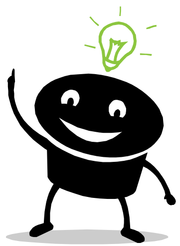
Implementarea algoritmului FORD-FULKERSON

Algoritmul Ford-Fulkerson

- Cum determinăm un lanţ f-nesaturat?

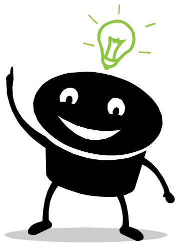
Algoritmul Ford-Fulkerson



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



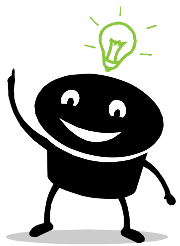
Spre exemplu prin parcurgerea grafului pornind din vârful s şi considerând doar arce cu capacitatea reziduală pozitivă (în raport cu lanţurile construite prin parcurgere, memorate cu vectorul tata)

- **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

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)

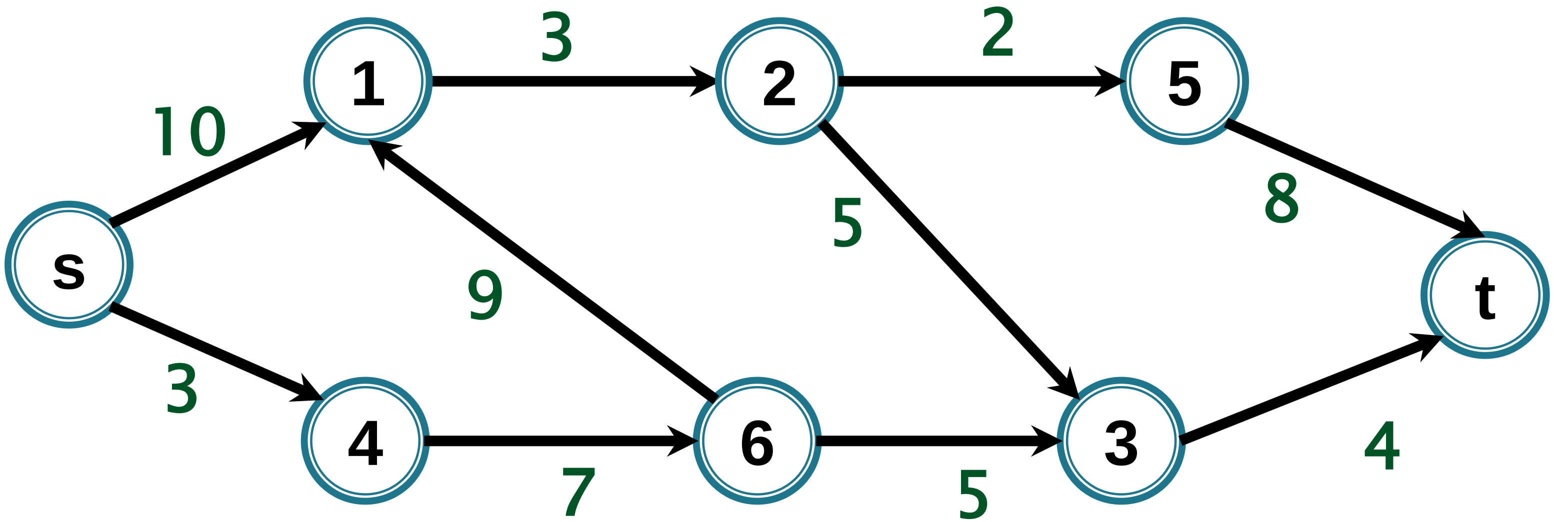
Implementare. Algoritmul Edmonds-Karp

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

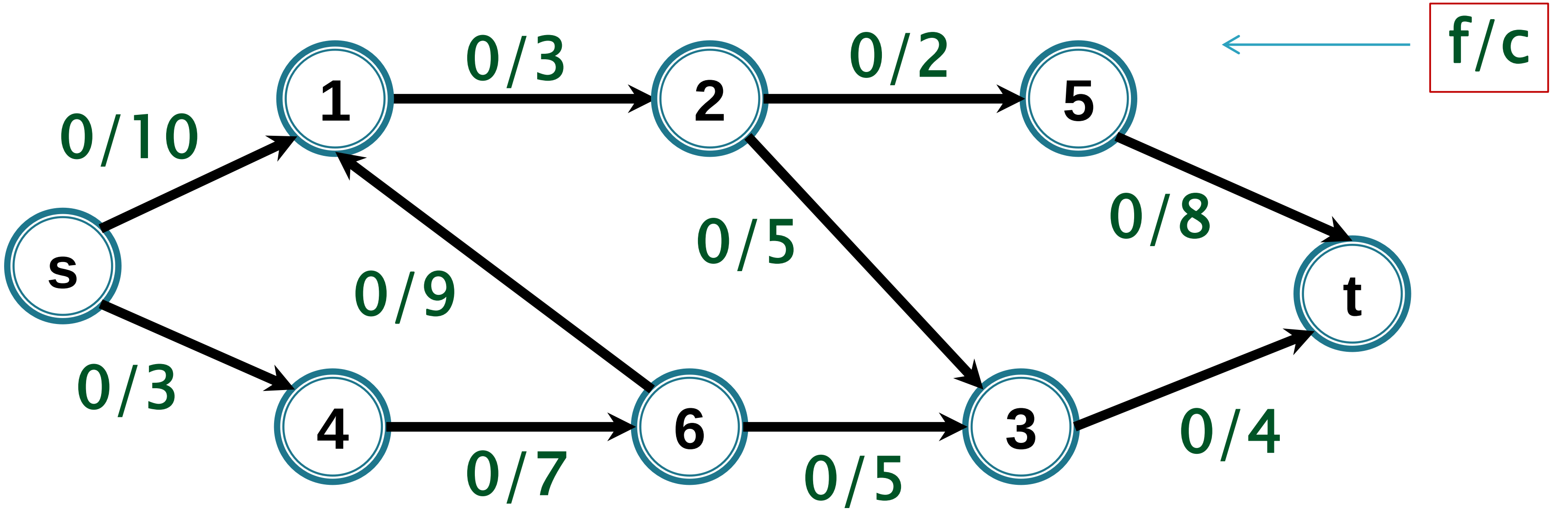
Exemplu

Algoritmul EDMONDS-KARP



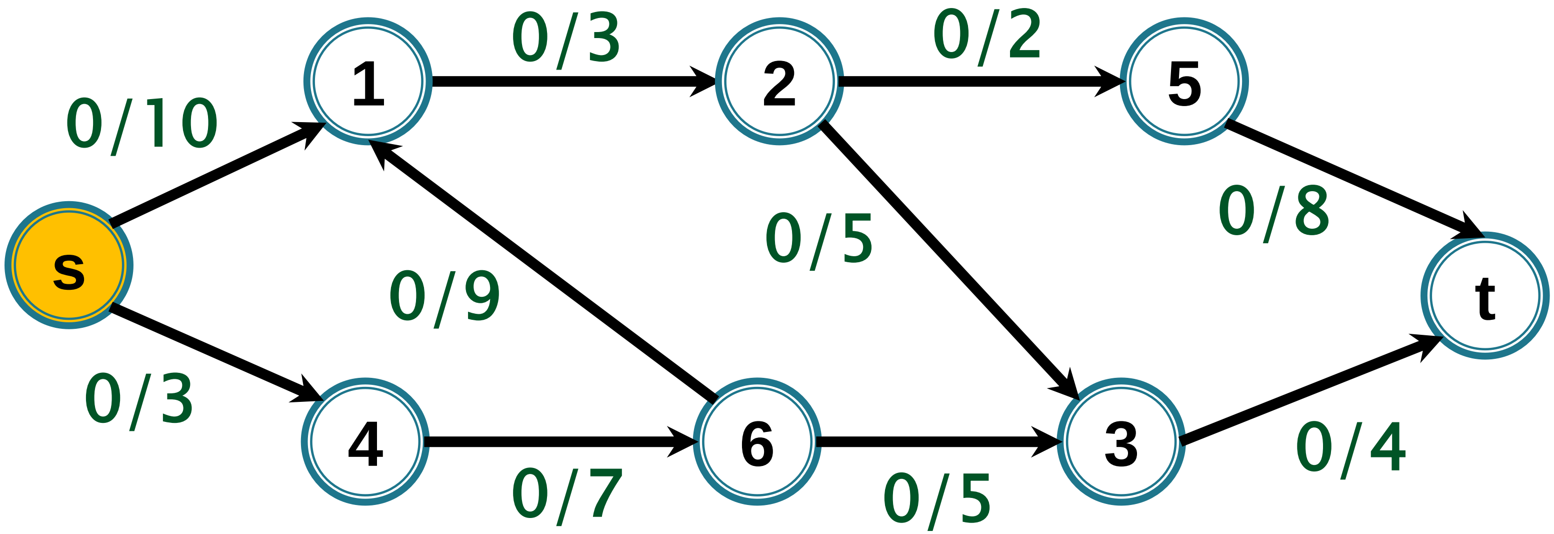
s
1
2
5
t
4
6
3
10
3
2
5
9
8
3
7
5
4

initializeaza_flux_nul

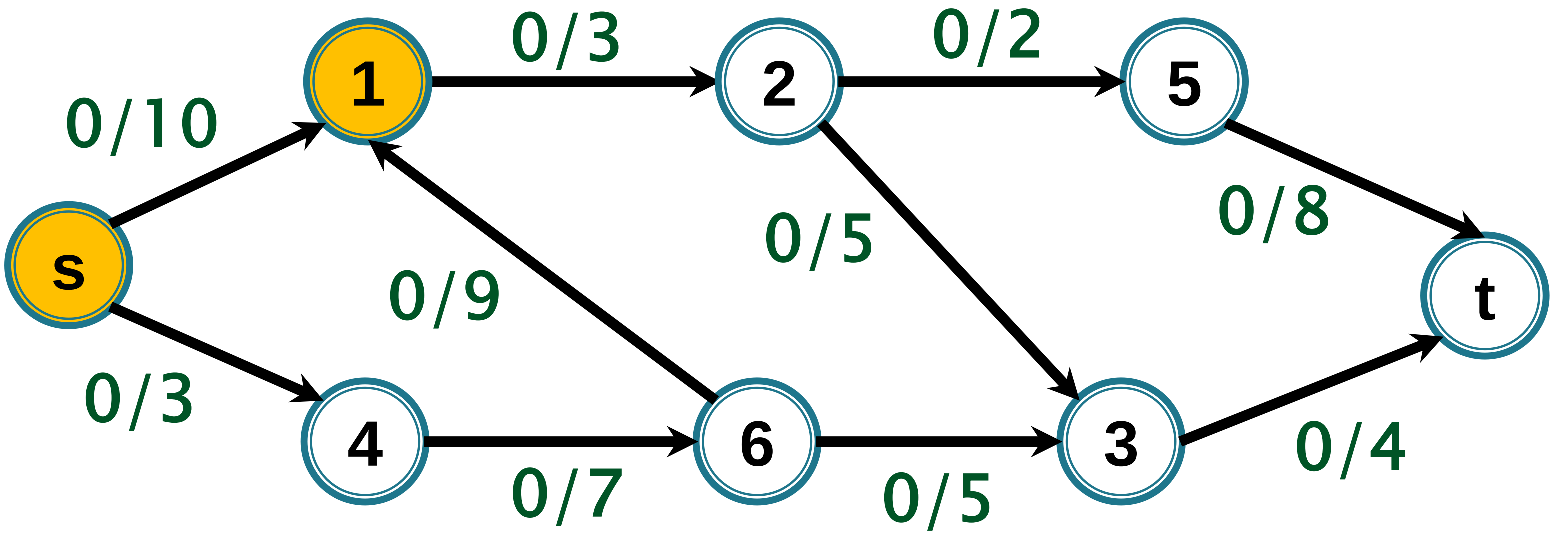


f/c
s
1
2
5
t
4
6
3
0/10
0/3
0/3
0/2
0/8
0/5
0/9
0/4
0/7
0/5

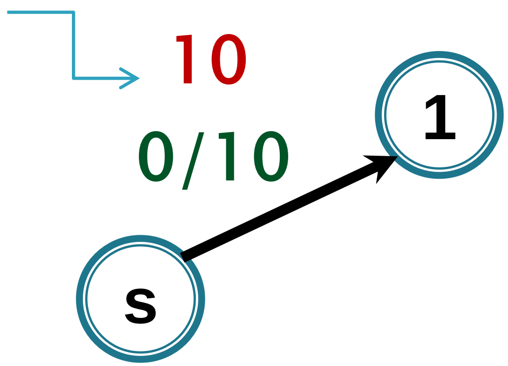
construieste_s-t_lant_nesat_BF

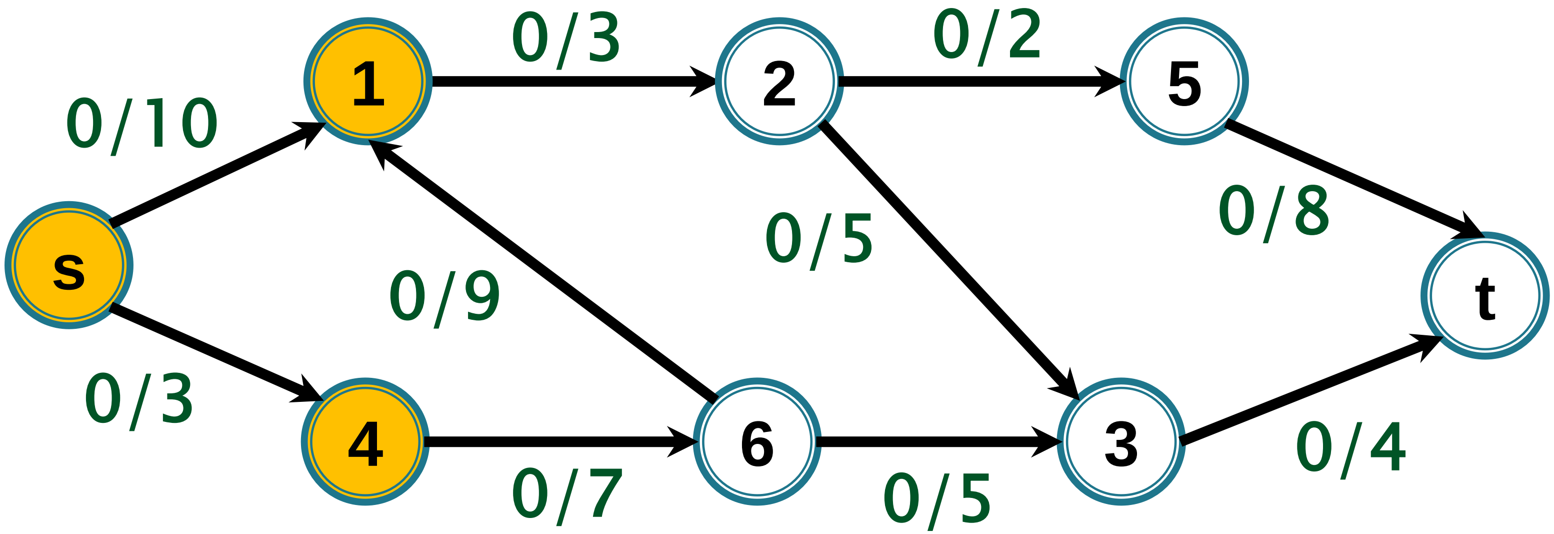

s
1
2
5
t
4
6
3
0/10
0/3
0/2
0/8
0/5
0/9
0/3
0/7
0/5
0/4

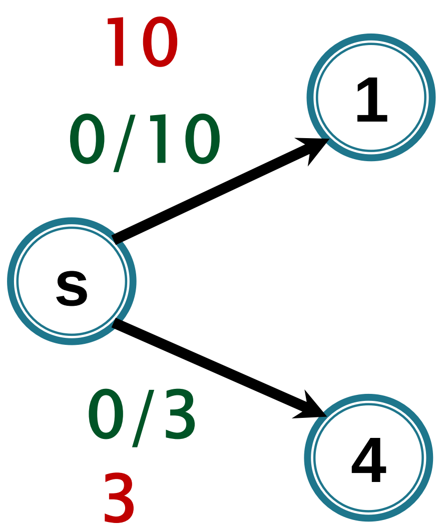

s

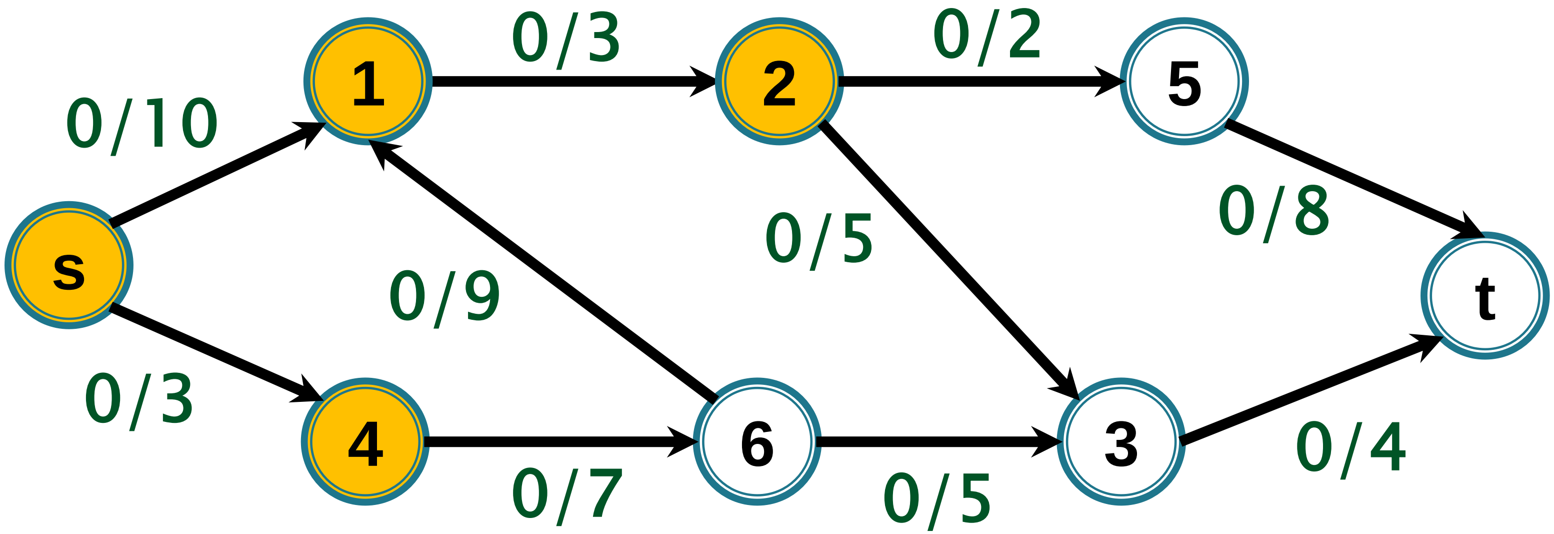


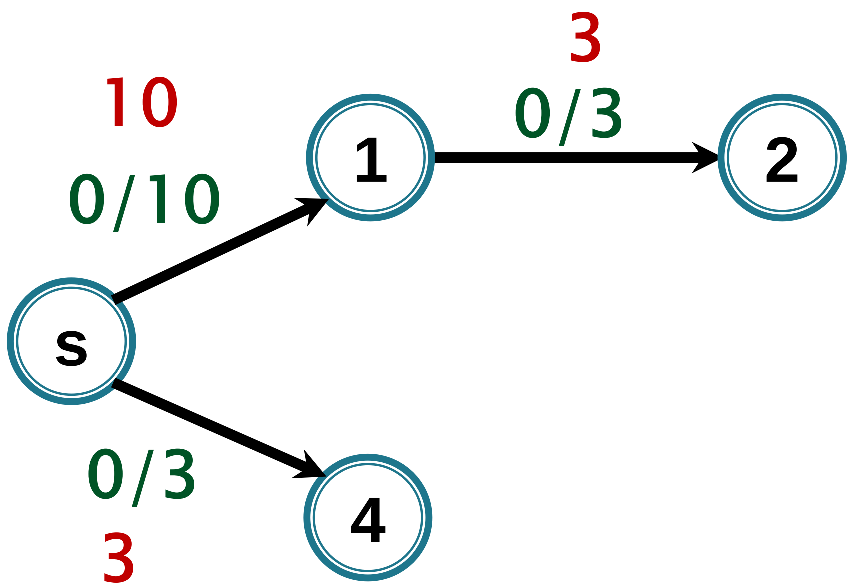
Capacitatea reziduală

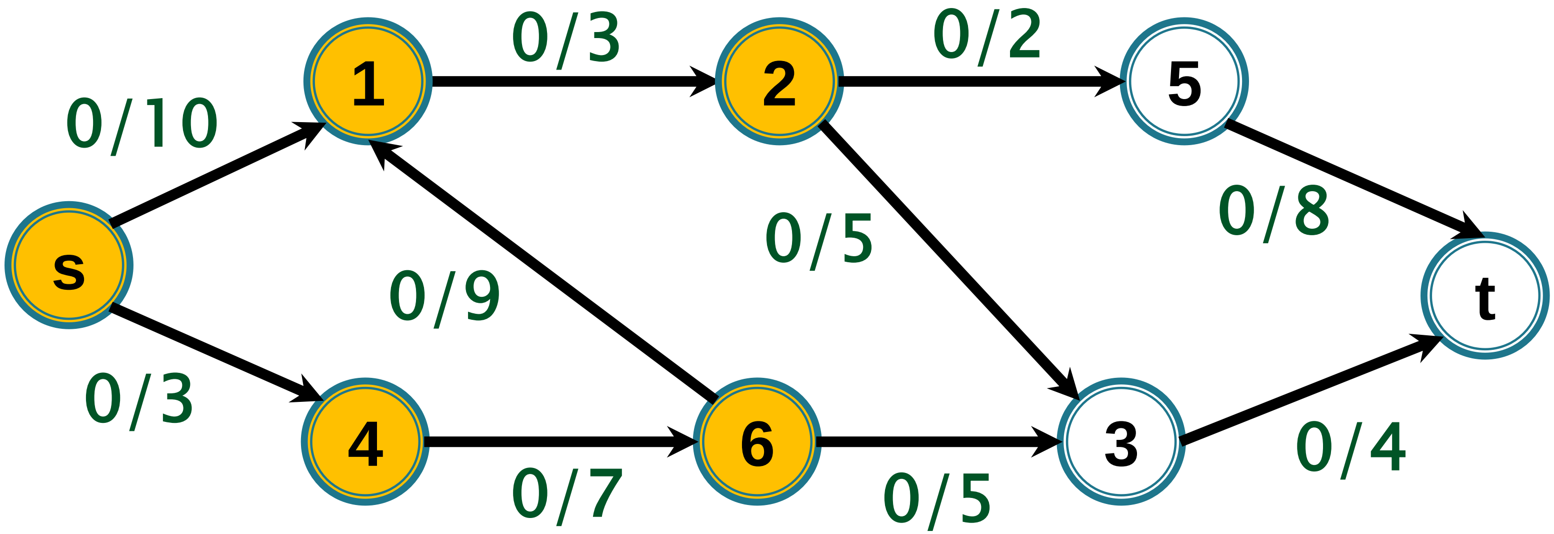


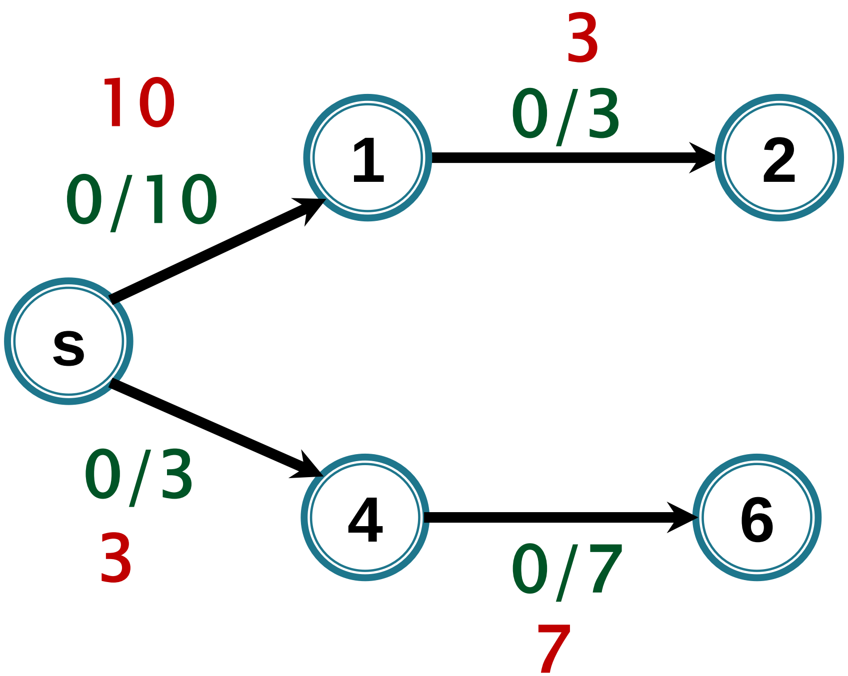

0/10
0/3
0/3
0/2
0/5
0/8
0/9
0/7
0/5
0/4
s
1
2
5
4
6
3
t

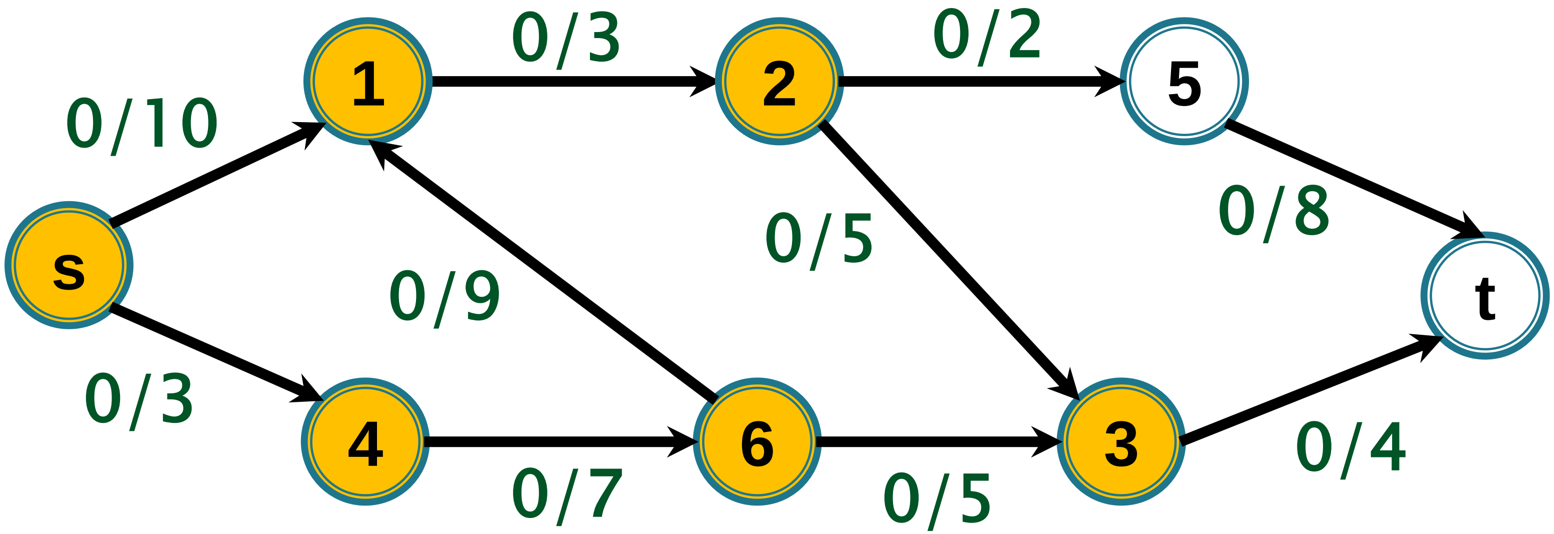

10
0/10
0/3
3
s
1
4


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

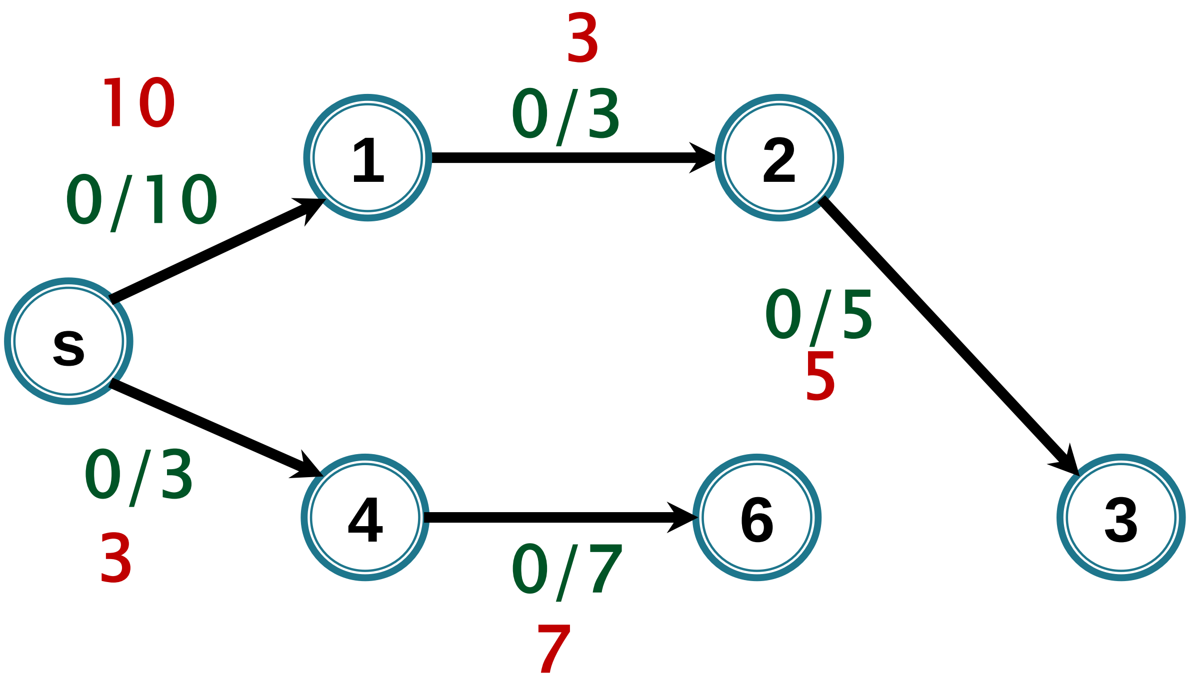

3
10
0/3
0/10
0/3
3
s
1
2
4


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

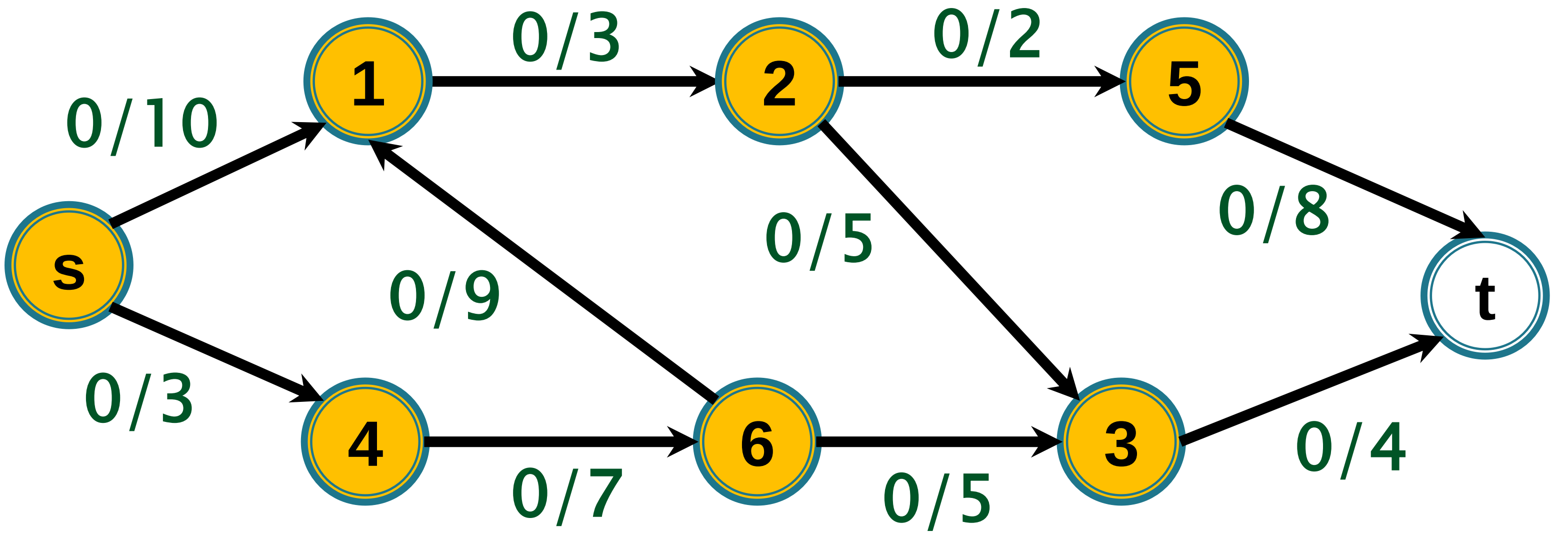

3
10
0/3
1
2
0/10
s
0/3
4
6
3
0/7
7



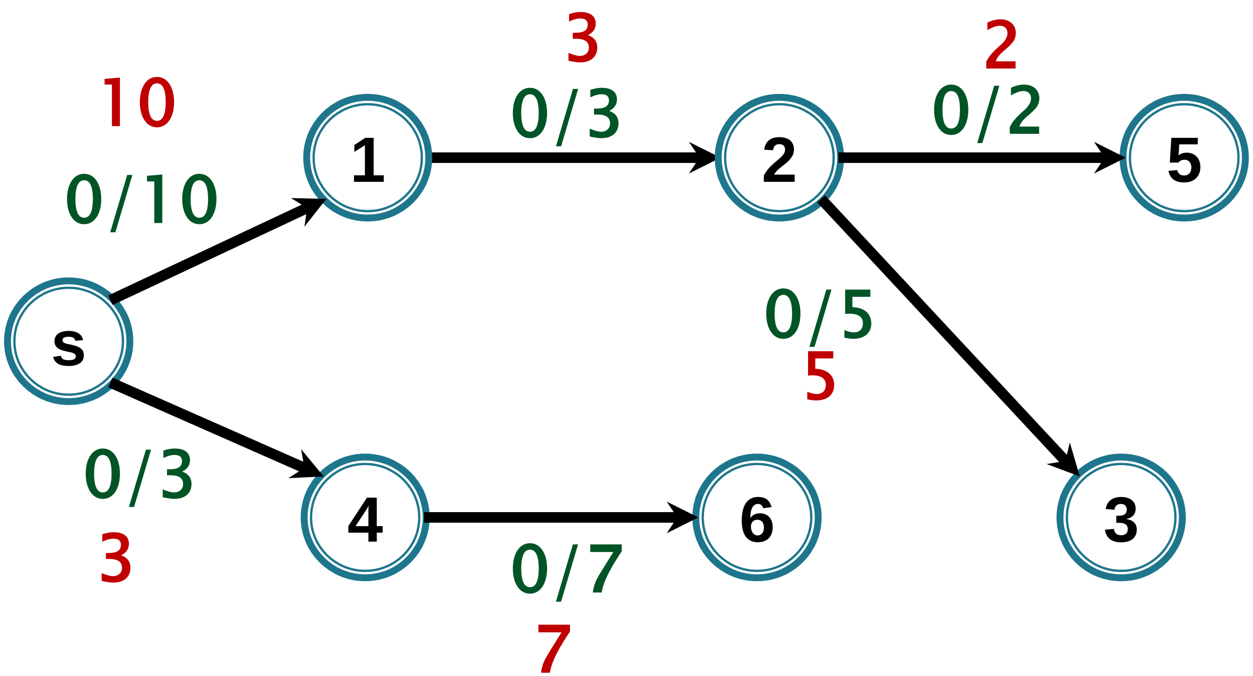
0/10
0/3
0/3
0/2
0/9
0/5
0/8
0/7
0/5
0/4
s
1
2
5
t
4
6
3



