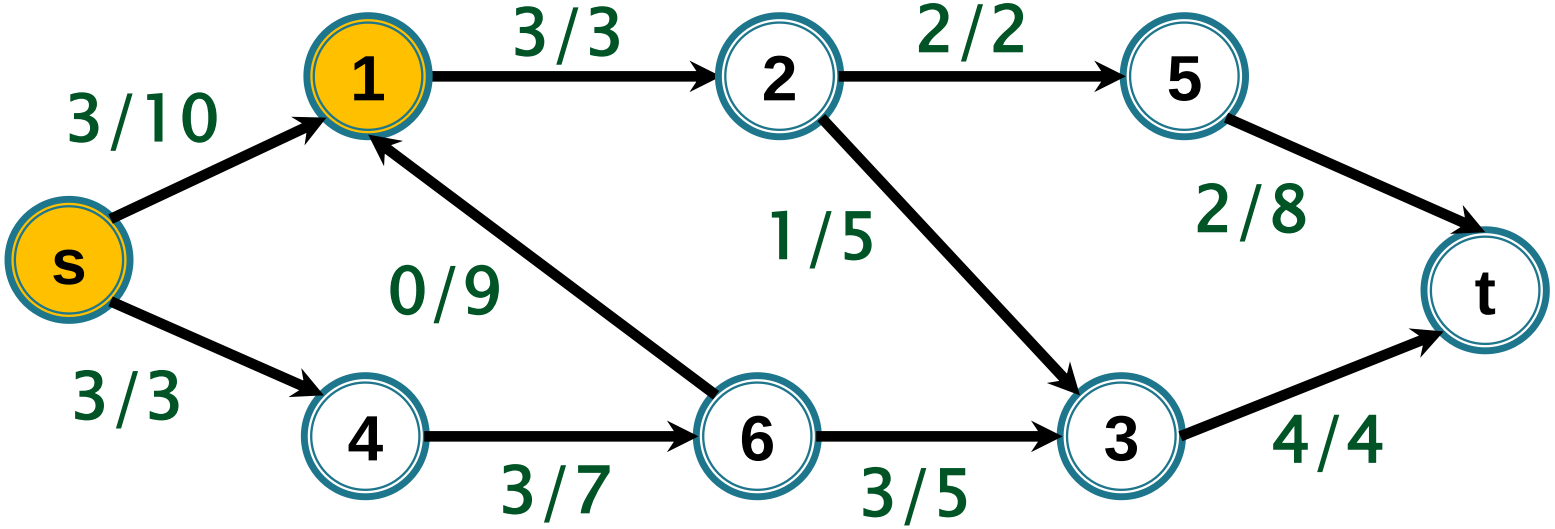

s
1
2
3
4
5
6
t
3/10
3/3
2/2
1/5
2/8
0/9
3/3
4/4
3/7
3/5

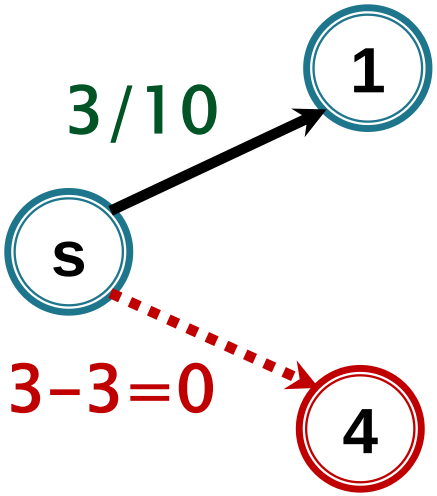

s
1
2
5
t
4
6
3
3/10
3/3
2/2
2/8
1/5
0/9
3/3
3/7
3/5
4/4

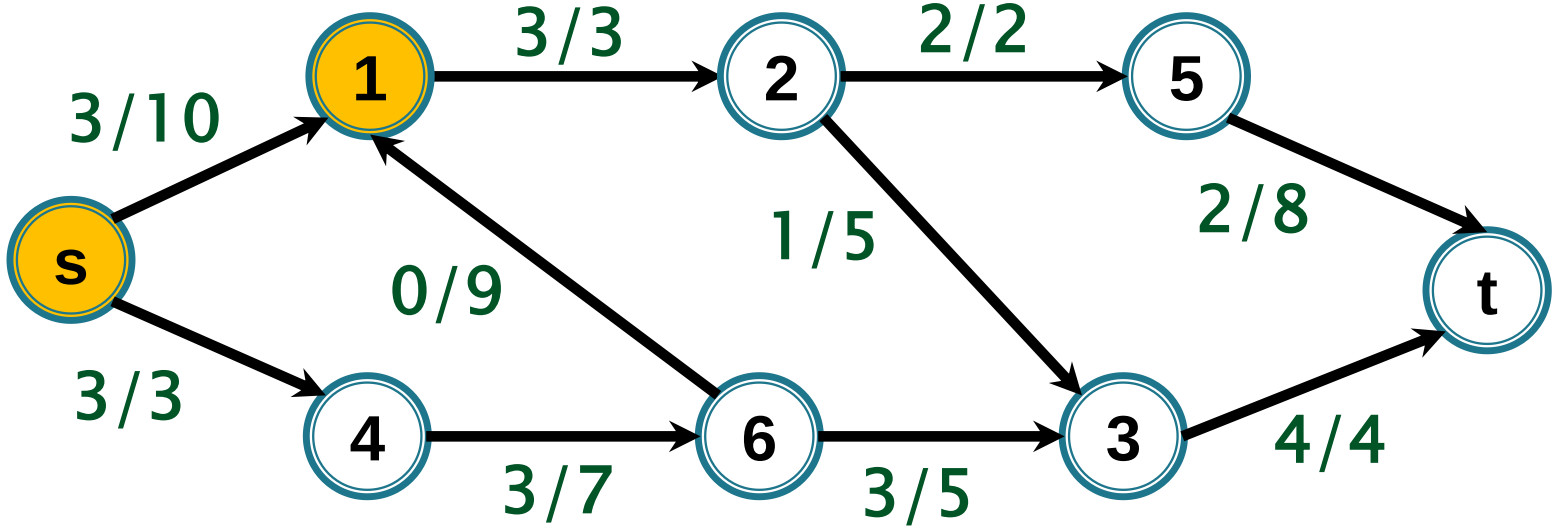

s

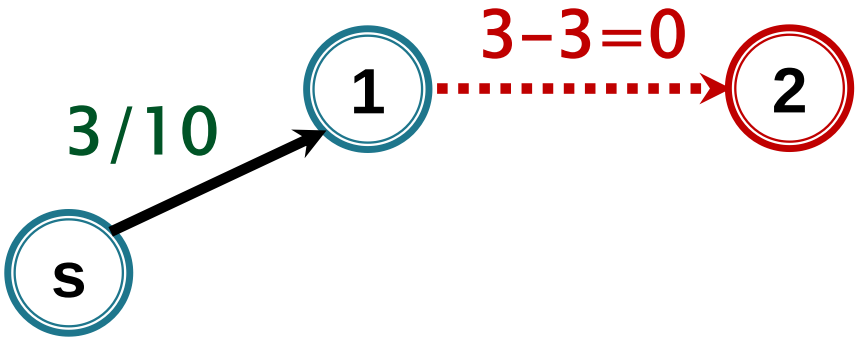

s
1
2
5
t
4
6
3
3/10
3/3
2/2
2/8
1/5
0/9
3/3
3/7
3/5
4/4

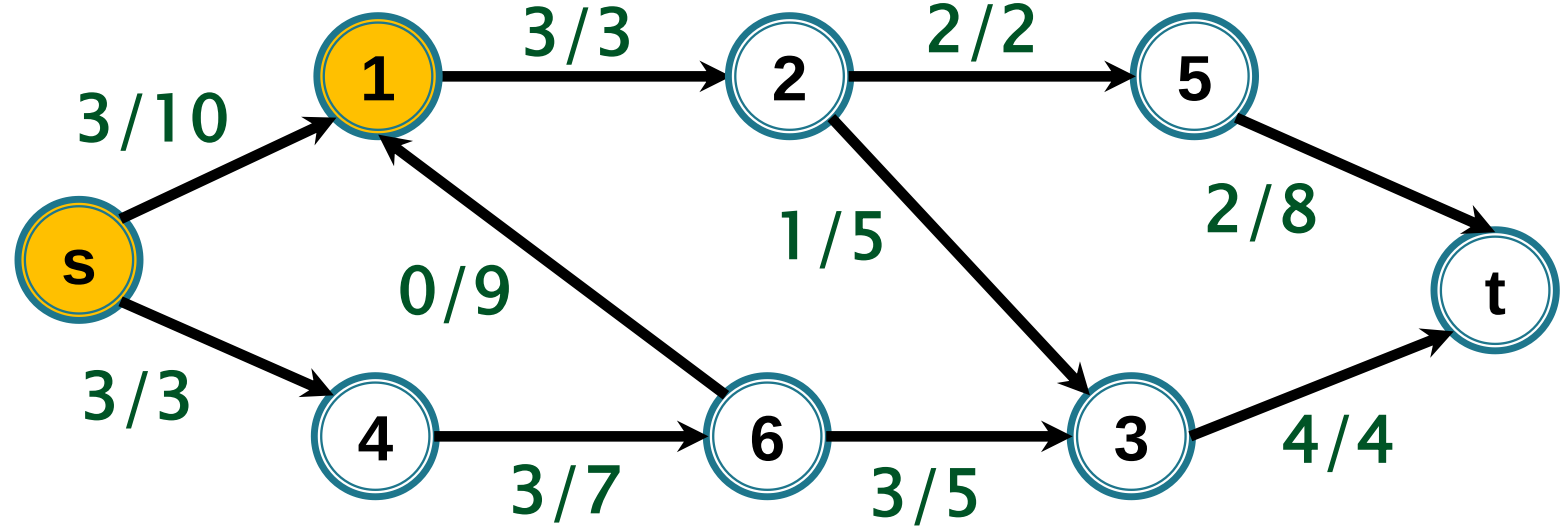

s
1
3/10

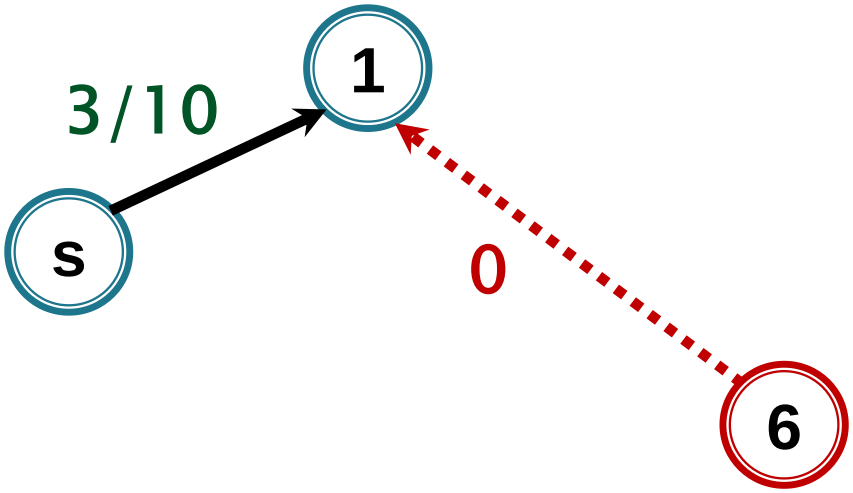

1
2
5
s
t
4
6
3
3/3
2/2
3/10
1/5
2/8
0/9
3/3
4/4
3/7
3/5


1
3/10
s
3-3=0
4


1
2
5
s
t
4
6
3
3/3
2/2
3/10
1/5
2/8
0/9
3/3
4/4
3/7
3/5


3-3=0
1
2
3/10
s


1
2
5
s
t
4
6
3
3/3
2/2
3/10
1/5
2/8
0/9
3/3
4/4
3/7
3/5


1
3/10
s
0
6

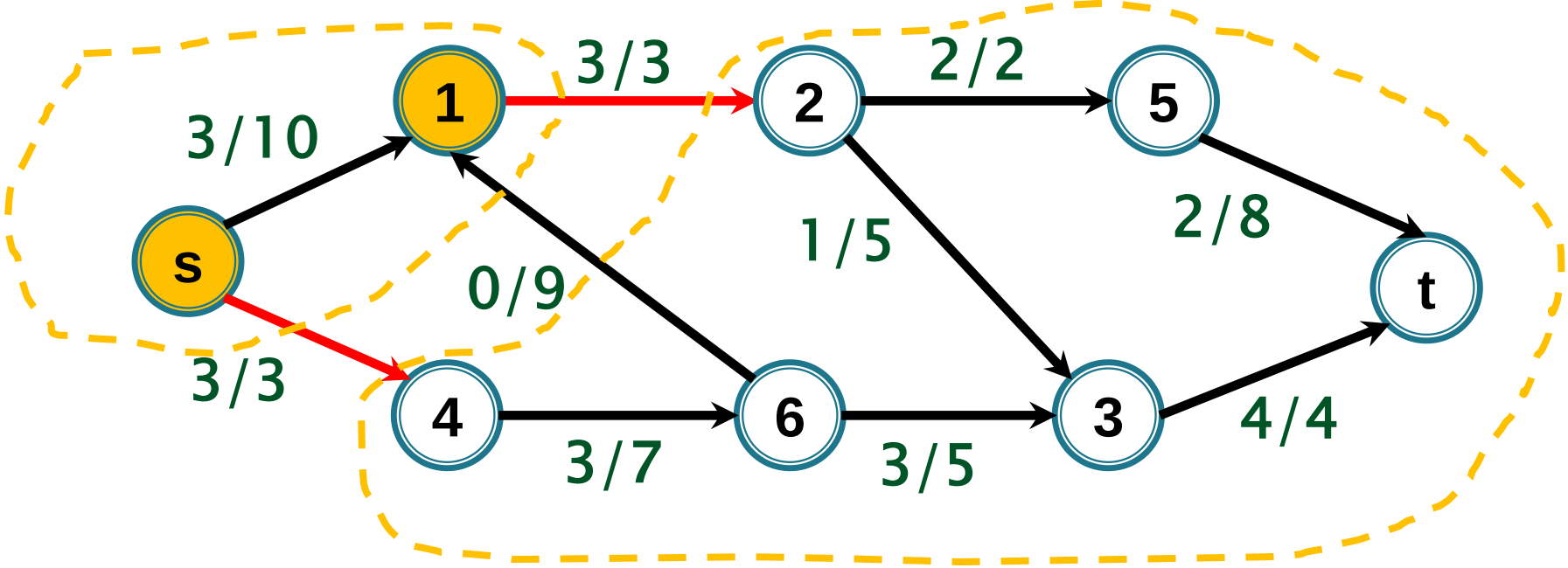
t nu este accesibil din s $\Rightarrow$ STOP

- f este flux maxim

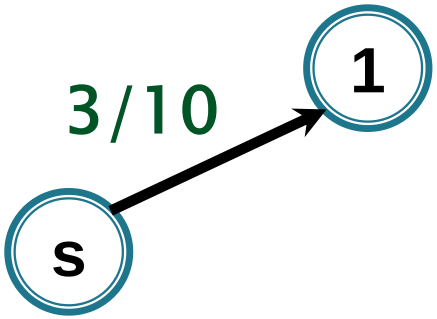
t nu este accesibil din s ⇒ STOP

