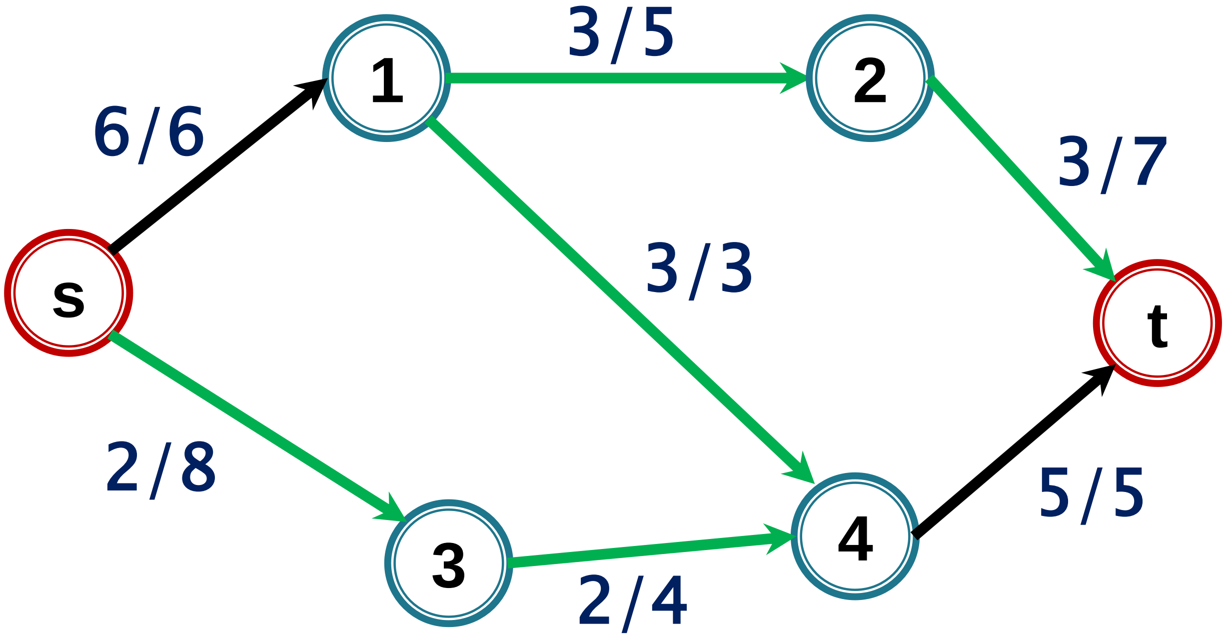
Algoritmul FORD-FULKERSON de determinare a unui flux maxim

+ a unei tăieturi minime

Algoritmul Ford-Fulkerson

Amintim:

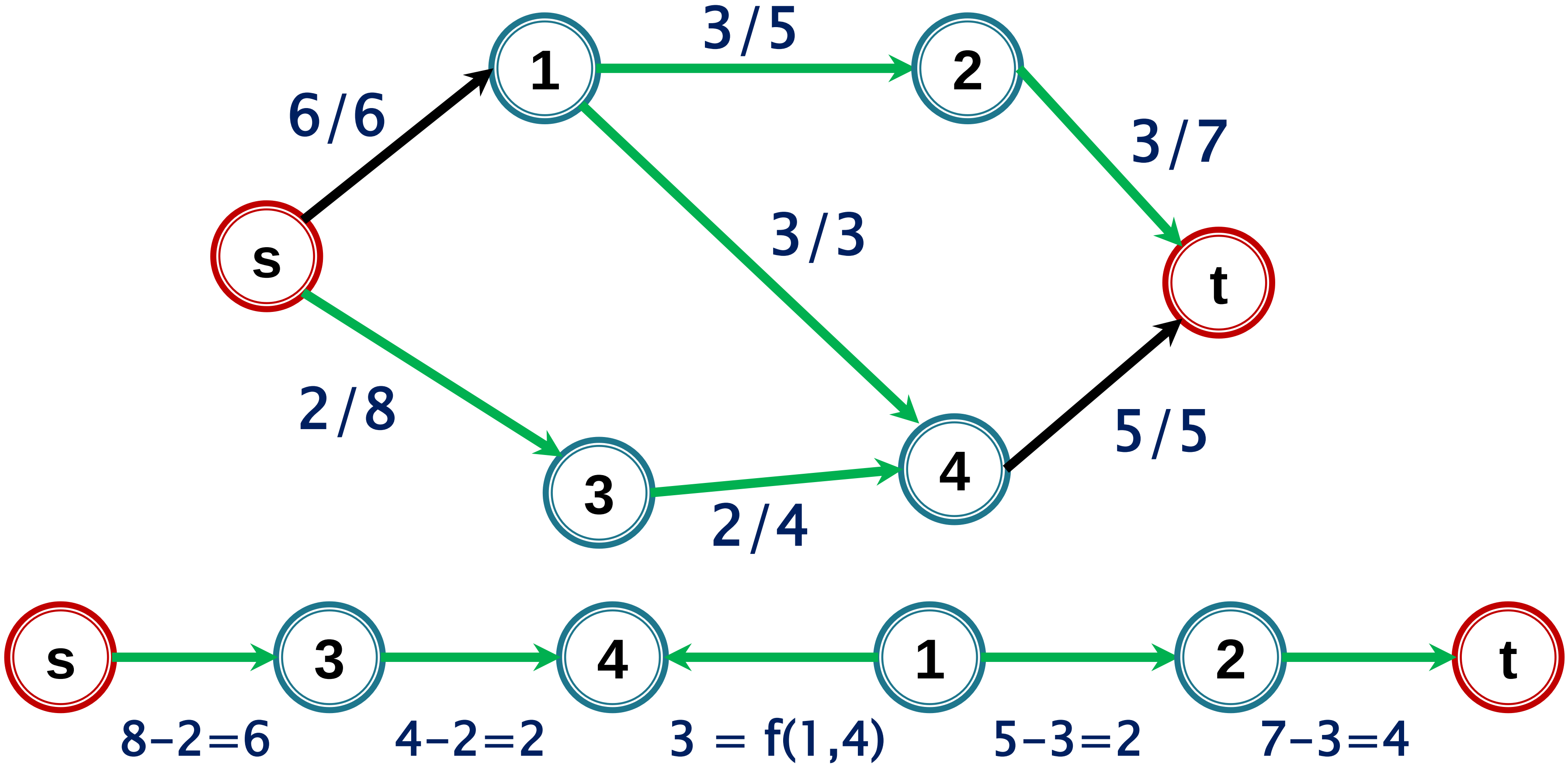
- **s-t lanț f-nesaturat**
  - arc direct
  - arc invers
  - capacitate reziduală arc, lanţ
- Operația de **revizuire a fluxului** de-a lungul unui s-t lanț *f-nesaturat*
- **Tăietură în rețea**
  - capacitatea unei tăieturi



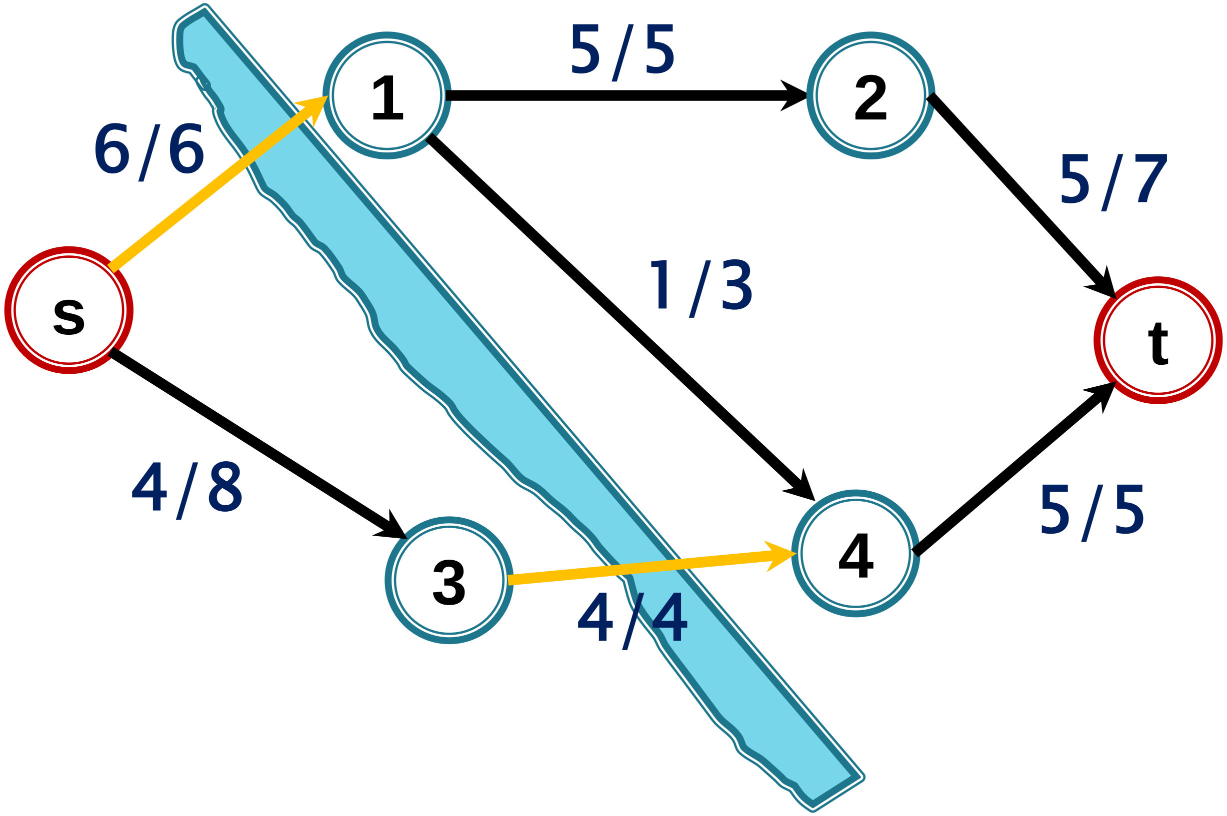
arc în sens invers,
putem trimite înapoi 3 unități de flux

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

capacități reziduale



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



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 (**tăietură**)