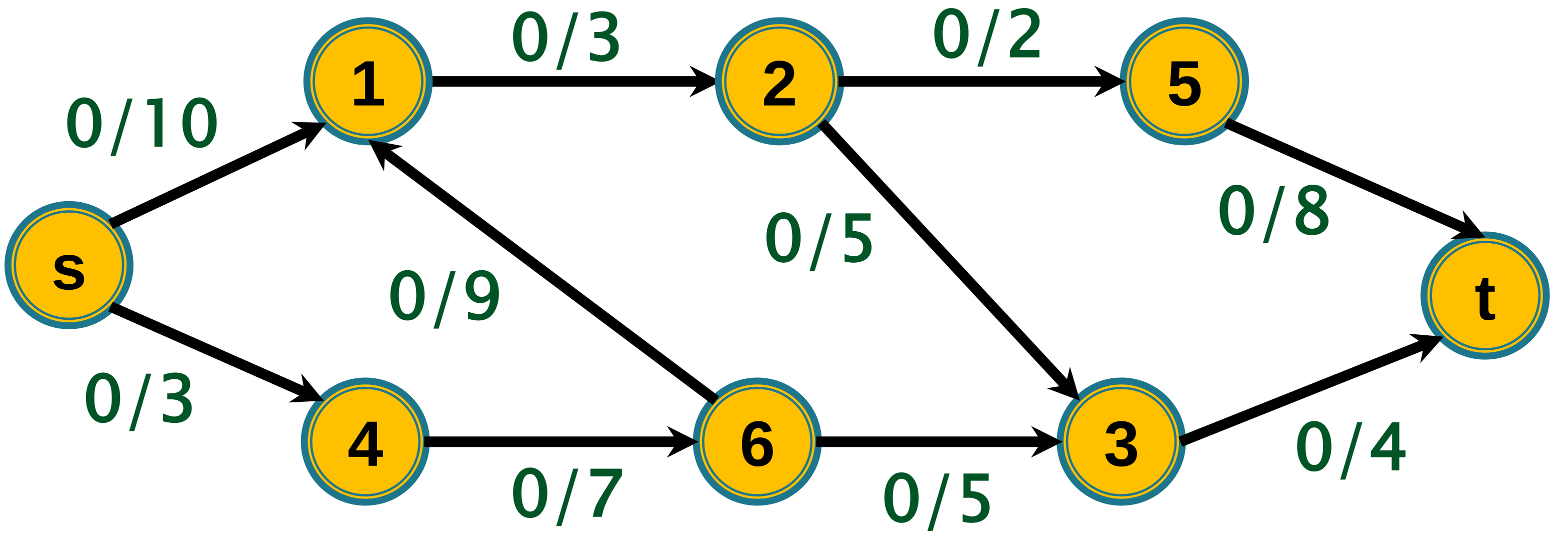
10
0/10
3
0/3
0/5
5
0/3
3
0/7
7
s
1
2
4
6
3

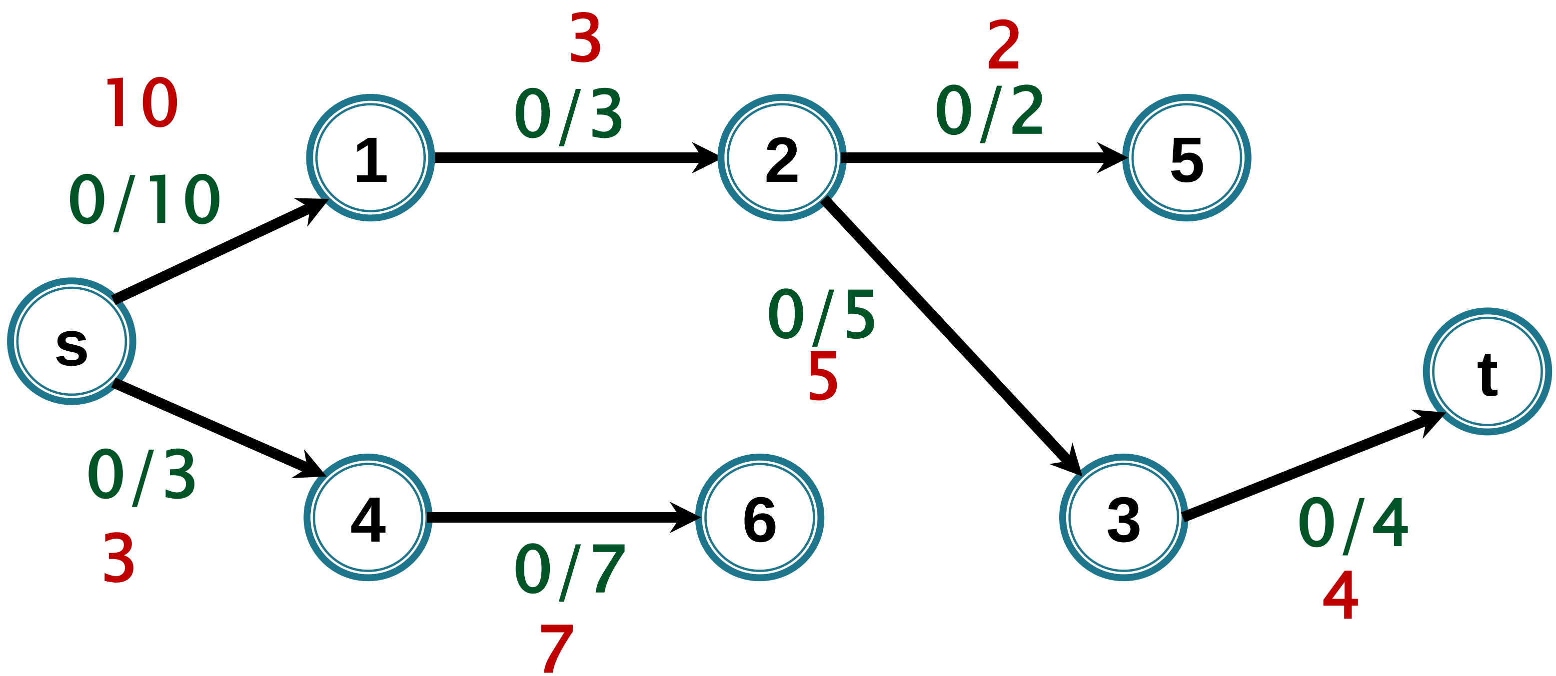


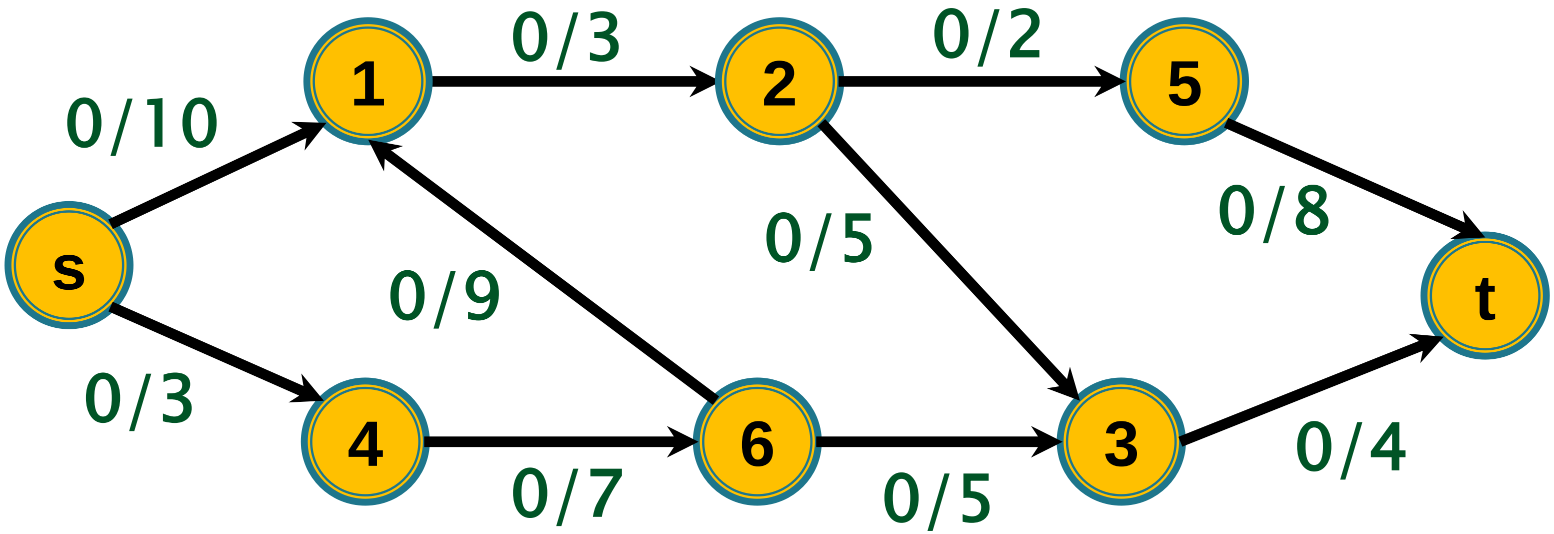
0/10
0/3
0/2
0/5
0/8
0/9
0/3
0/7
0/5
0/4
s
1
2
5
t
4
6
3



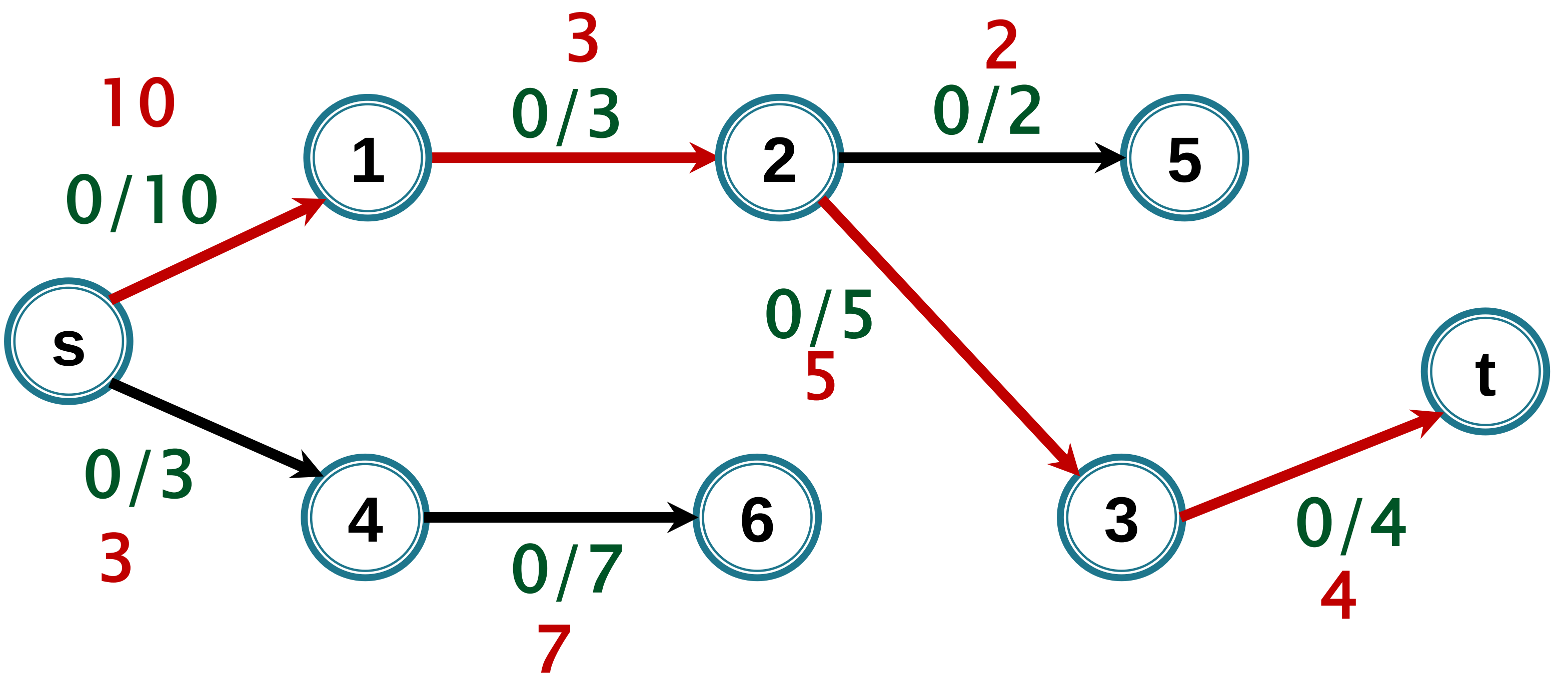
10
0/10
3
0/3
2
0/2
0/5
5
0/3
3
0/7
7
s
1
2
5
4
6
3


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

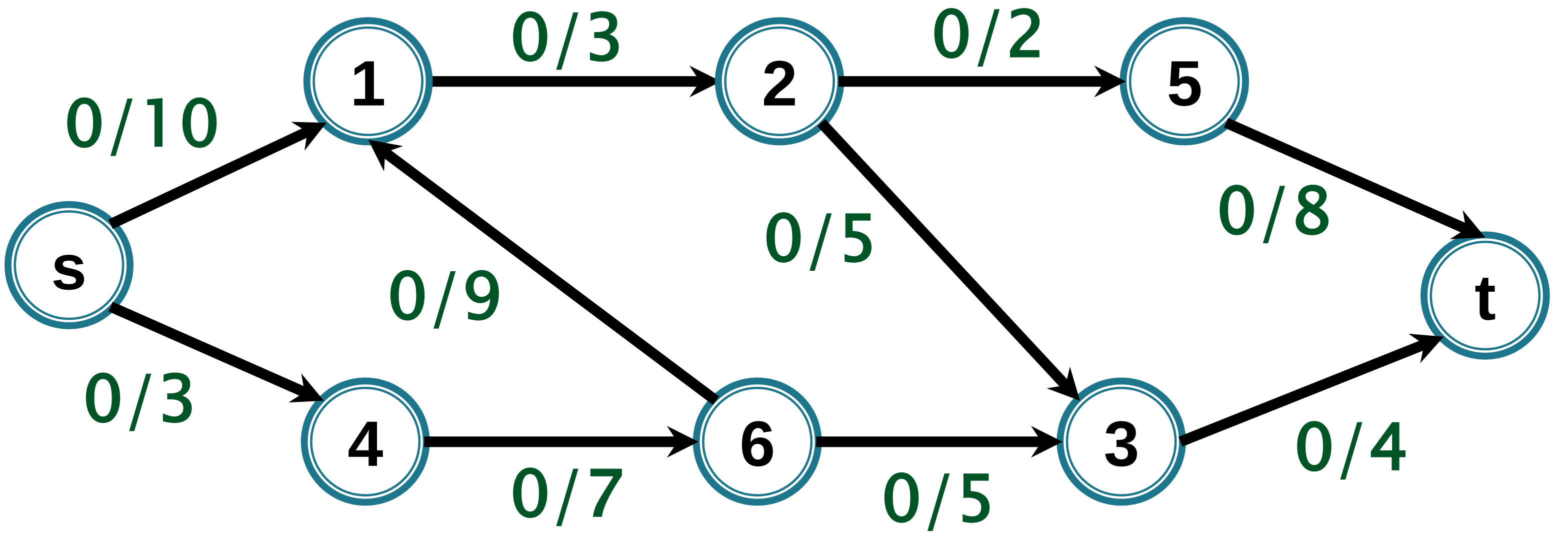


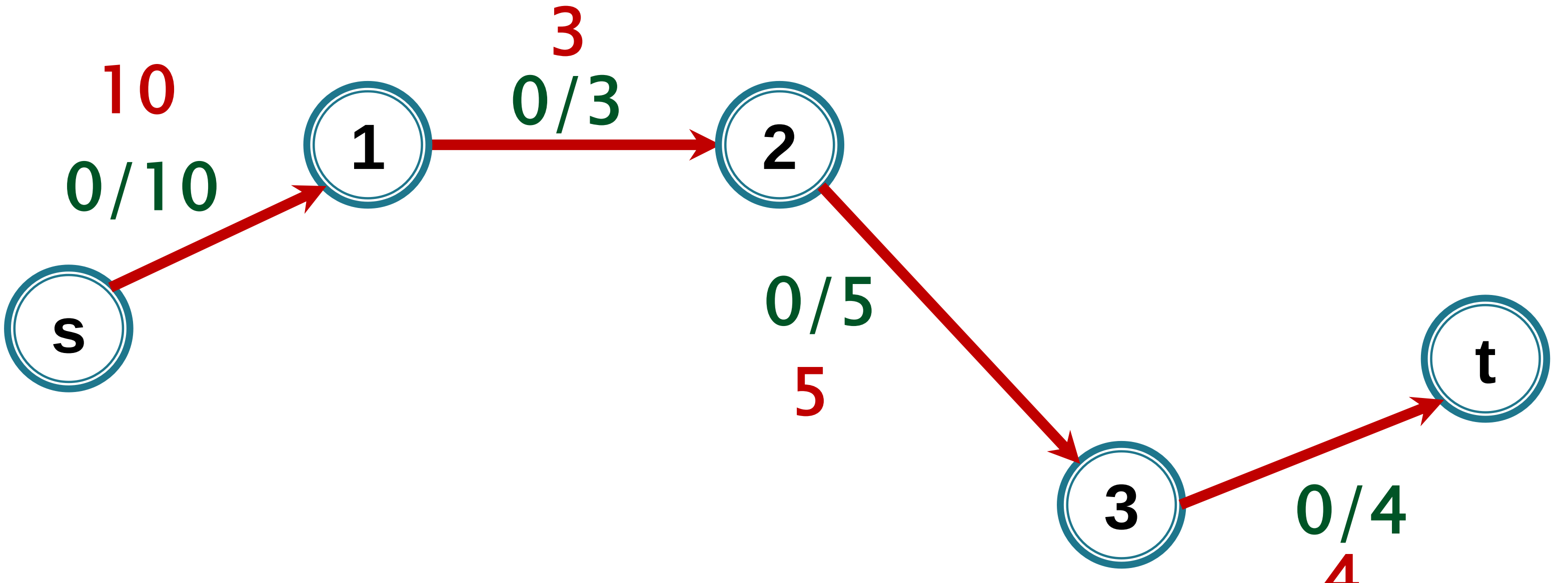
s
1
2
5
t
4
6
3
0/10
0/3
0/2
0/8
0/5
0/9
0/3
0/7
0/5
0/4



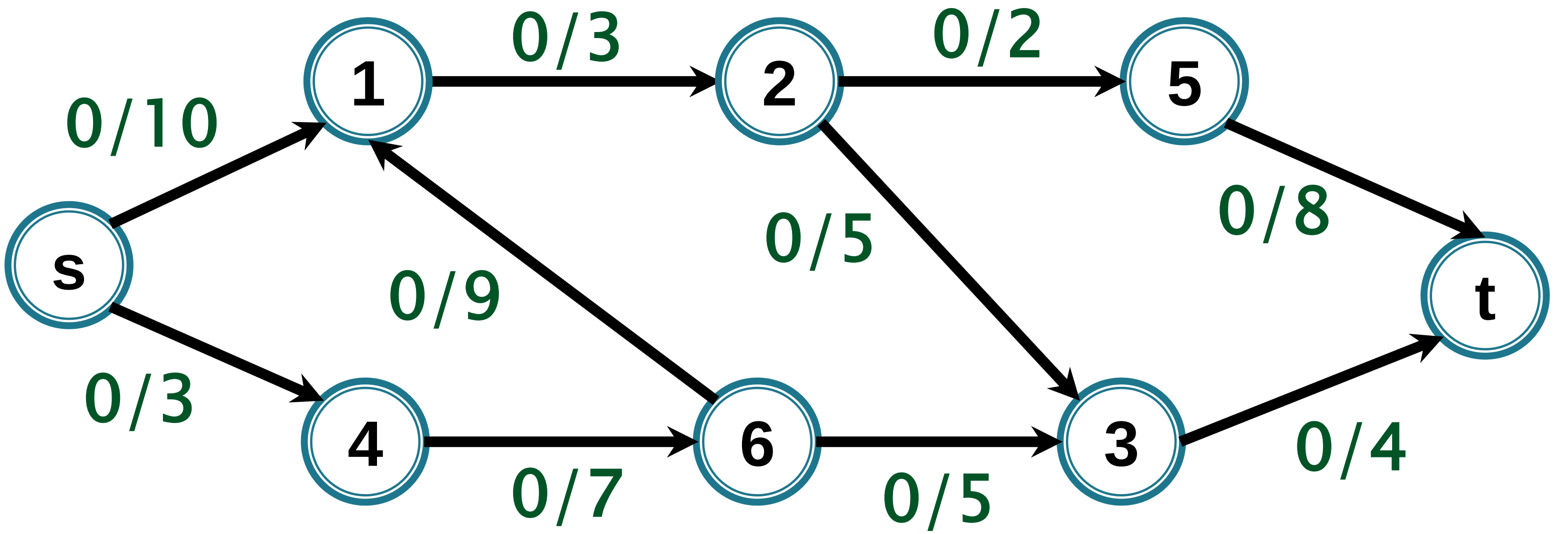
s
1
2
5
t
4
6
3
10
0/10
3
0/3
2
0/2
0/5
5
0/3
3
0/7
7
0/4
4