- 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 minimă



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

construieste_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∈V) executa tata[v] ←0; viz[v] ←0
  coada C ← ∅
  adauga(s, C)
  viz[s]← 1
  cat timp C ≠ ∅ executa
     i ←  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

$\Leftrightarrow$

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

Varianta 2 de implementare

revizuirea fluxului folosind s-t drumuri în G_f (în graful rezidual)

Algoritmul Edmonds-Karp – implementare folosind graf rezidual

Schema devine:

initializeaza_flux_nul()

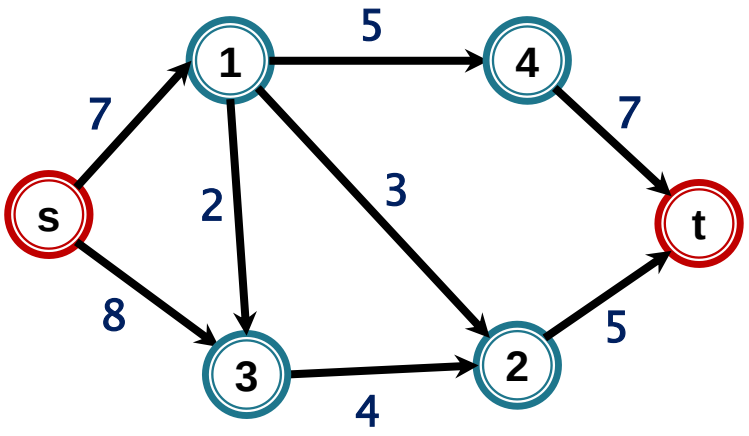
cat timp (**construieste_s-t_drum_in G_f_BF**()=true) executa

revizuieste_flux_lant()

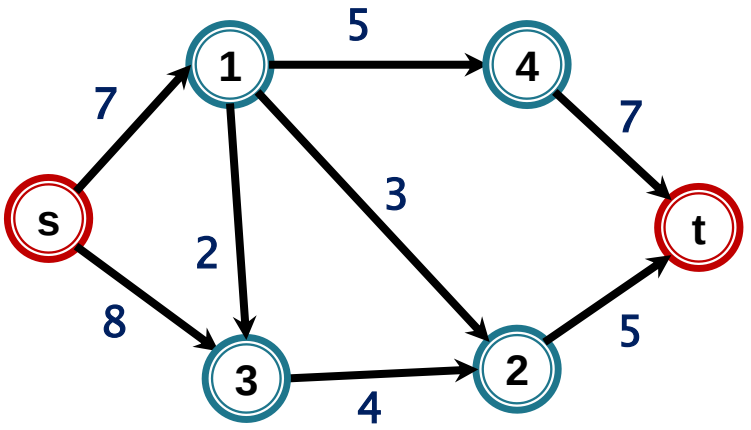
actualizeaza G_f

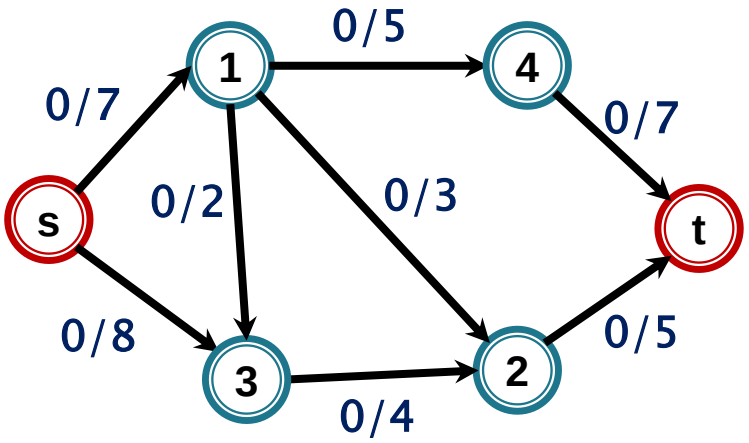
afiseaza_flux()