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

- Fie $f \equiv 0$ 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

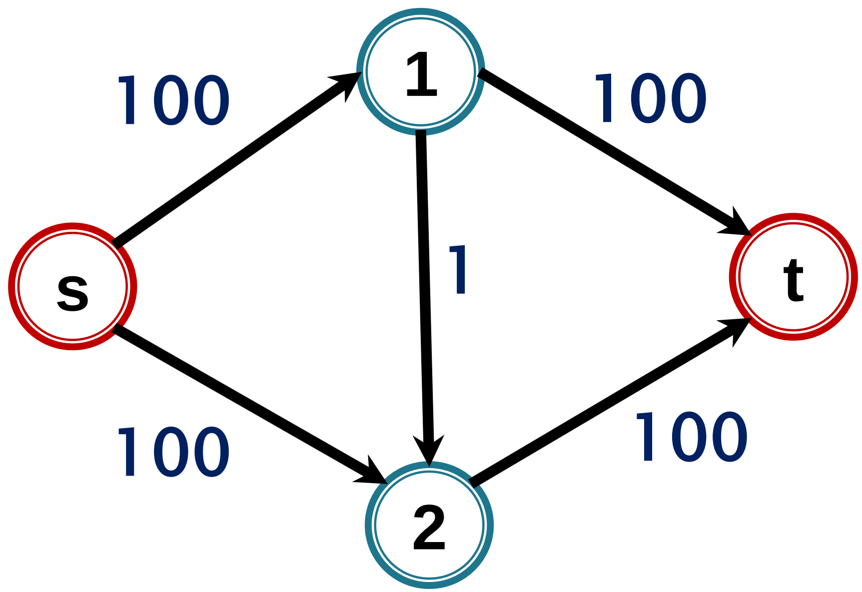
Algoritmul FORD-FULKERSON

Complexitate

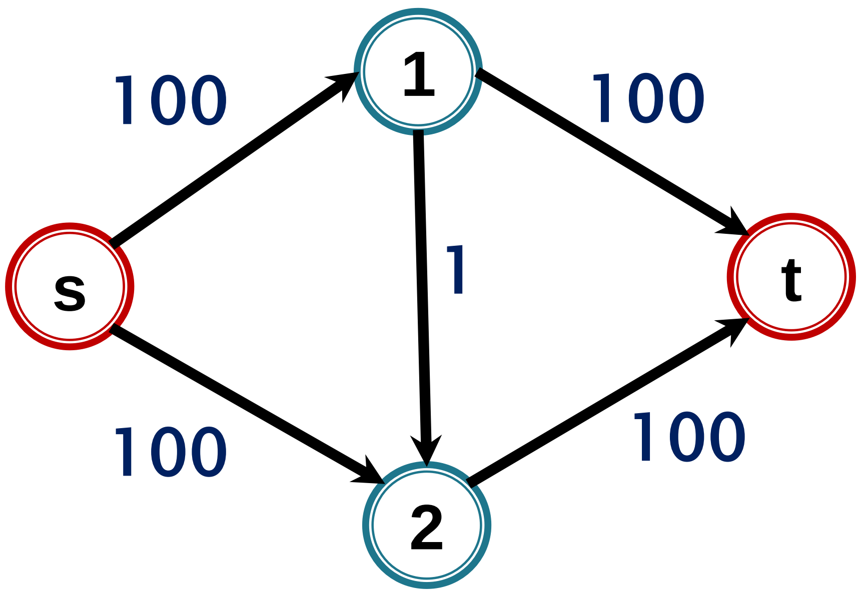
Algoritmul Ford-Fulkerson

- 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)?

Criteriul după care construim lanţul f-nesaturat influenţează numărul de etape



Criteriul după care construim lanţul f-nesaturat influenţează numărul de etape



Pasul 1: [s, 1, 2, t] – i(P)=1

Pasul 2: [s, 2, 1, t] – i(P)=1

Pasul 3: [s, 1, 2, t] – i(P)=1

Pasul 4: [s, 2, 1, t] – i(P)=1

...

Algoritm FORD – FULKERSON

- Complexitate

Algoritm FORD – FULKERSON

- Complexitate O(mL), unde

$$L = \sum_{su \in E} c(su)$$

Algoritmul FORD-FULKERSON

Corectitudine

Algoritmul Ford-Fulkerson

- **Fluxul determinat de algoritm are valoare maximă, sau putem determina un flux de valoare mai mare prin alte metode?**

  - Trebuie să arătăm că

    $\nexists$ s-t lanţ f-nesaturat $\Rightarrow$ f flux maxim

Algoritmul Ford-Fulkerson

- Vom demonstra că
  - val(f) $\leq$ c(K) pentru orice f flux, K tăietură
  -

Algoritmul Ford-Fulkerson

- Vom demonstra că
  - $val(f) \le c(K)$ pentru orice f flux, K tăietură
  - $\nexists$ s-t lanţ f-nesaturat $\Rightarrow$ $\exists$ K cu $val(f) = c(K)$ $\Rightarrow$

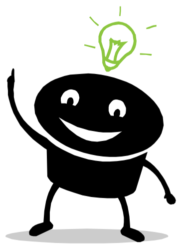
    $\Rightarrow$ f flux maxim

Implementarea algoritmului FORD-FULKERSON

Algoritmul Ford-Fulkerson

- Cum determinăm un lanţ f-nesaturat?

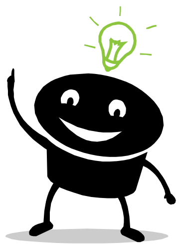
Algoritmul Ford-Fulkerson



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



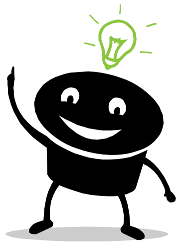
Spre exemplu prin parcurgerea grafului pornind din vârful s şi considerând doar arce cu capacitatea reziduală pozitivă (în raport cu lanţurile construite prin parcurgere, memorate cu vectorul tata)

- **Parcurgerea BF** $\Rightarrow$

  determinăm s-t lanţuri 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

<u>**Schema**</u>:

```
initializeaza_flux_nul()
cat timp (construieste_s-t_lant_nesat_BF()=true) executa
    revizuieste_flux_lant()
afiseaza_flux()
```

Implementare. Algoritmul Edmonds-Karp

construieste_s-t_lant_nesat_BF() - construieste un s-t lanţ nesaturat prin parcurgerea BF din s
(a grafului rezidual)

- sunt considerate în parcurgere **doar arce pe care se poate modifica fluxul**, adică având capacitate reziduală pozitivă

Implementare. Algoritmul Edmonds-Karp

construieste_s-t_lant_nesat_BF() – construieste un s-t lanţ nesaturat prin parcurgerea BF din s

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
    -
    -

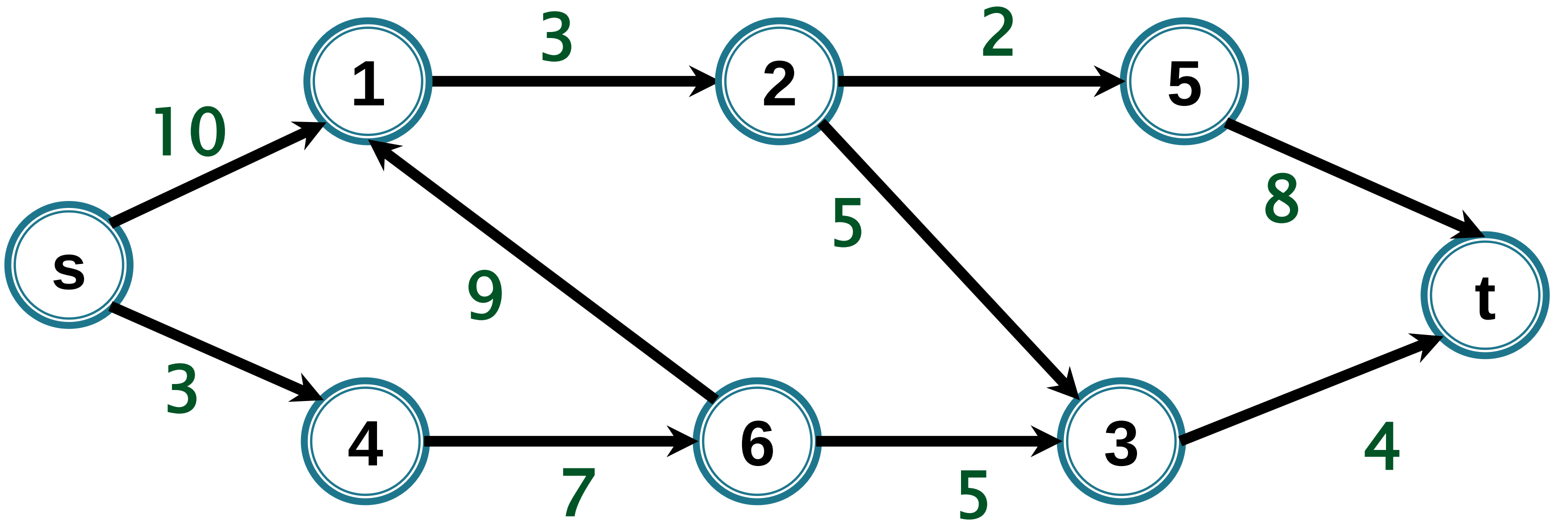
Implementare. Algoritmul Edmonds-Karp

revizuieste_flux_lant()

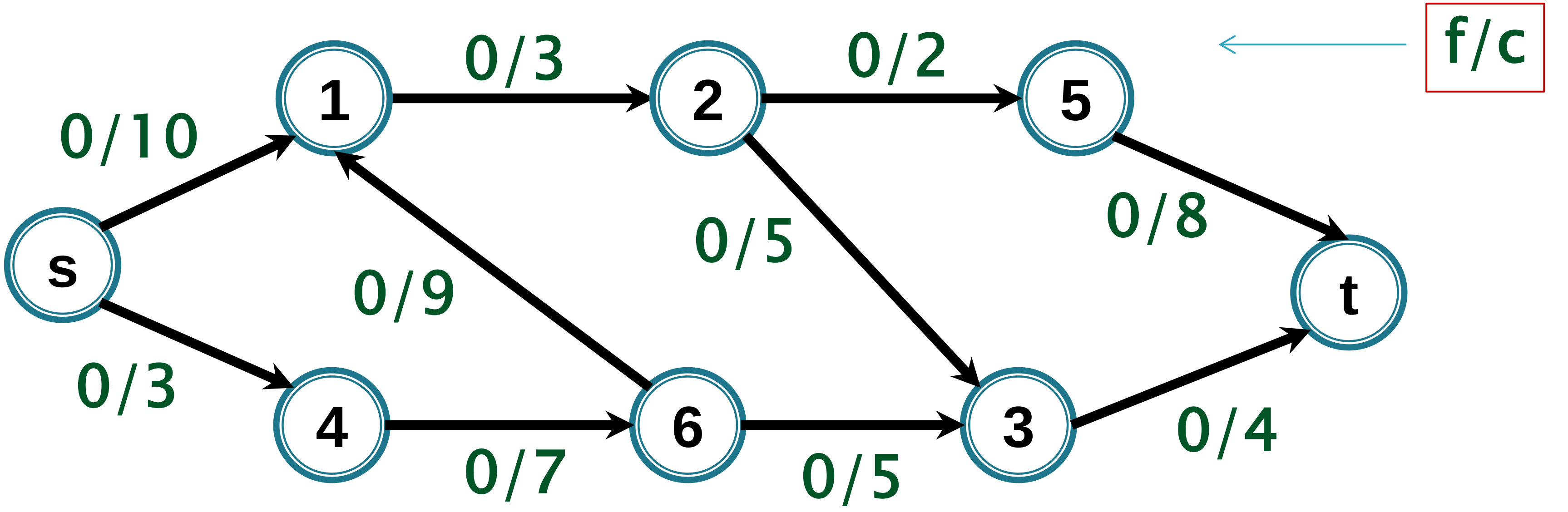
- **fie P s-t lanțul găsit în construieste_s-t_lant_nesat_BF()**
- **calculăm i(P)**
- **pentru fiecare arc *e* al lanțului P**
  - creștem cu i(P) fluxul pe *e* dacă este arc direct
  - scădem cu i(P) fluxul pe *e* dacă este arc invers