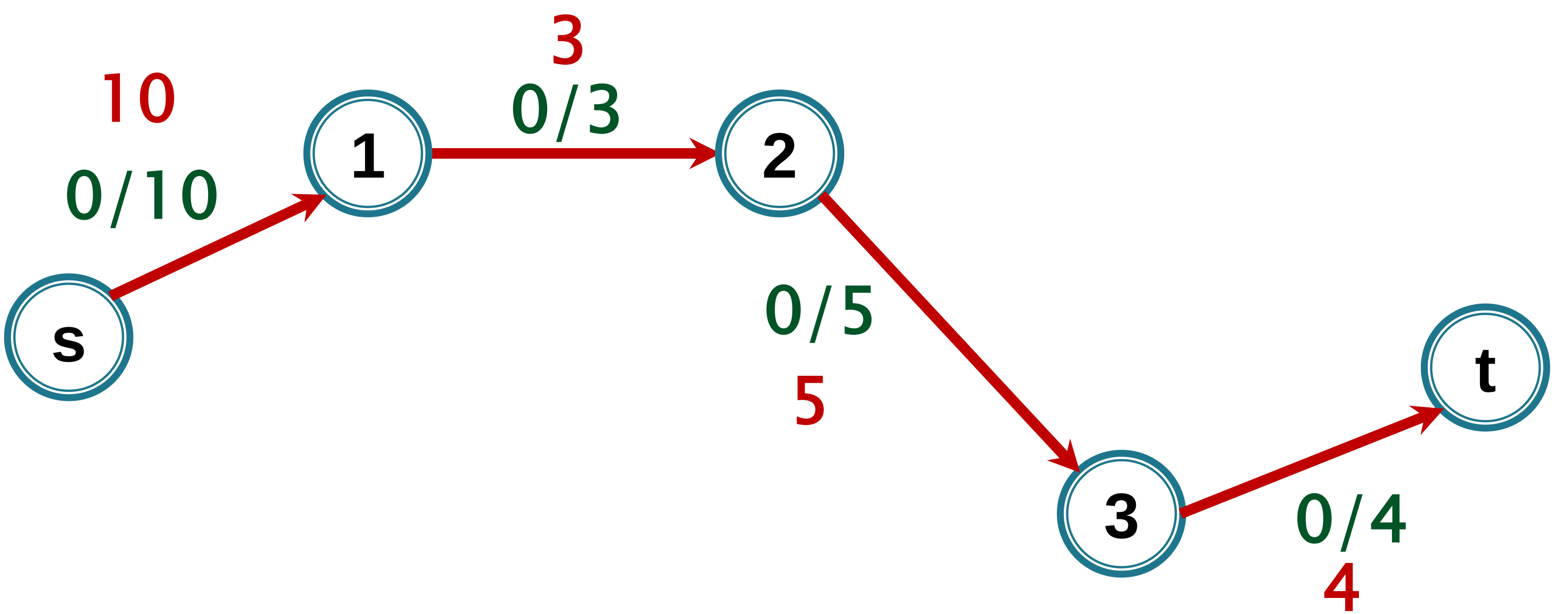
revizuieste_flux_lant



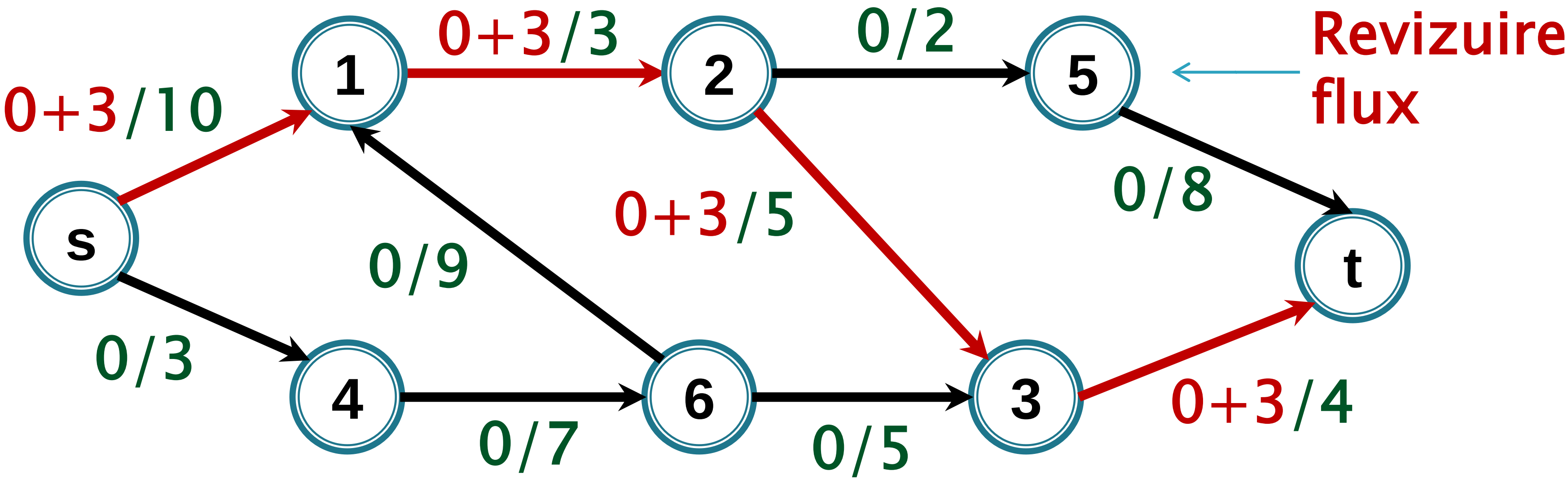


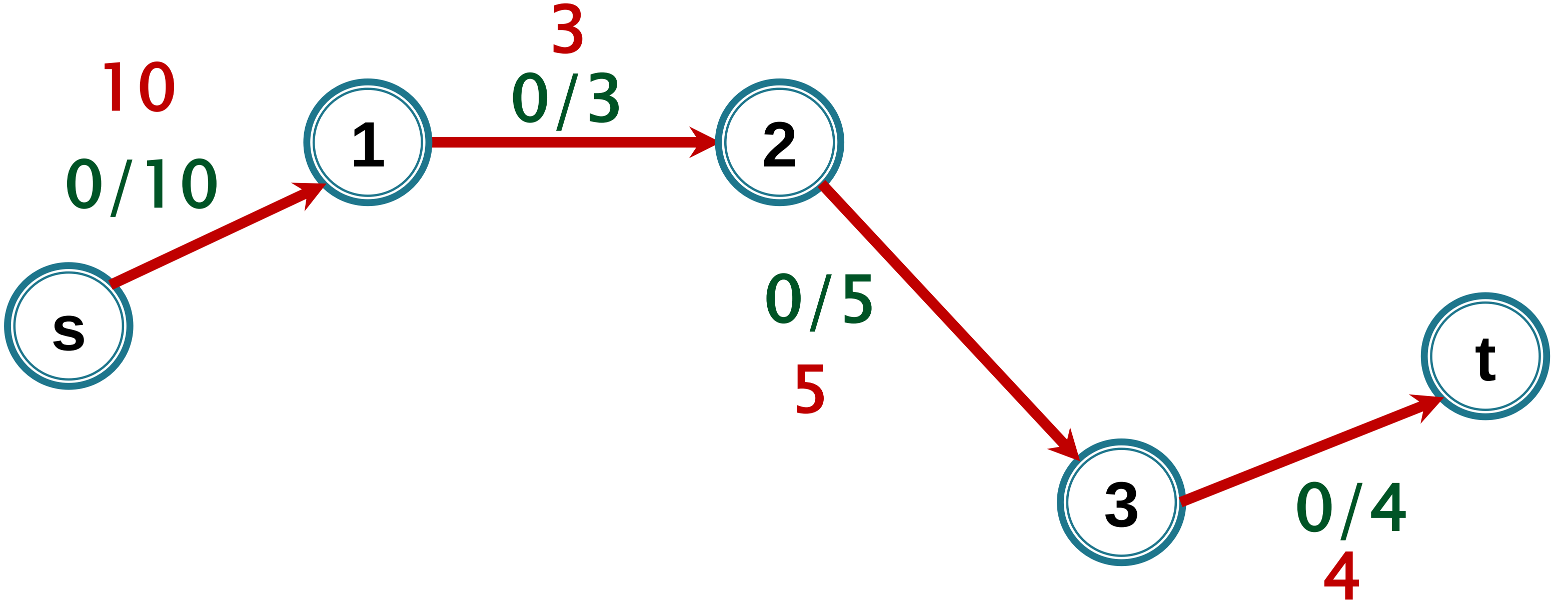
$i(P) = ?$



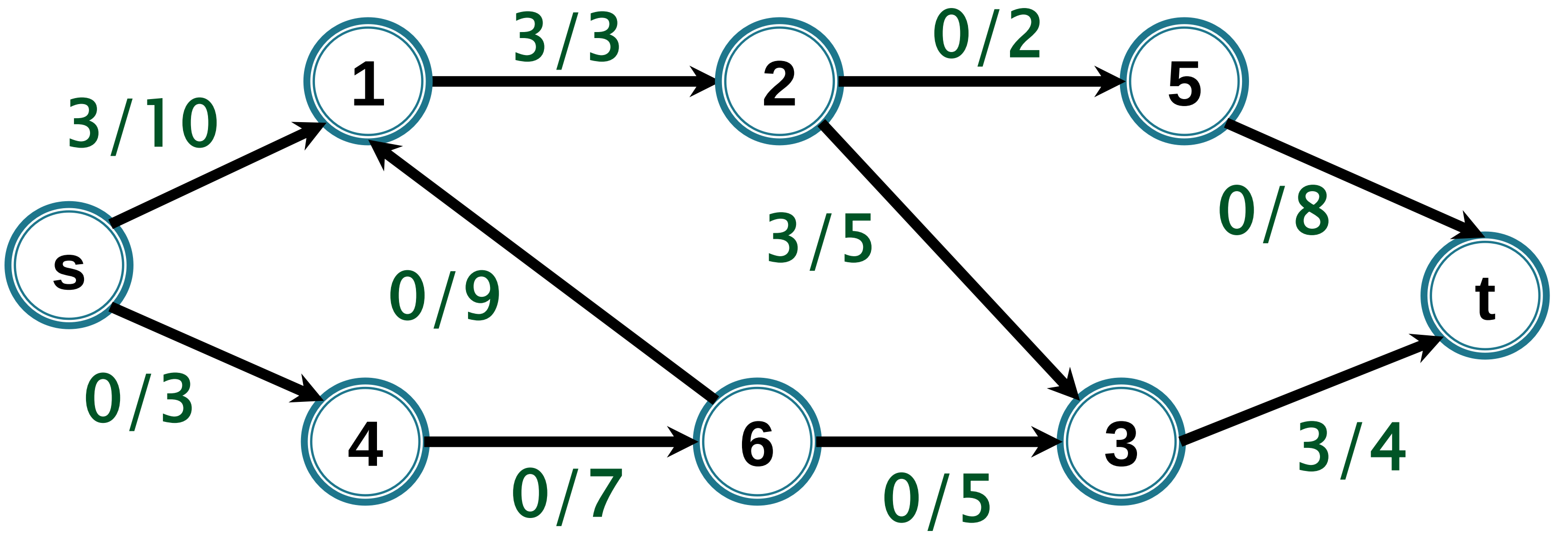


$$i(P) = \min\{10, 3, 5, 4\} = 3$$



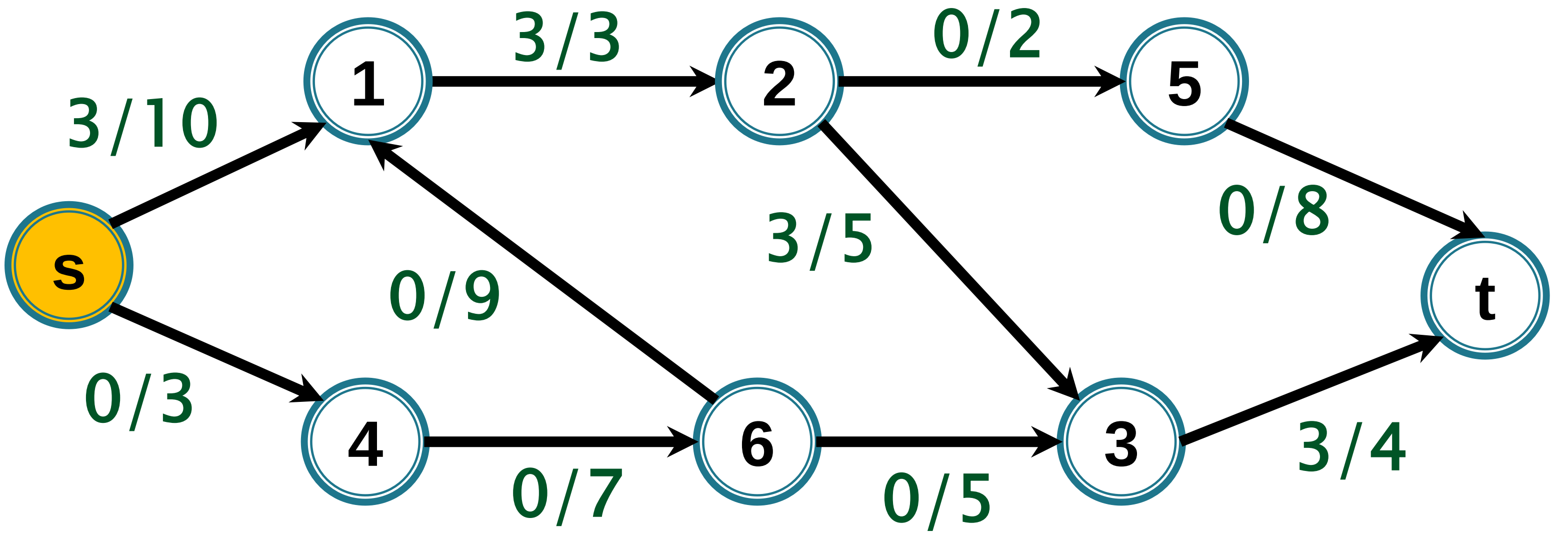


$$i(P) = \min\{10, 3, 5, 4\} = 3$$

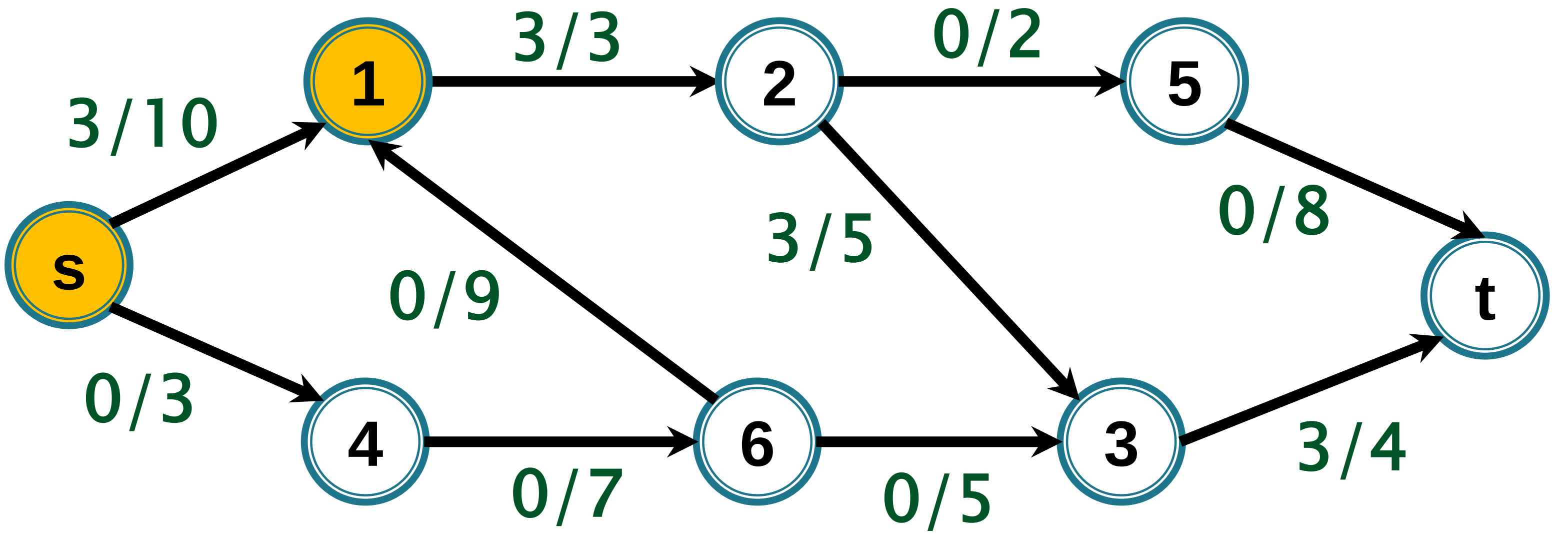


s
1
2
5
t
4
6
3
3/10
3/3
0/2
0/8
3/5
0/9
0/3
0/7
0/5
3/4

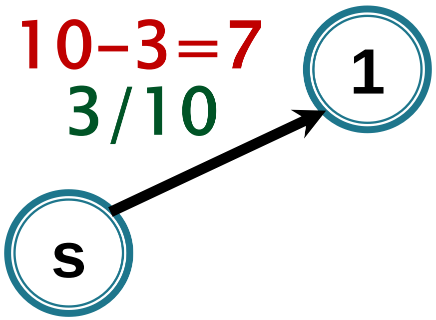
construieste_s-t_lant_nesat_BF


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

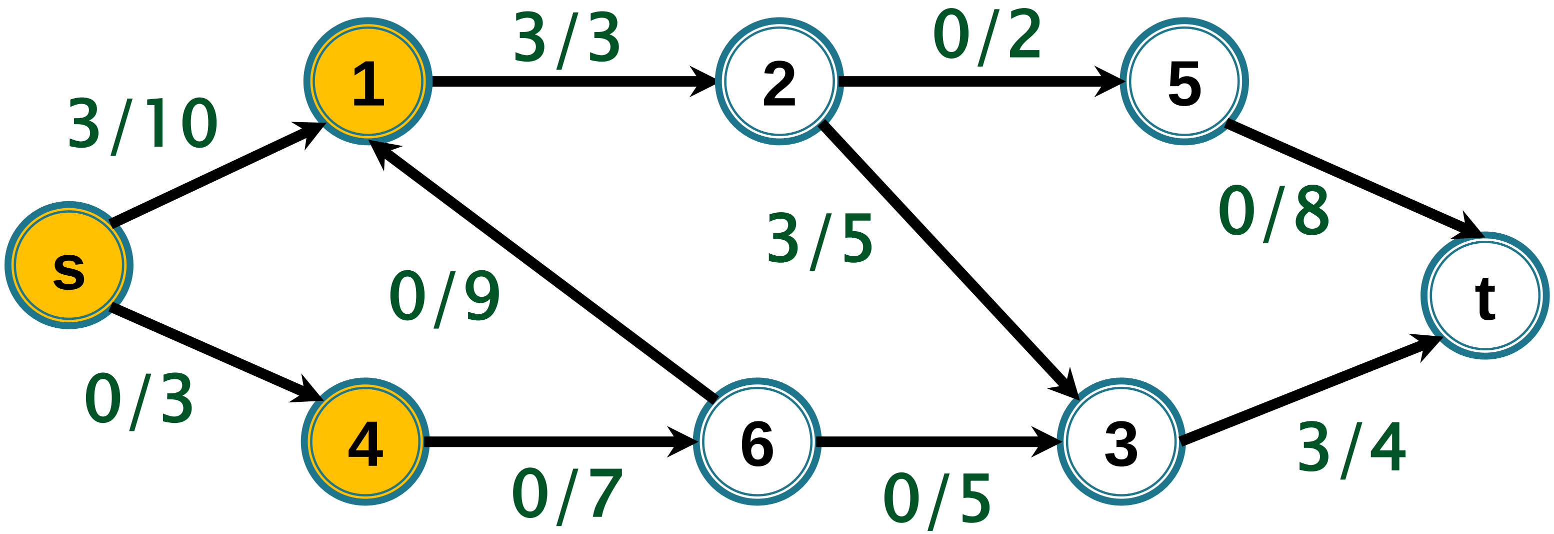

s



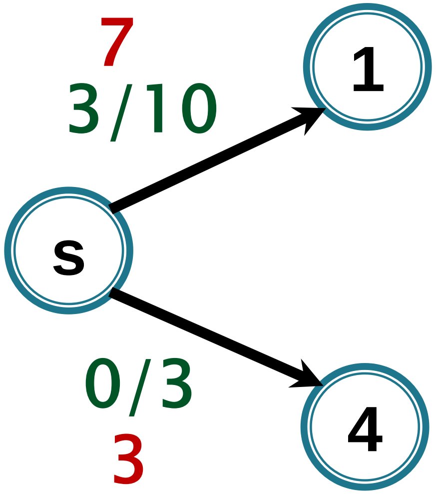
s
1
2
5
t
4
6
3
3/10
3/3
0/2
0/8
3/5
0/9
0/3
0/7
0/5
3/4



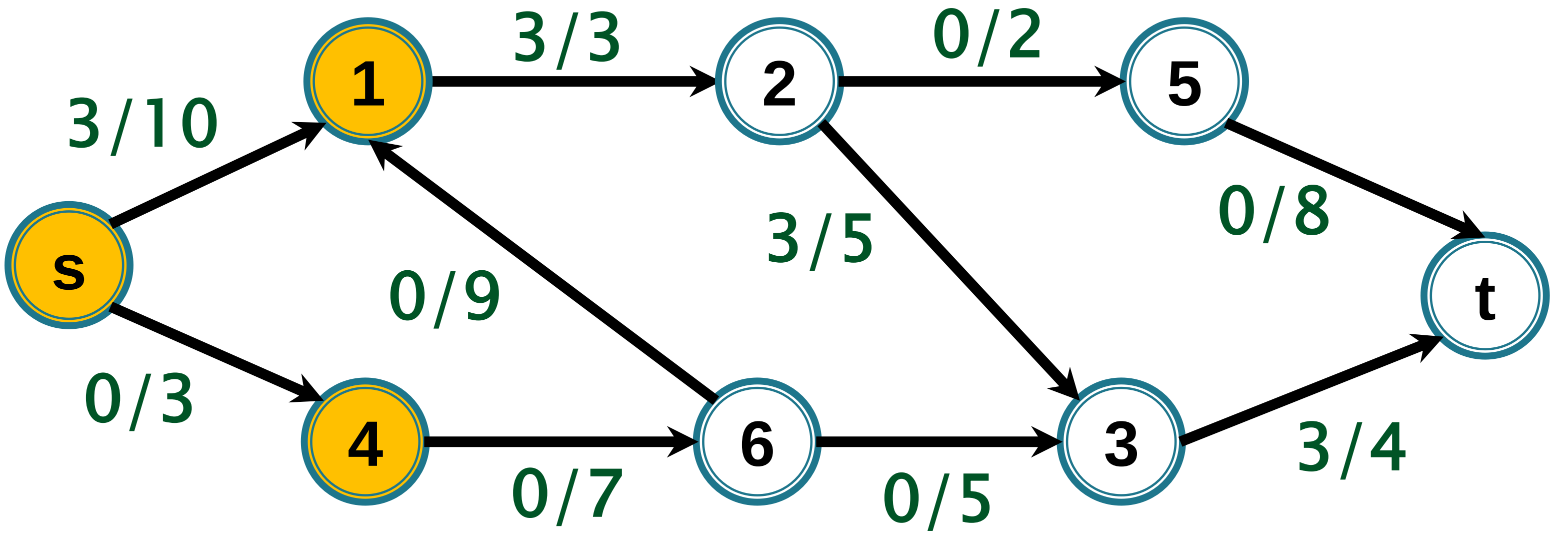
10-3=7
3/10
1
s

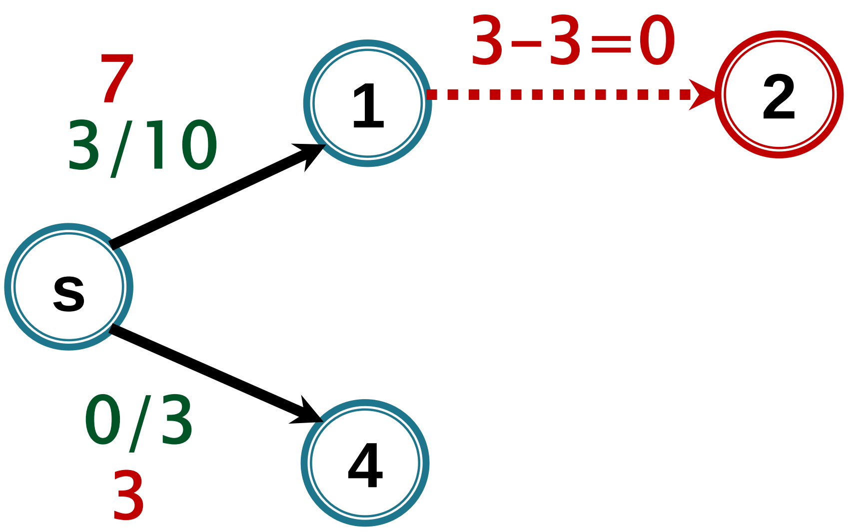


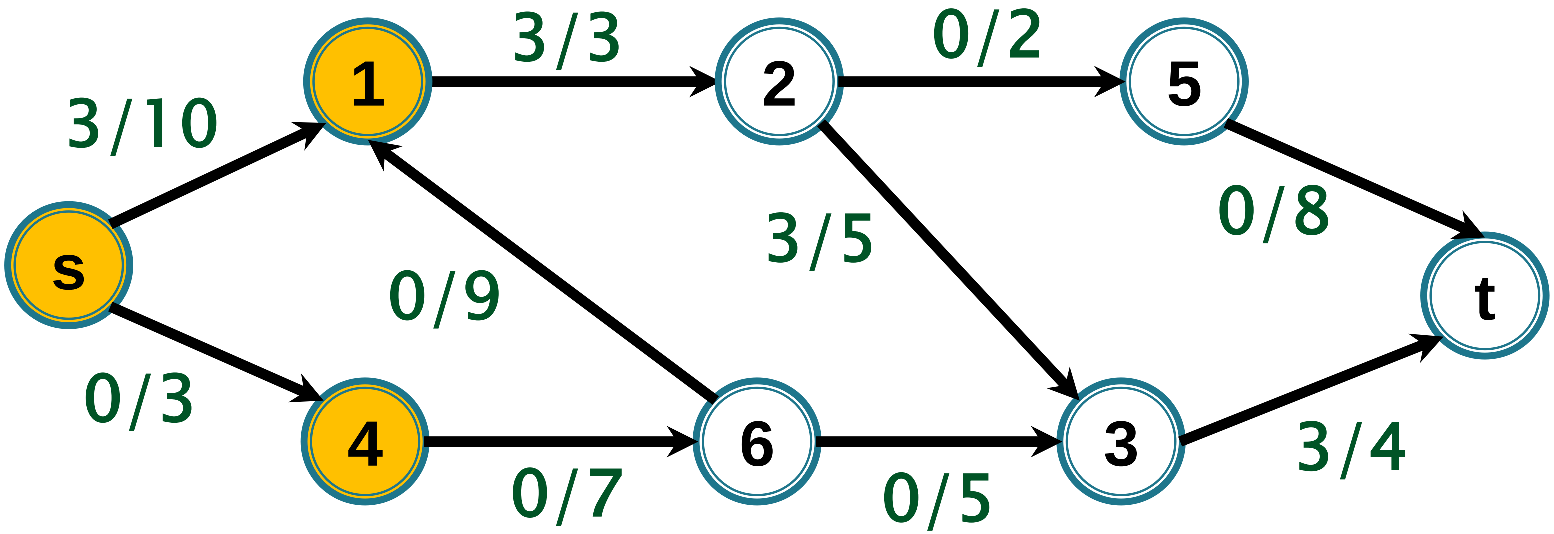
s
1
2
5
t
4
6
3
3/10
3/3
0/2
0/8
3/5
0/9
0/3
0/7
0/5
3/4

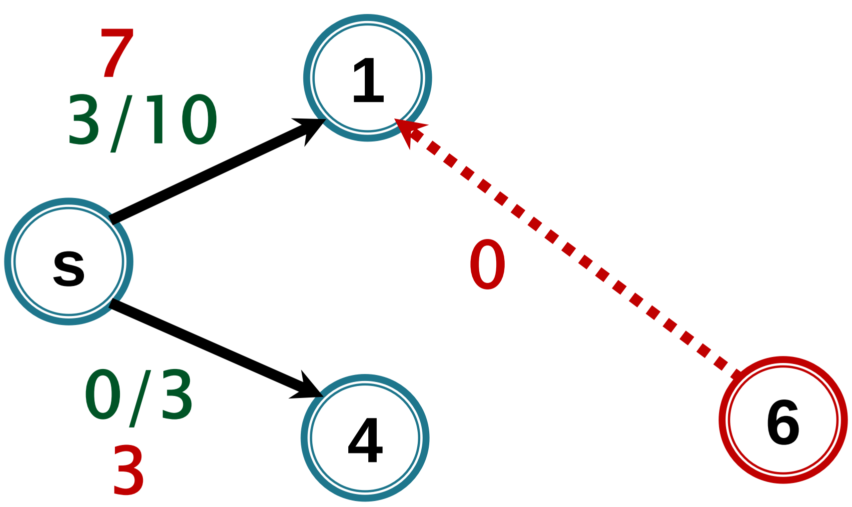


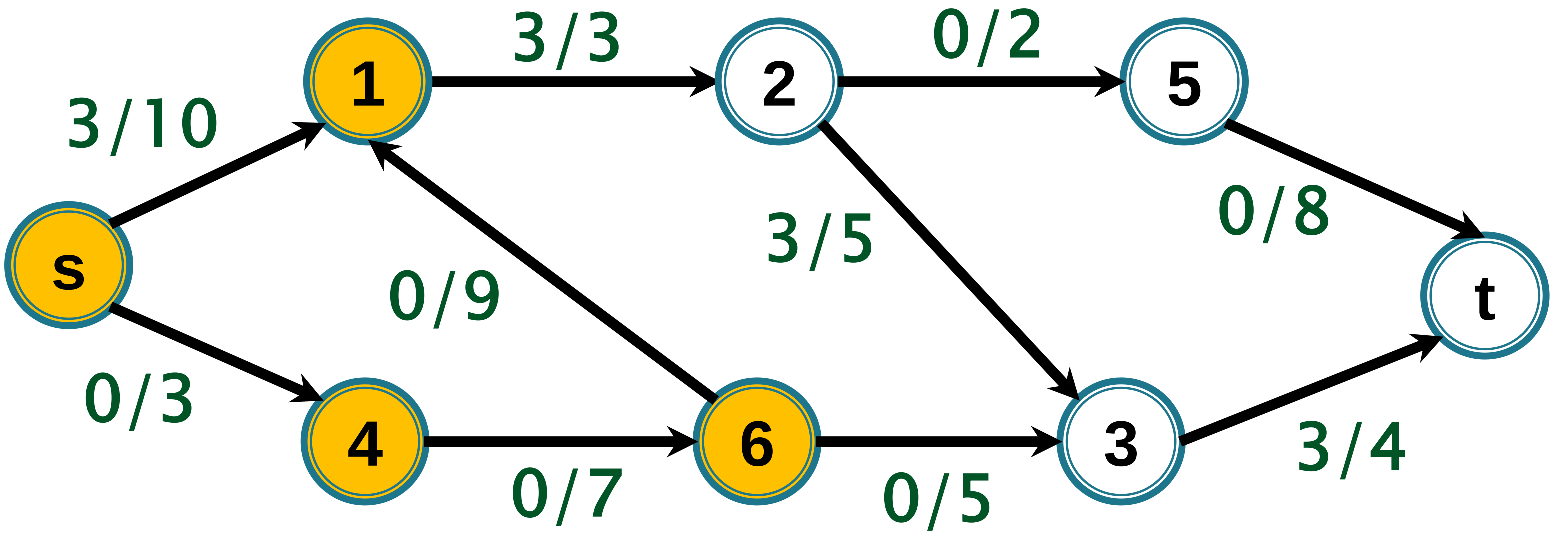
7
3/10
1
s
0/3
3
4


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

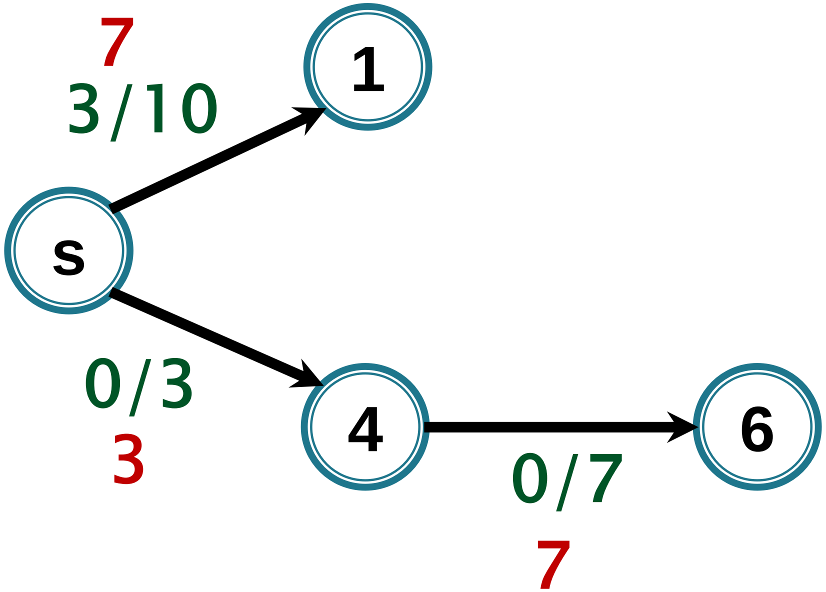

3-3=0
7
3/10
1
2
s
0/3
3
4


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

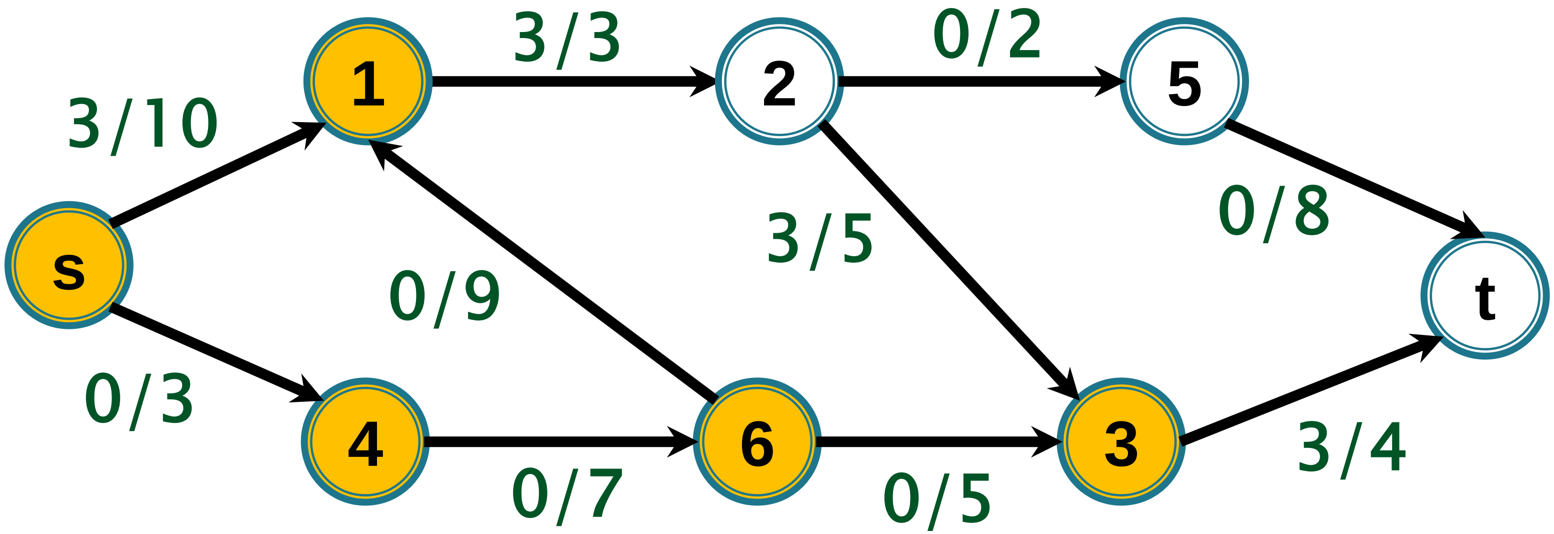

7
3/10
1
s
0
0/3
3
4
6

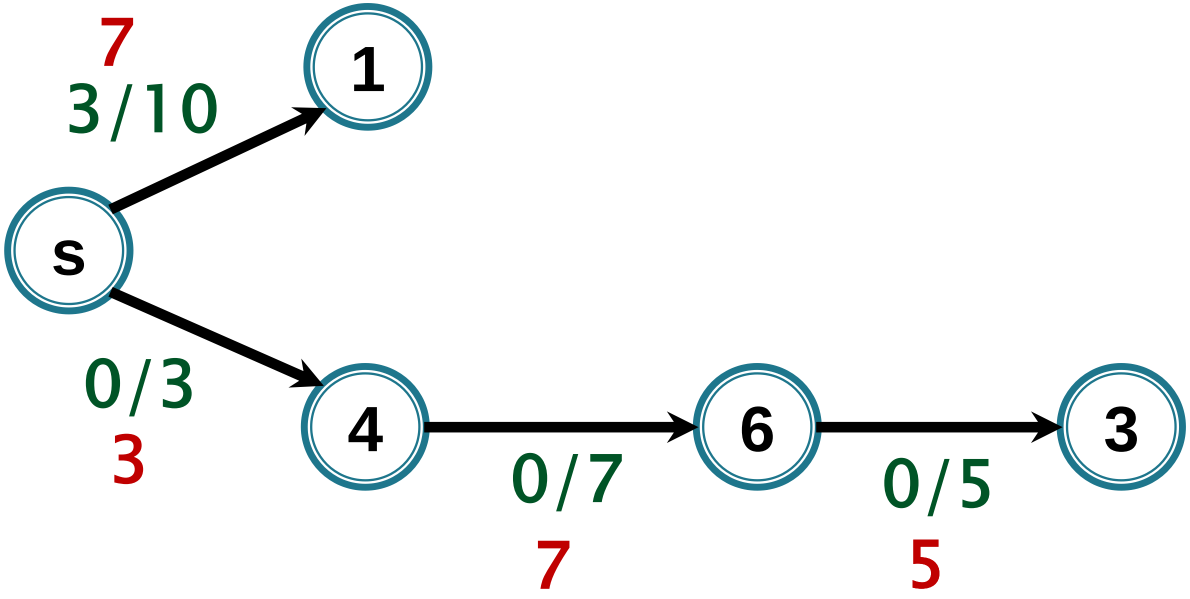


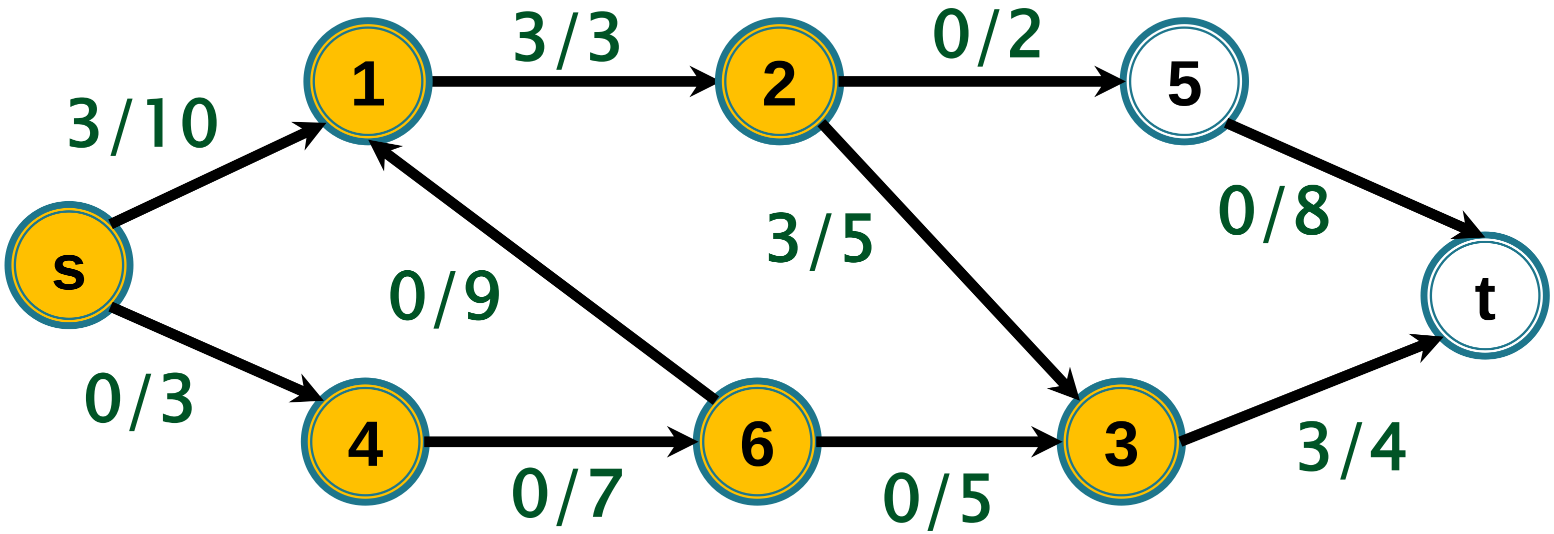
s
1
2
5
t
4
6
3
3/10
3/3
0/2
0/8
3/5
0/9
0/3
0/7
0/5
3/4