Detaliem această schemă

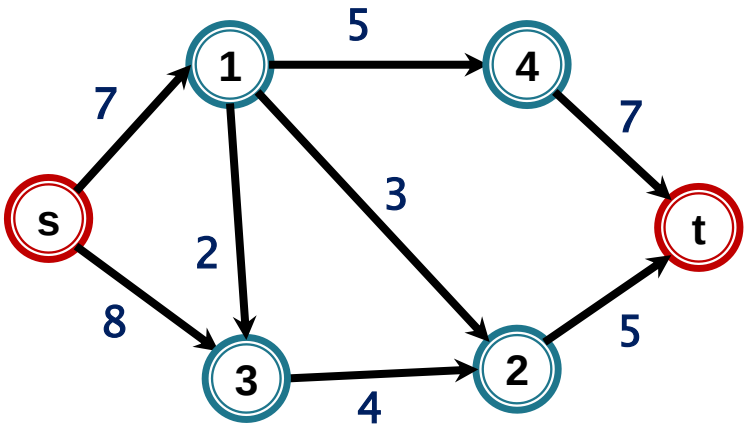


Graful rezidual G_f

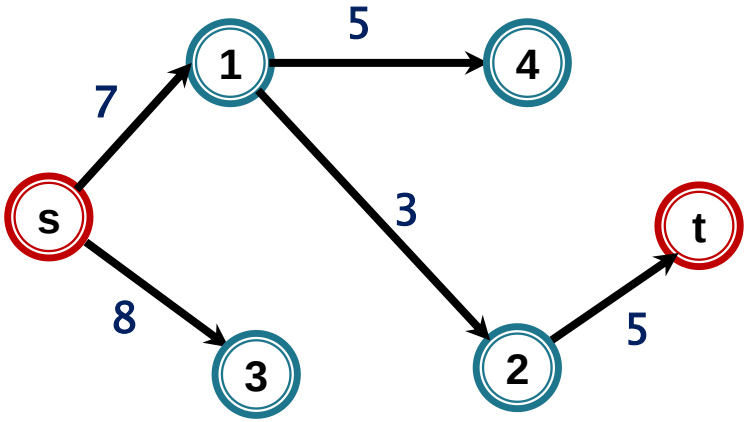




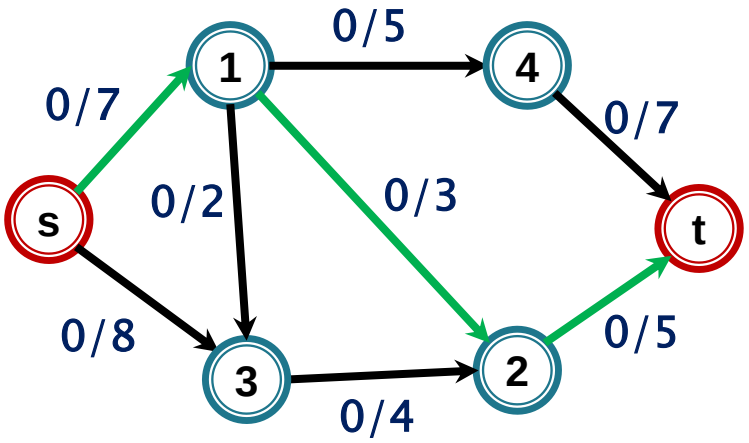
Graful rezidual G_f

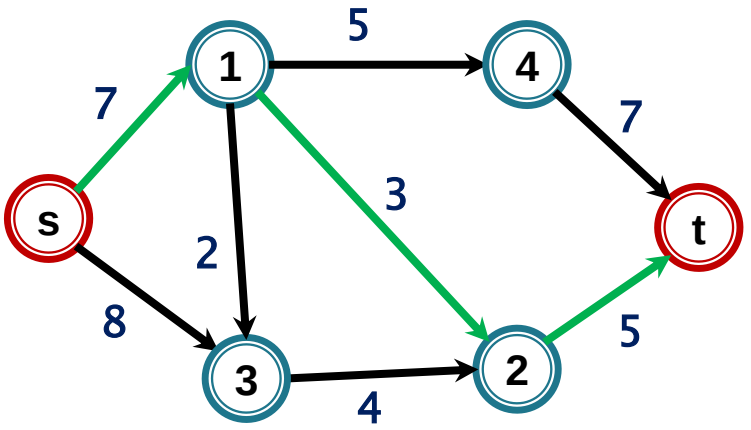


BF(s) – în graful rezidual

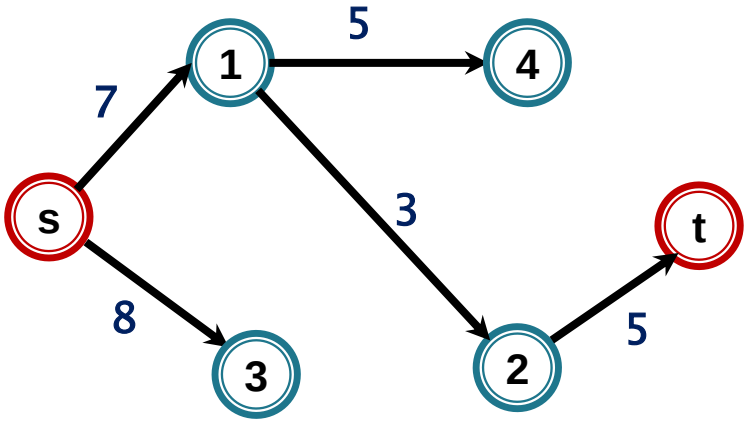


Graful rezidual G_f





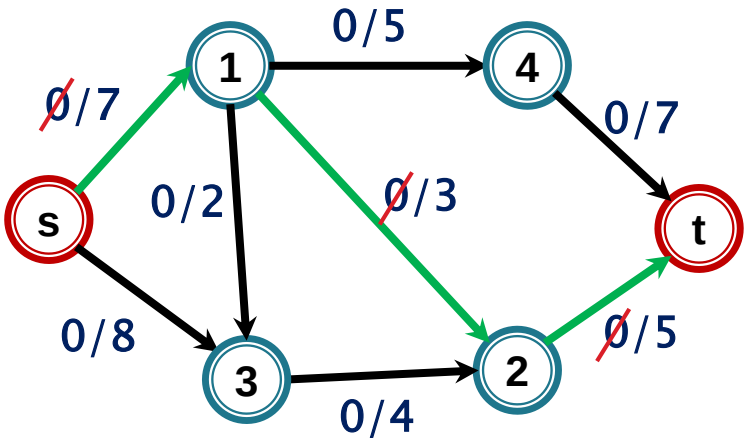
BF(s) – în graful rezidual

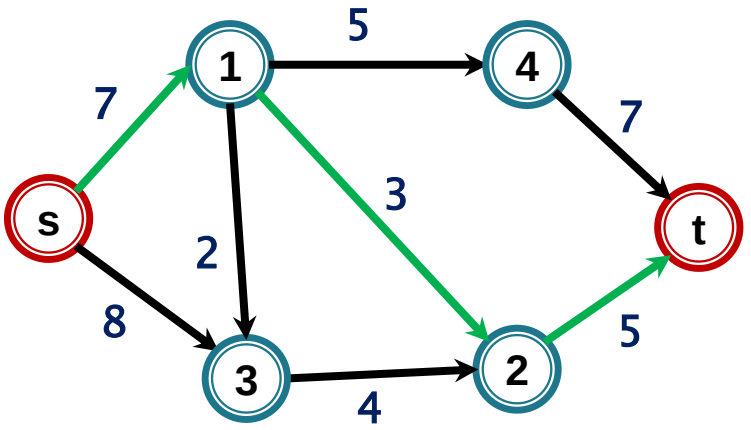


Drumul de creștere [s, 1, 2, t] – capacitate reziduală 3

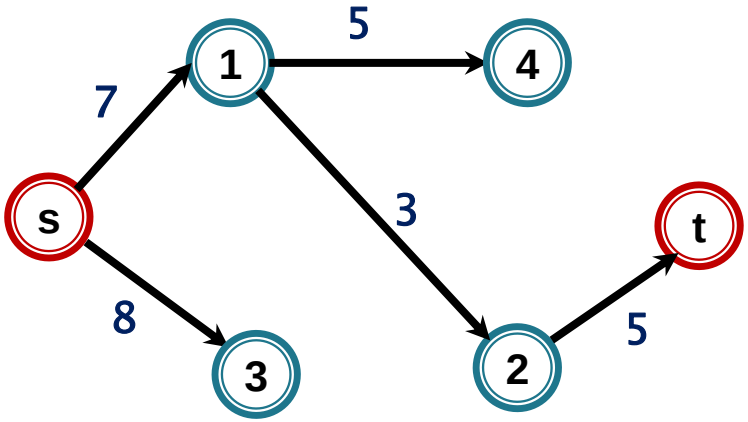
Revizuim fluxul

Graful rezidual G_f





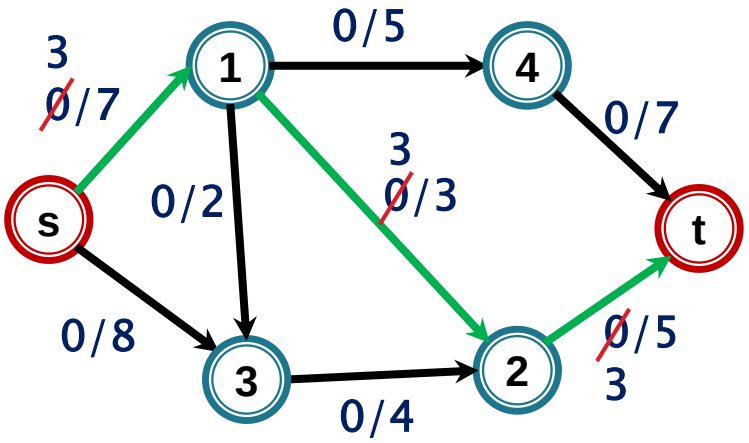
BF(s) – în graful rezidual

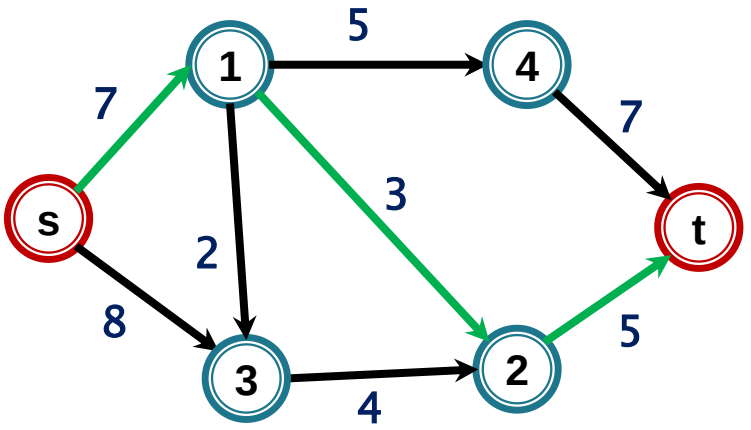


Drumul de creștere [s, 1, 2, t] – capacitate reziduală 3

Revizuim fluxul

Graful rezidual G_f





Actualizăm rețeaua reziduală

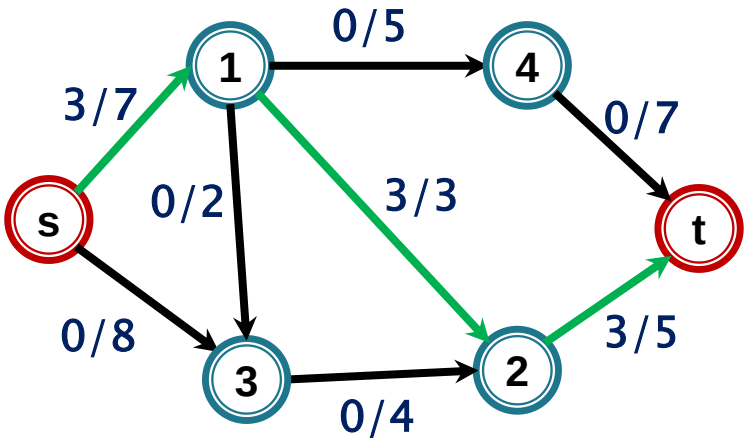
pentru e ∈ E(P) ⊆ E(G_f)

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

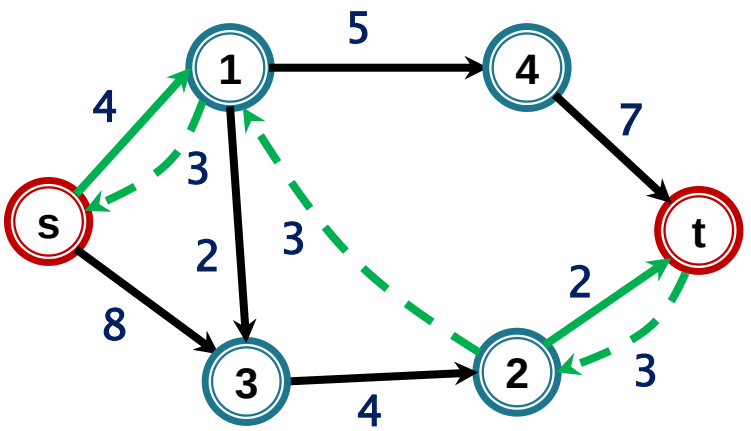
dacă $c_f(e)=0$ elimina e din G_f (se ignora in parcurgere)

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

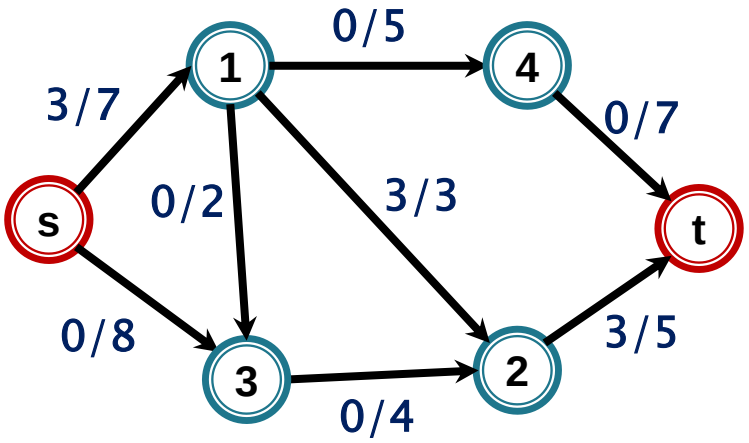
dacă $c_f(e^{-1})>0$ și $e^{-1} \notin G_f$ atunci adauga e^{-1} la G_f