Exemplu

Algoritmul EDMONDS-KARP

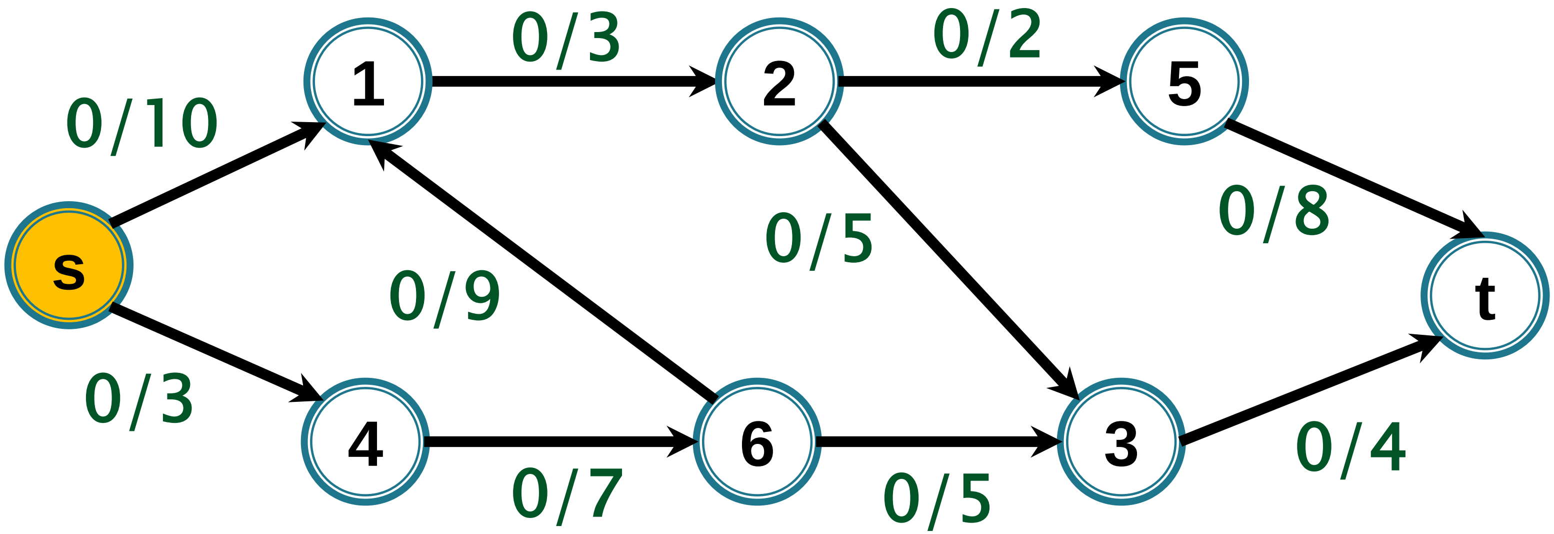

s
1
2
5
t
4
6
3
10
3
3
2
8
5
9
7
5
4

initializeaza_flux_nul

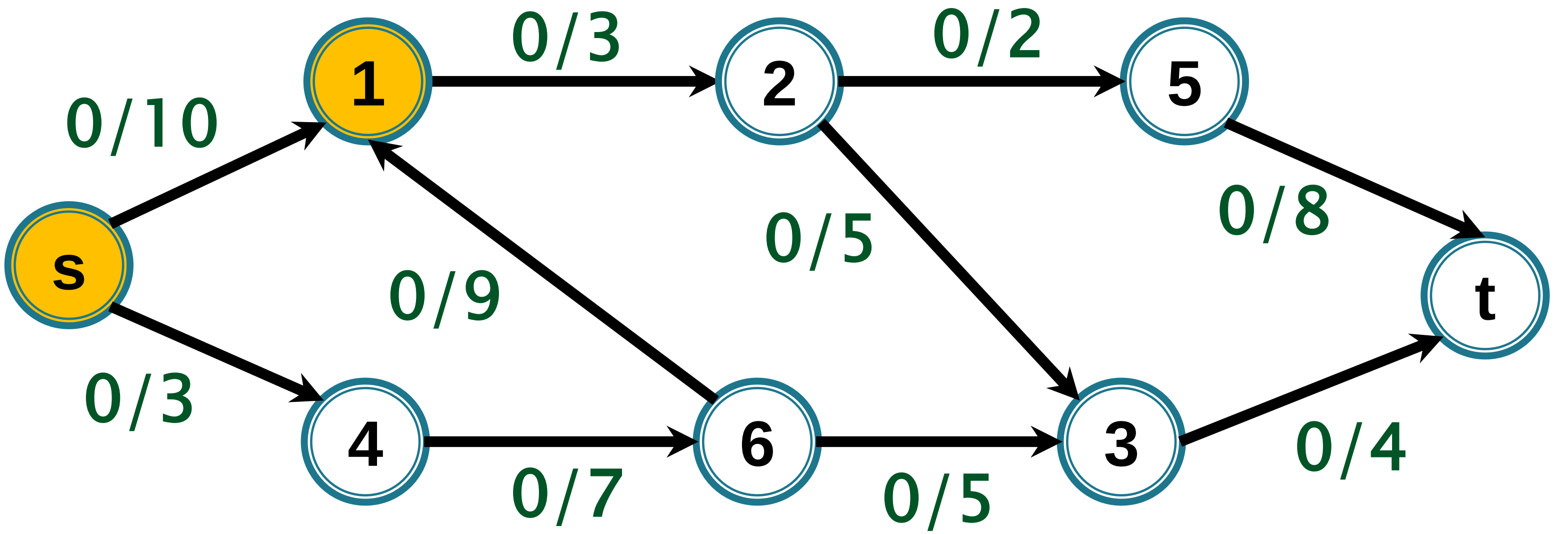


f/c
0/10
0/3
0/2
0/5
0/8
0/9
0/3
0/7
0/5
0/4
s
1
2
5
t
4
6
3

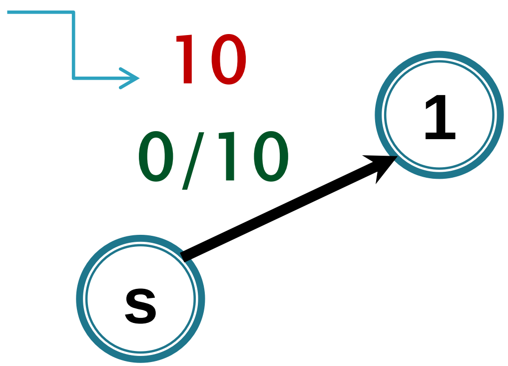
construieste_s-t_lant_nesat_BF