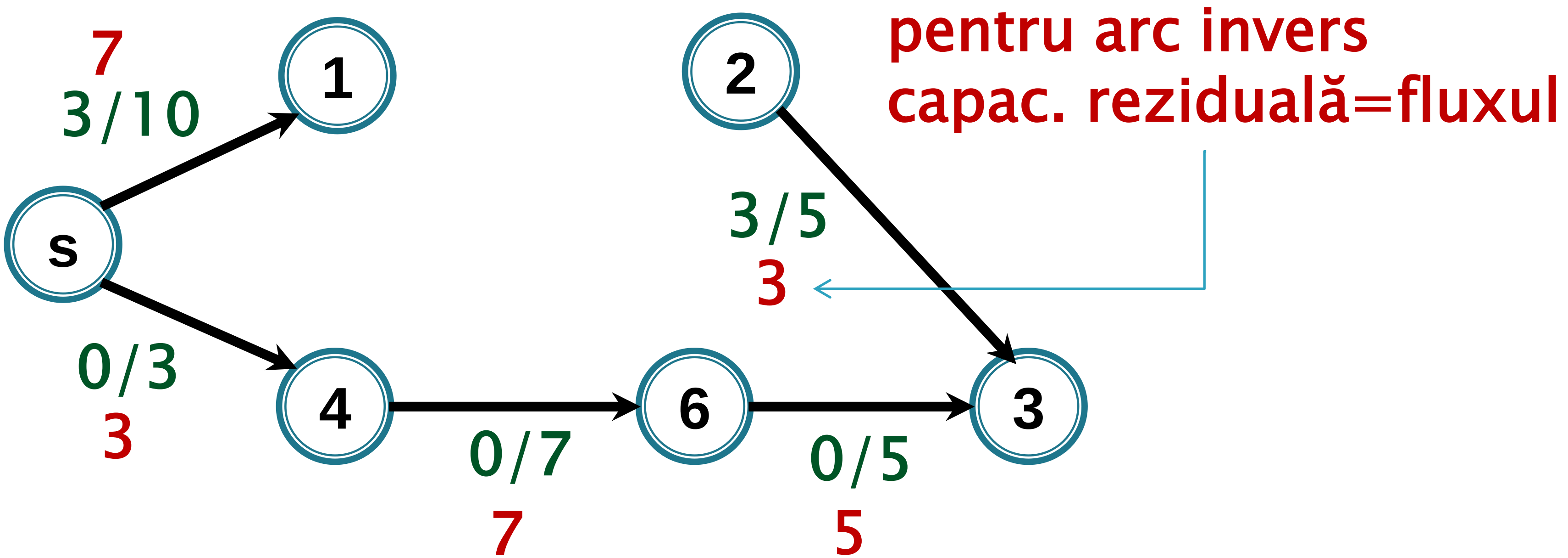
7
3/10
1
s
0/3
3
4
6
0/7
7


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

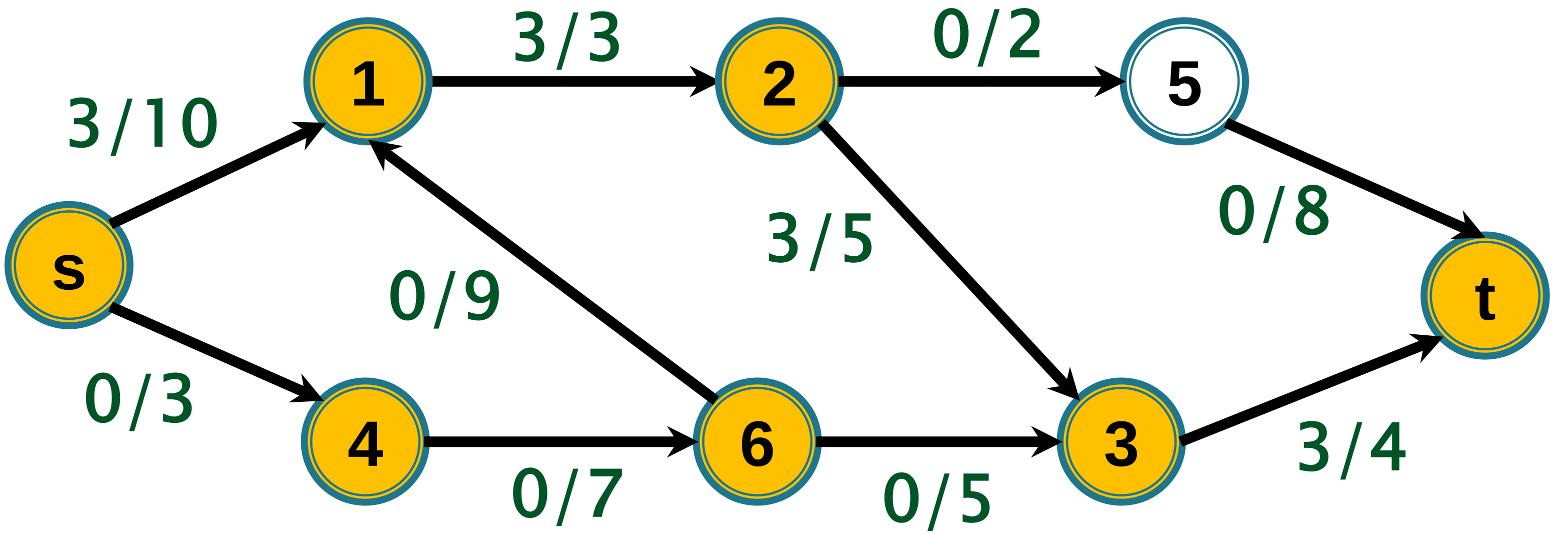

7
3/10
1
s
0/3
3
4
6
3
0/7
0/5
7
5

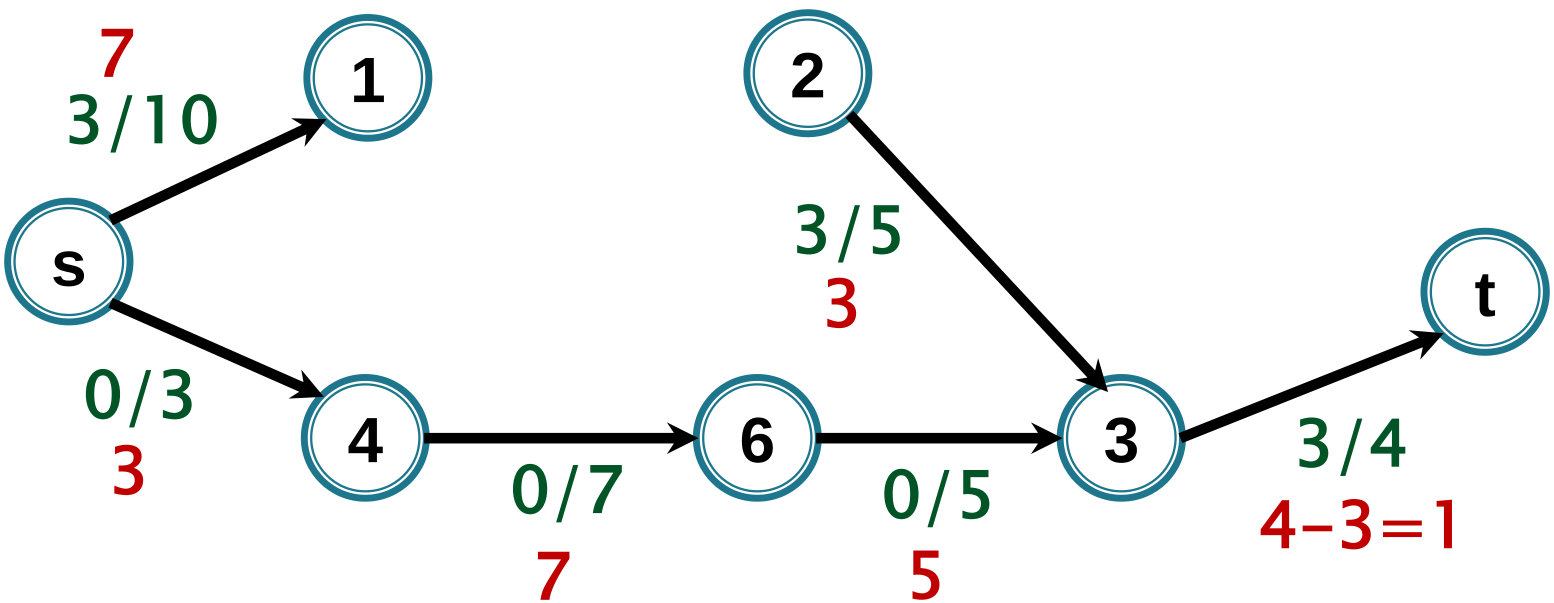


s
1
2
5
t
4
6
3
3/10
3/3
0/2
0/8
3/5
0/9
0/3
0/7
0/5
3/4

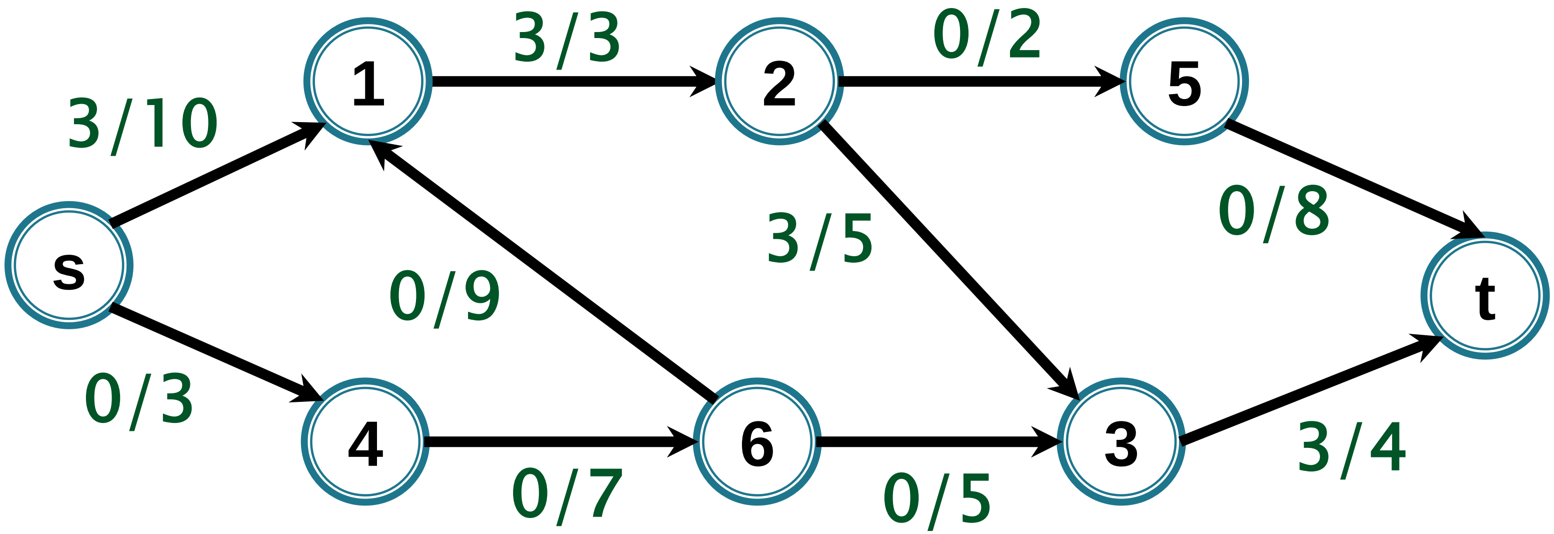


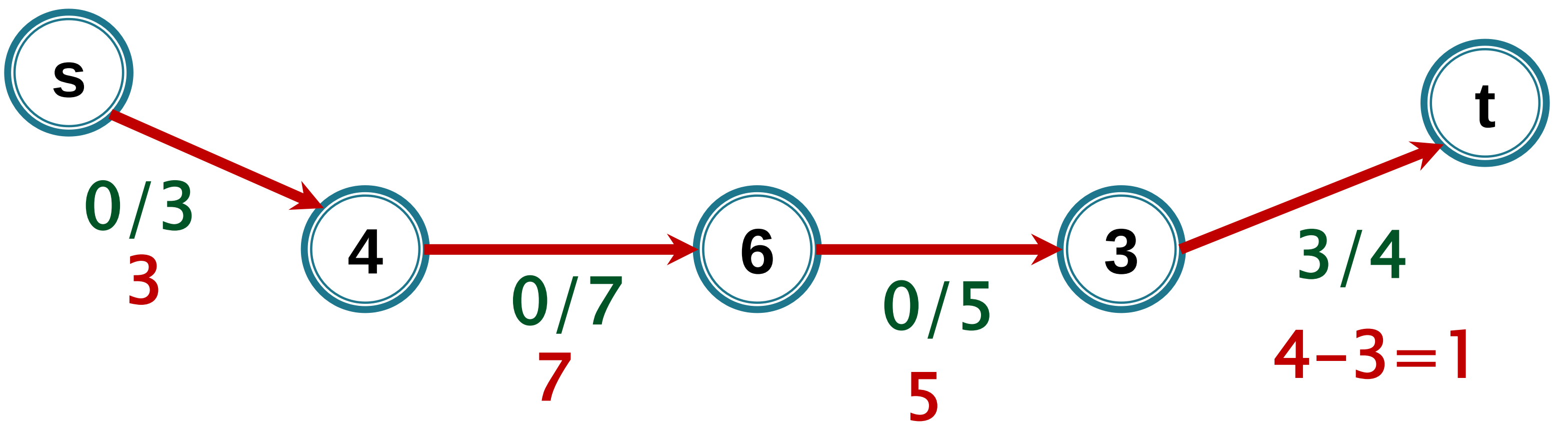
pentru arc invers
capac. reziduală=fluxul
s
1
2
4
6
3
7
3/10
3/5
3
0/3
3
0/7
7
0/5
5

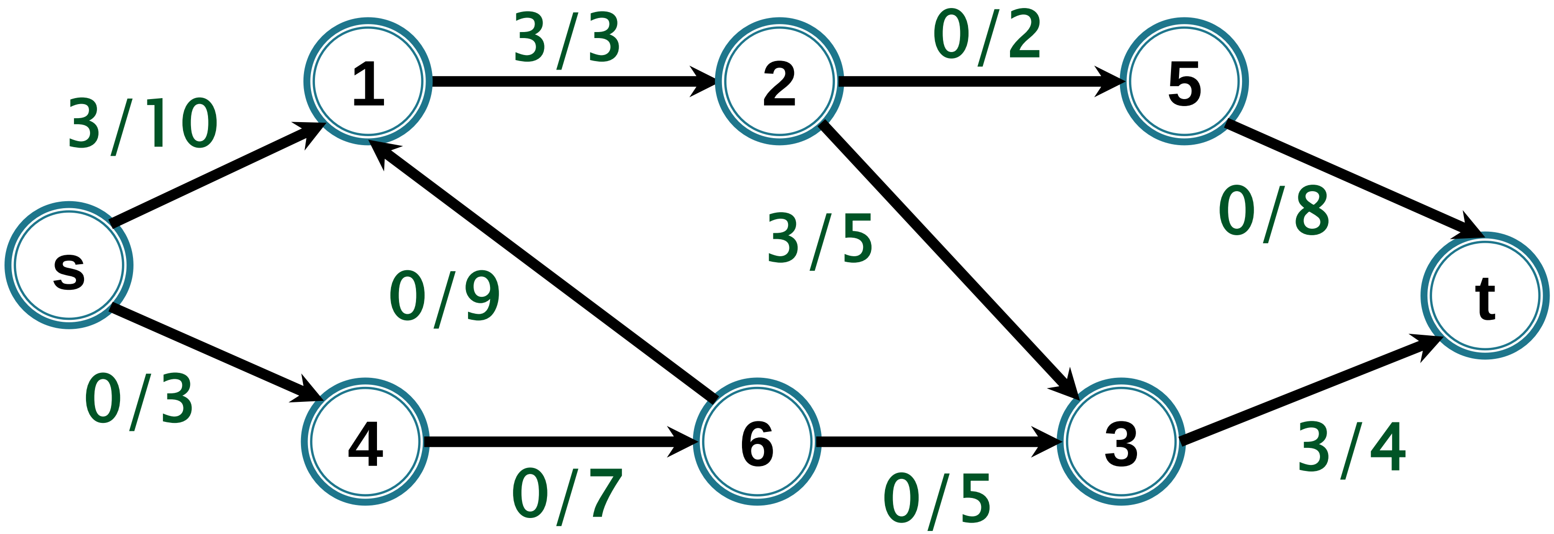

3/3
0/2
1
2
5
3/10
s
0/8
t
3/5
0/9
0/3
4
6
3
3/4
0/7
0/5

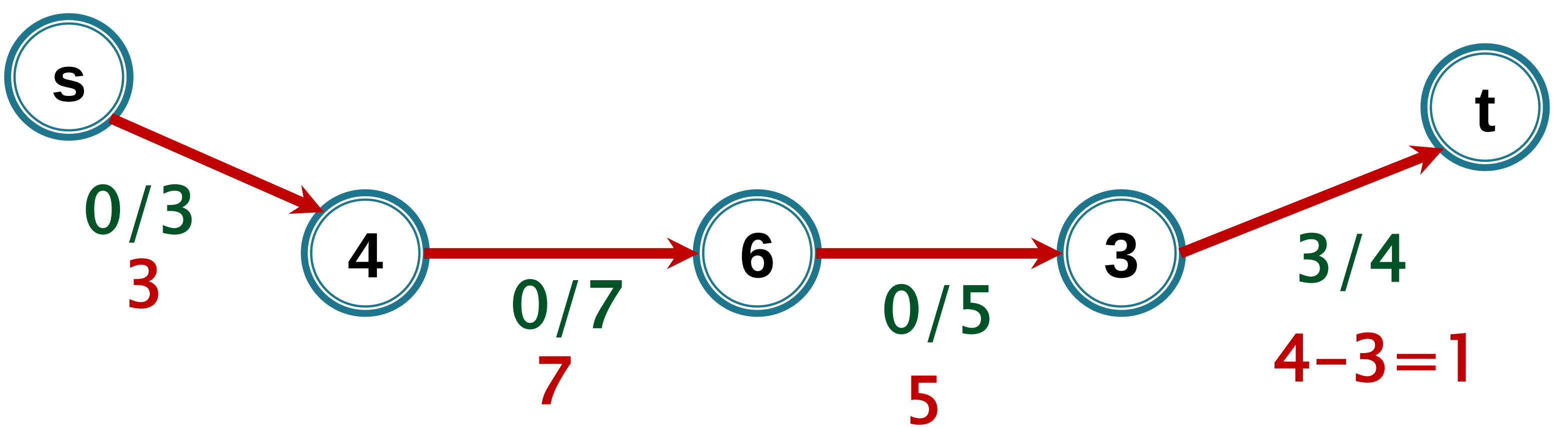

7
3/10
1
2
s
3/5
3
t
0/3
3
4
6
3
3/4
0/7
0/5
4−3=1
7
5

revizuieste_flux_lant

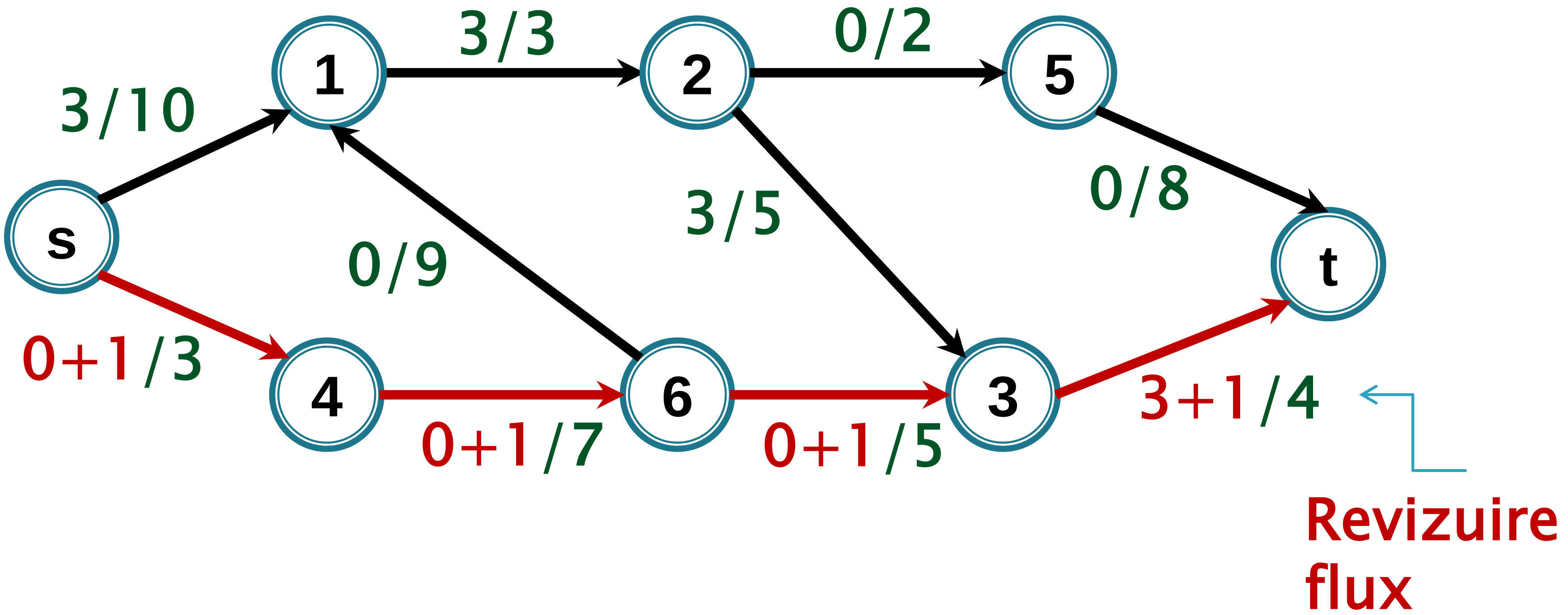

3/3
0/2
3/10
1
2
5
s
0/9
3/5
0/8
t
0/3
4
6
3
3/4
0/7
0/5


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

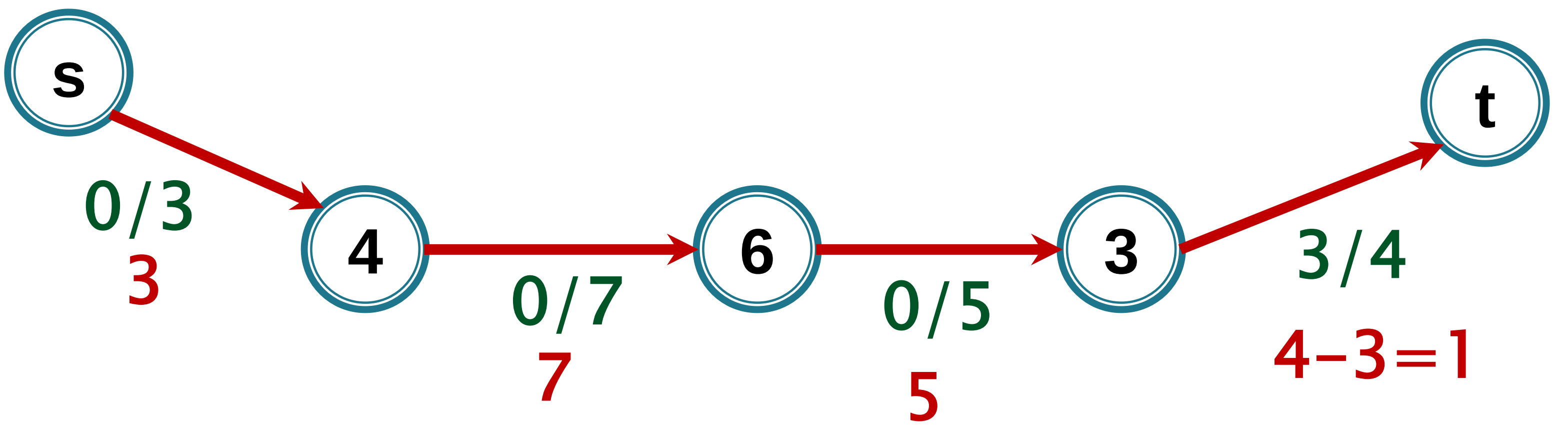




$i(P) = \min\{3, 7, 5, 1\} = 1$

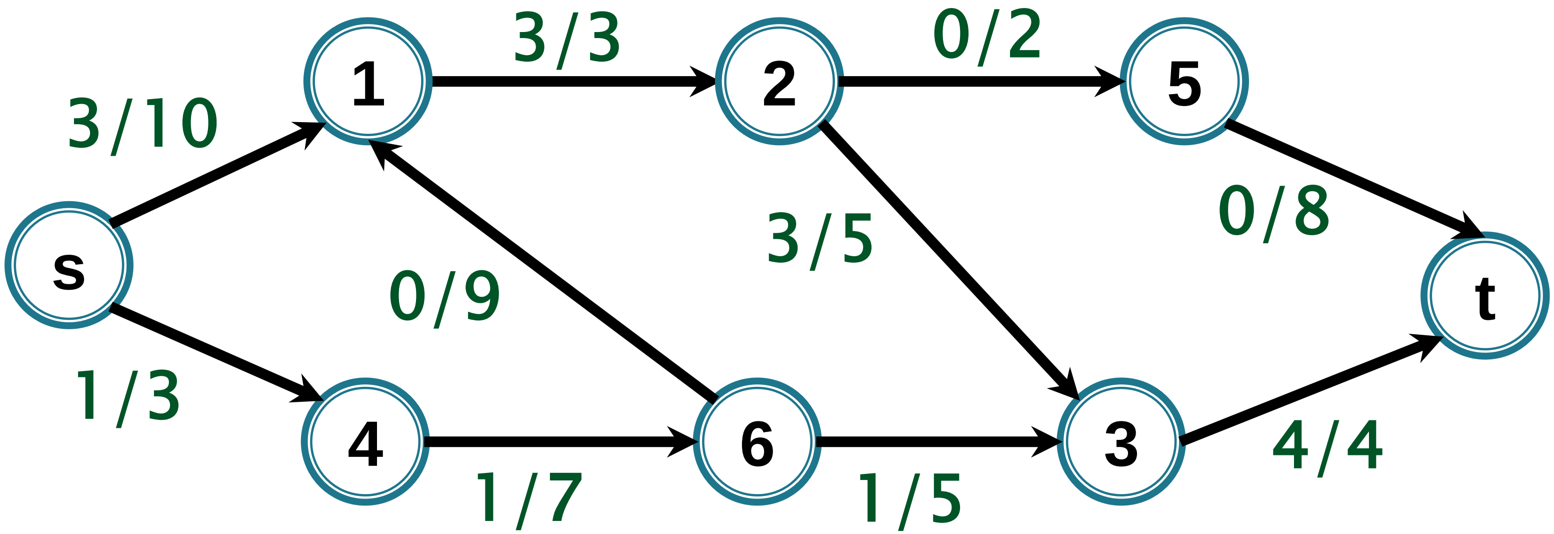


s
1
2
5
t
4
6
3
3/10
3/3
0/2
0/8
3/5
0/9
0+1/3
0+1/7
0+1/5
3+1/4
Revizuire flux



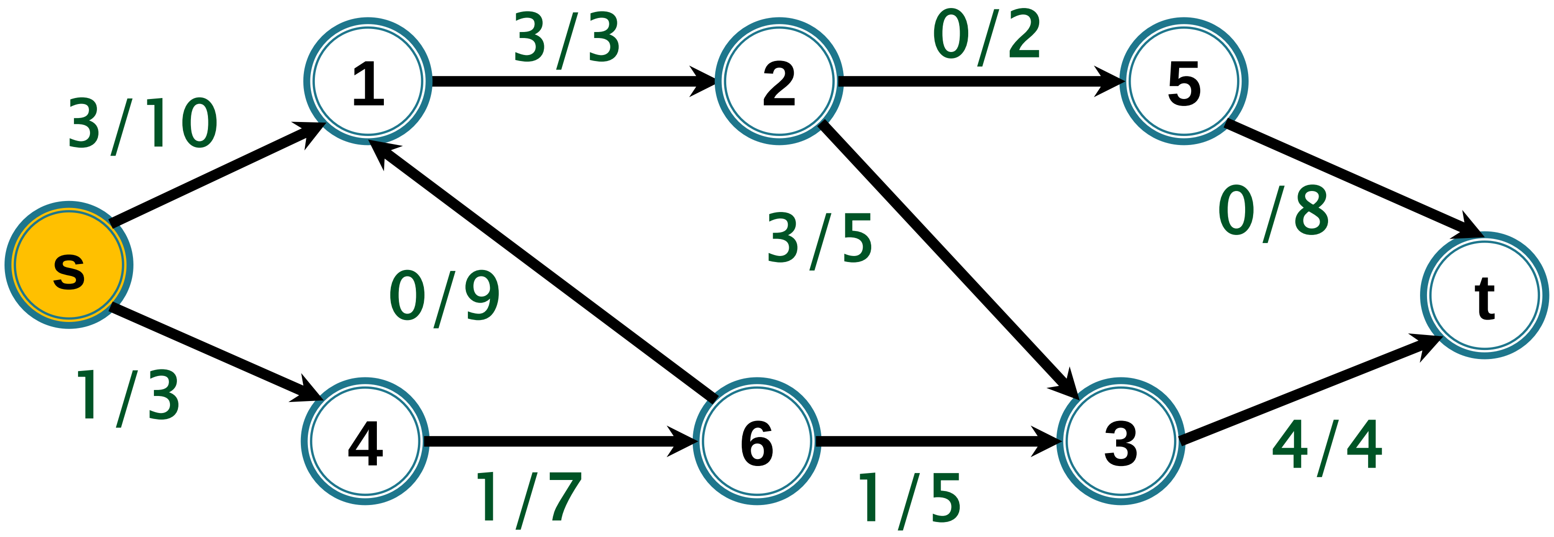
s
4
6
3
t
0/3
3
0/7
7
0/5
5
3/4
4−3=1

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

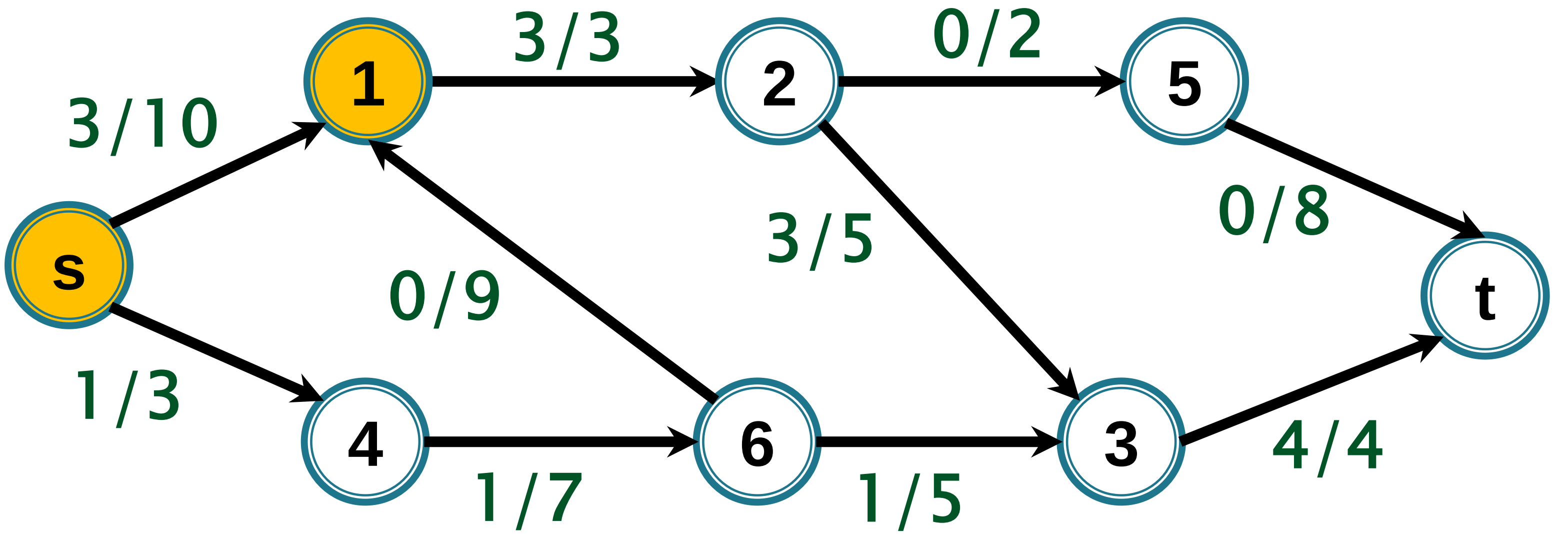


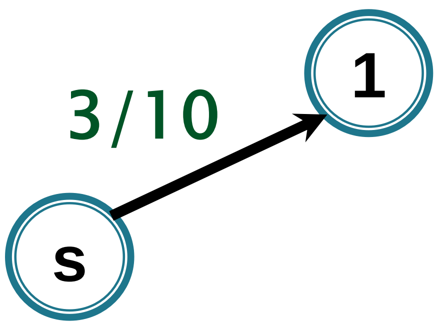
s
1
2
5
t
4
6
3
3/10
3/3
0/2
0/8
3/5
0/9
1/3
1/7
1/5
4/4

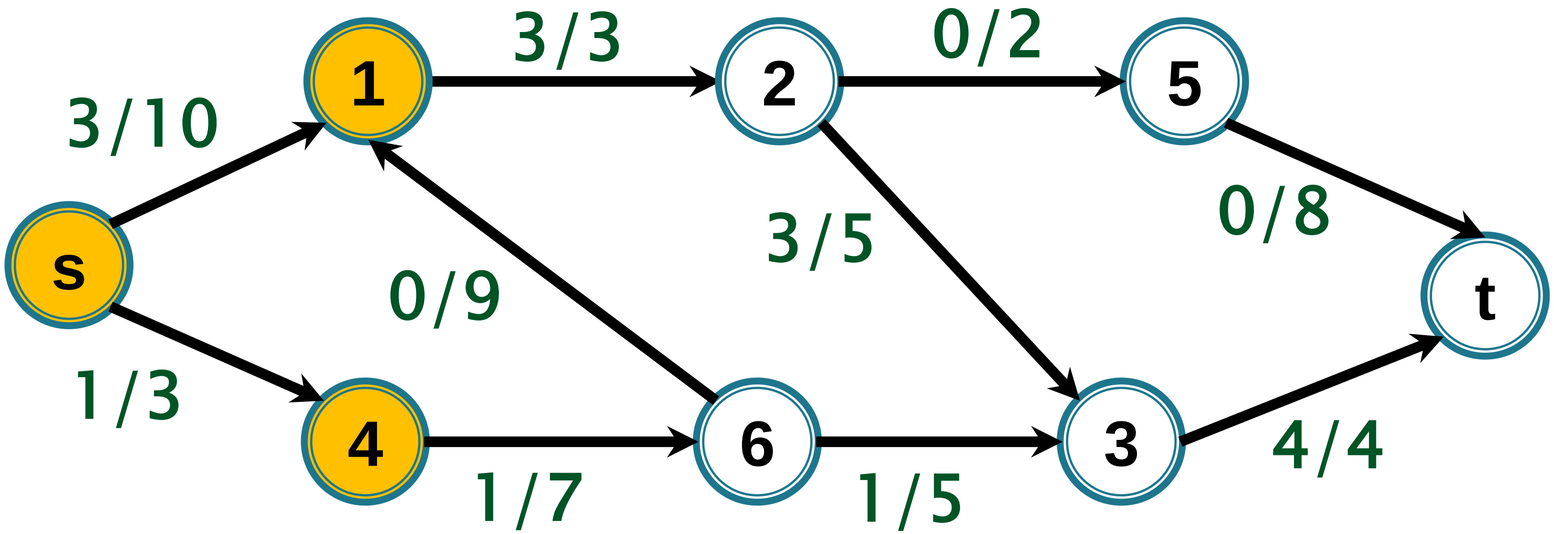
construieste_s-t_lant_nesat_BF


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

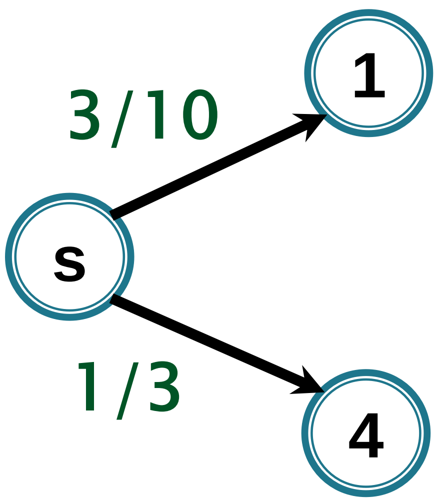

s


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

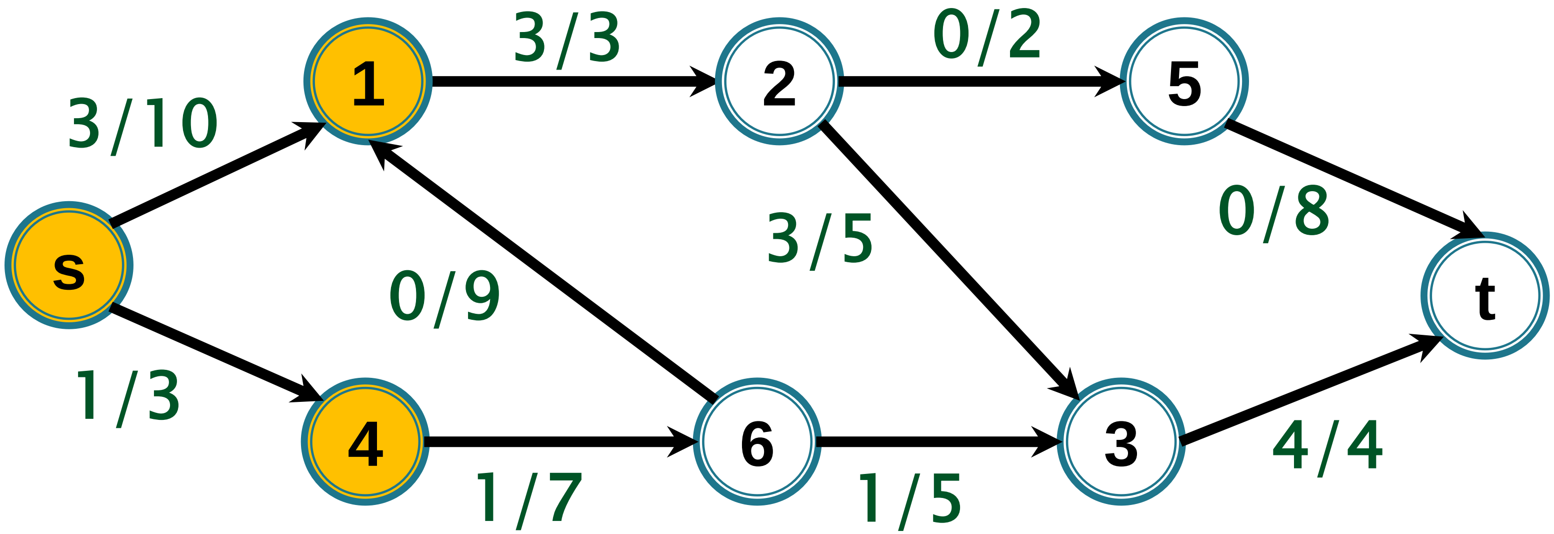

s
1
3/10



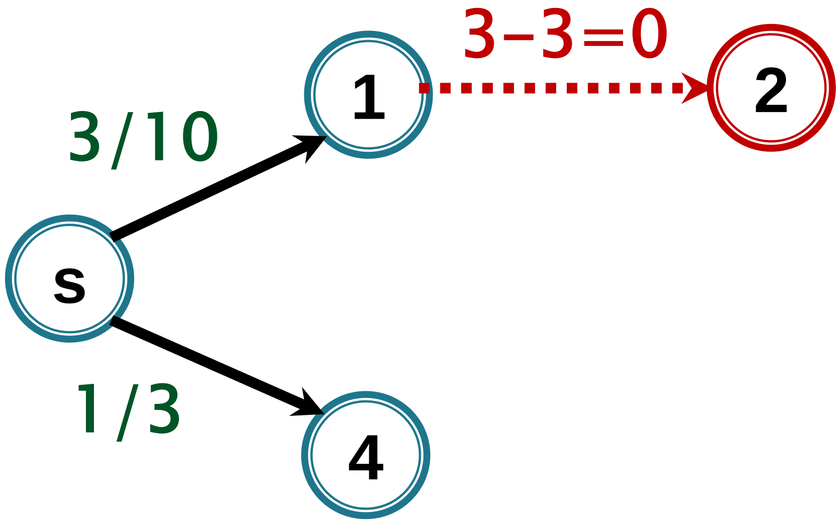
3/10
1
3/3
2
0/2
5
s
0/9
3/5
0/8
t
1/3
4
1/7
6
1/5
3
4/4