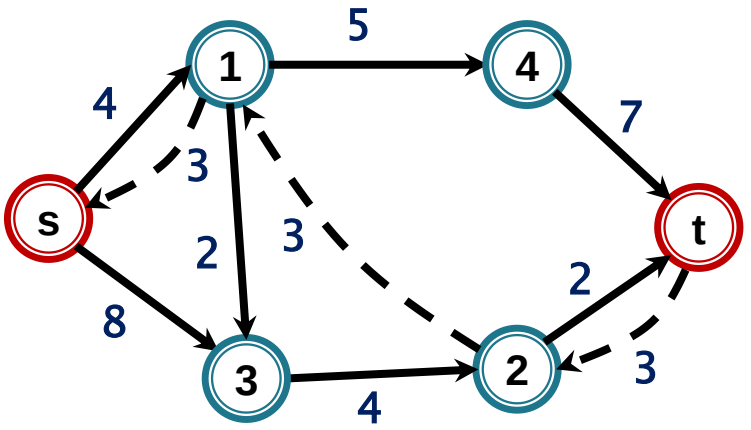


Graful rezidual G_f

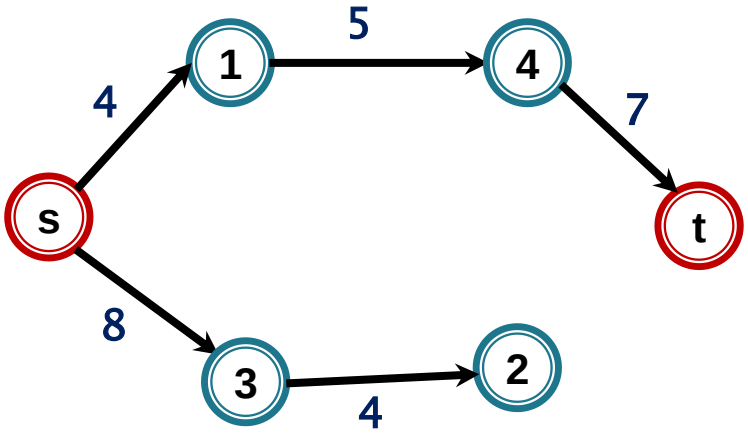


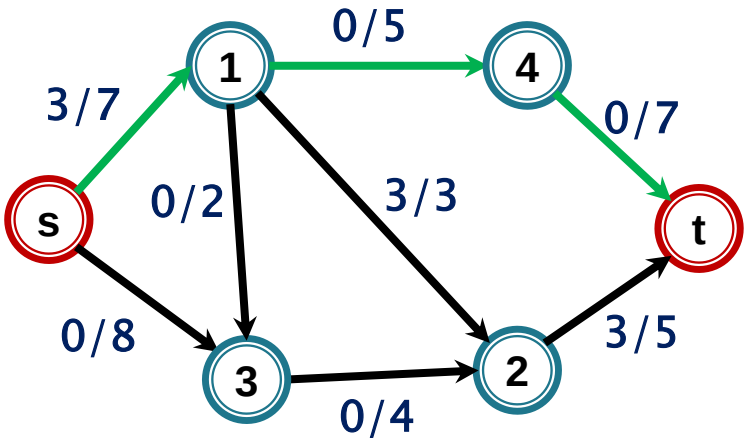
Graful rezidual G_f



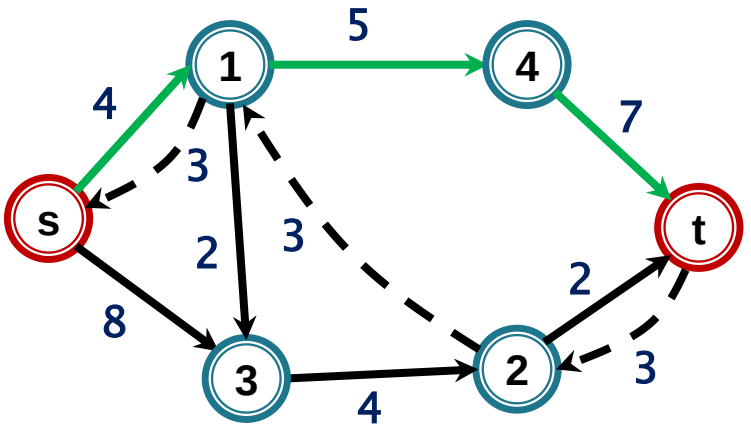


BF(s) – în graful rezidual

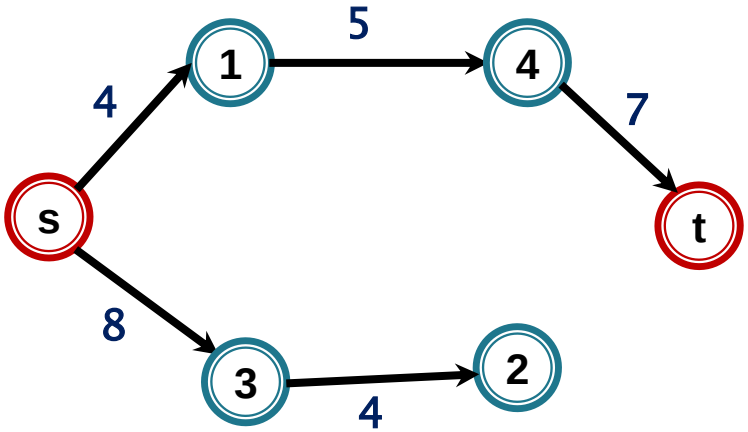




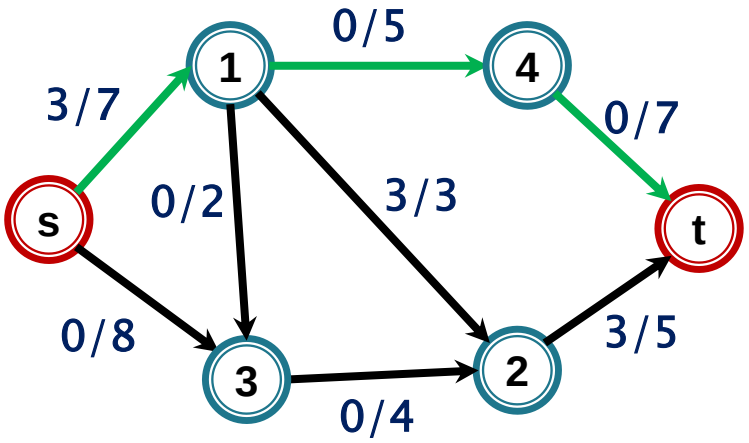
Graful rezidual G_f

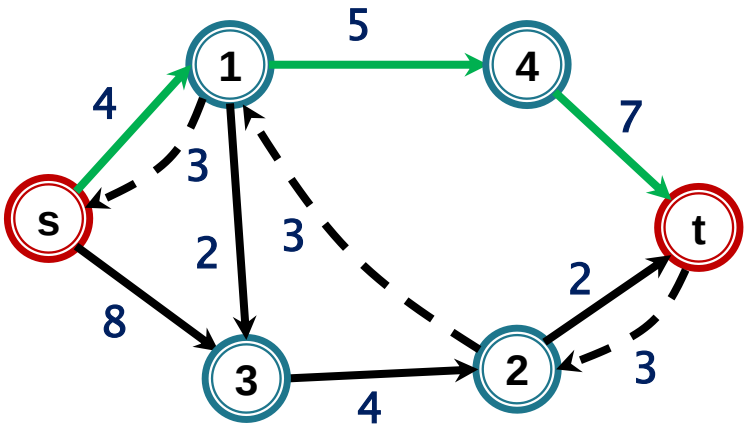


BF(s) – în graful rezidual

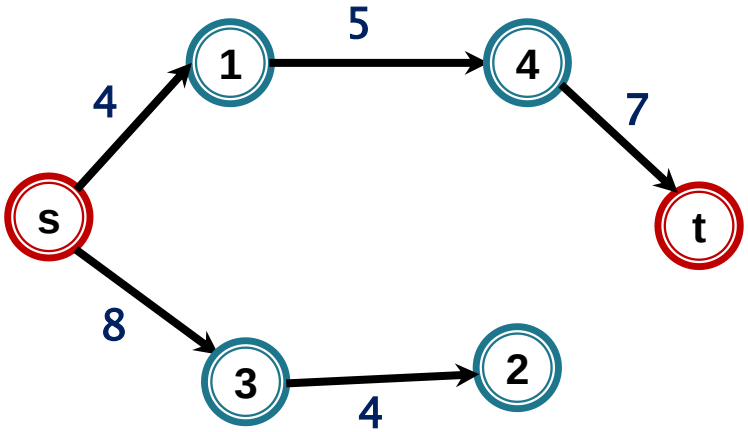


Graful rezidual G_f

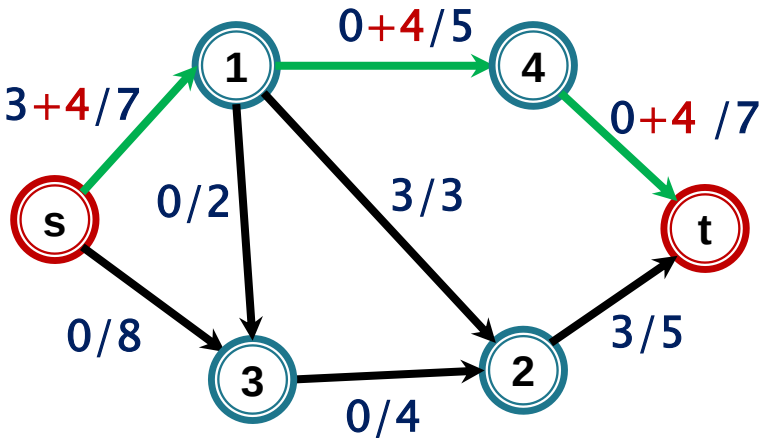




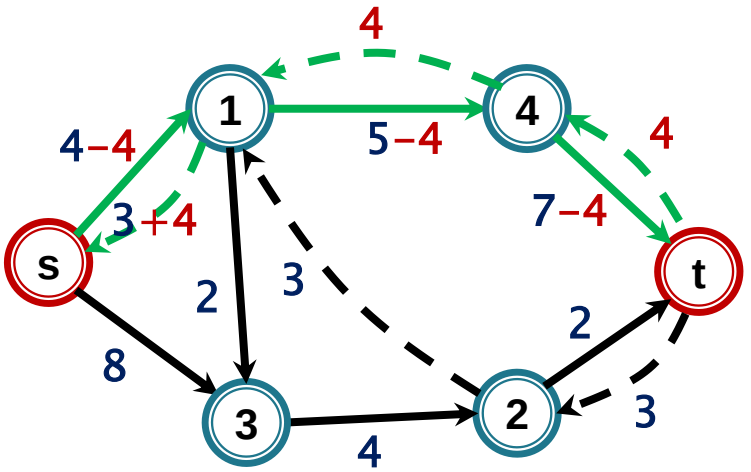
BF(s) – în graful rezidual



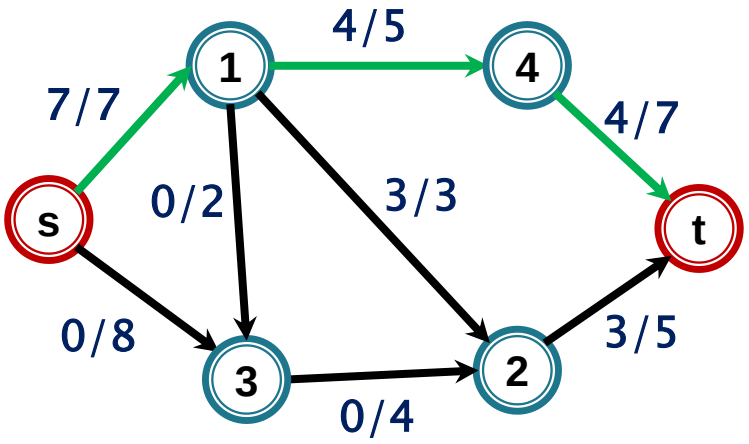
Drumul de creștere [s, 1, 4, t] – capacitate reziduală 4