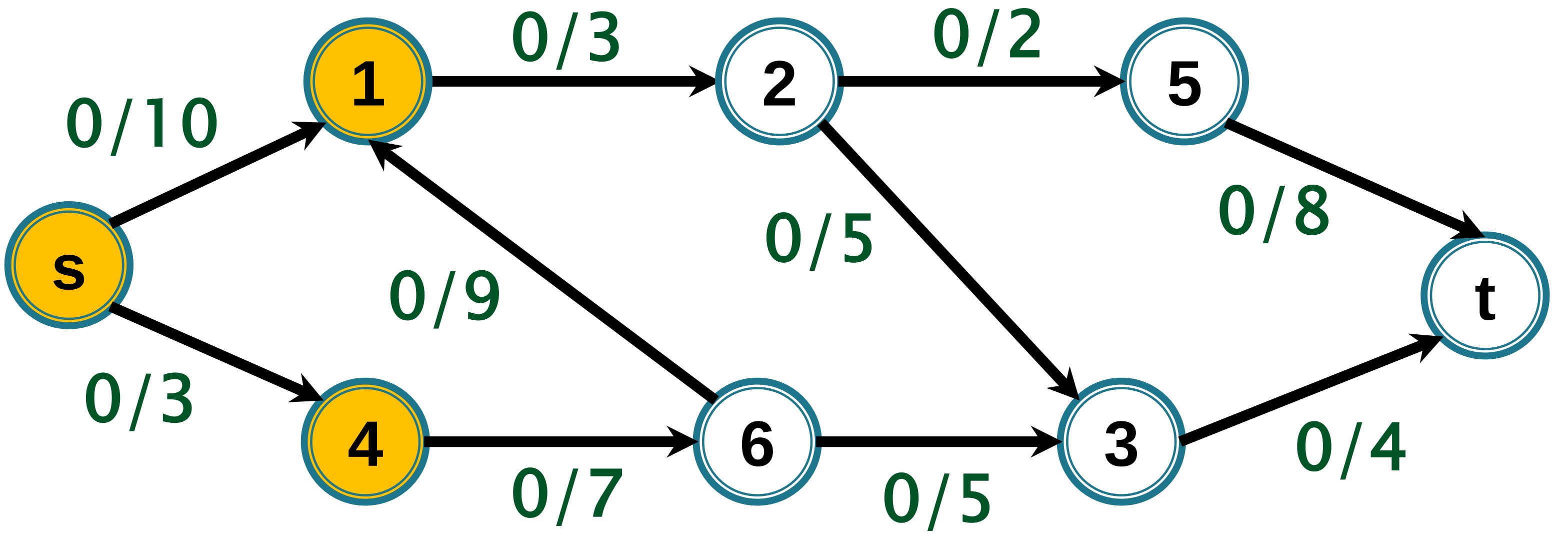

s
1
2
5
t
4
6
3
0/10
0/3
0/3
0/2
0/9
0/5
0/8
0/7
0/5
0/4


s

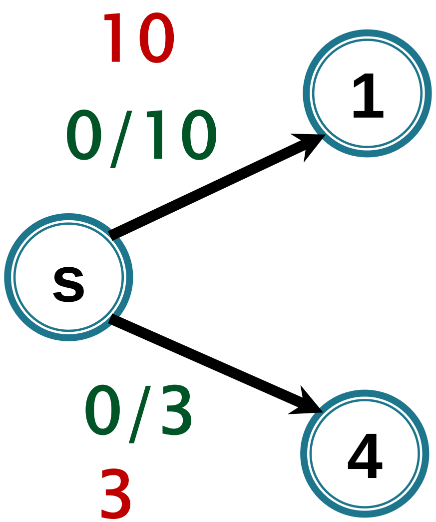


Capacitatea reziduală

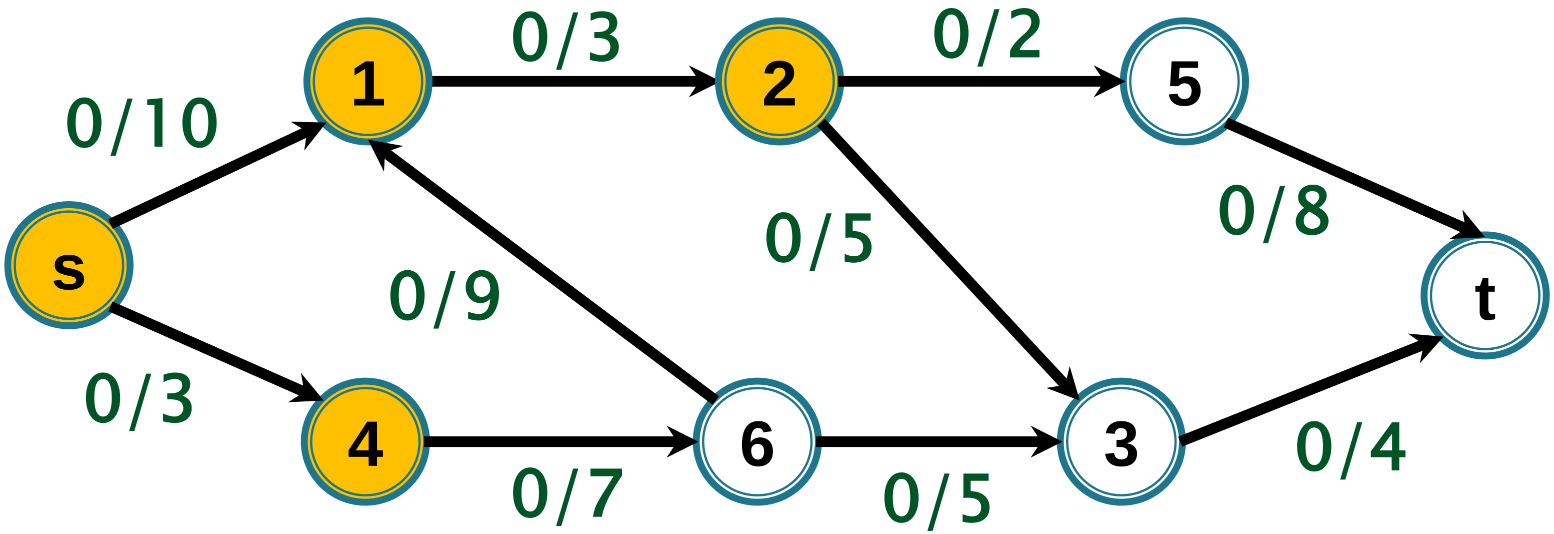


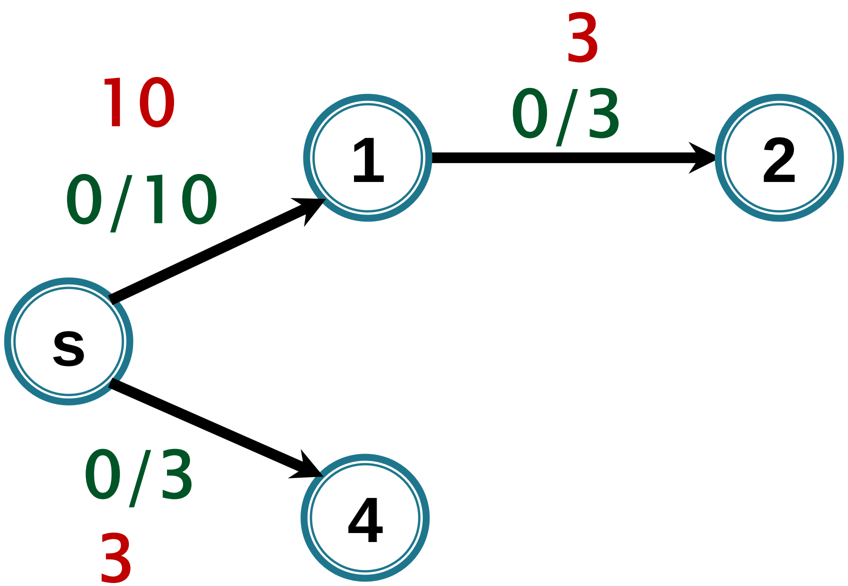


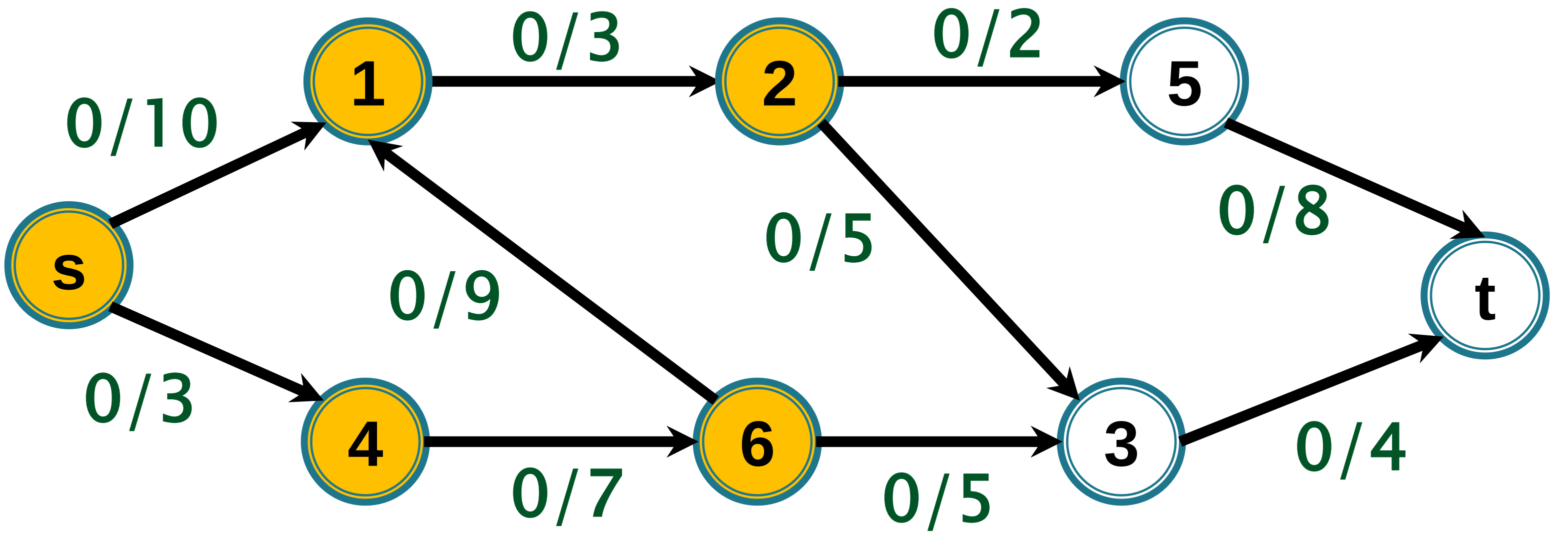
s
1
2
5
t
4
6
3
0/10
0/3
0/3
0/2
0/9
0/5
0/8
0/7
0/5
0/4