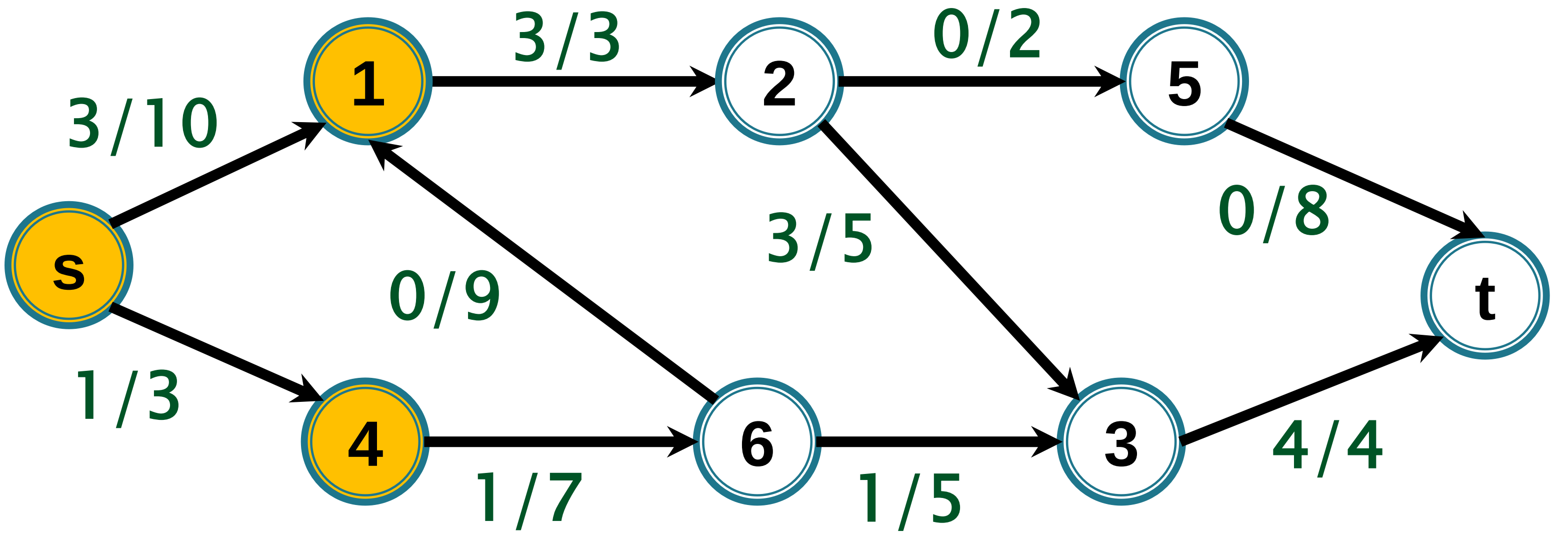
3/10
1
s
1/3
4

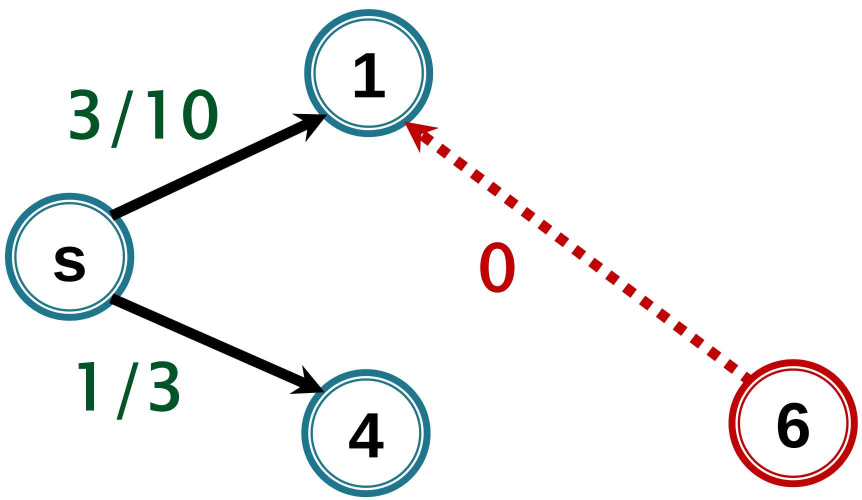


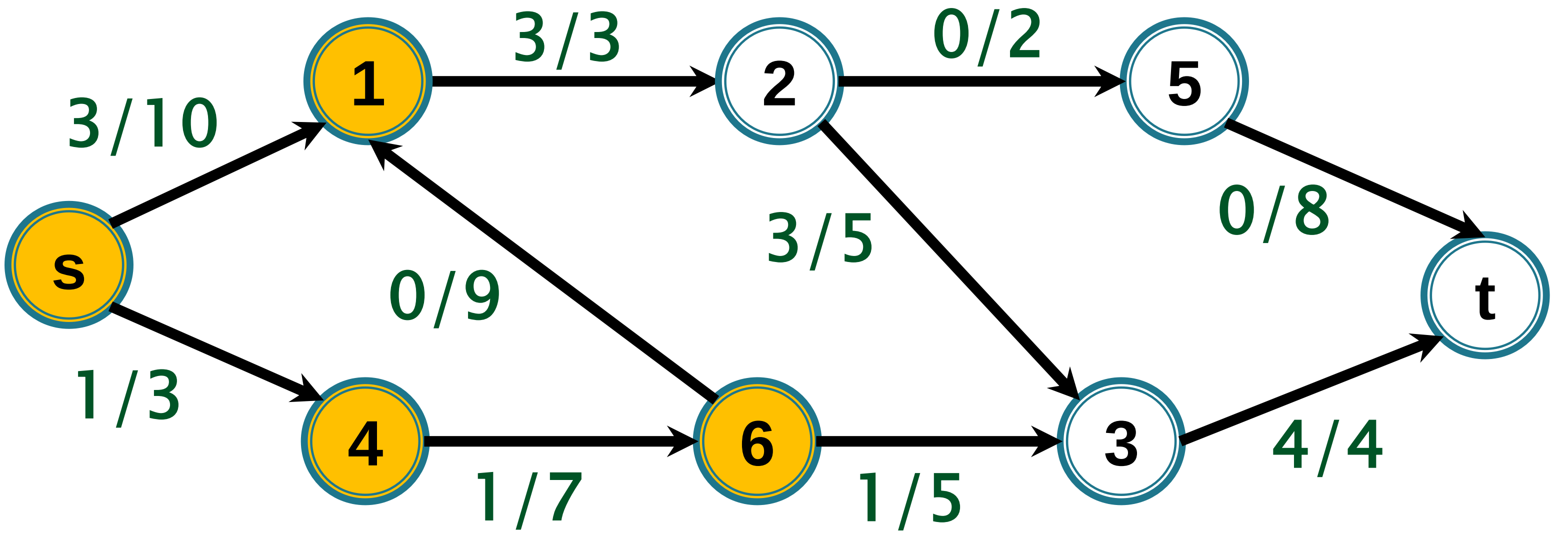
s
1
2
5
t
4
6
3
3/10
1/3
3/3
0/2
0/9
3/5
0/8
1/7
1/5
4/4

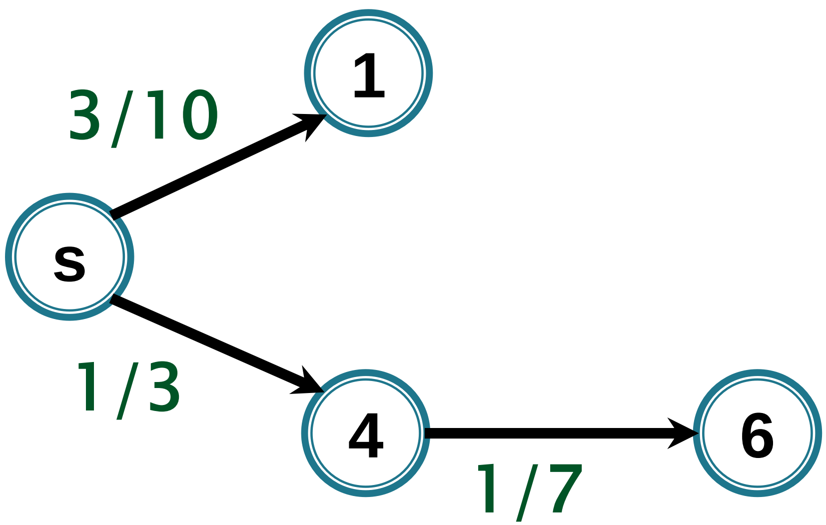


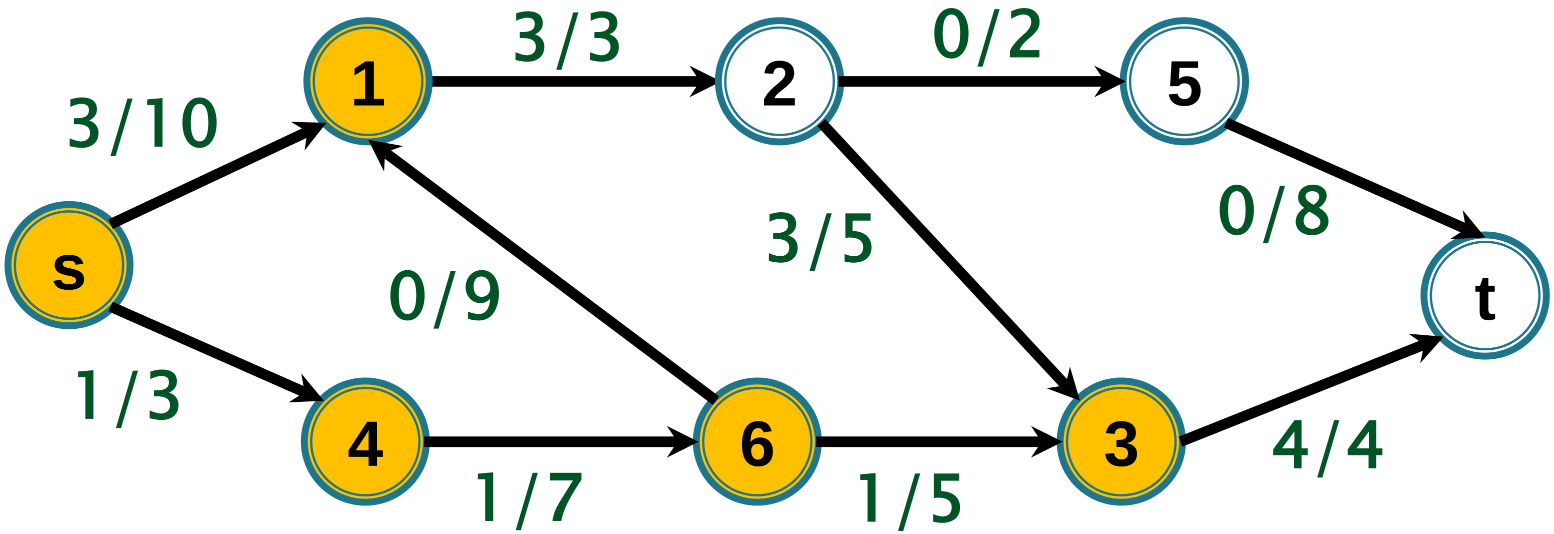
s
1
2
4
3/10
1/3
3-3=0


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

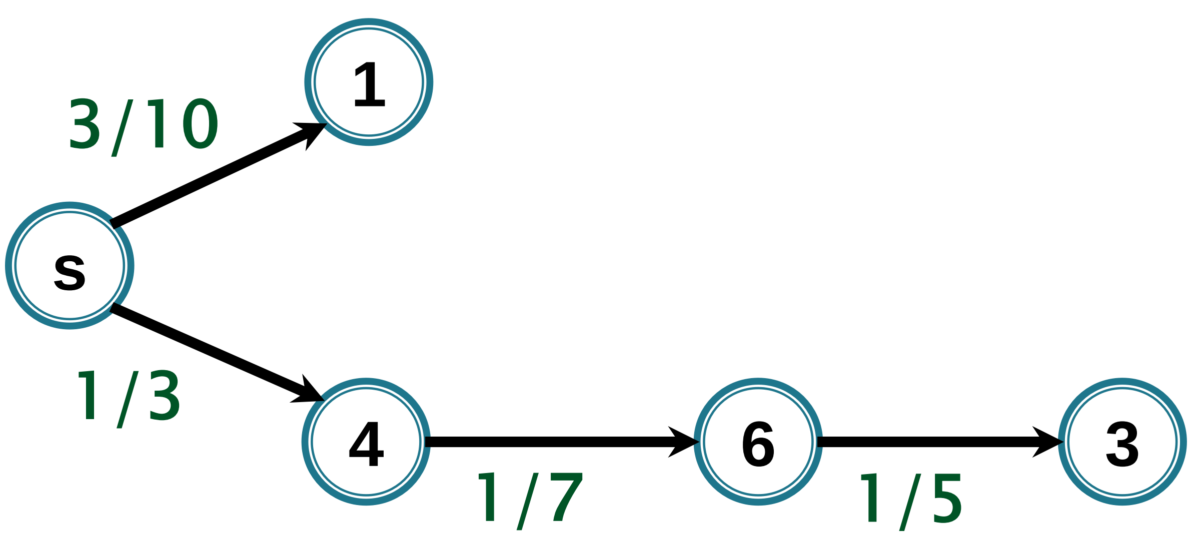

s
1
4
6
3/10
1/3
0


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

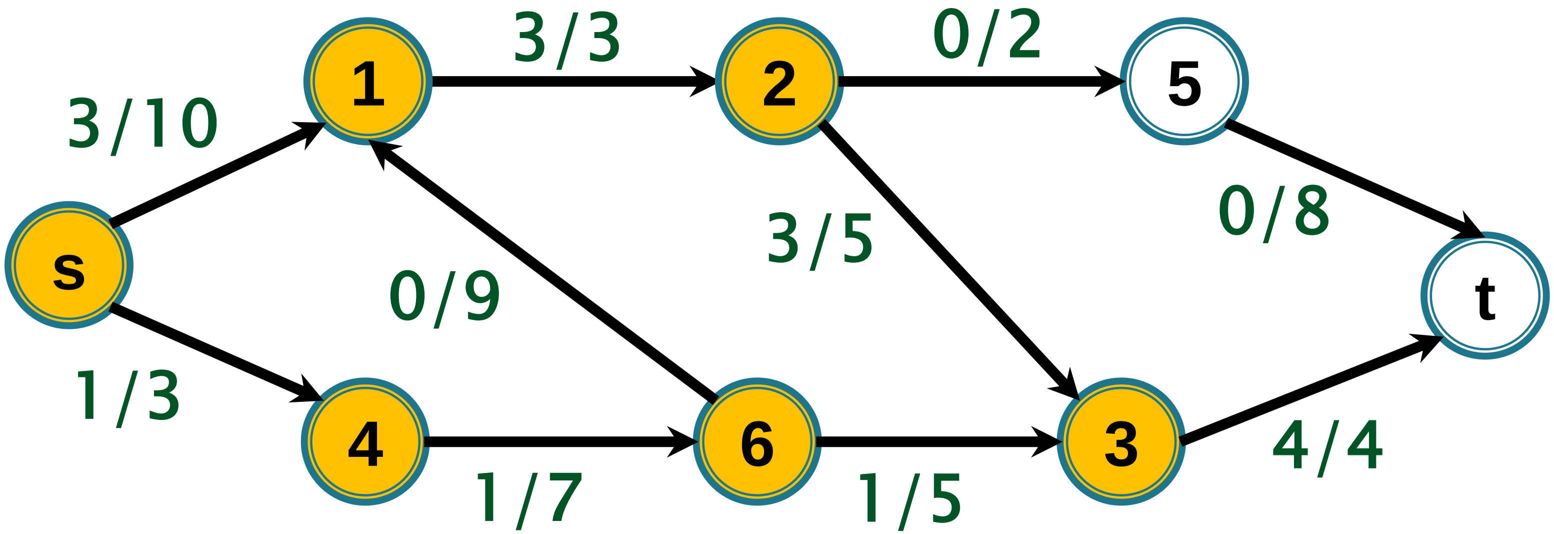

1
3/10
s
1/3
4
6
1/7

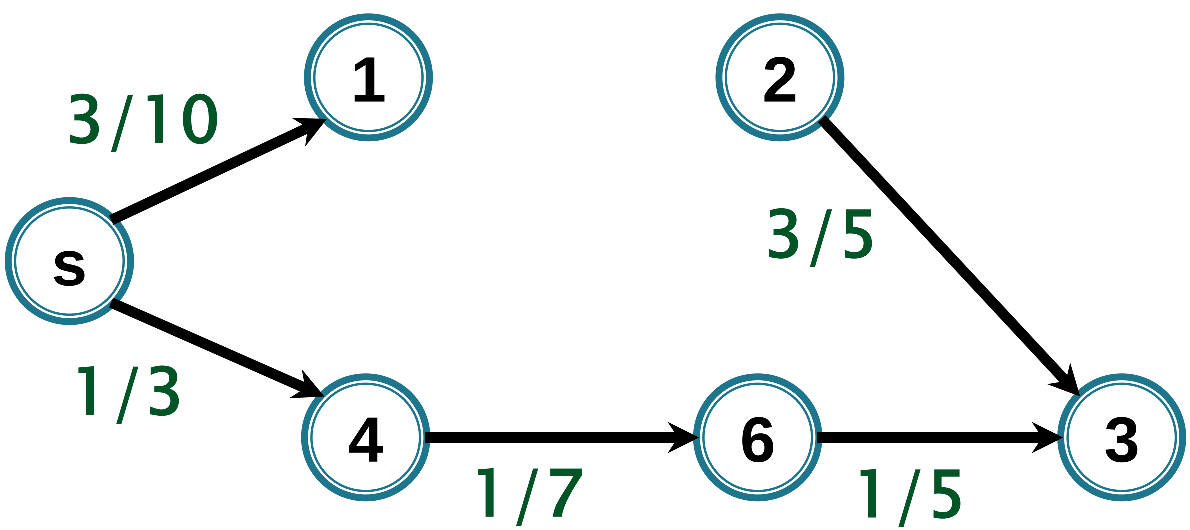


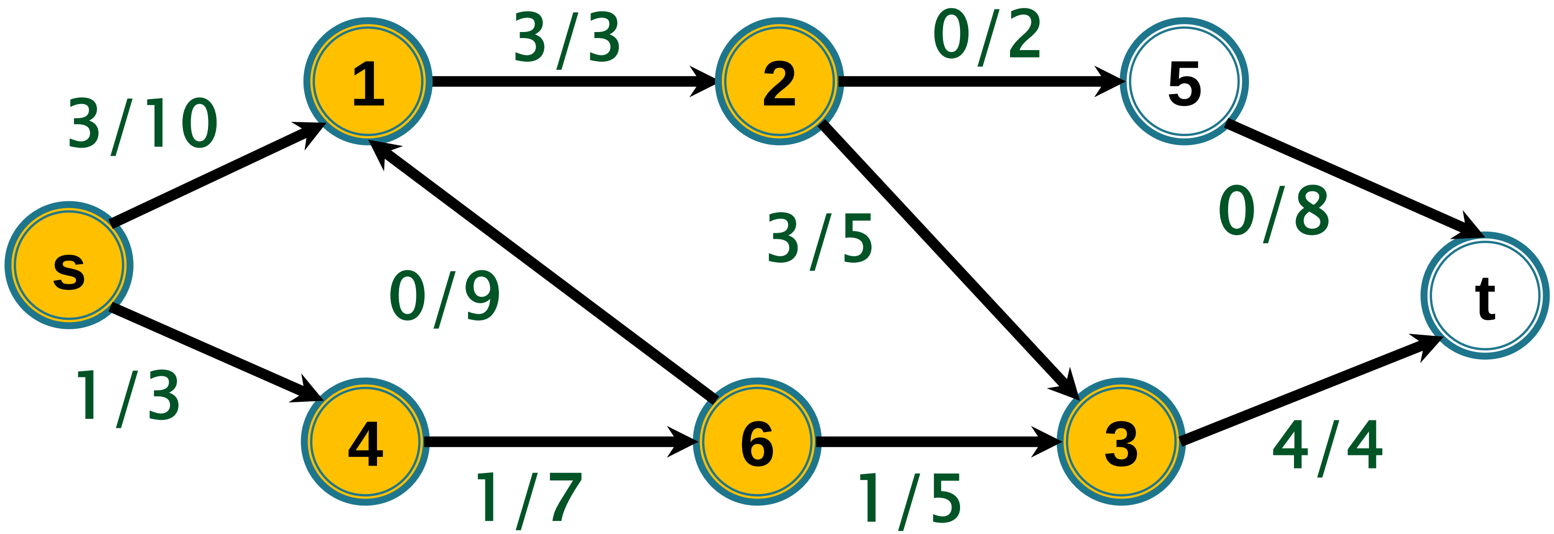
s
1
2
5
t
4
6
3
3/10
3/3
0/2
0/8
3/5
0/9
1/3
1/7
1/5
4/4

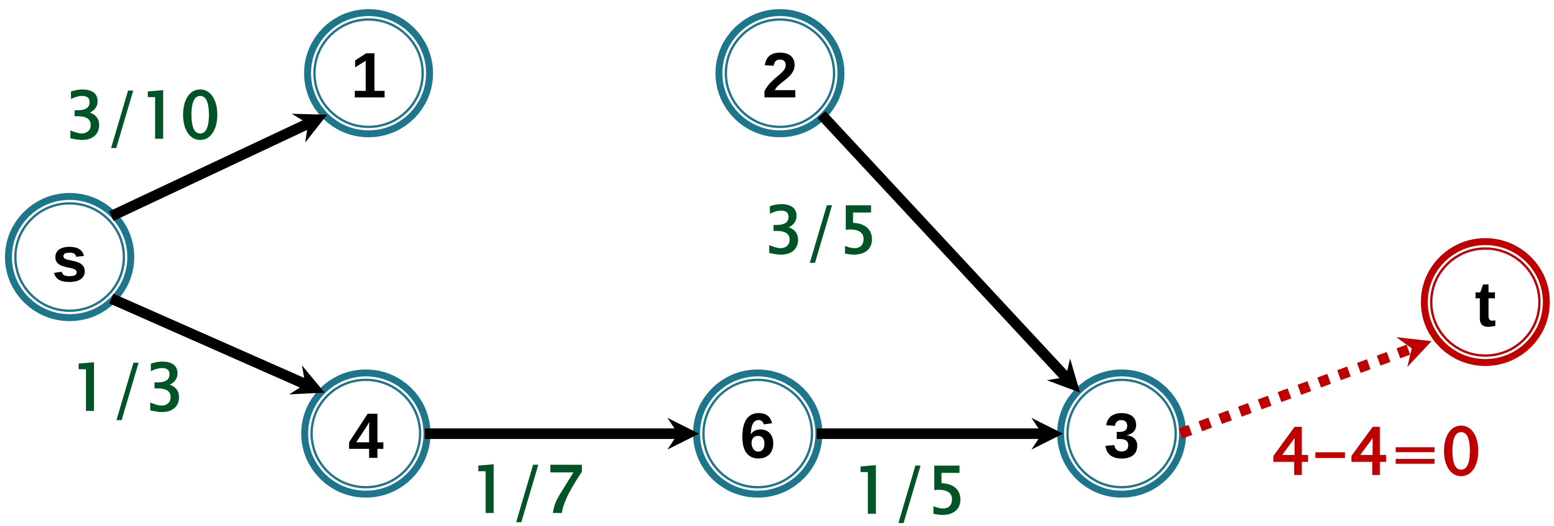


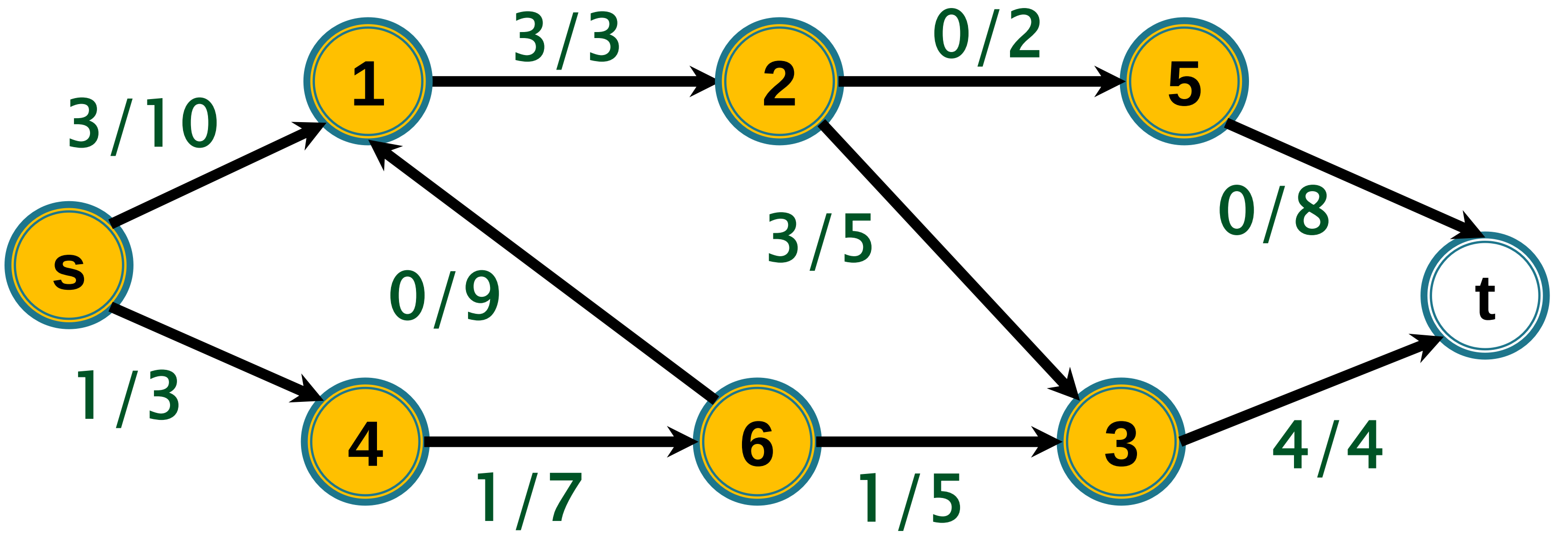
s
1
4
6
3
3/10
1/3
1/7
1/5


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

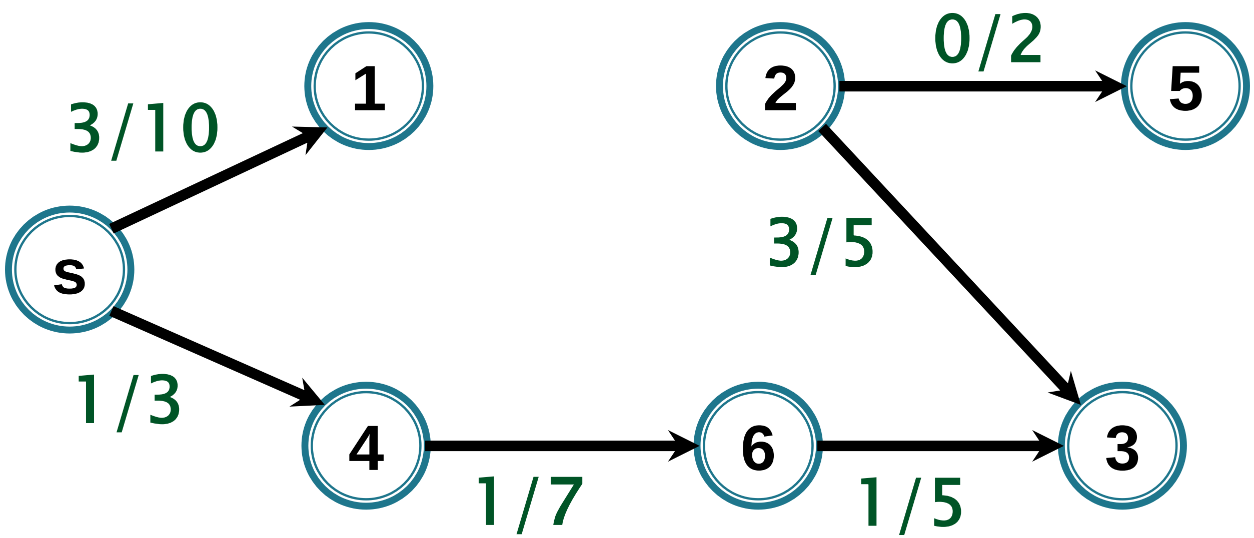

3/10
s
1
2
3/5
1/3
4
6
3
1/7
1/5


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

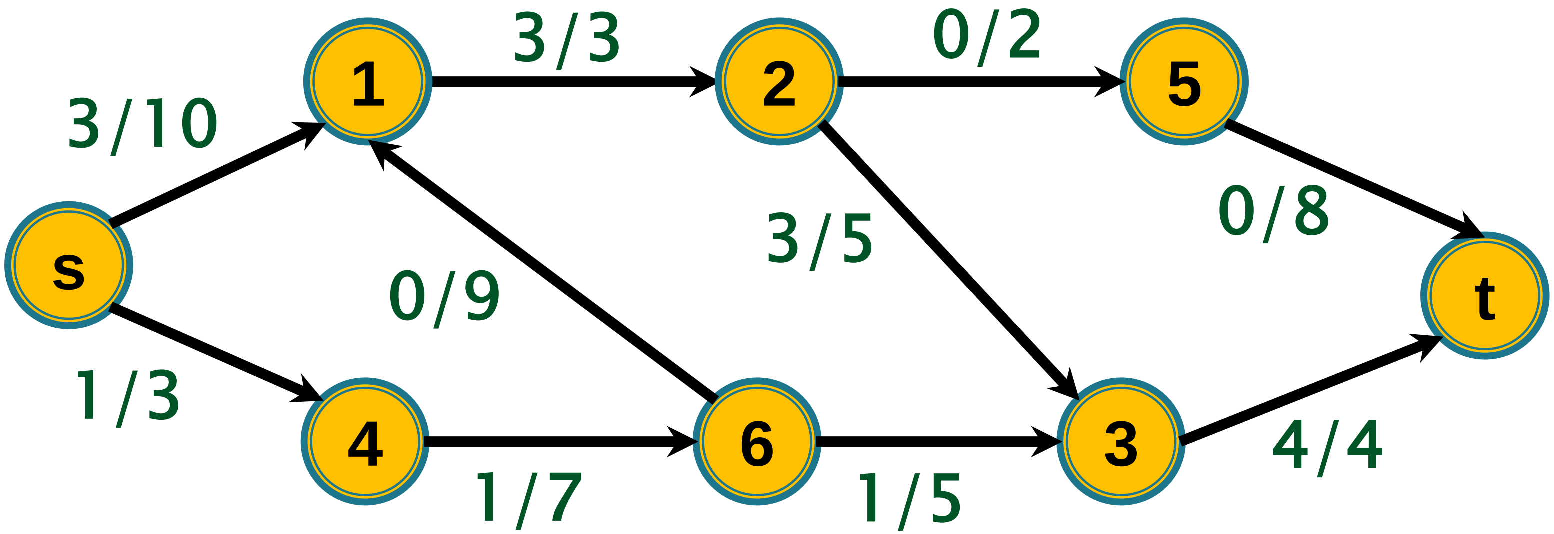

3/10
1
2
s
3/5
t
1/3
4
6
3
4−4=0
1/7
1/5



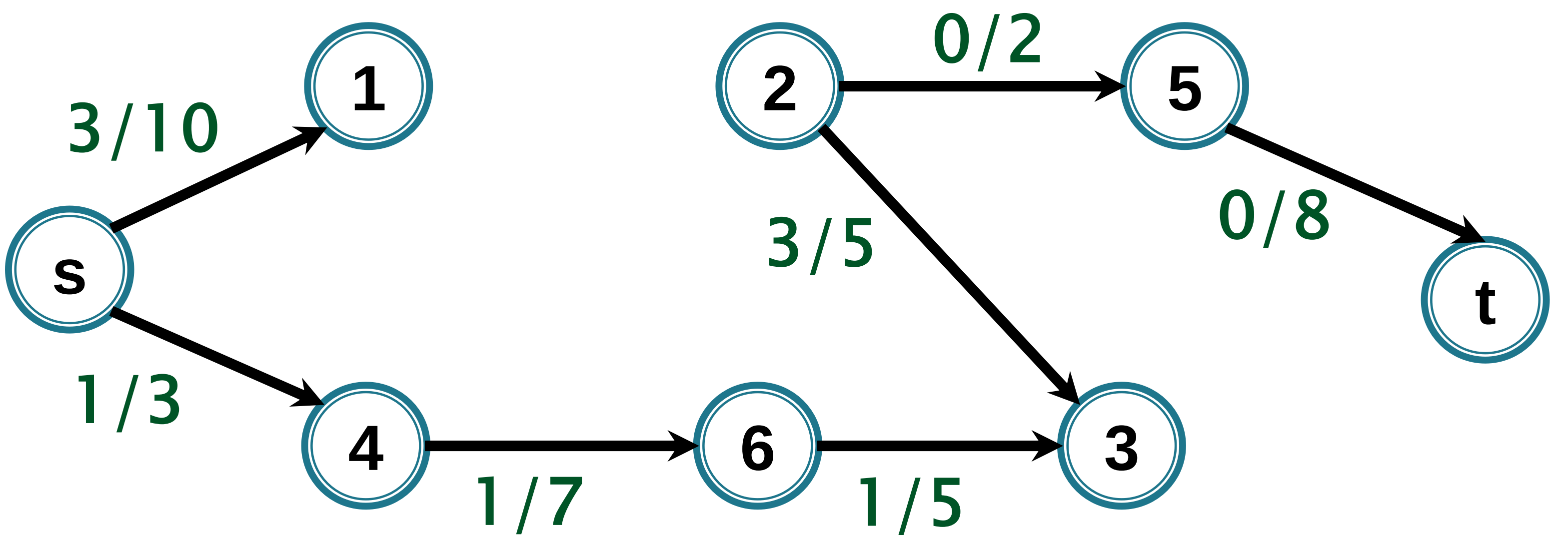
s
1
2
5
t
4
6
3
3/10
3/3
0/2
0/8
0/9
3/5
1/3
1/7
1/5
4/4