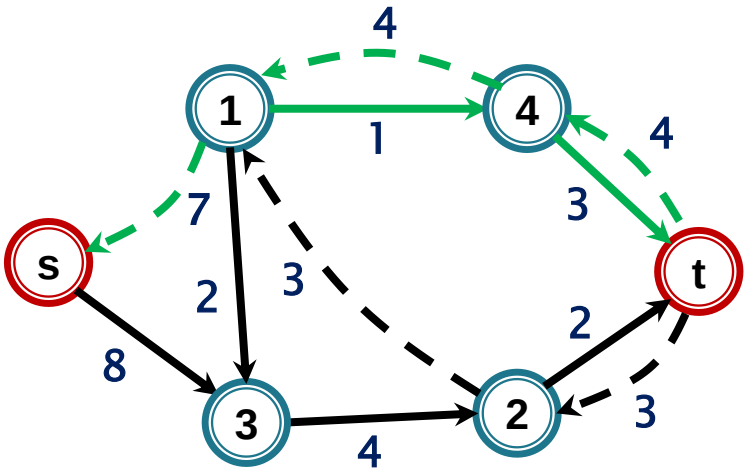
Graful rezidual G_f

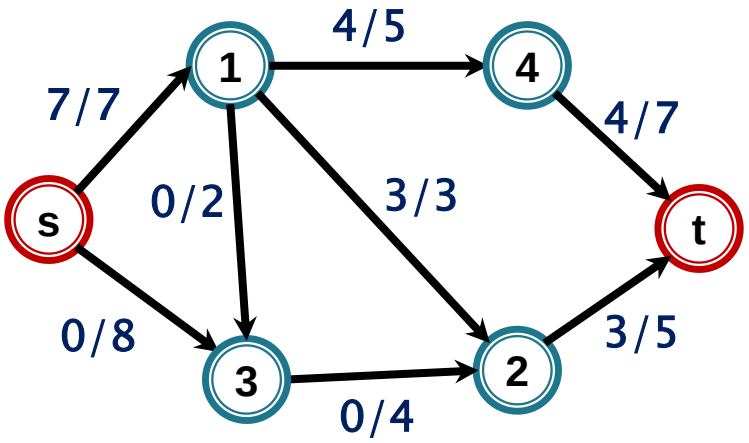


actualizare G_f:

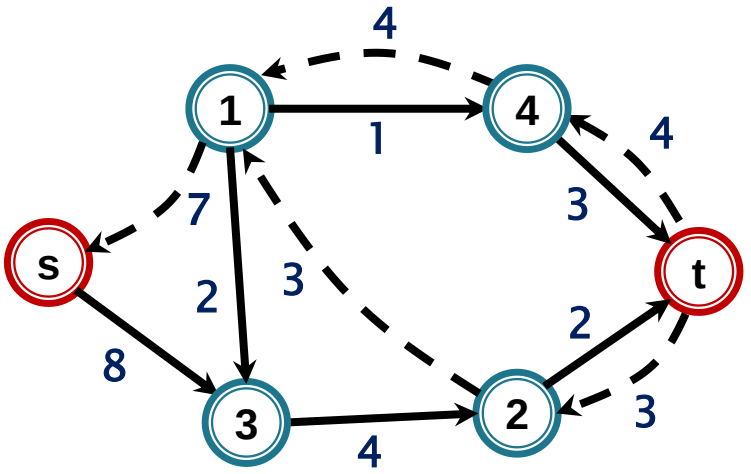


Graful rezidual G_f

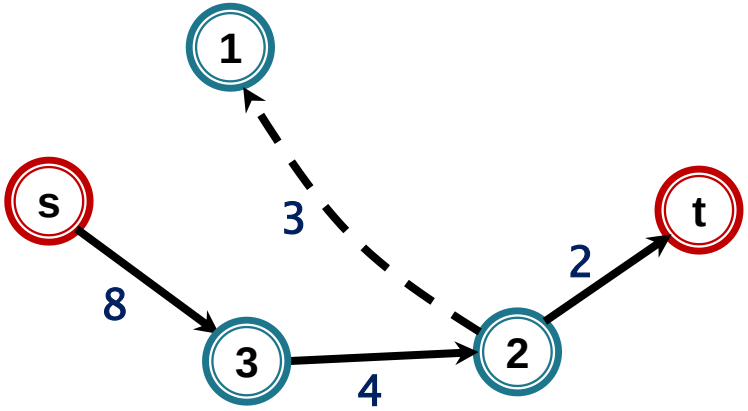


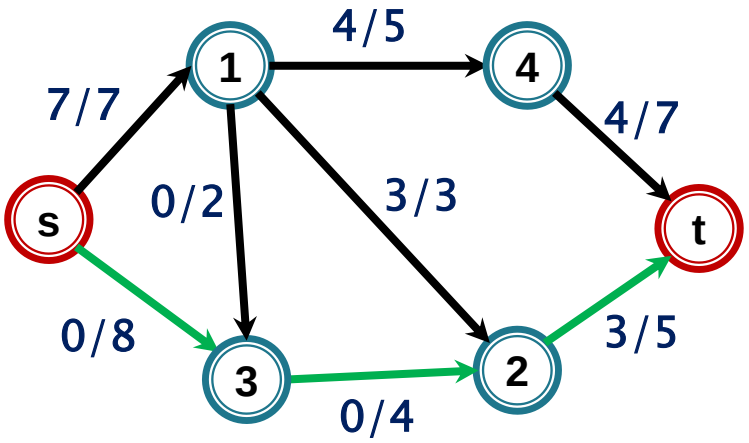


Graful rezidual G_f

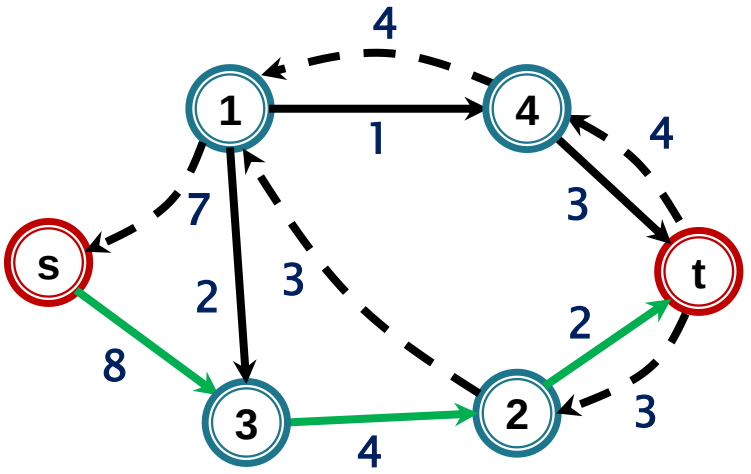


BF(s) – în graful rezidual

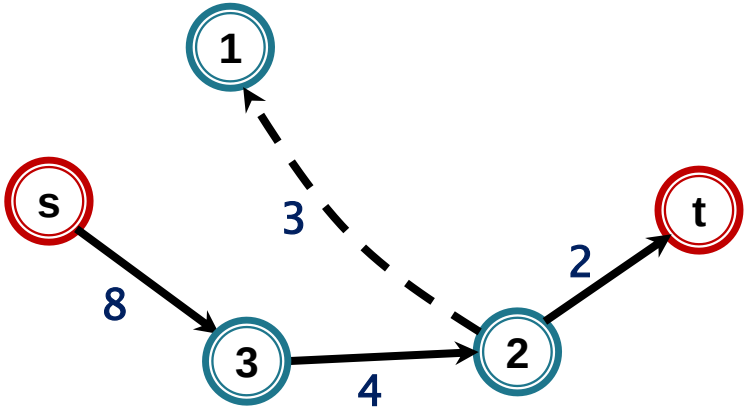




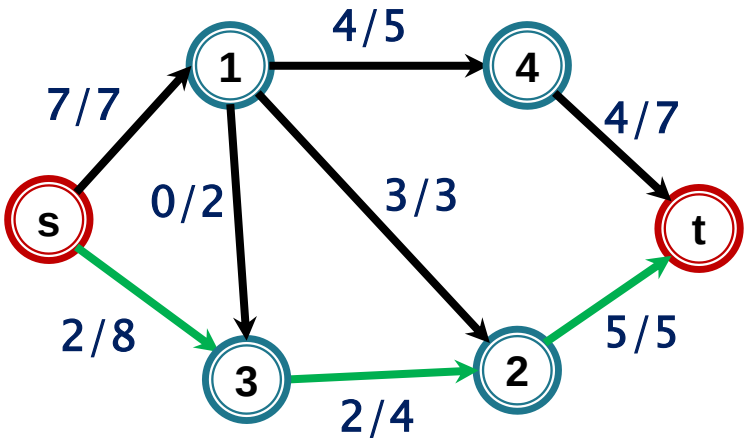
Graful rezidual G_f



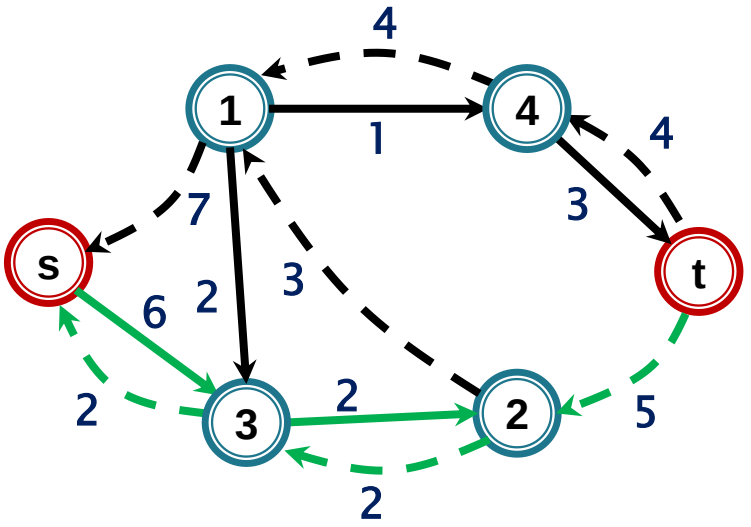
BF(s) – în graful rezidual

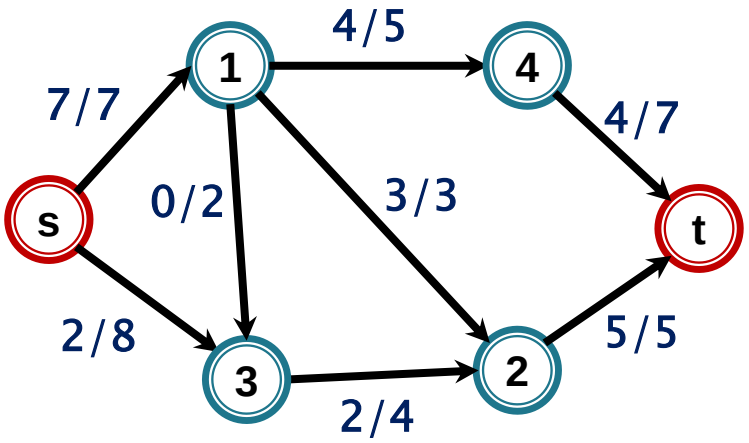


Drumul de creștere [s, 3, 2, t] – capacitate rezidulă 2

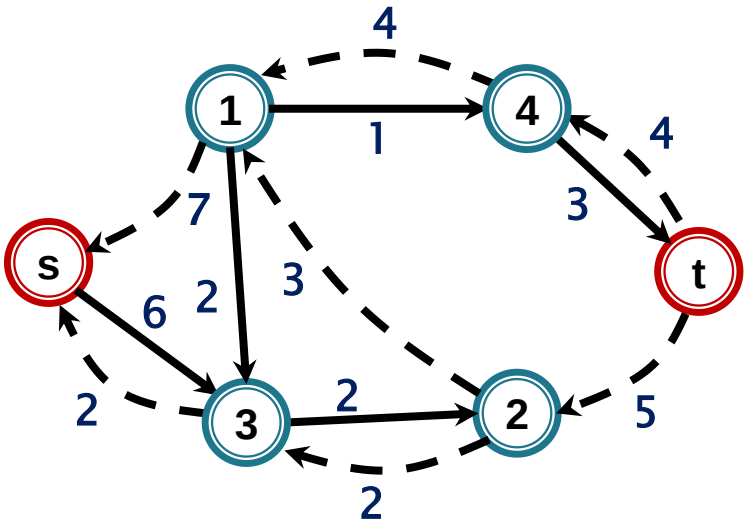