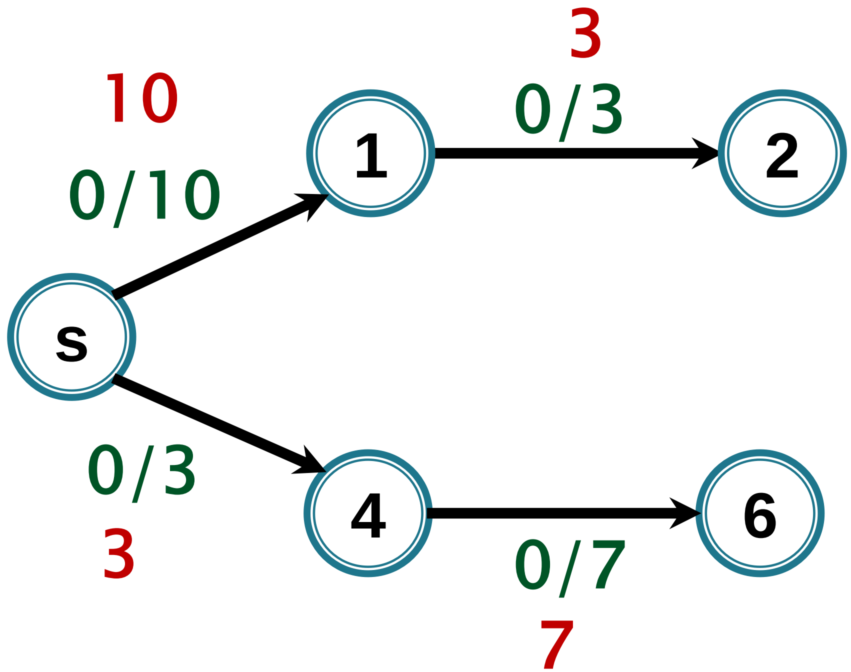
10
0/10
s
1
4
0/3
3


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

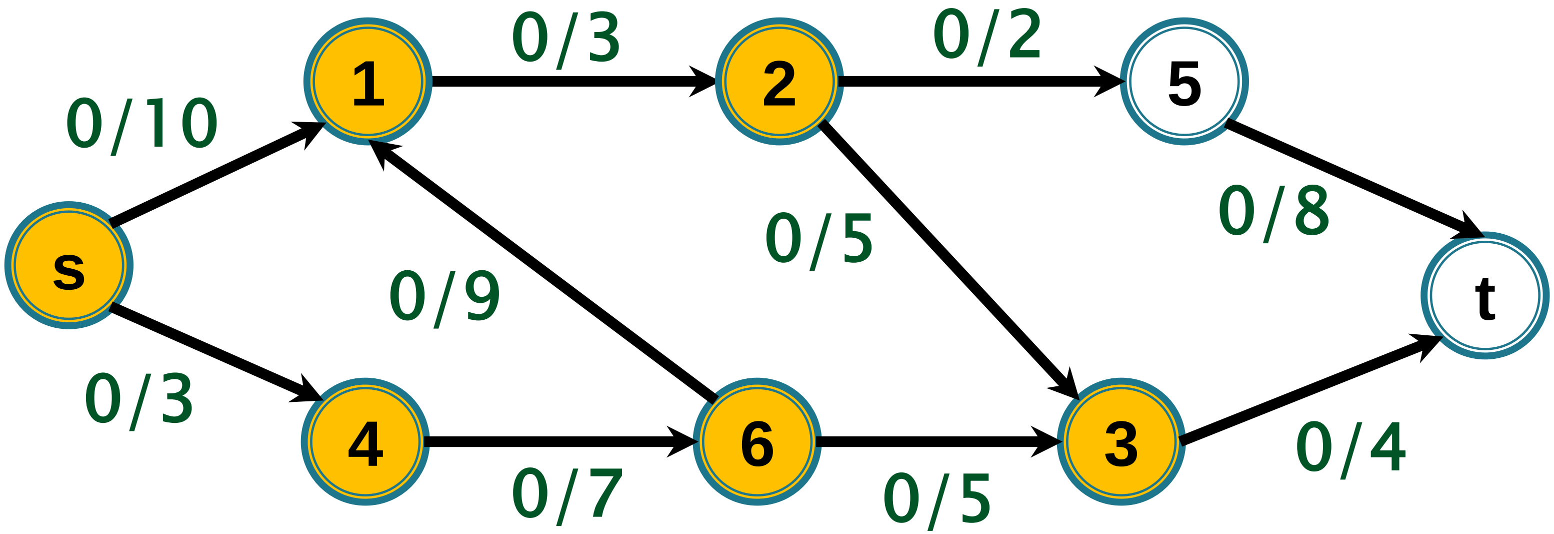

3
10
0/3
1
2
0/10
s
0/3
4
3



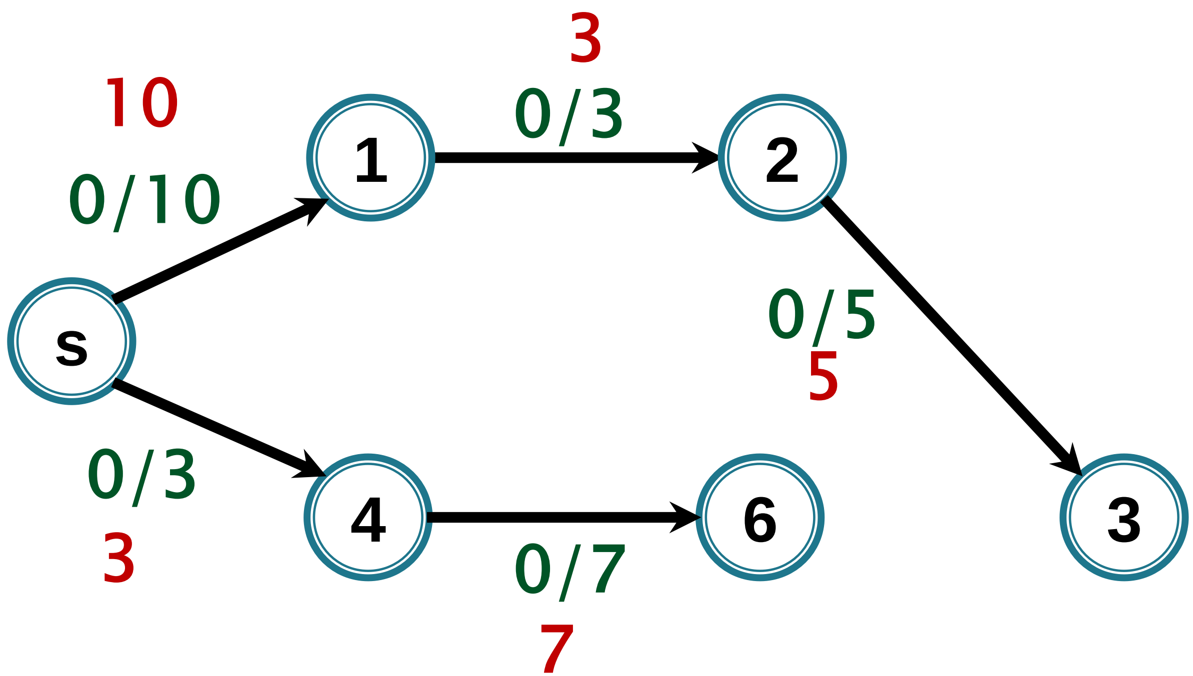
0/3
0/2
0/10
1
2
5
0/8
0/5
s
0/9
t
0/3
4
6
3
0/4
0/7
0/5



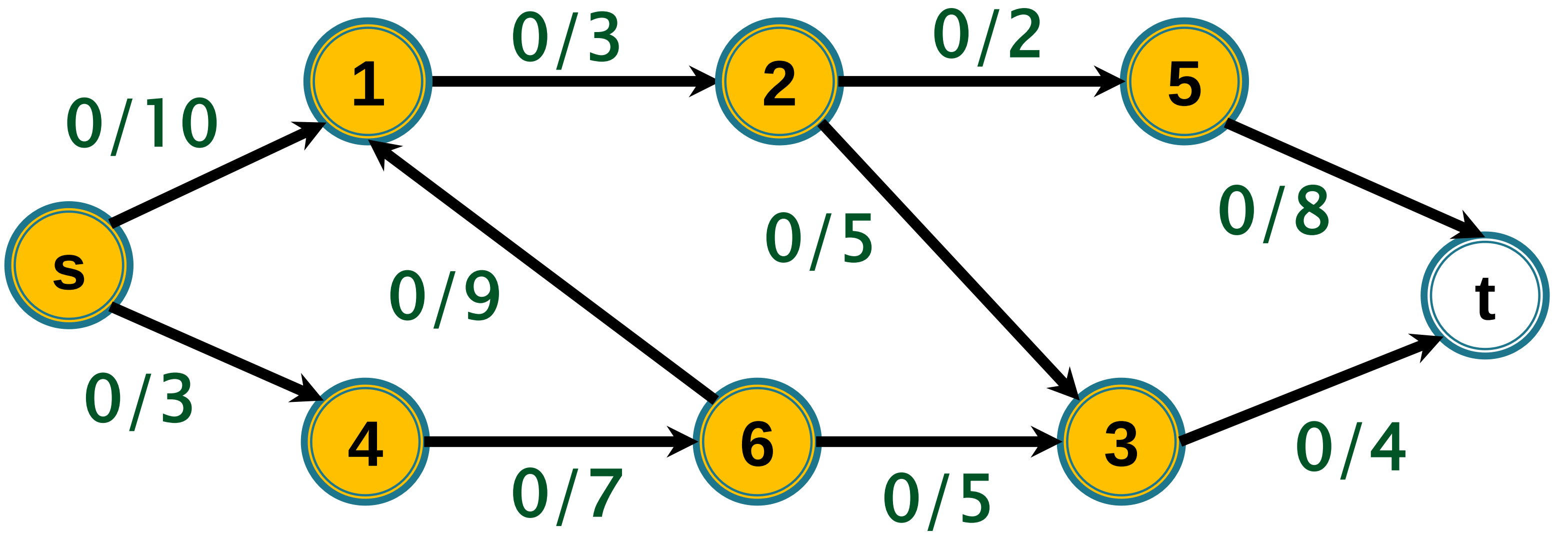
3
10
0/3
1
2
0/10
s
0/3
4
6
3
0/7
7



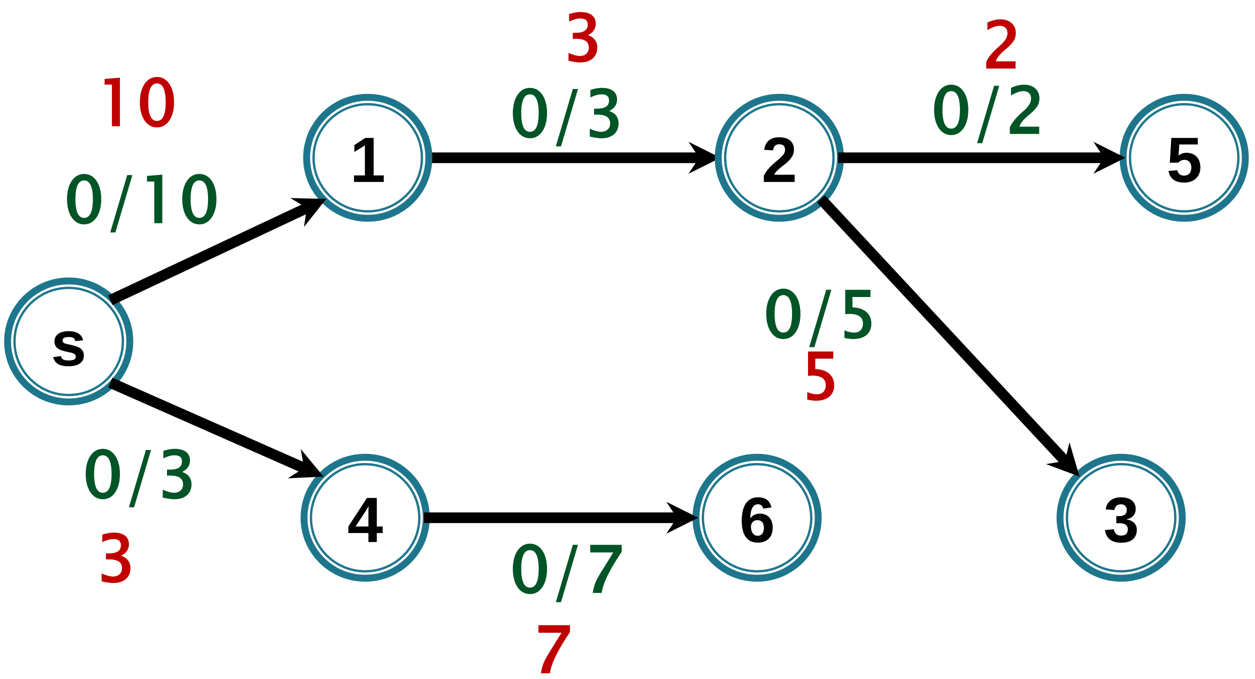
s
1
2
5
t
4
6
3
0/10
0/3
0/3
0/2
0/8
0/5
0/9
0/7
0/5
0/4