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

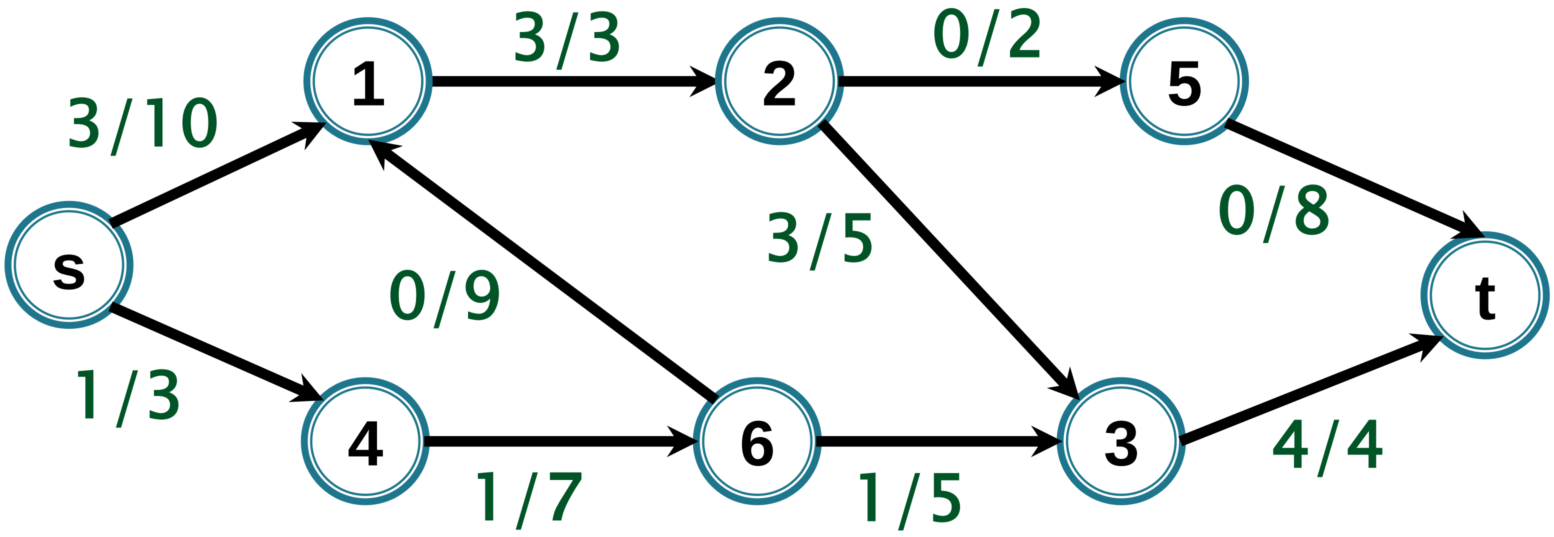


1
2
5
s
t
4
6
3
3/10
3/3
0/2
0/8
3/5
0/9
1/3
1/7
1/5
4/4

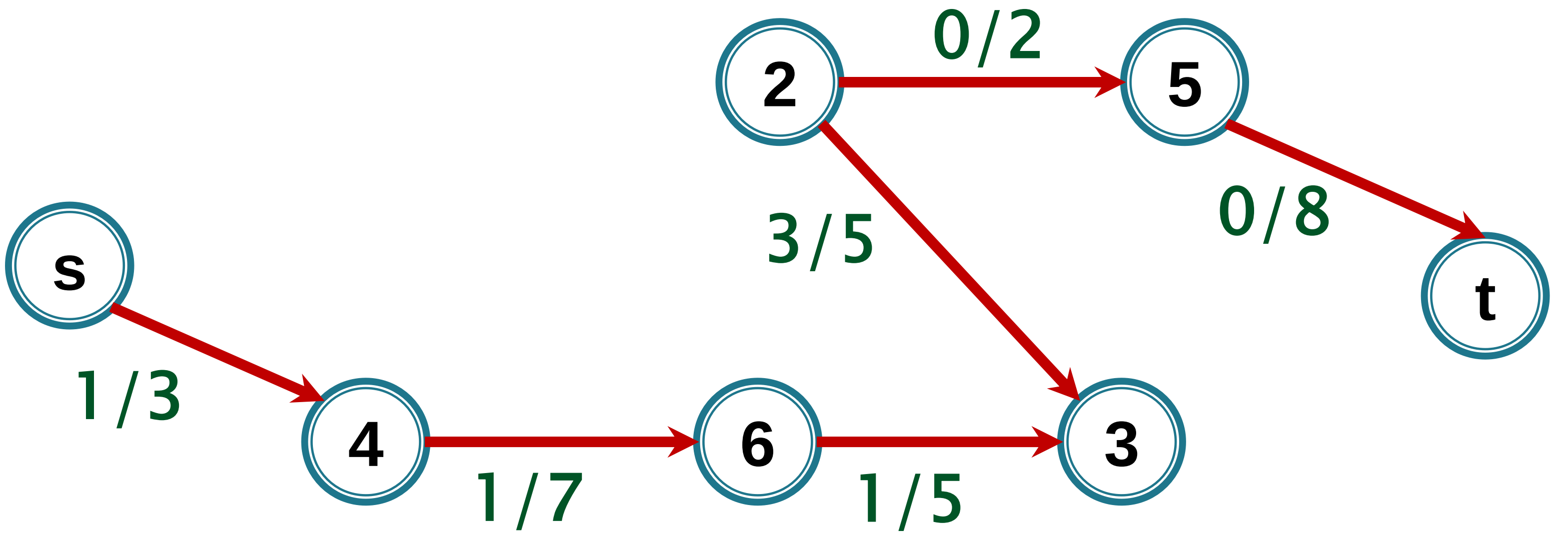


1
2
5
s
t
4
6
3
3/10
0/2
0/8
3/5
1/3
1/7
1/5

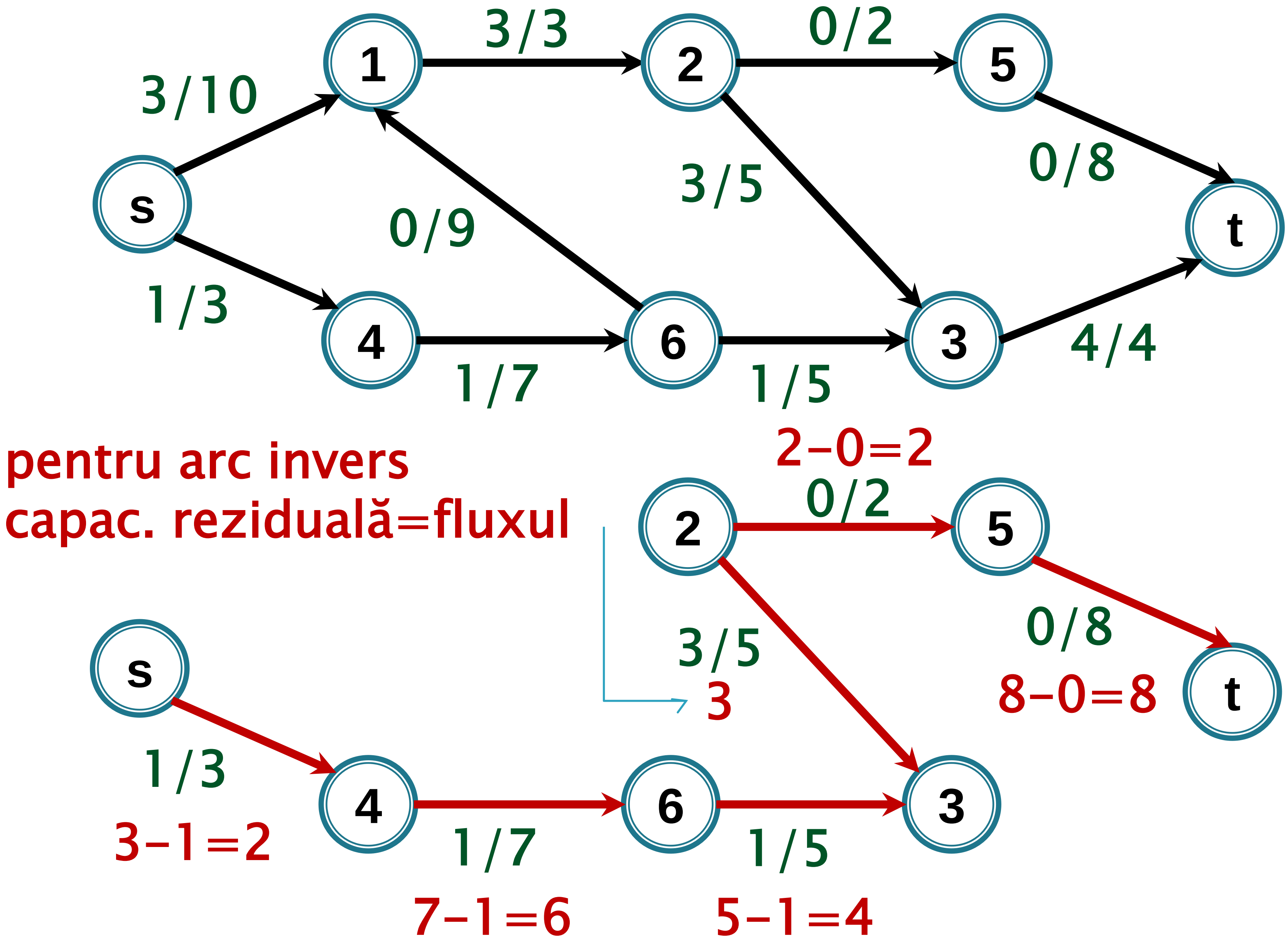
revizuieste_flux_lant



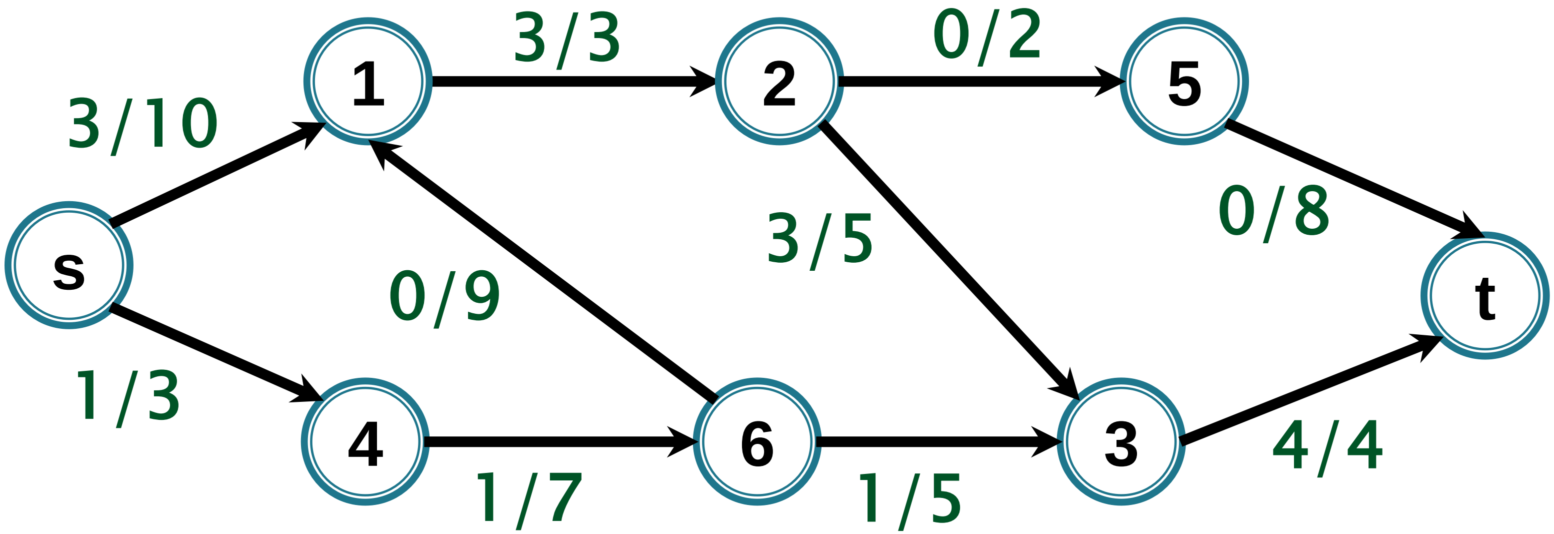
s
1
2
5
t
4
6
3
3/10
3/3
0/2
0/8
3/5
0/9
1/3
1/7
1/5
4/4

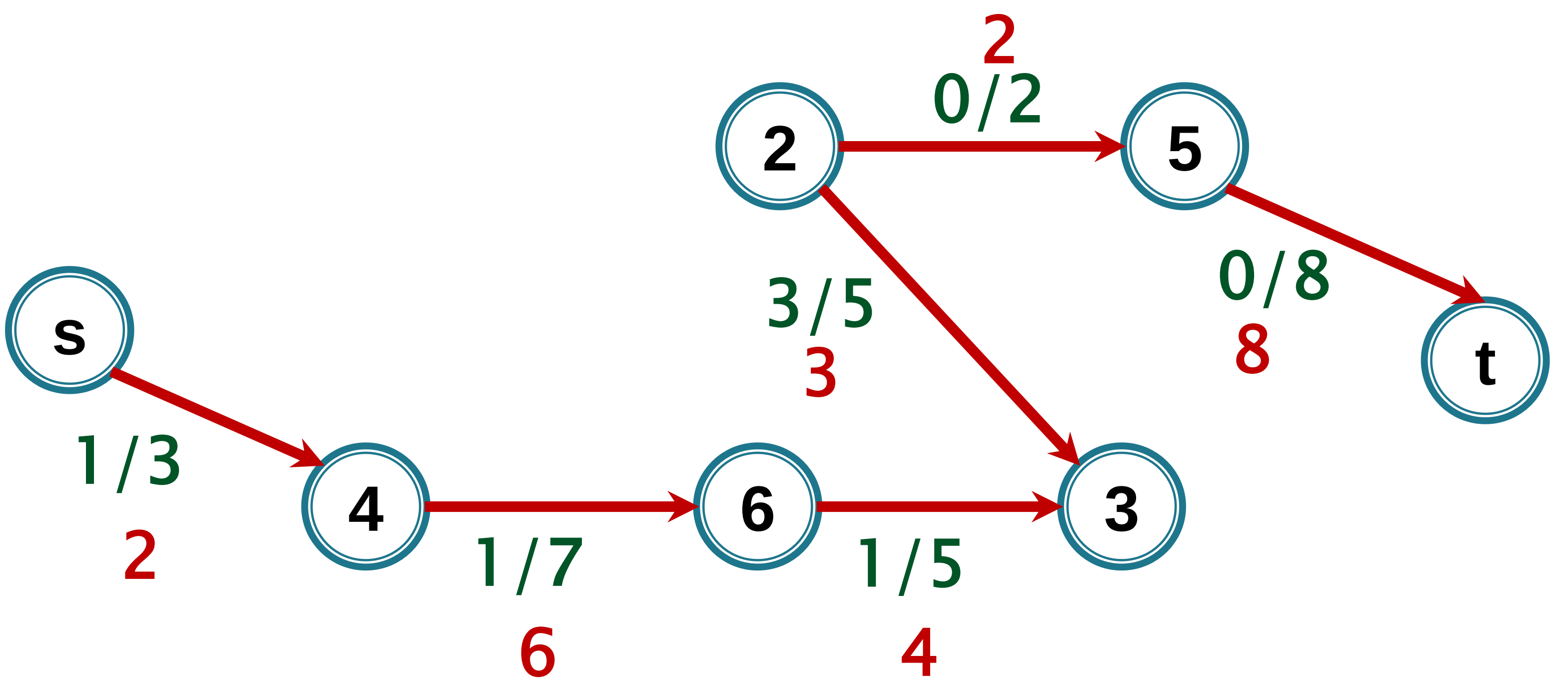


2
5
s
t
4
6
3
0/2
0/8
3/5
1/3
1/7
1/5

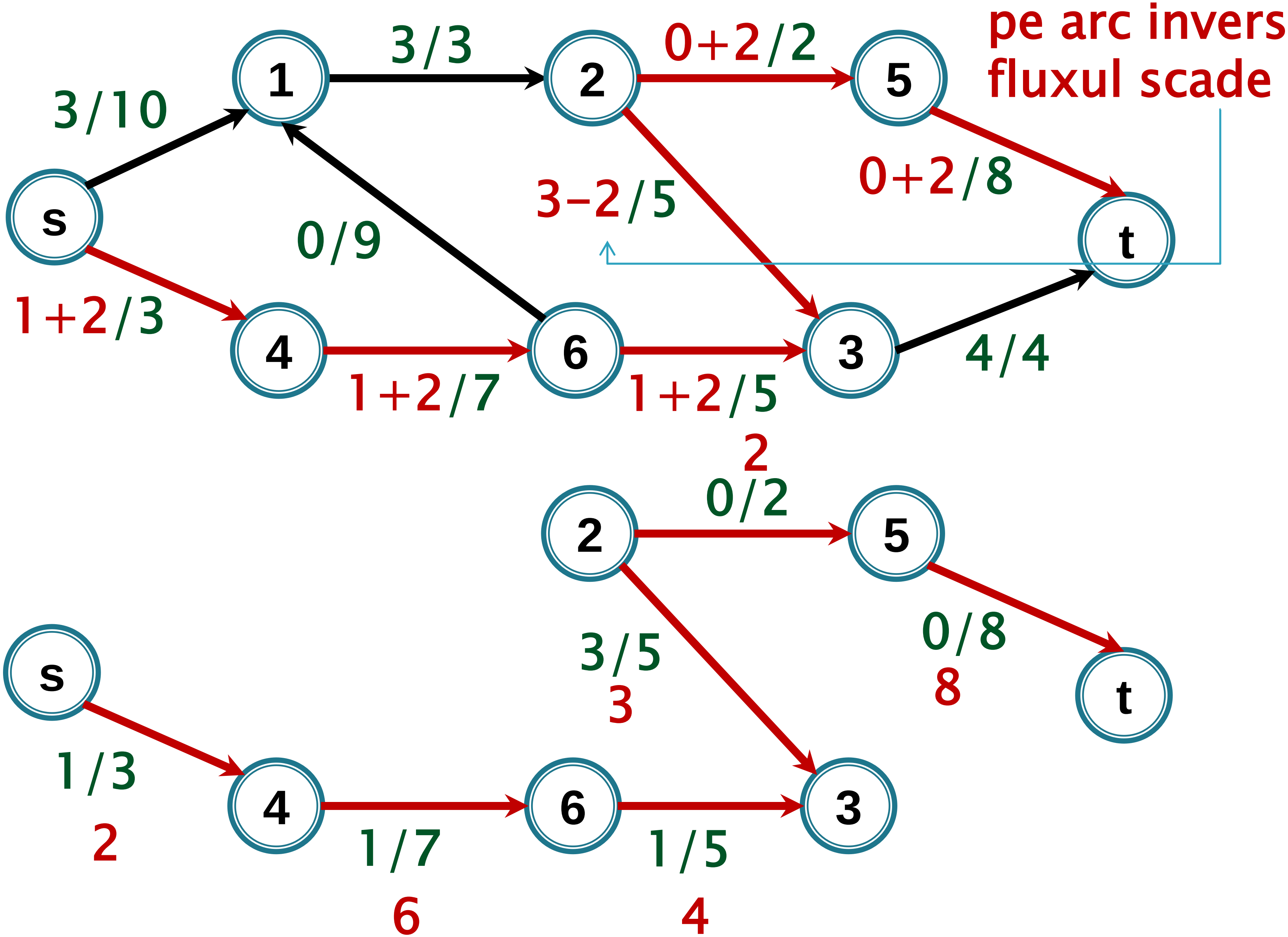


s
1
2
5
t
4
6
3
3/10
3/3
0/2
0/8
3/5
0/9
1/3
1/7
1/5
4/4
pentru arc invers
capac. reziduală=fluxul
2-0=2
0/2
3/5
3
0/8
8-0=8
1/3
3-1=2
1/7
7-1=6
1/5
5-1=4

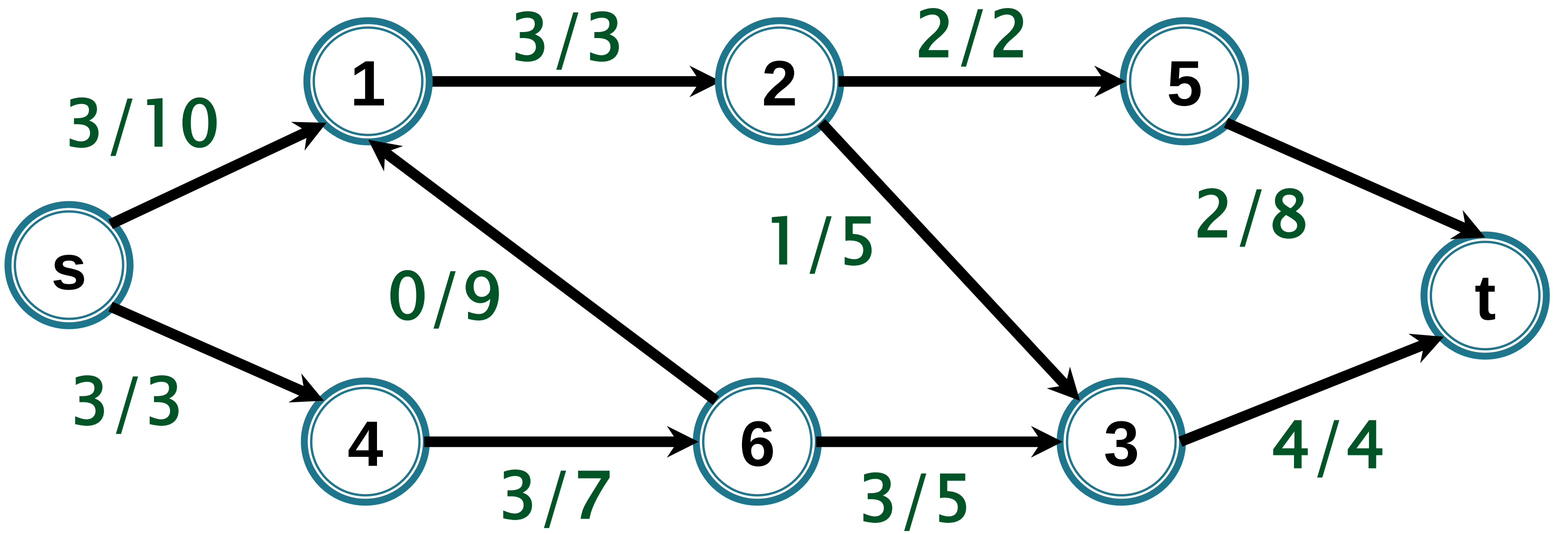




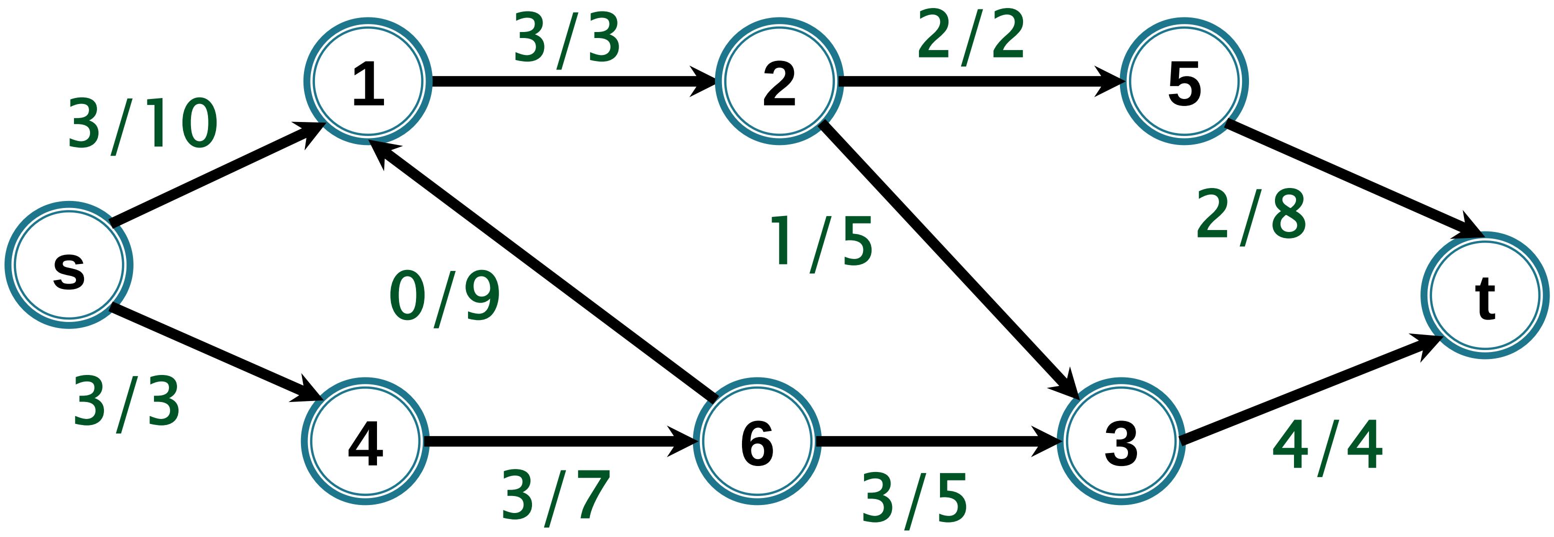
$$i(P) = \min\{2, 6, 4, 3, 2, 8\} = 2$$



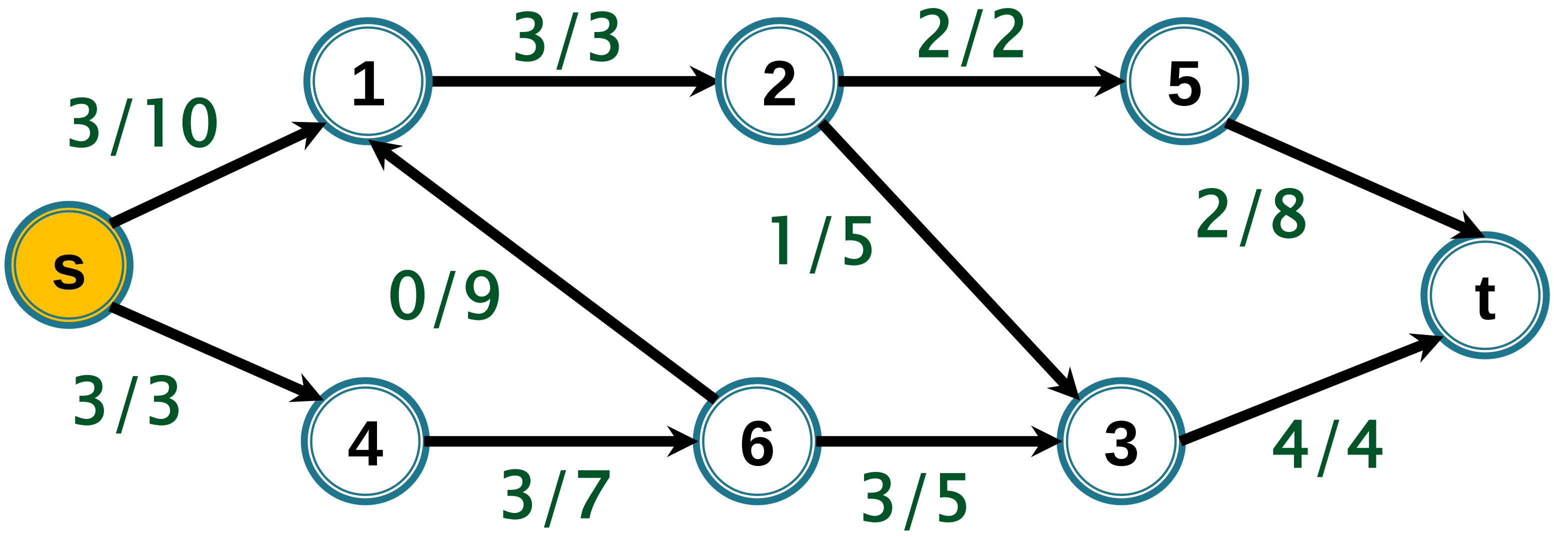
$$i(P) = \min\{2, 6, 4, 3, 2, 8\} = 2$$


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

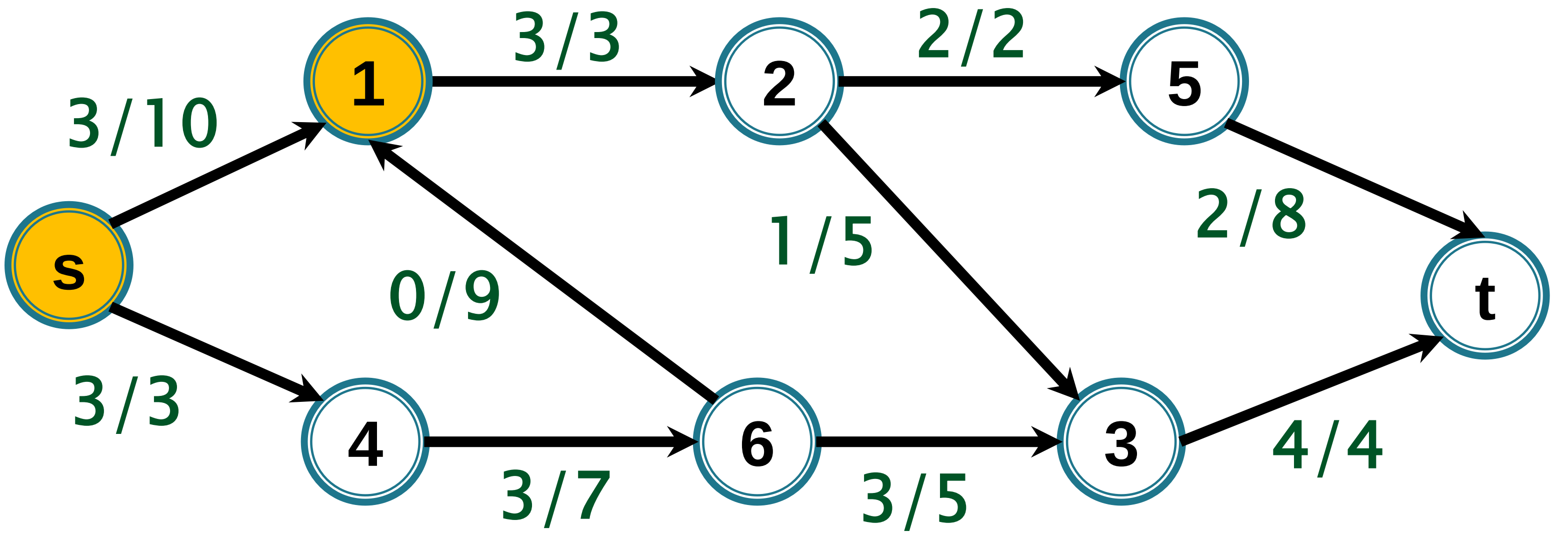
construieste_s-t_lant_nesat_BF



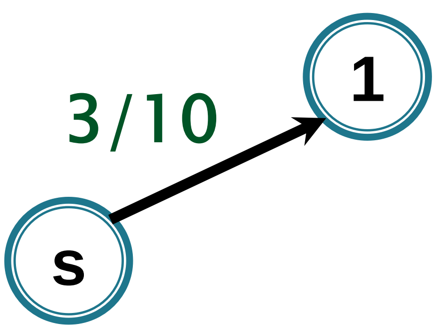
s
1
2
5
t
4
6
3
3/10
3/3
2/2
2/8
1/5
0/9
3/3
3/7
3/5
4/4


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

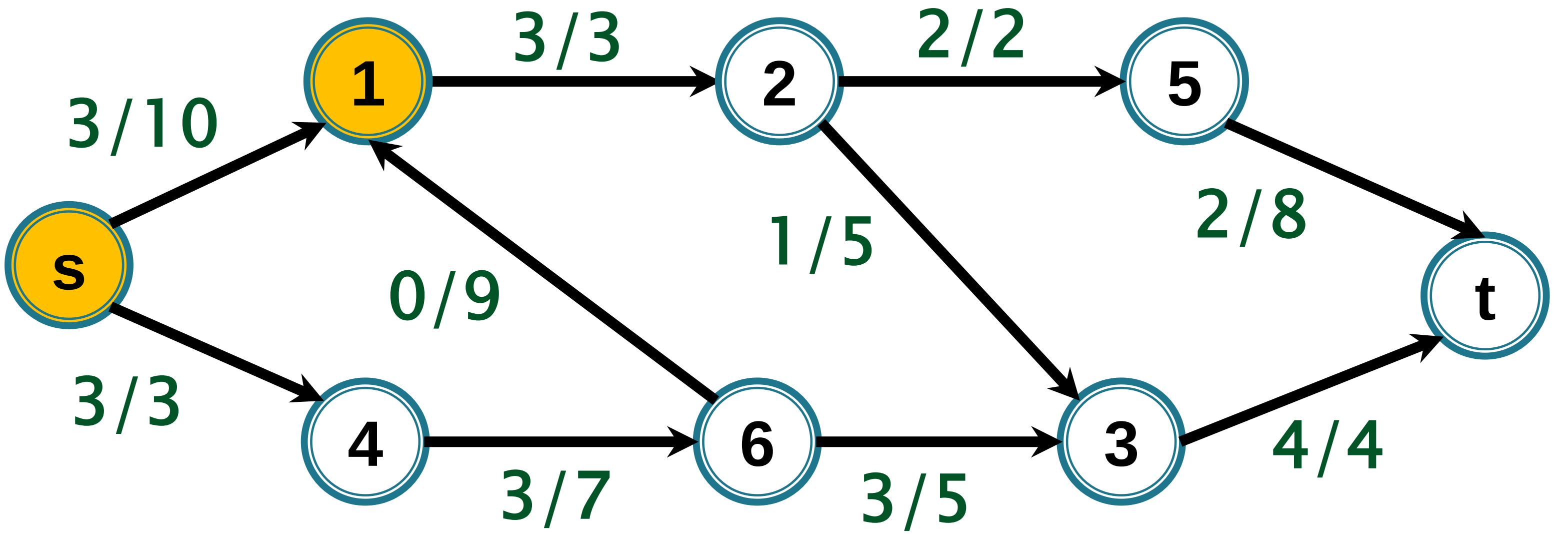

s

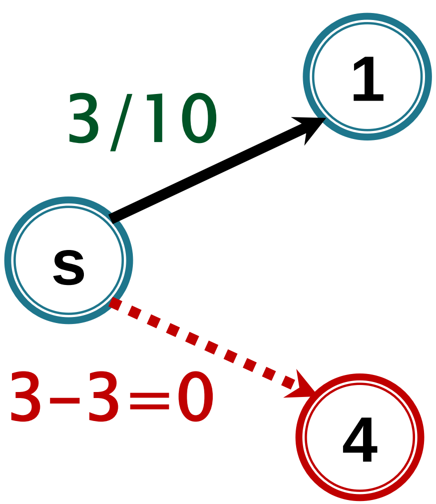


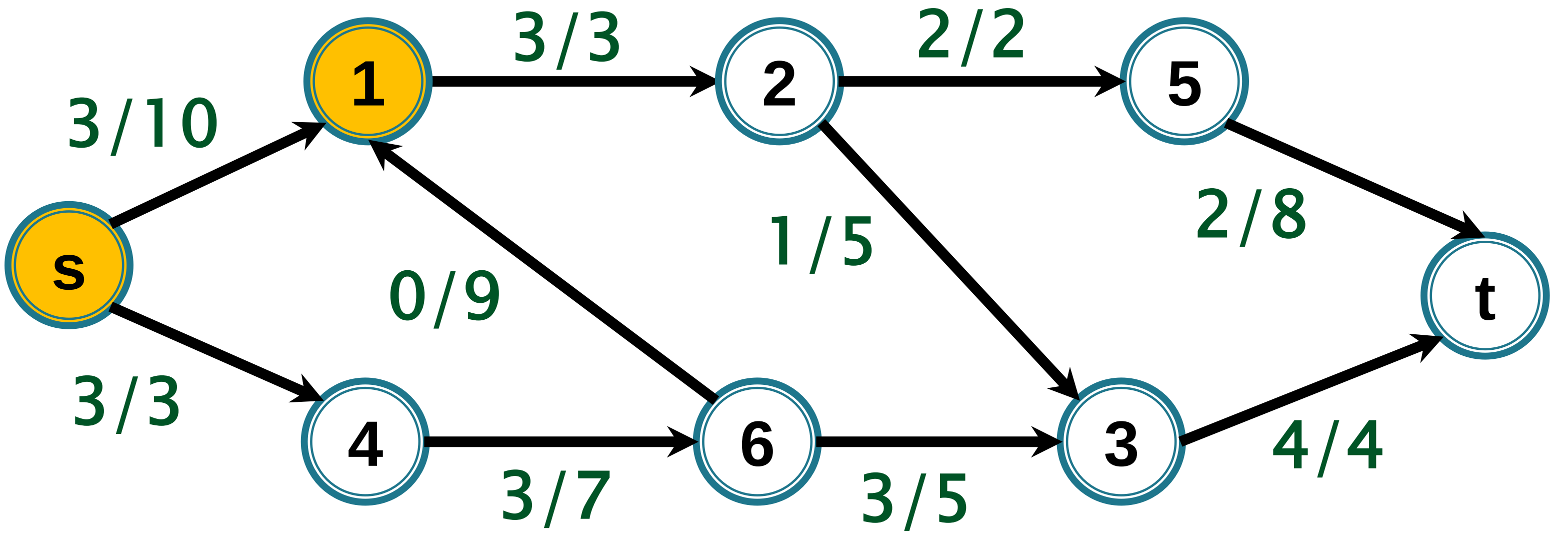
s
1
2
5
t
4
6
3
3/10
3/3
2/2
2/8
1/5
0/9
3/3
3/7
3/5
4/4