Graful rezidual G_f

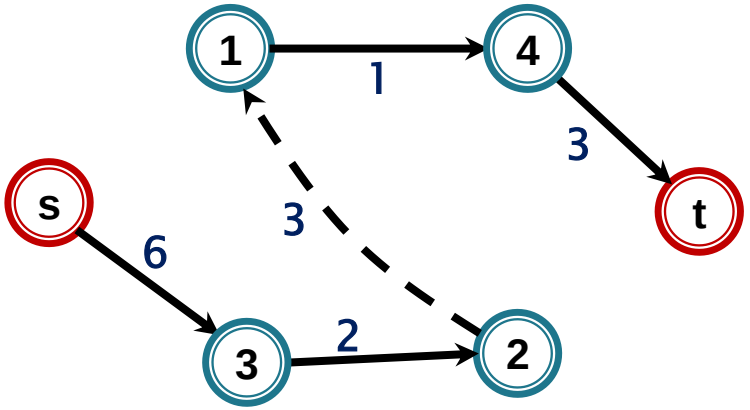


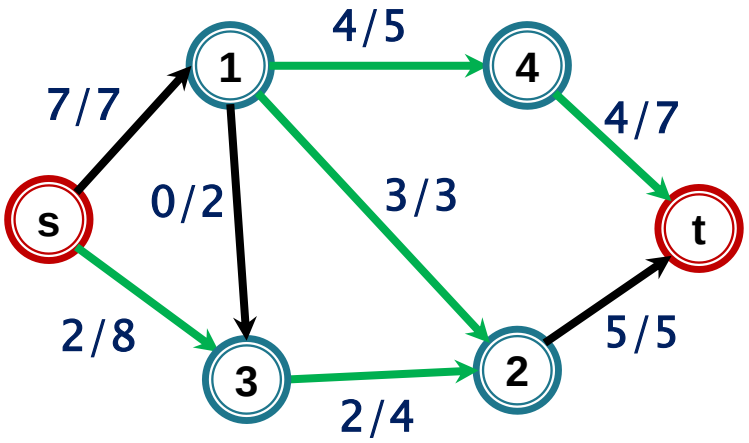


Graful rezidual G_f

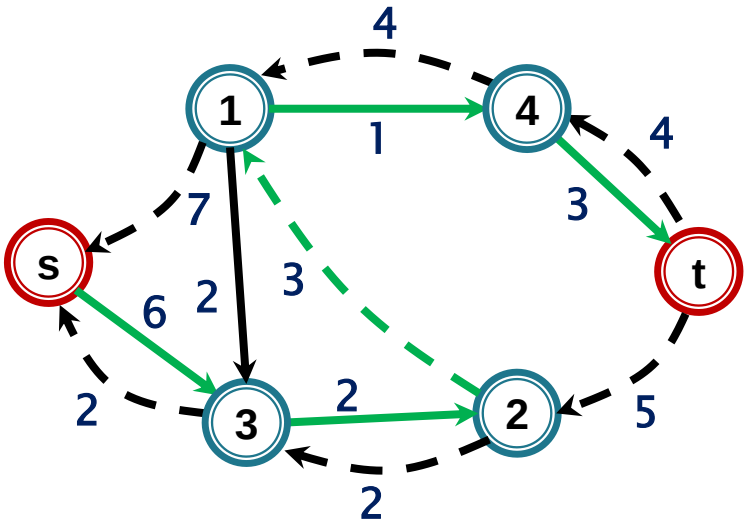


BF(s) – pe graful rezidual

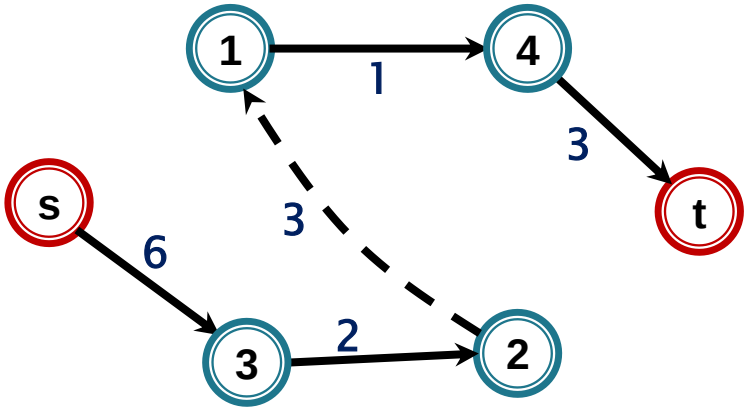




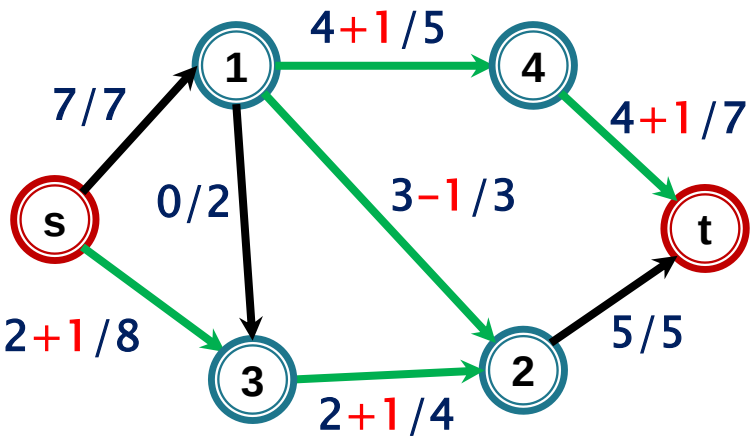
Graful rezidual G_f



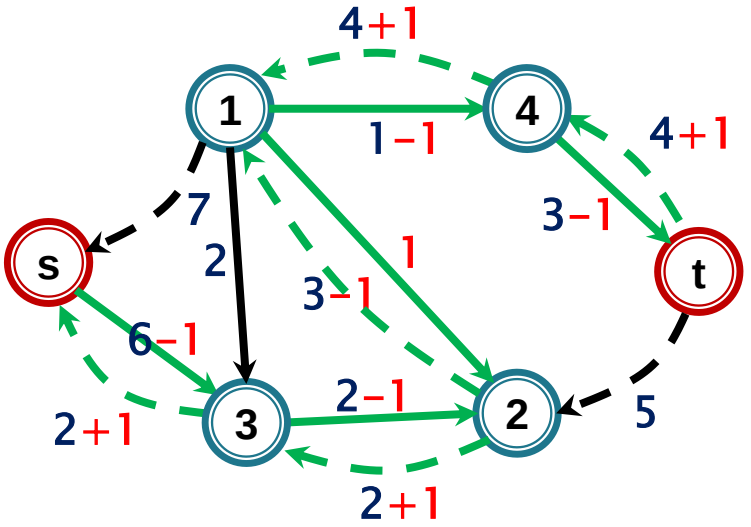
BF(s) – în graful rezidual



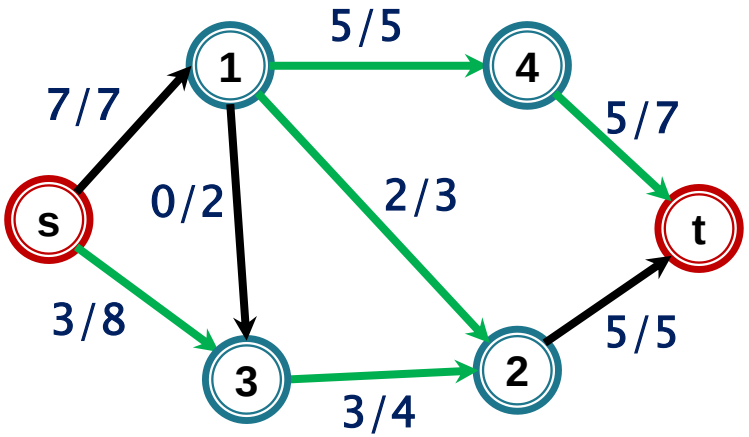
Drumul de creștere [s, 3, 2, 1, 4, t] – capacitate rezidulă 1



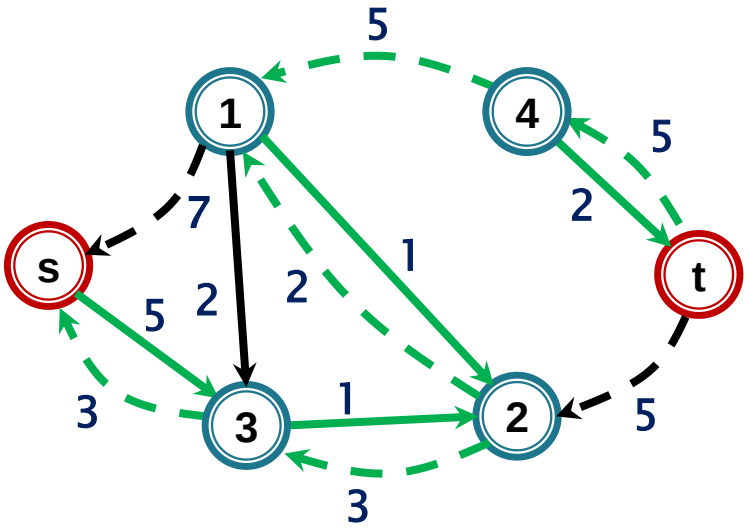