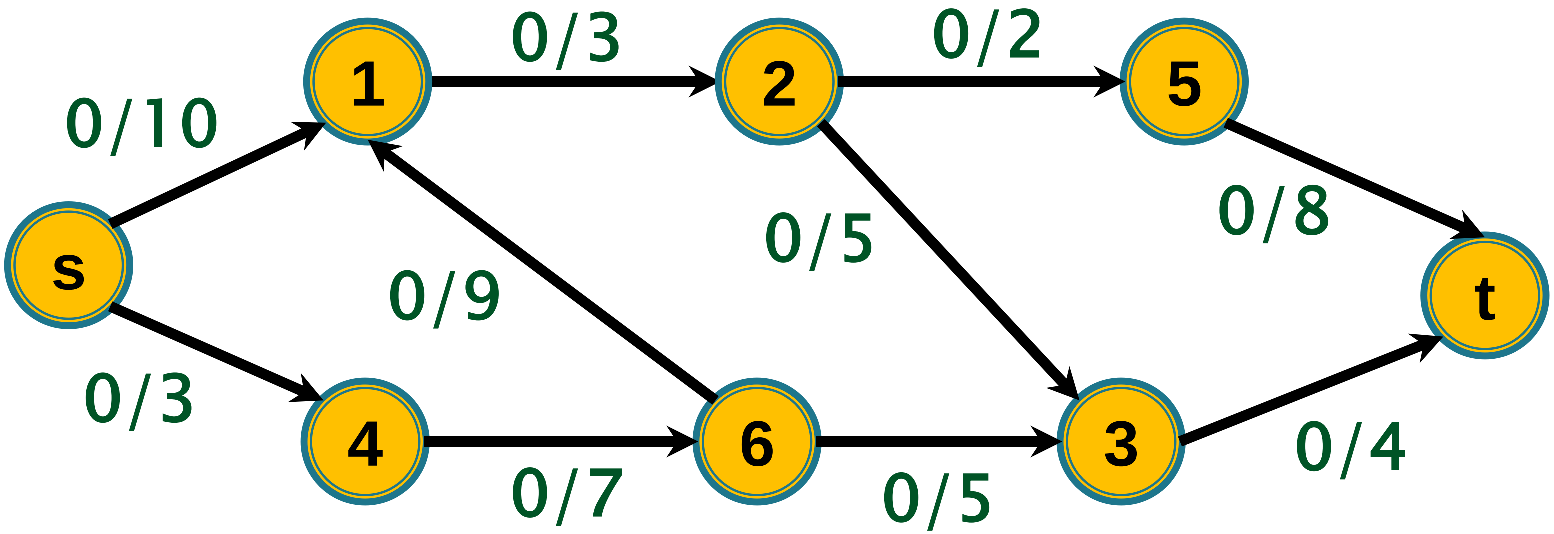
s
1
2
4
6
3
3
0/3
10
0/10
0/5
5
0/3
3
0/7
7

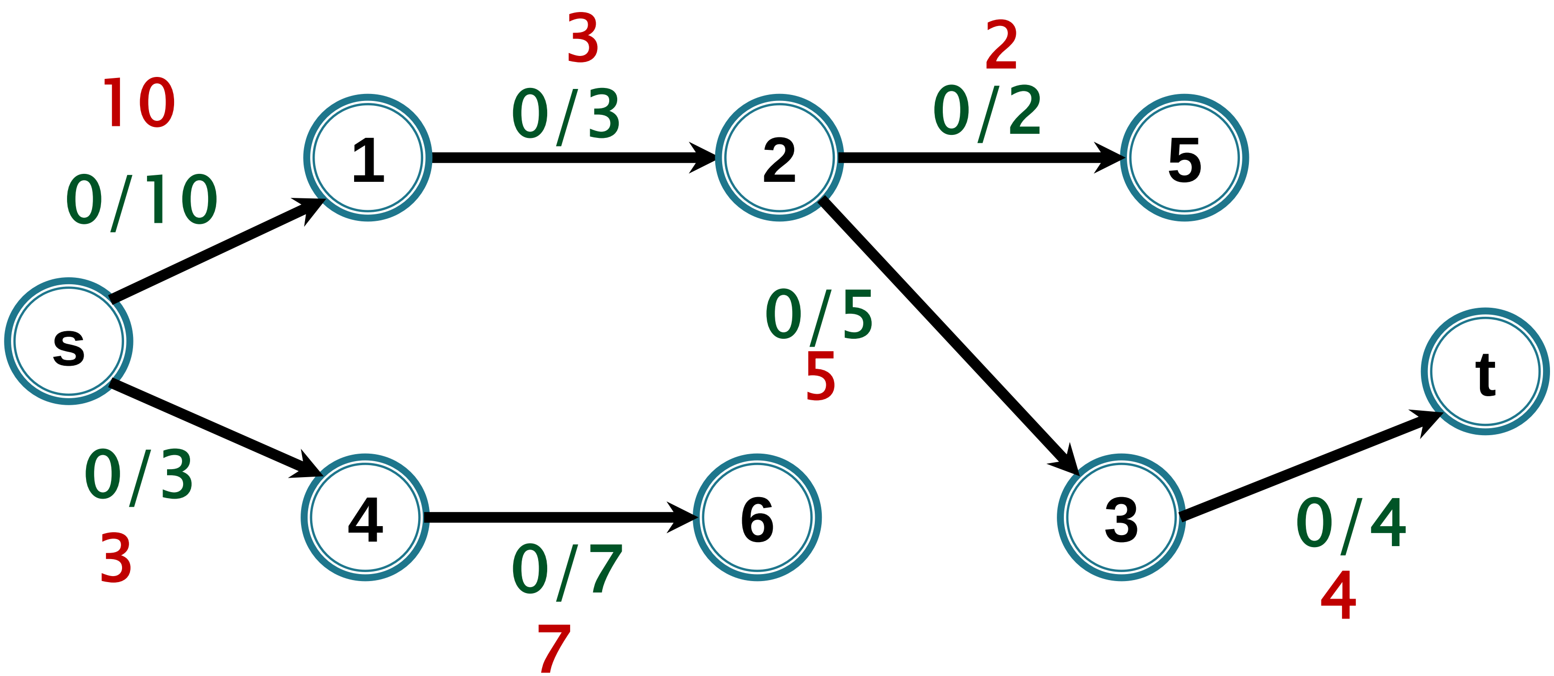


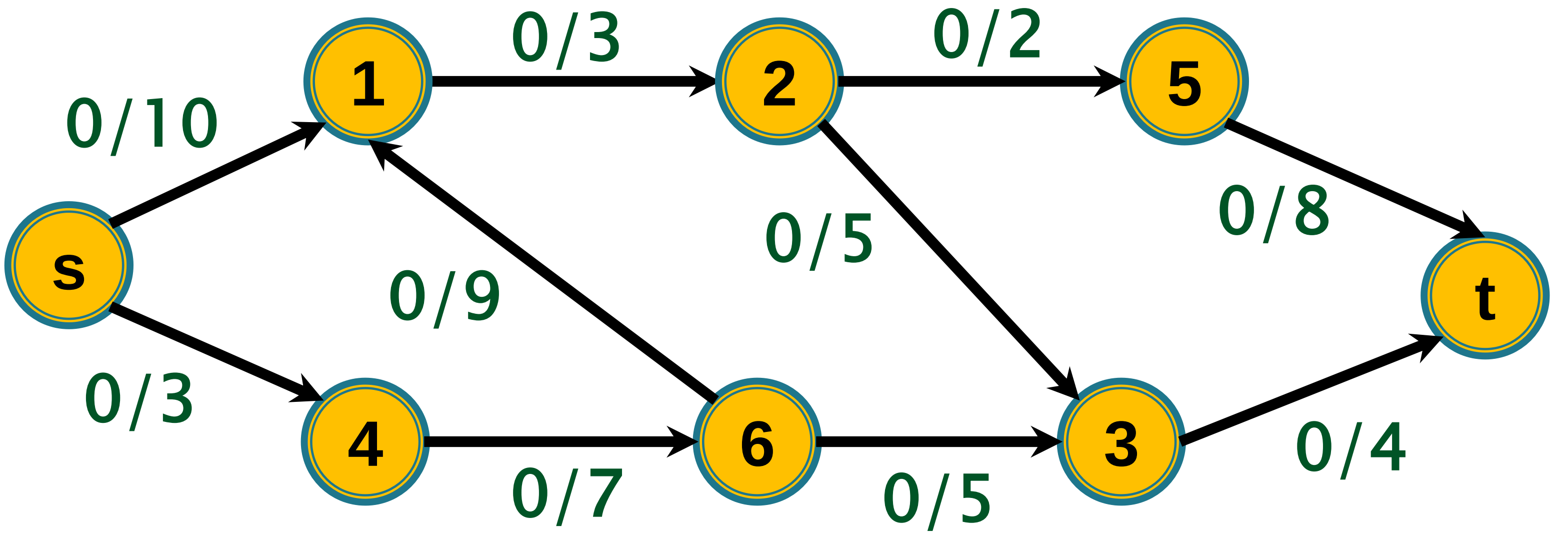
s
1
2
5
t
4
6
3
0/10
0/3
0/2
0/9
0/5
0/8
0/3
0/7
0/5
0/4