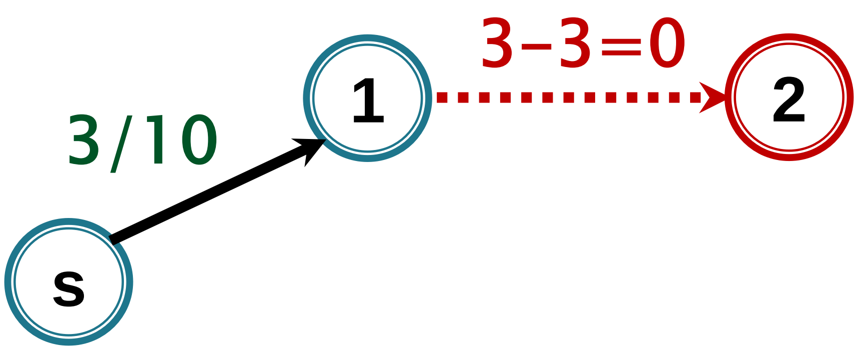


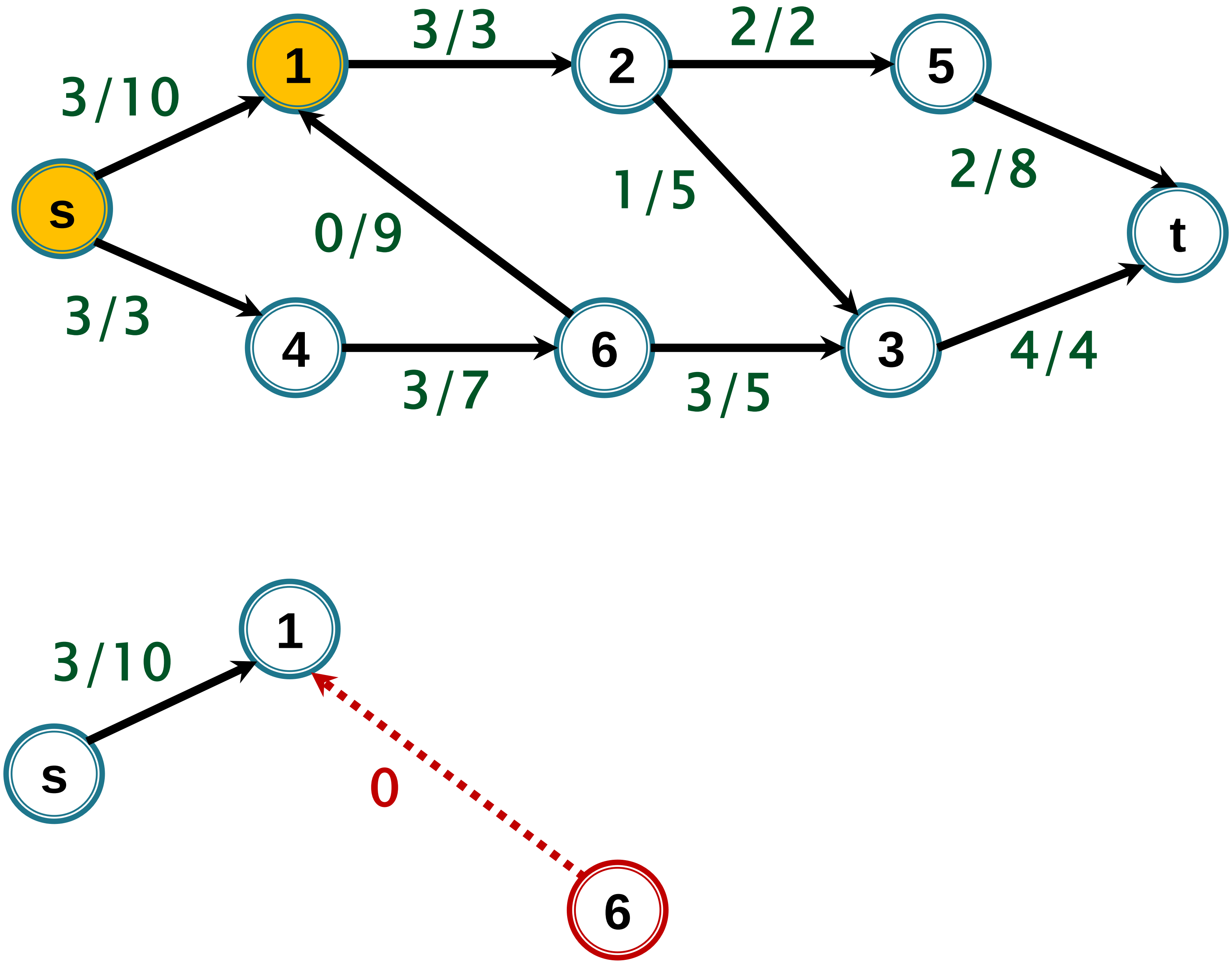
s
1
3/10


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


1
s
4
3/10
3–3=0


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


3/10
3−3=0
s
1
2



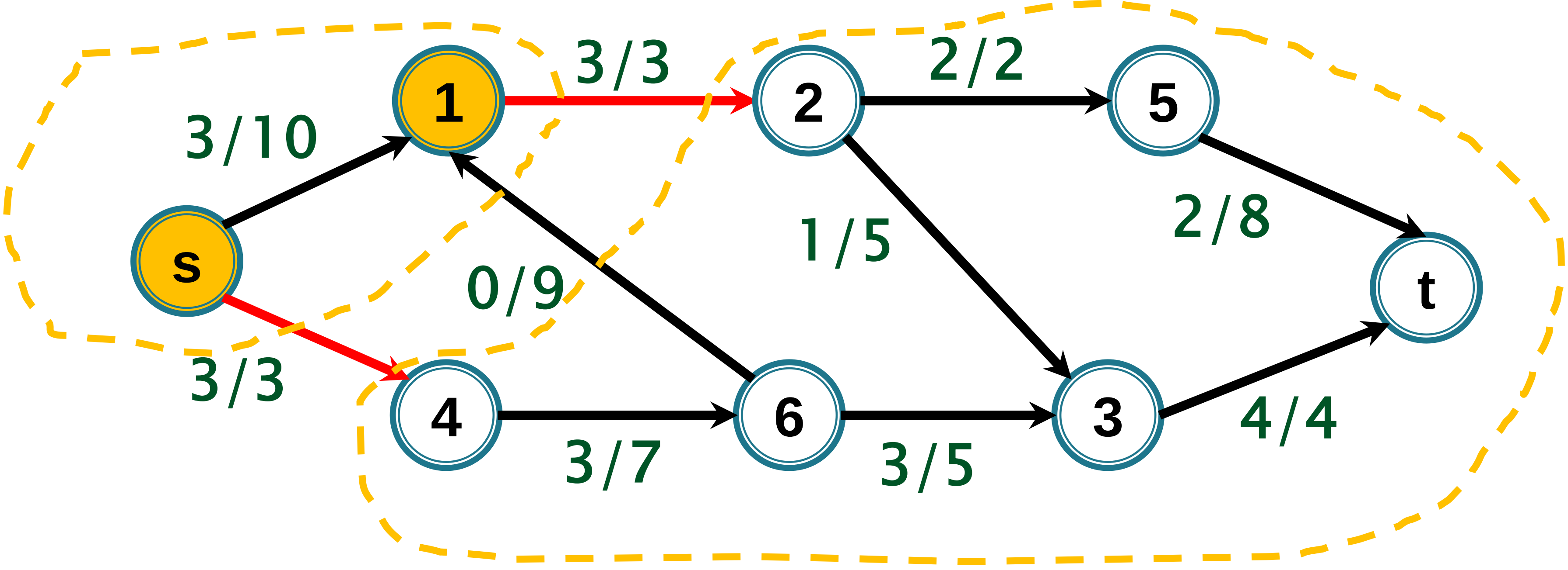
3/10
1
3/3
2
2/2
5
s
0/9
1/5
2/8
t
3/3
4
6
3
4/4
3/7
3/5
3/10
1
s
0
6

- t nu este accesibil din s $\Rightarrow$ STOP
  - f este flux maxim

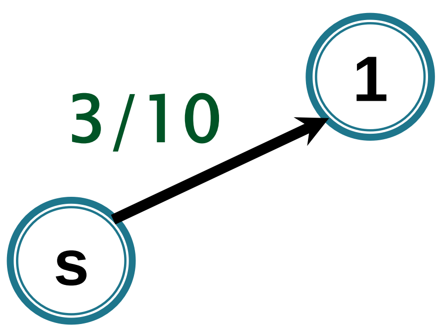
t nu este accesibil din s $\Rightarrow$ STOP

- 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**
- **<u>Convenție</u>** – 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

<u>Schema</u>:

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 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()

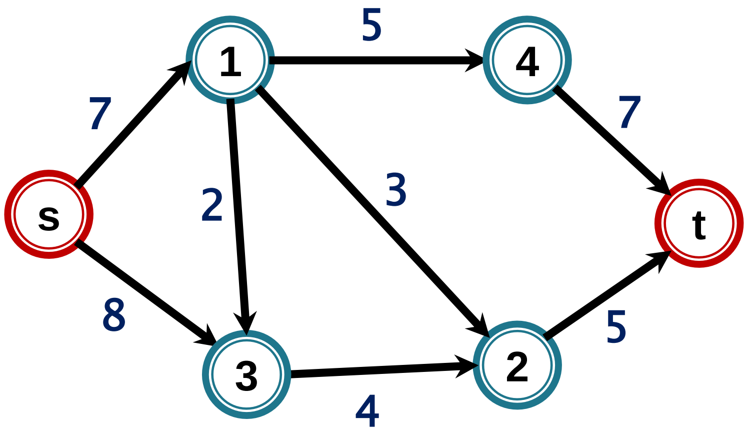
cat timp (**construieste_s-t_drum_in G_f_BF**()=true) executa

  revizuieste_flux_lant()

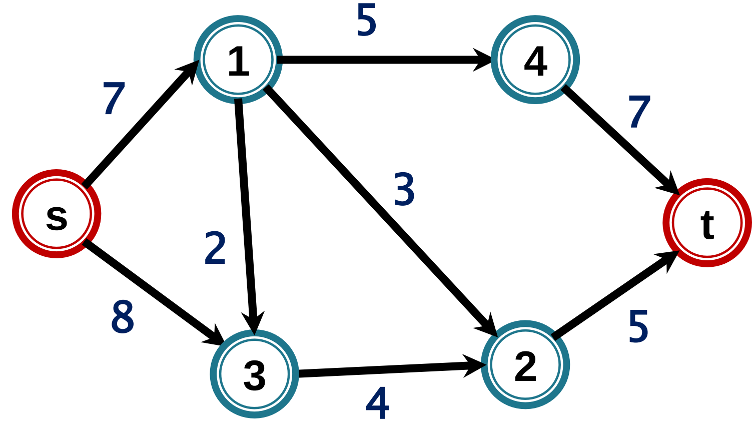
  actualizeaza G_f

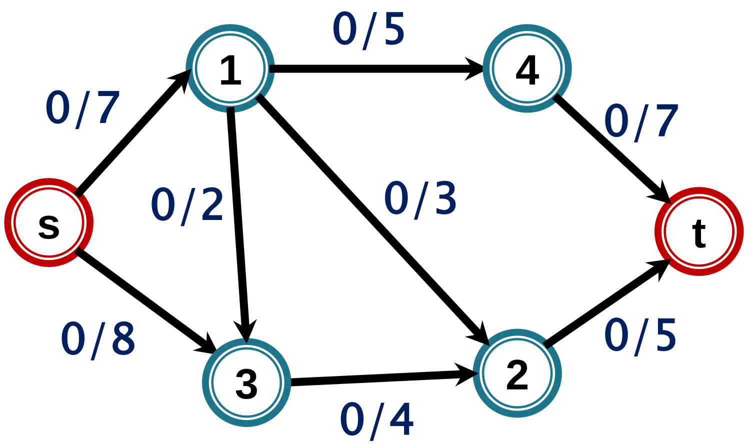
afiseaza_flux()