Graful rezidual G_f

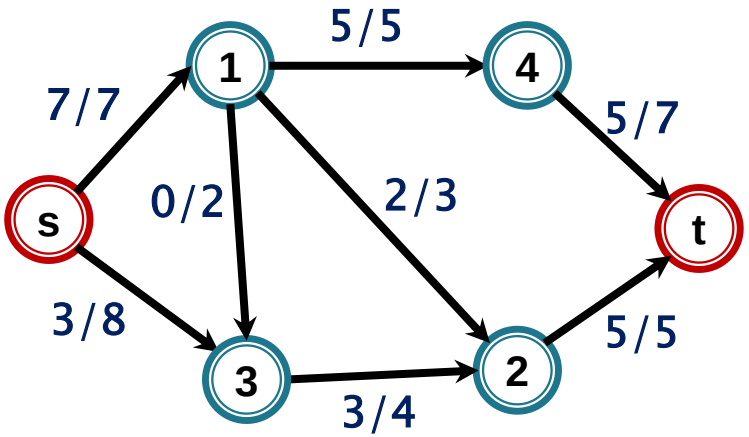


actualizare G_f:

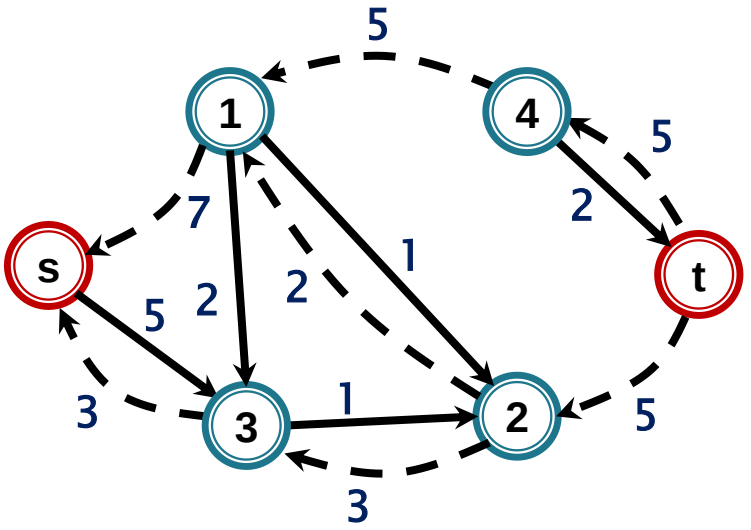


Graful rezidual G_f

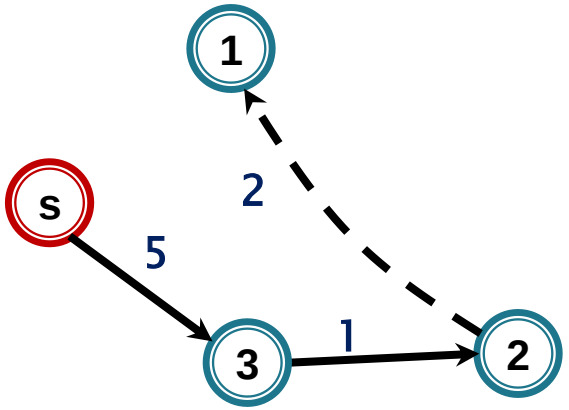


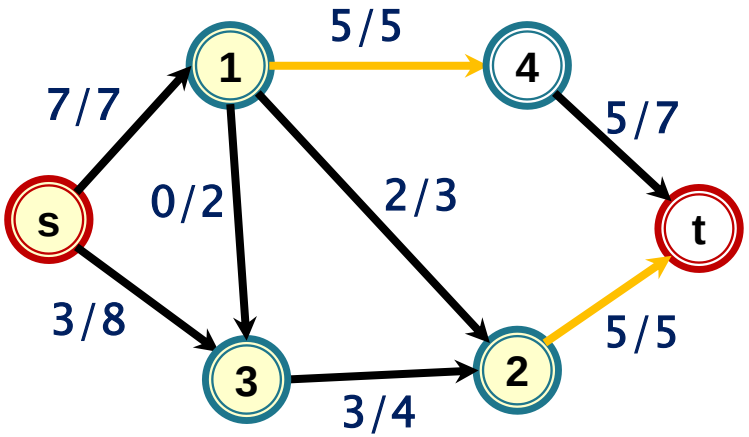


Graful rezidual G_f

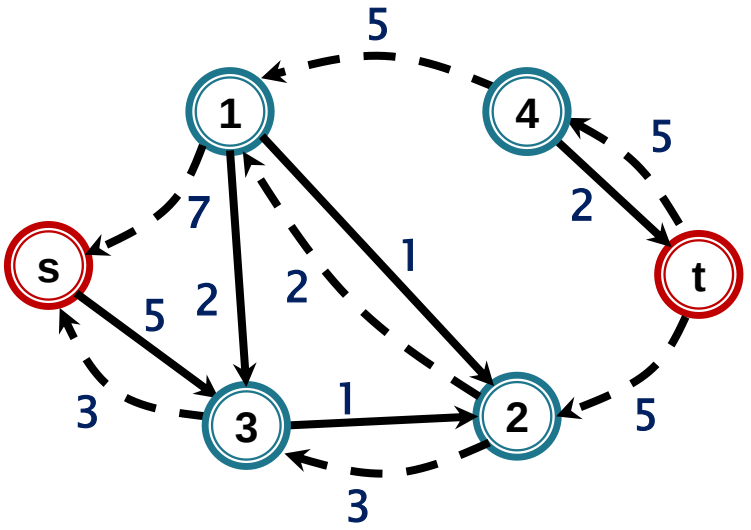


BF(s) – în graful rezidual

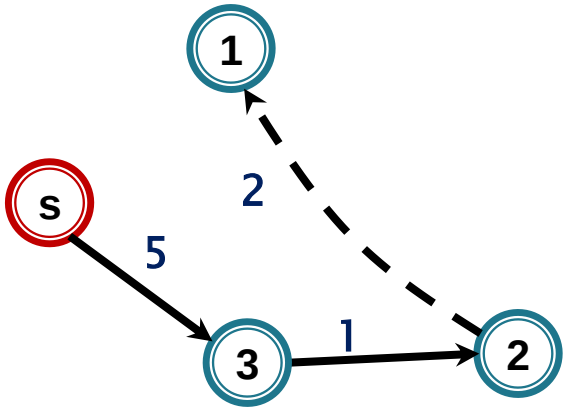




Graful rezidual G_f



BF(s) – în graful rezidual



Nu mai există drumul de creștere ⇒ 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 FORD-FULKERSON

CORECTITUDINE