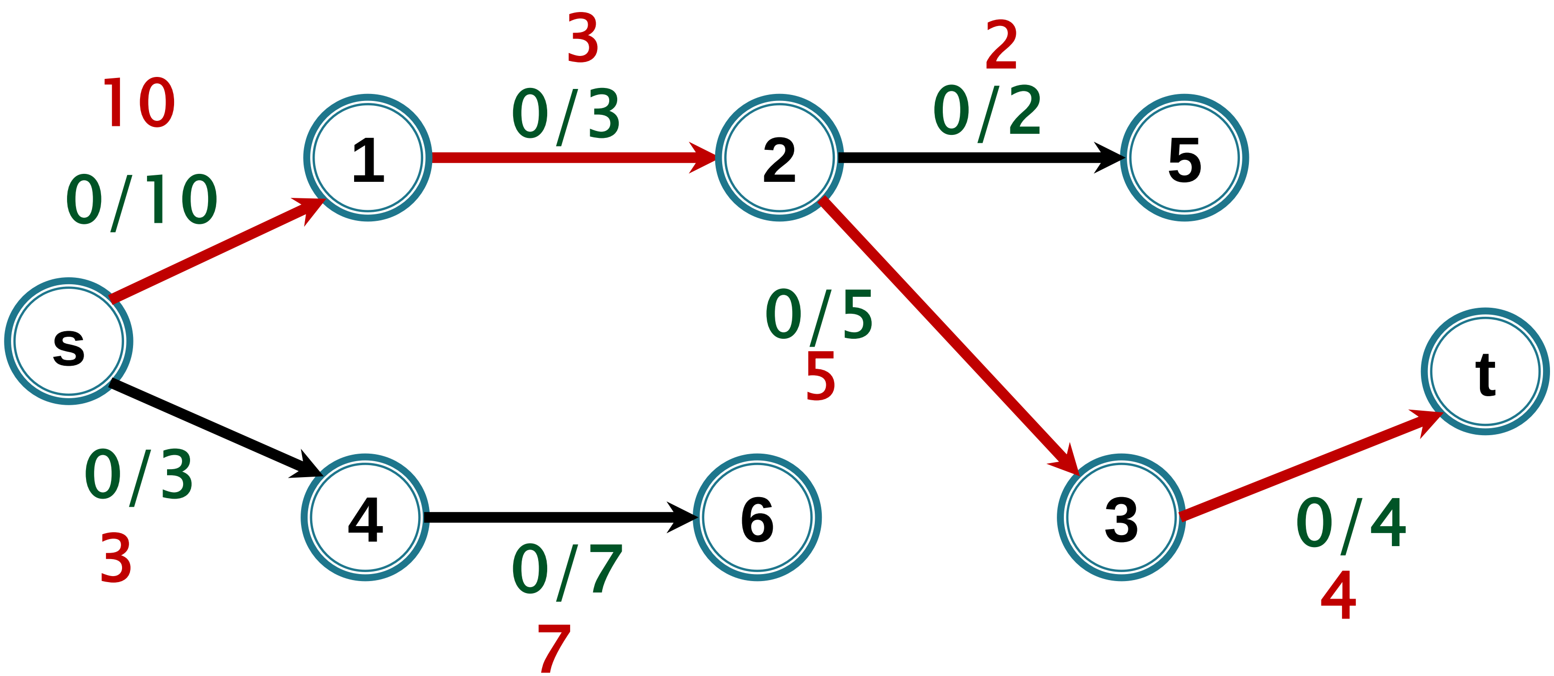


s
1
2
5
4
6
3
10
0/10
3
0/3
2
0/2
0/5
5
0/3
3
0/7
7

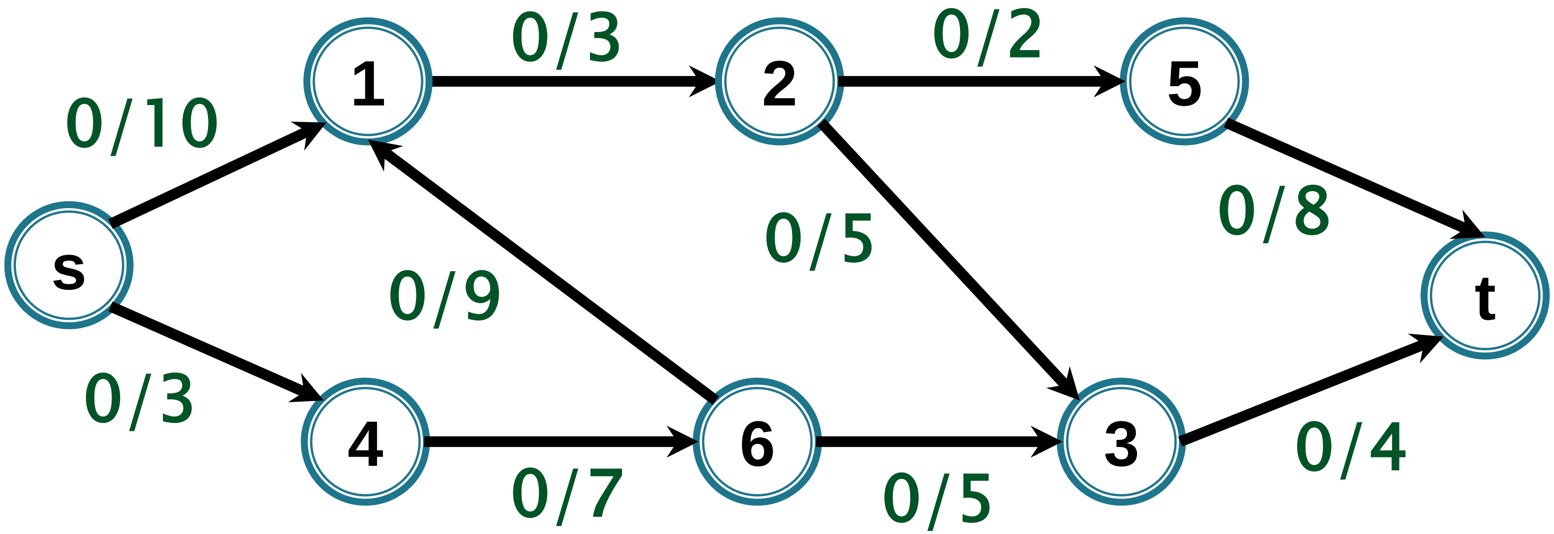

s
1
2
5
t
4
6
3
0/10
0/3
0/2
0/8
0/5
0/9
0/3
0/7
0/5
0/4

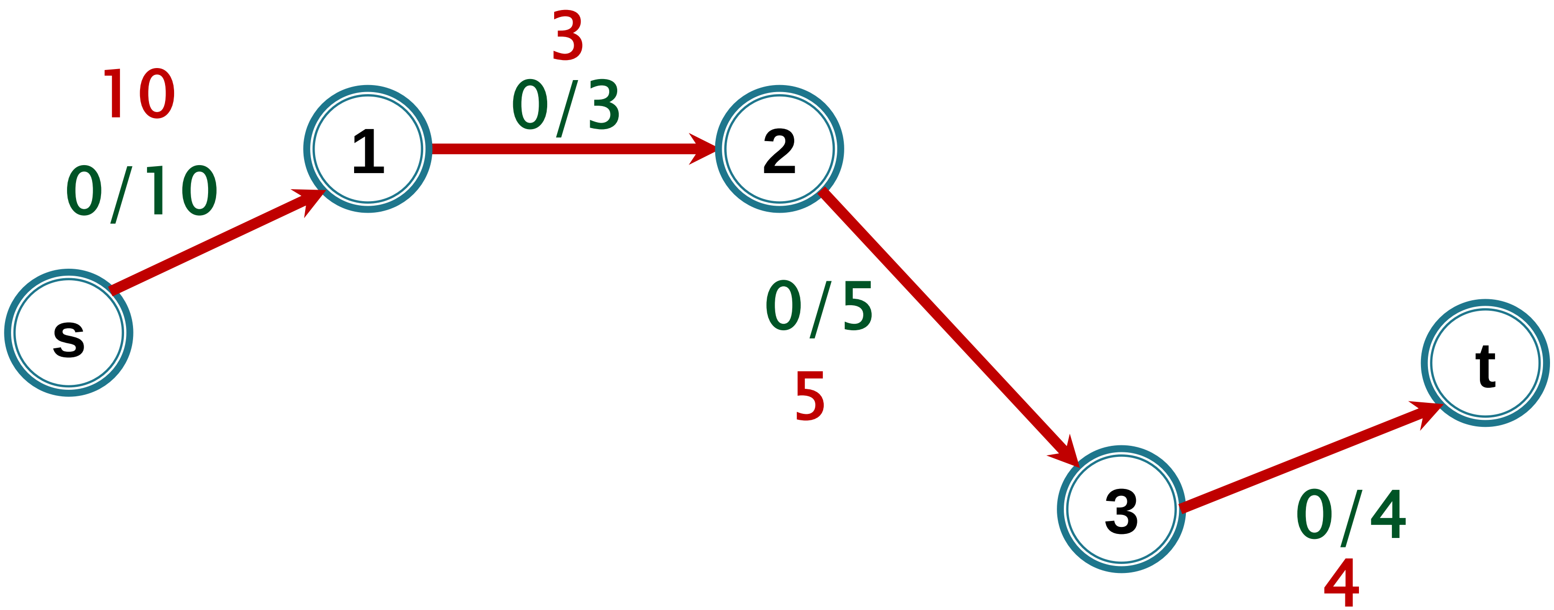

s
1
2
5
t
4
6
3
10
0/10
3
0/3
2
0/2
0/5
5
0/3
3
0/7
7
0/4
4


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