Detaliem această schemă

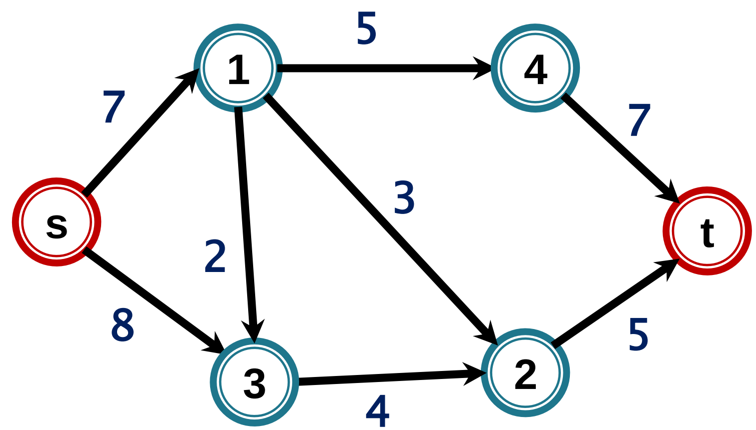


Graful rezidual G_f

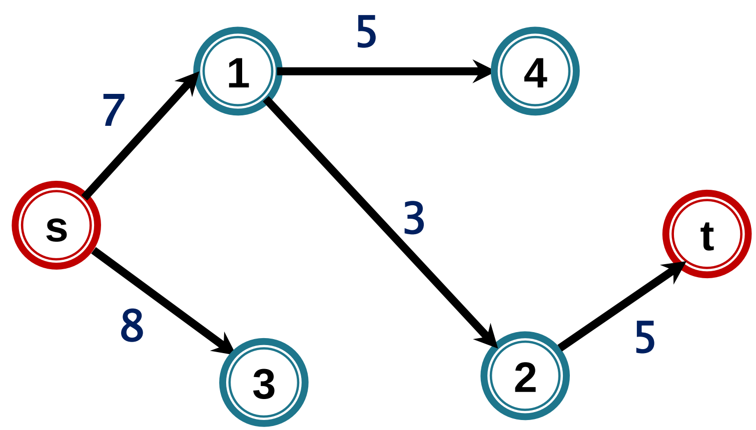


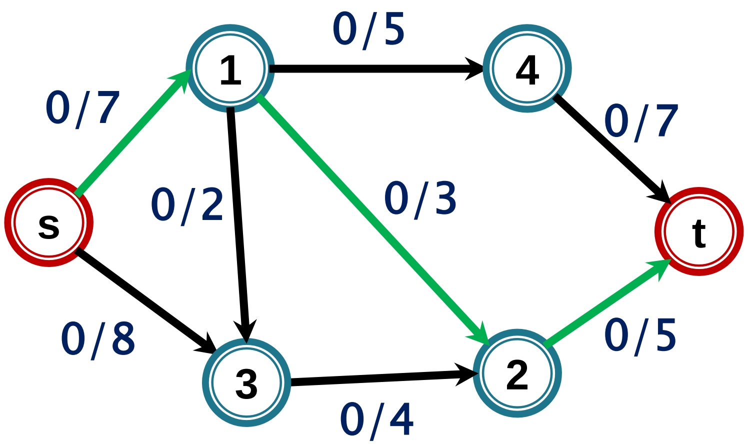


Graful rezidual G_f

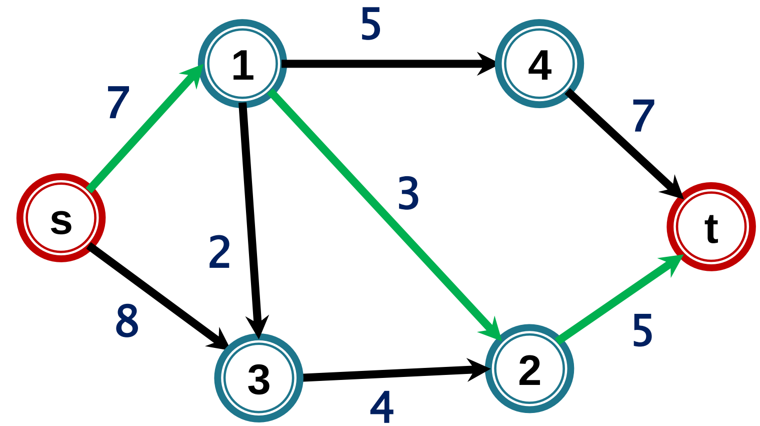


BF(s) – în graful rezidual

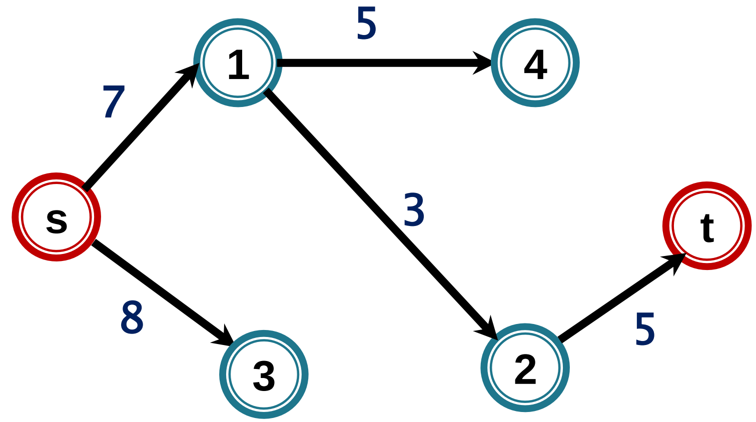




Graful rezidual G_f

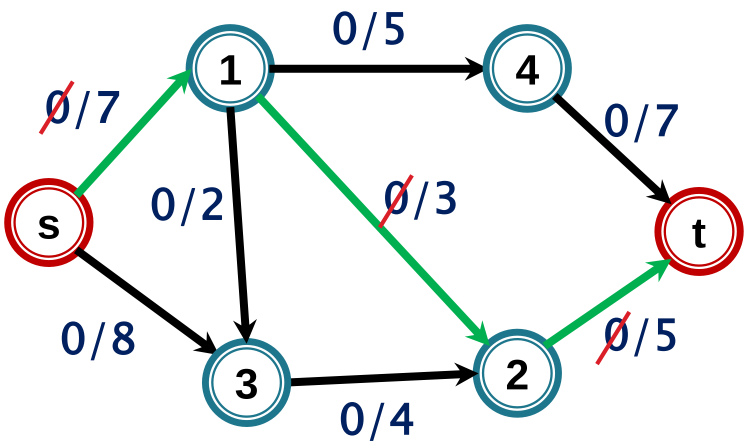


BF(s) – în graful rezidual

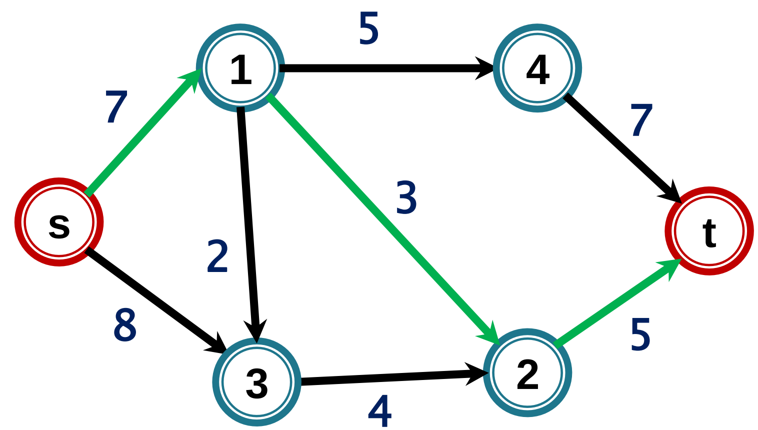


Drumul de creștere [s, 1, 2, t] – capacitate reziduală 3

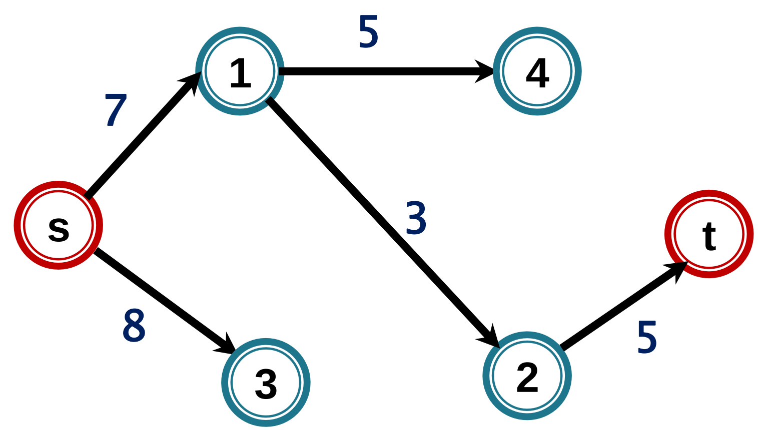
Revizuim fluxul



Graful rezidual G_f

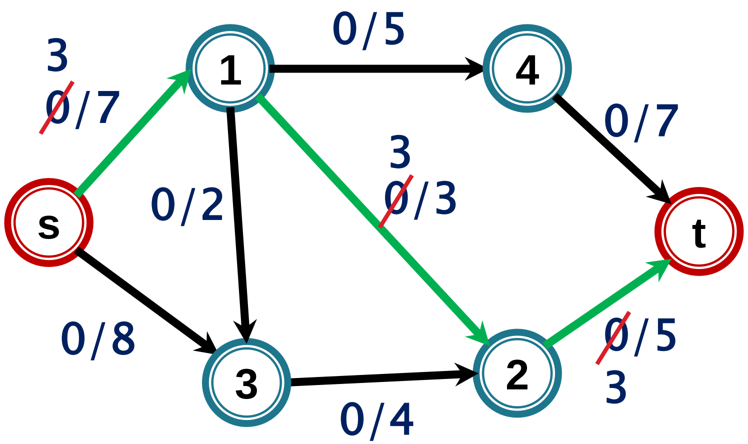


BF(s) – în graful rezidual

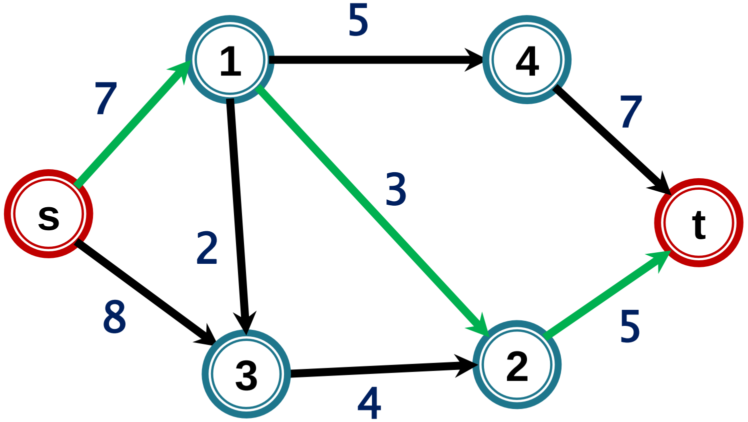


Drumul de creștere [s, 1, 2, t] – capacitate reziduală 3

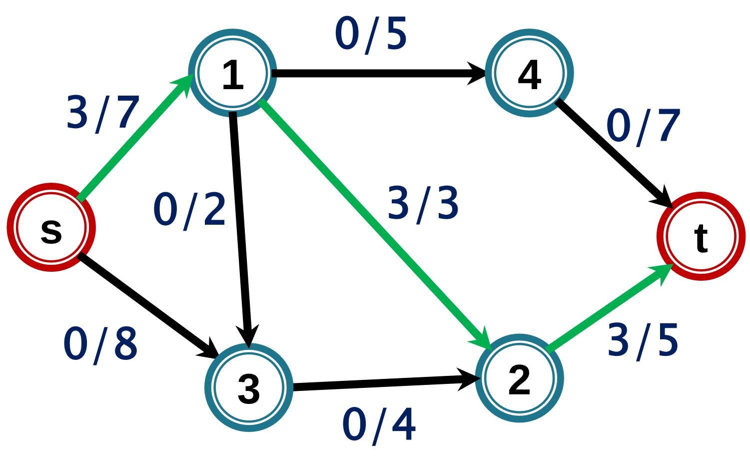
Revizuim fluxul



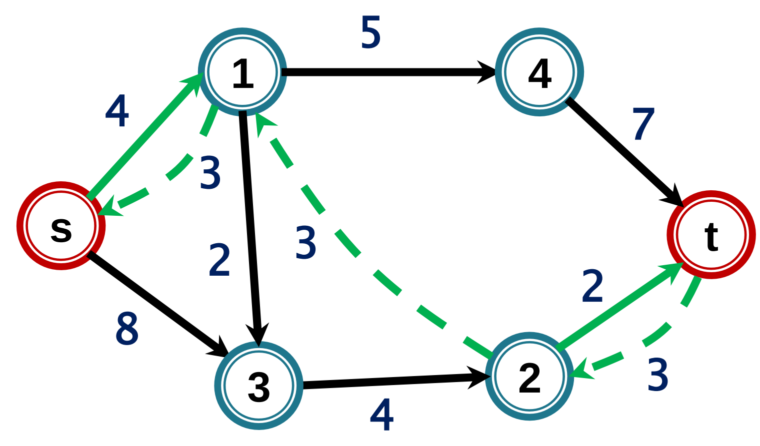
Graful rezidual G_f

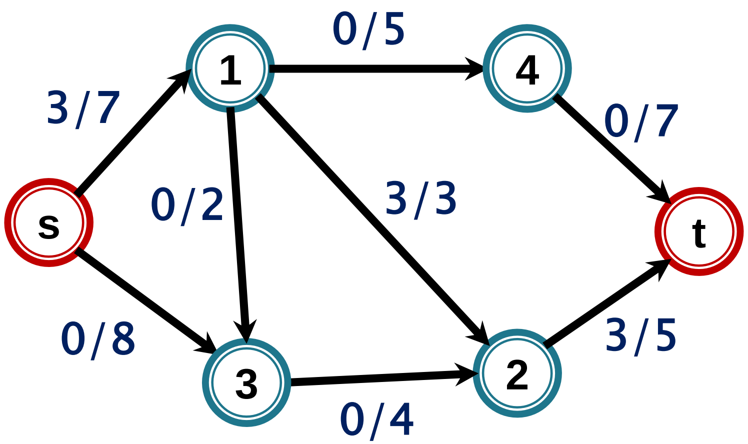


Actualizăm rețeaua reziduală

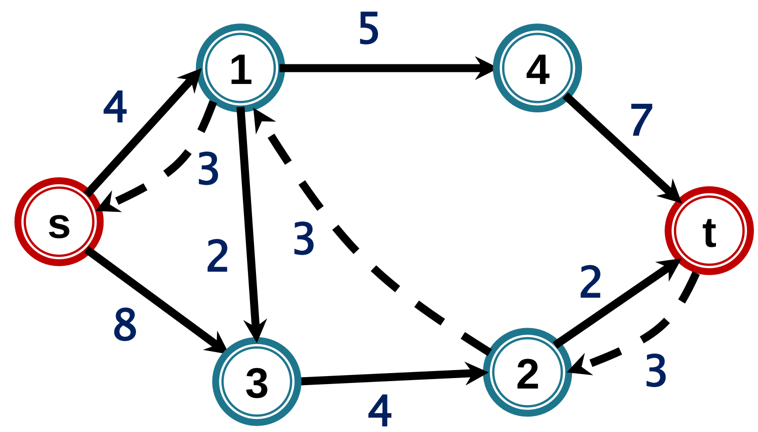


Graful rezidual G_f

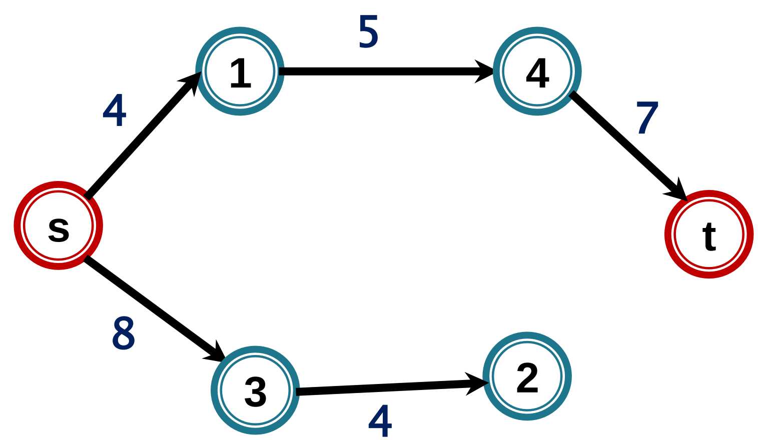




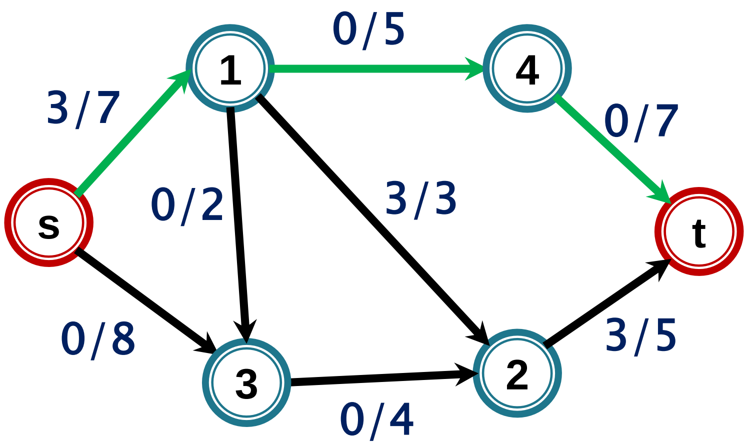
Graful rezidual G_f

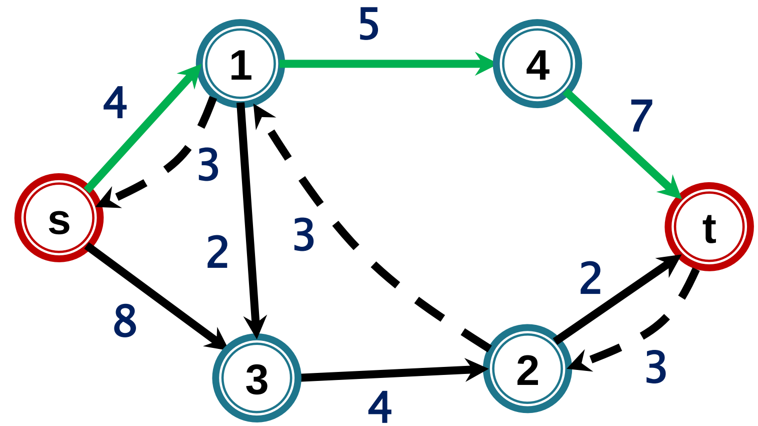


BF(s) – în graful rezidual

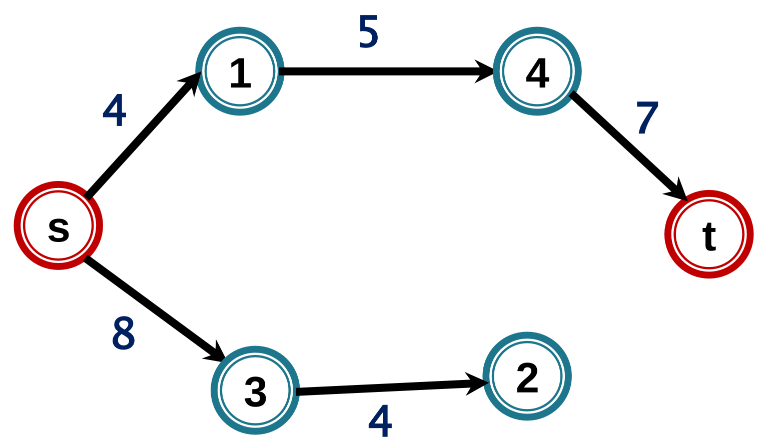


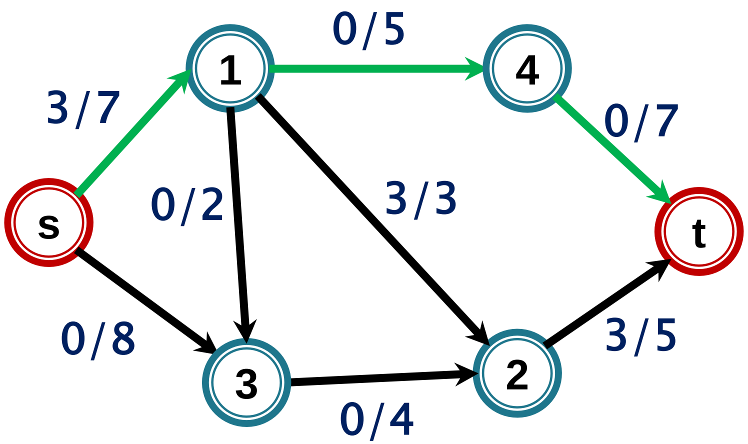
Graful rezidual G_f



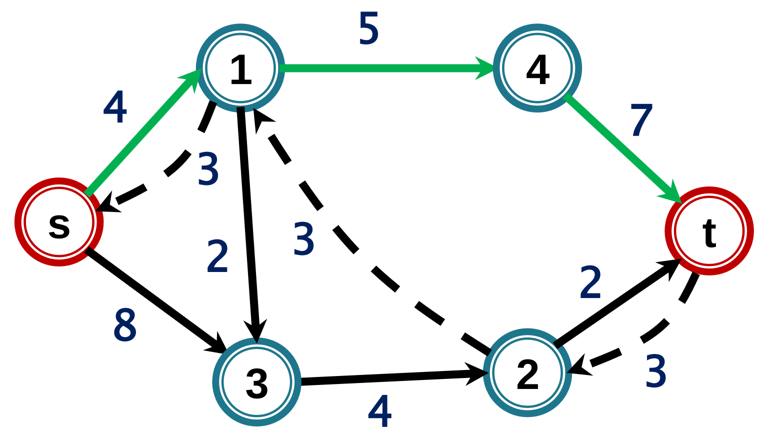


BF(s) – în graful rezidual

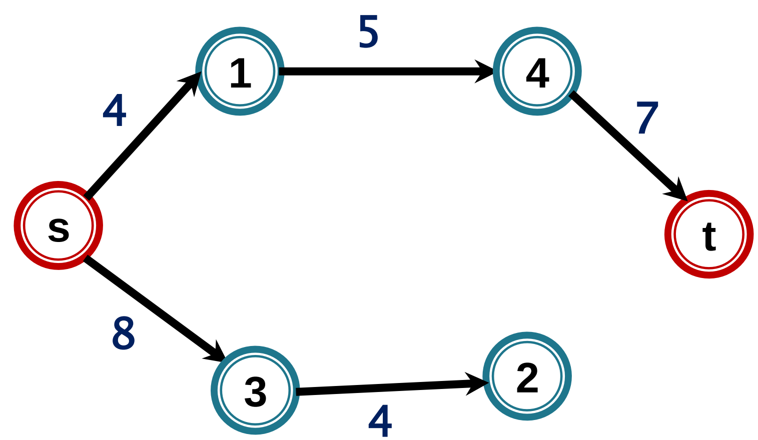




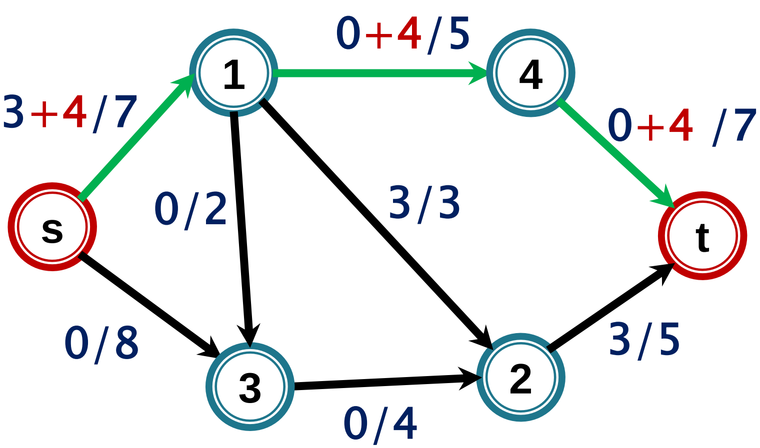
Graful rezidual G_f



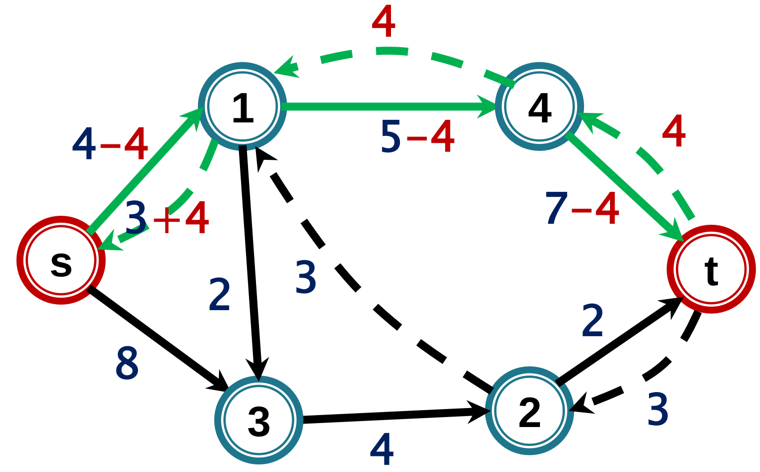
BF(s) – în graful rezidual

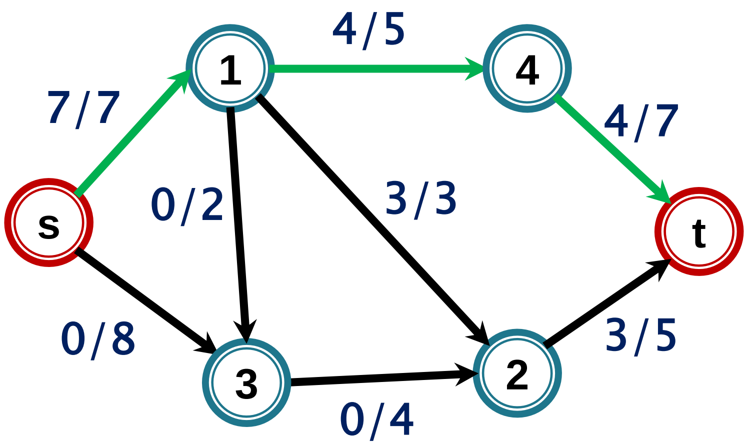


Drumul de creștere [s, 1, 4, t] – capacitate reziduală 4

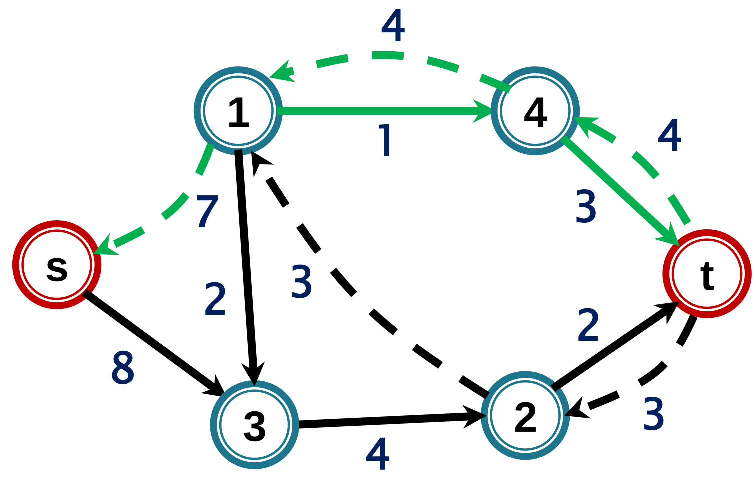


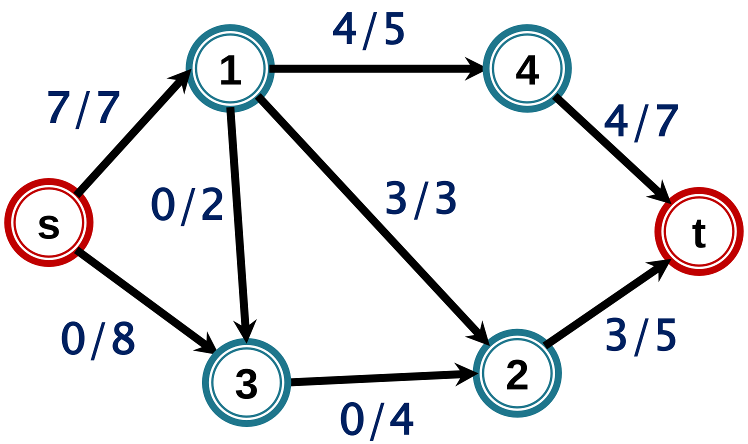
Graful rezidual G_f



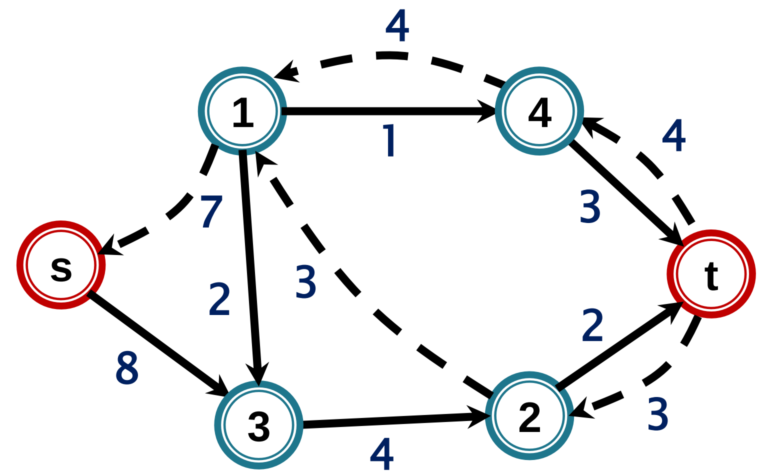


Graful rezidual G_f

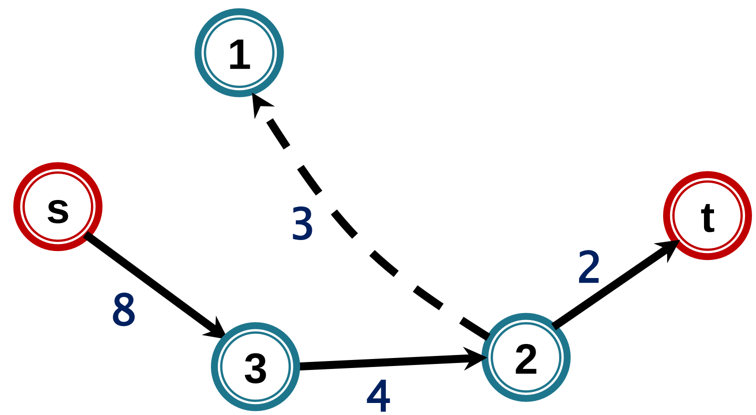


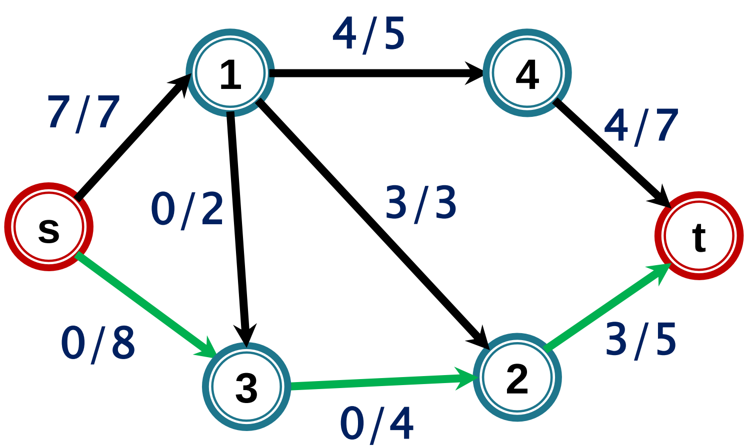


Graful rezidual G_f

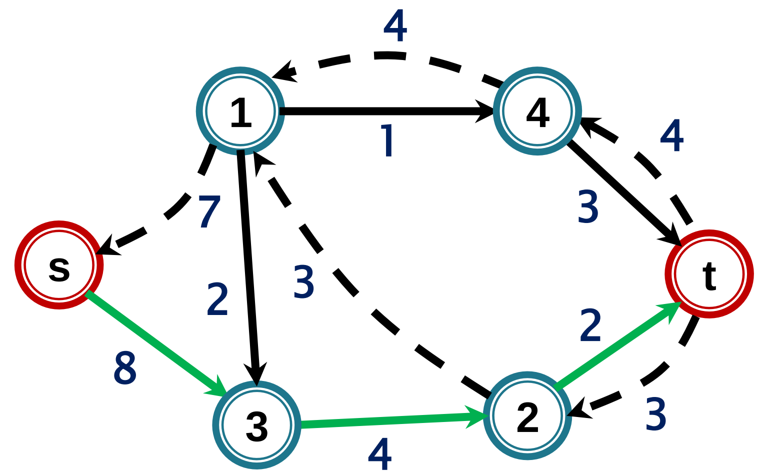


BF(s) – în graful rezidual

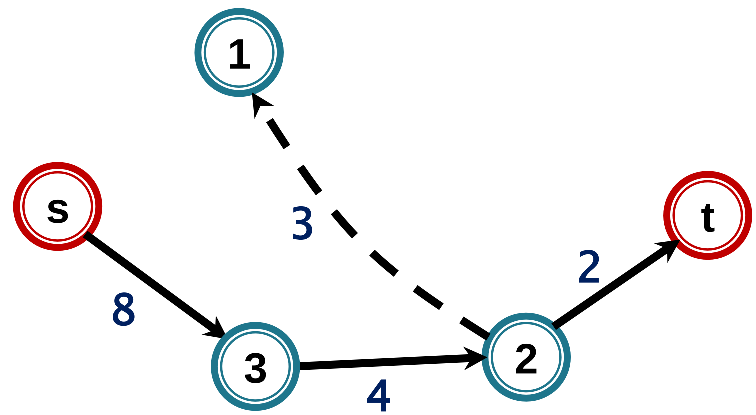




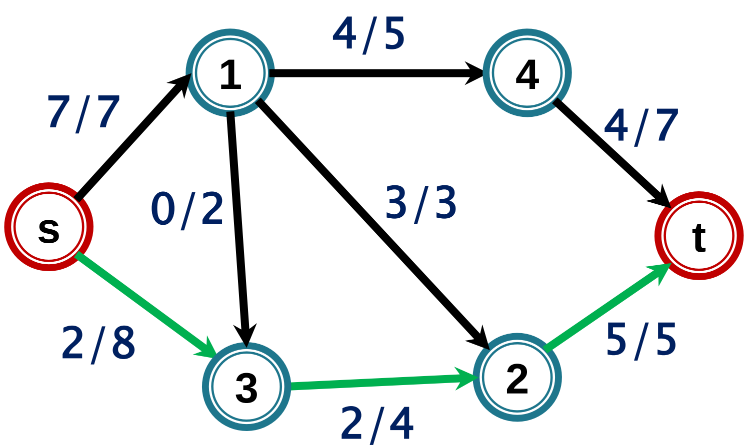
Graful rezidual G_f



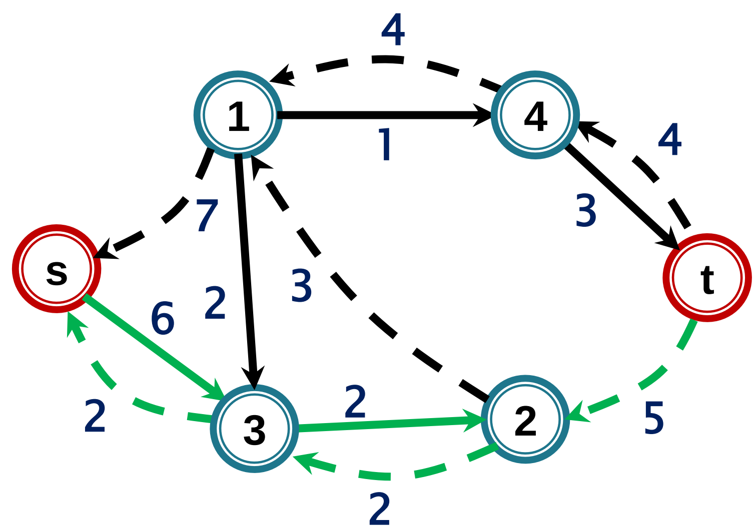
BF(s) – în graful rezidual

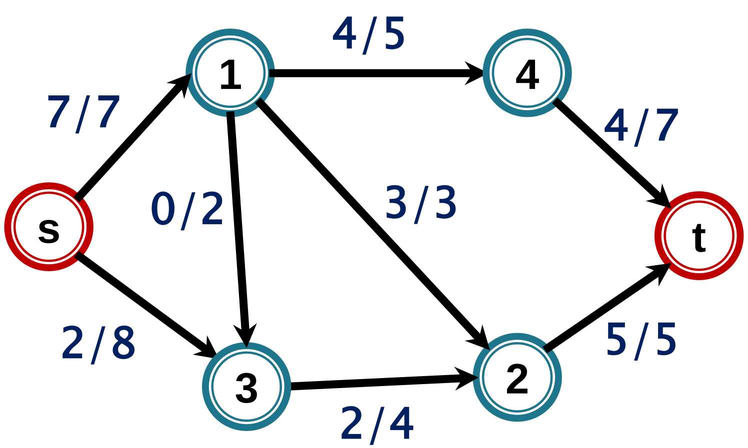


Drumul de creștere [s, 3, 2, t] – capacitate reziduală 2

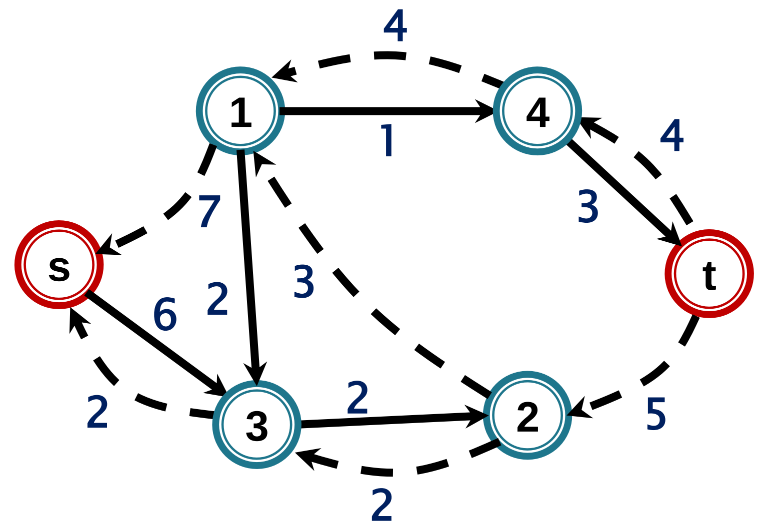


Graful rezidual G_f

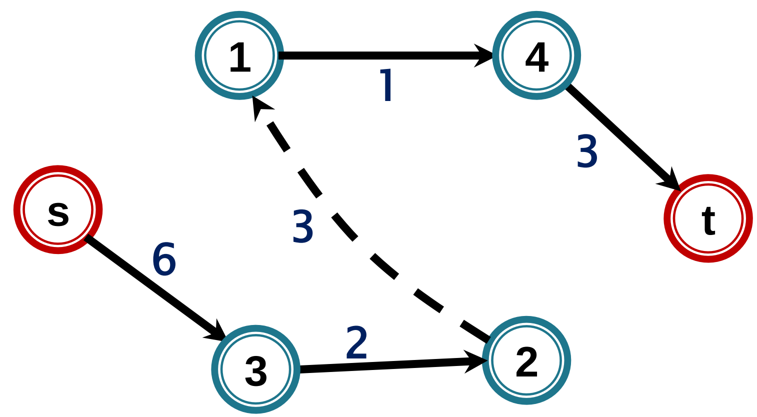


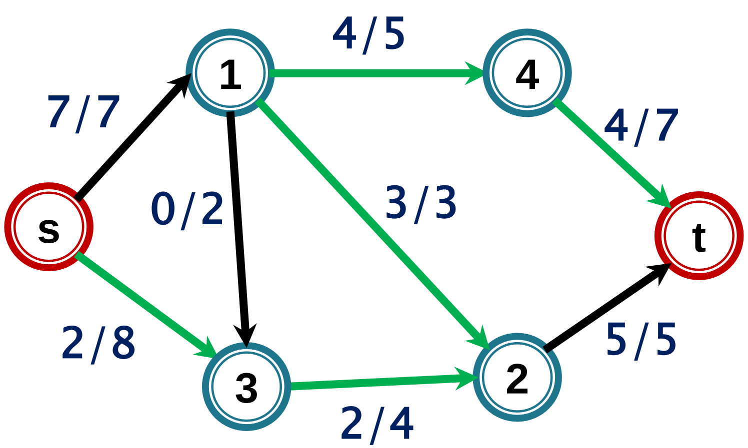


Graful rezidual G_f

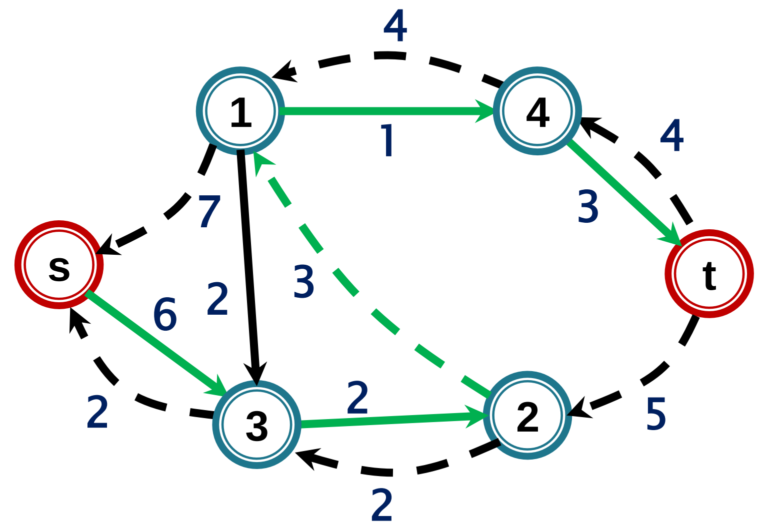


BF(s) - pe graful rezidual

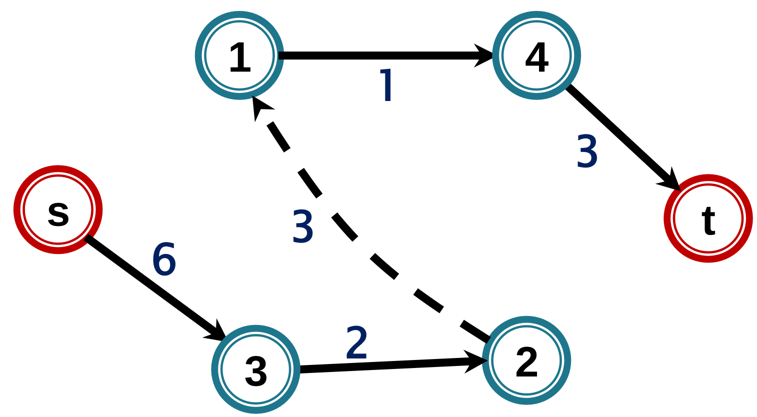




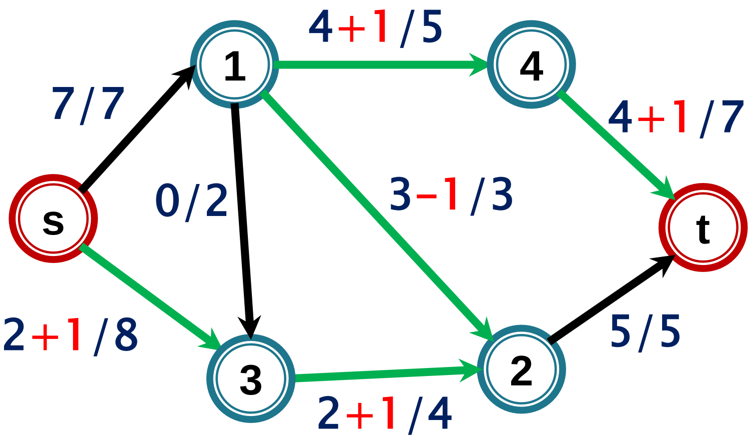
Graful rezidual G_f



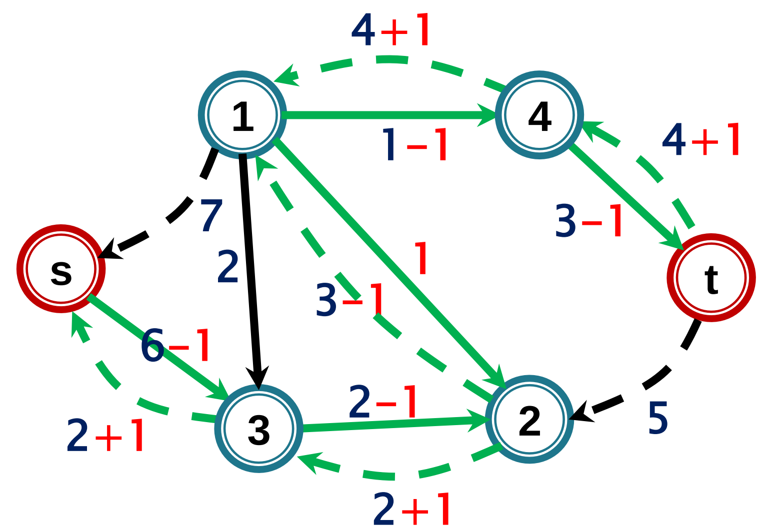
BF(s) – în graful rezidual



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



Graful rezidual G_f



actualizare G_f:

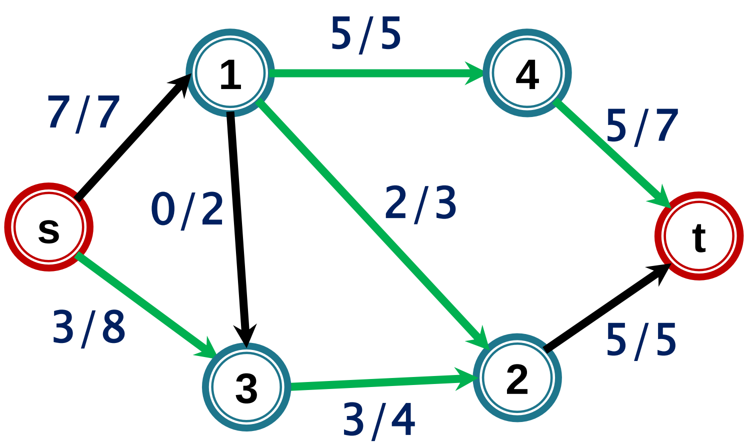
pentru e $\in$ E(P) $\subseteq$ E(G_f)

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

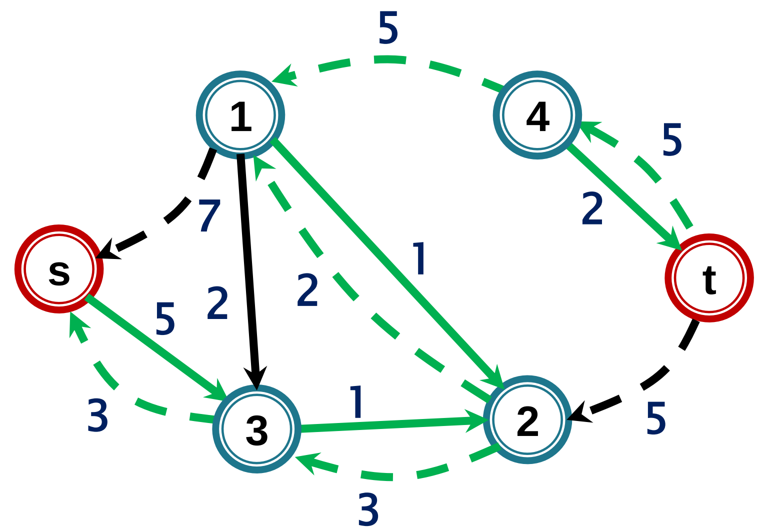
dacă $c_f(e)=0$ elimina e din G_f (se ignora in parcurgere)

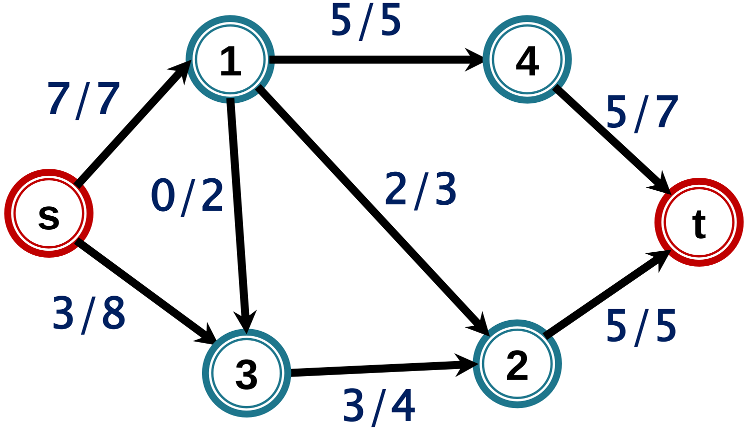
$c_f(e^{-1}) \leftarrow c_f(e^{-1})$ + cfP

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

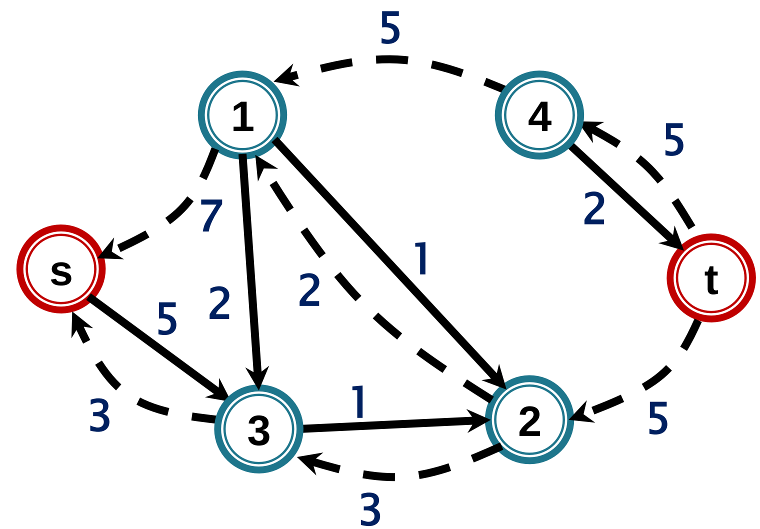


Graful rezidual G_f

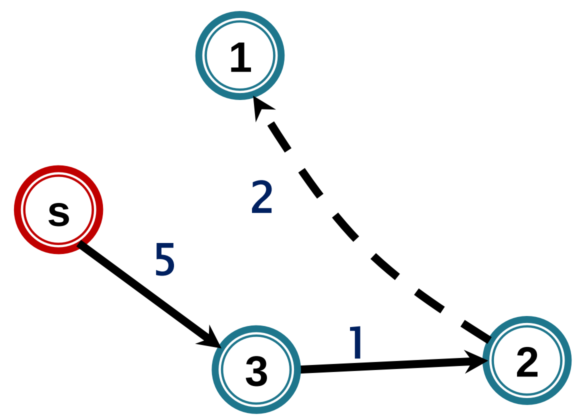


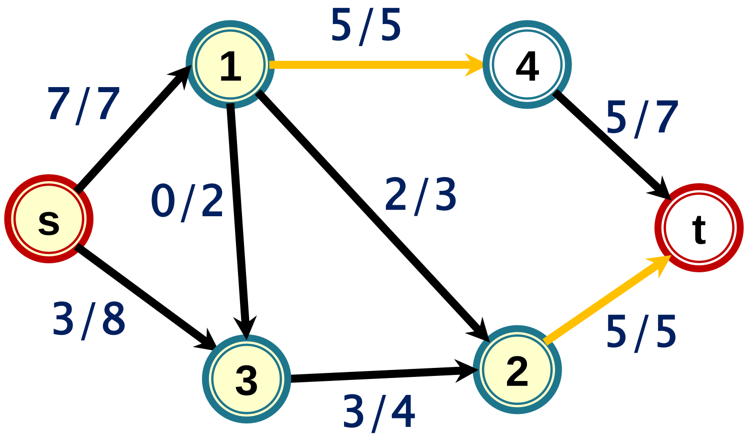


Graful rezidual G_f

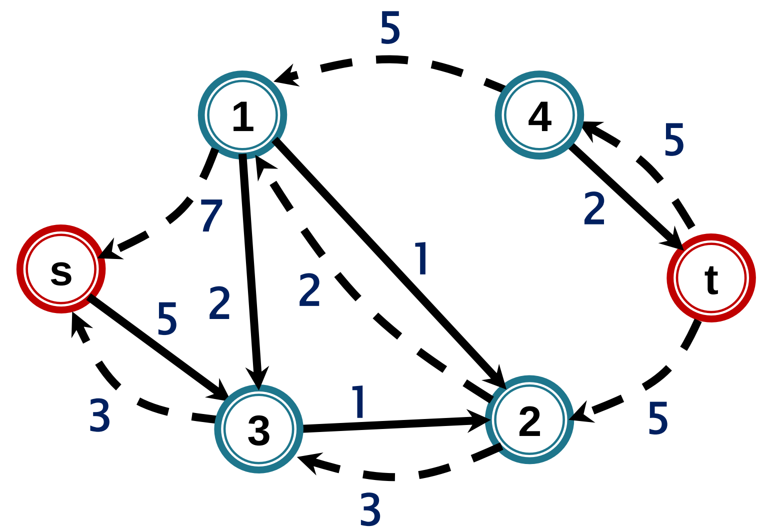


BF(s) – în graful rezidual

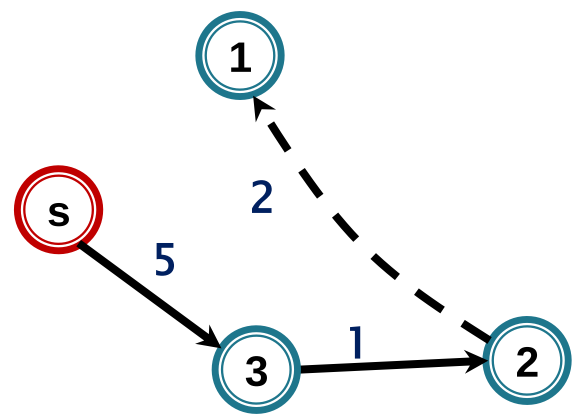




Graful rezidual G_f



BF(s) – în graful rezidual



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 D_f : S = {s, 1, 3, 2})