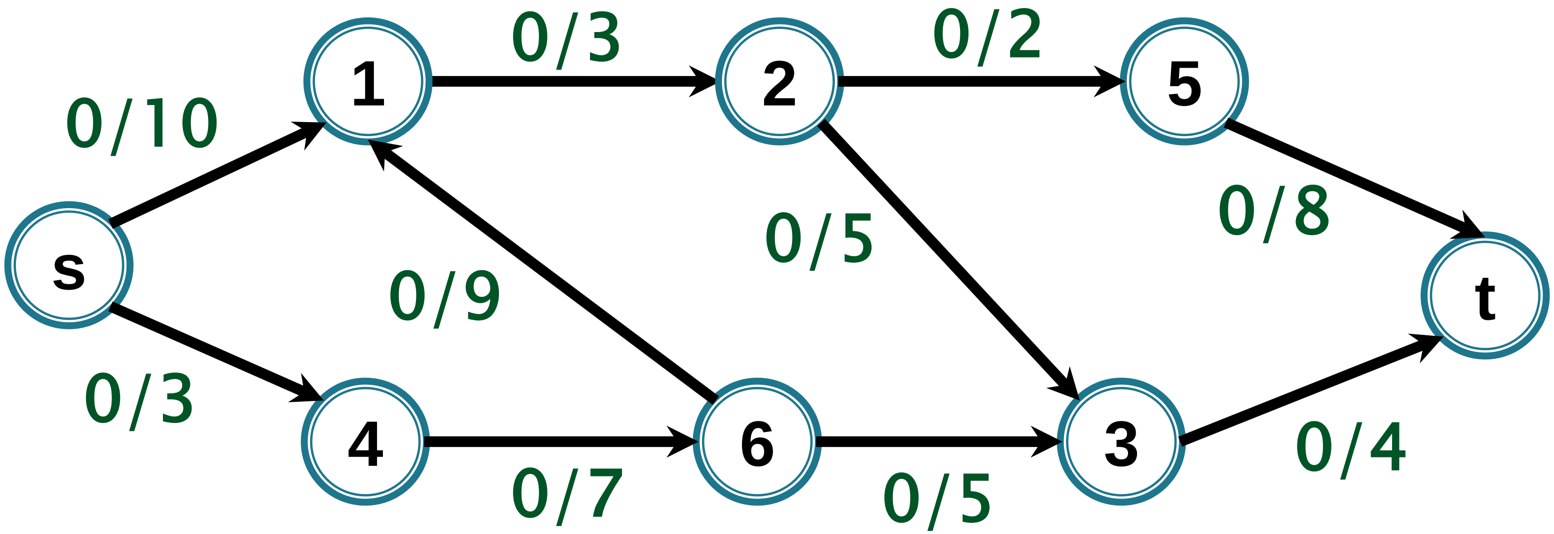

s
1
2
5
t
4
6
3
10
0/10
3
0/3
2
0/2
0/5
5
0/3
3
0/7
7
0/4
4

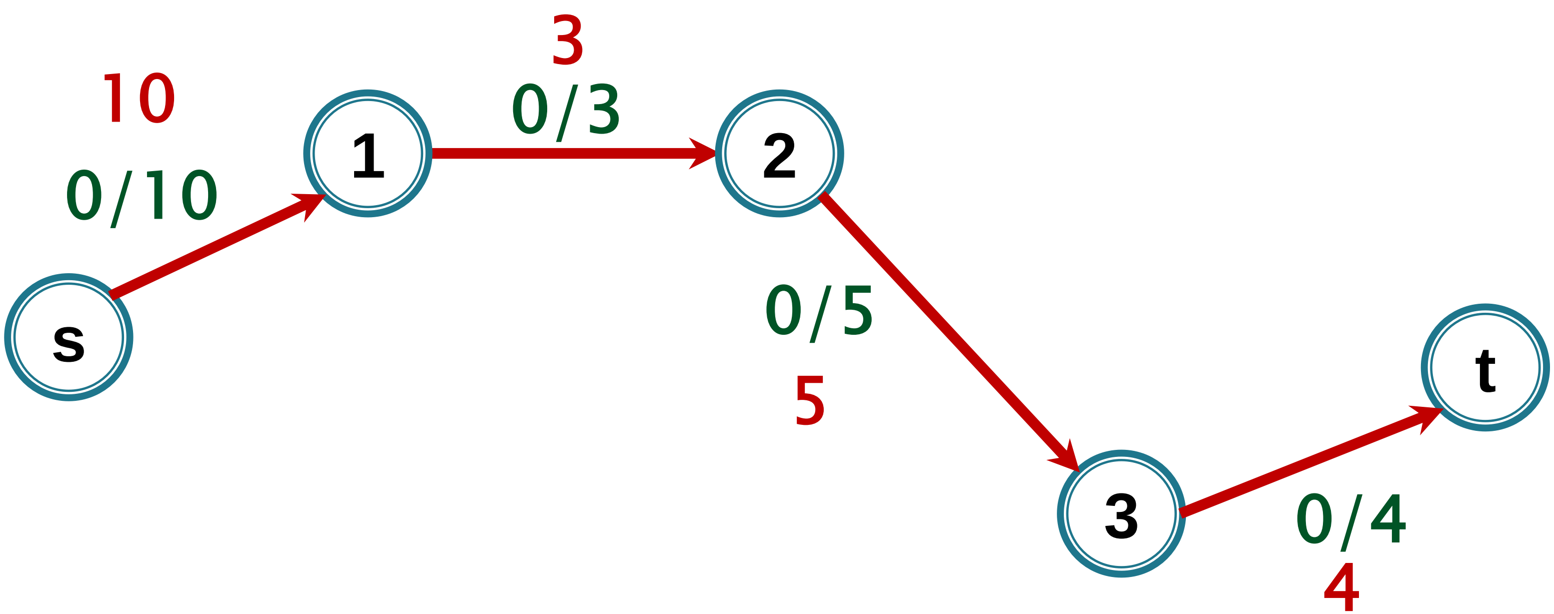
revizuieste_flux_lant



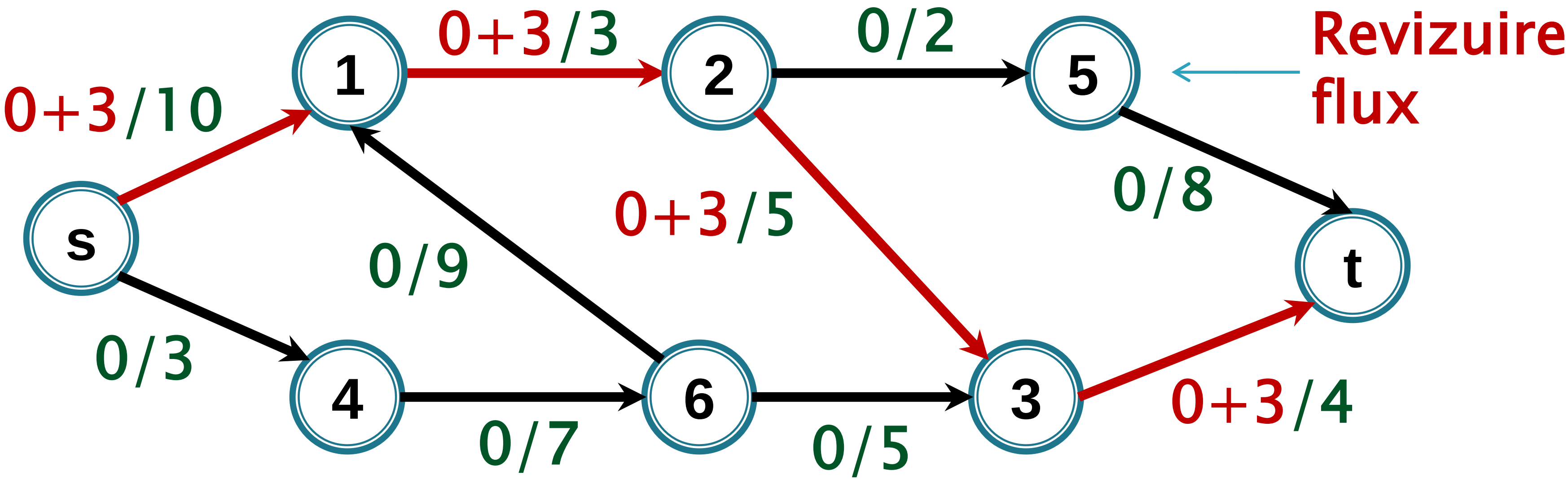


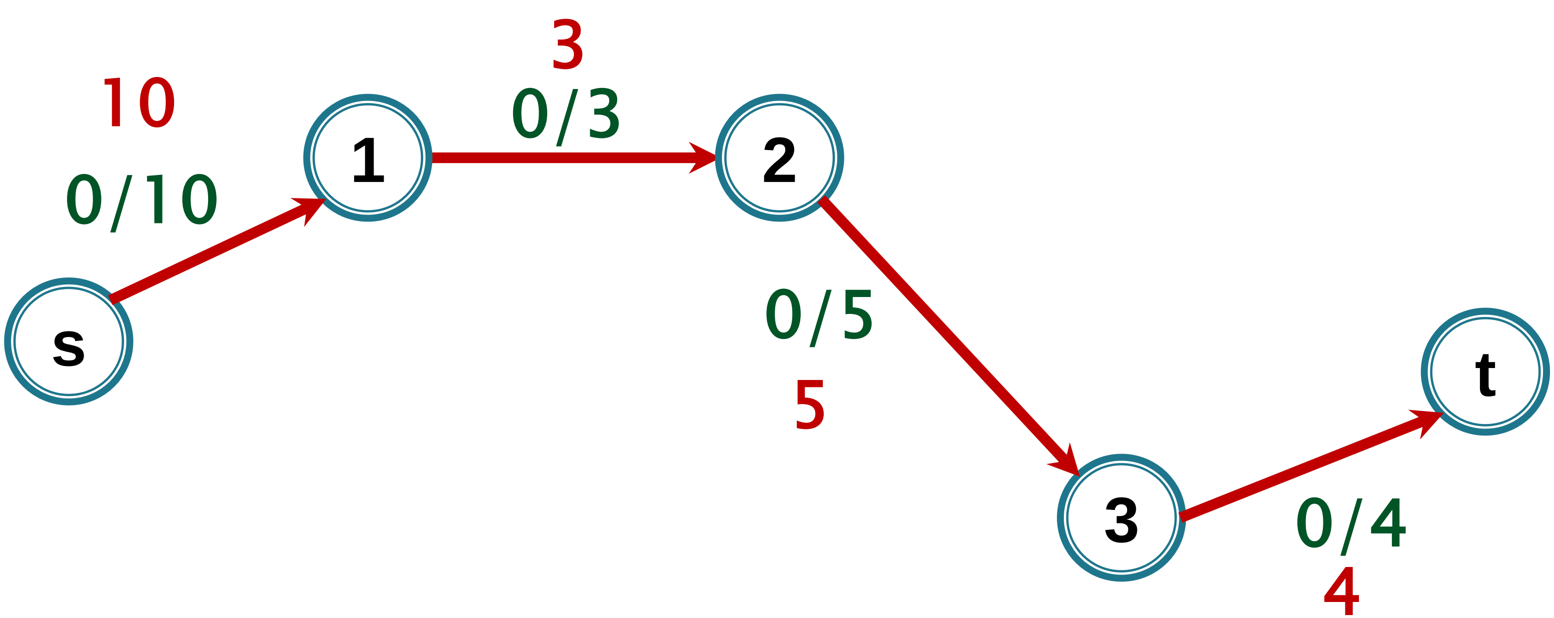
i(P) = ?



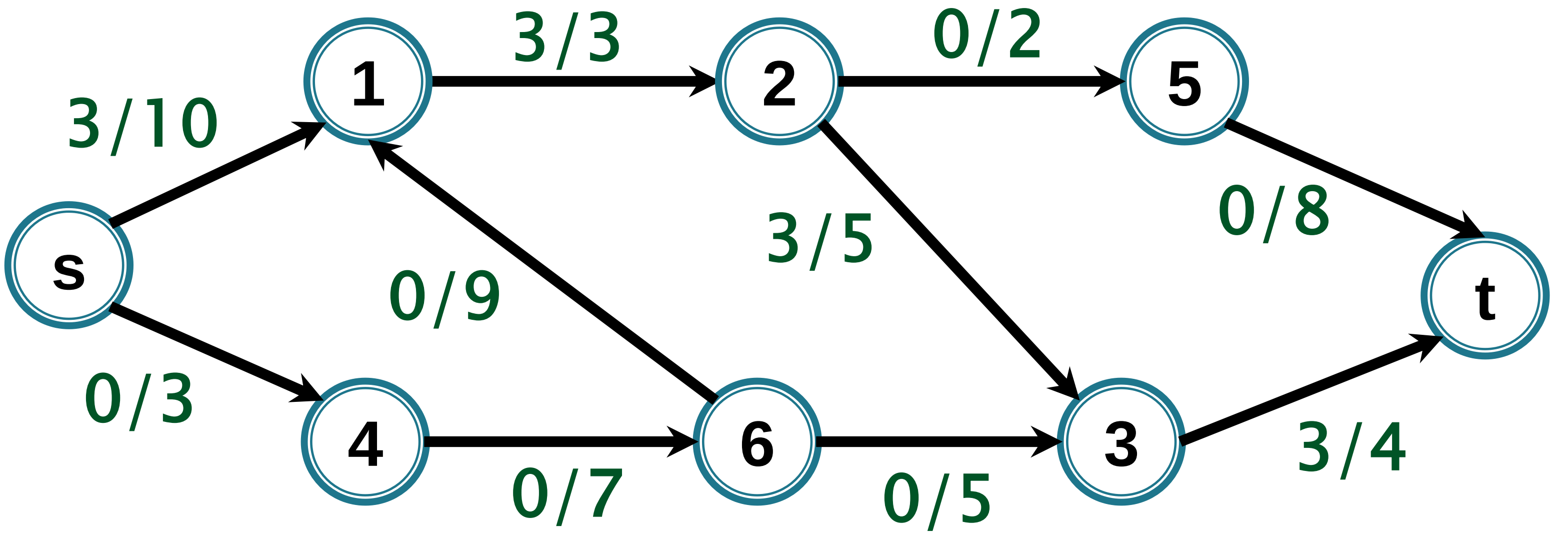


$$i(P) = \min\{10, 3, 5, 4\} = 3$$

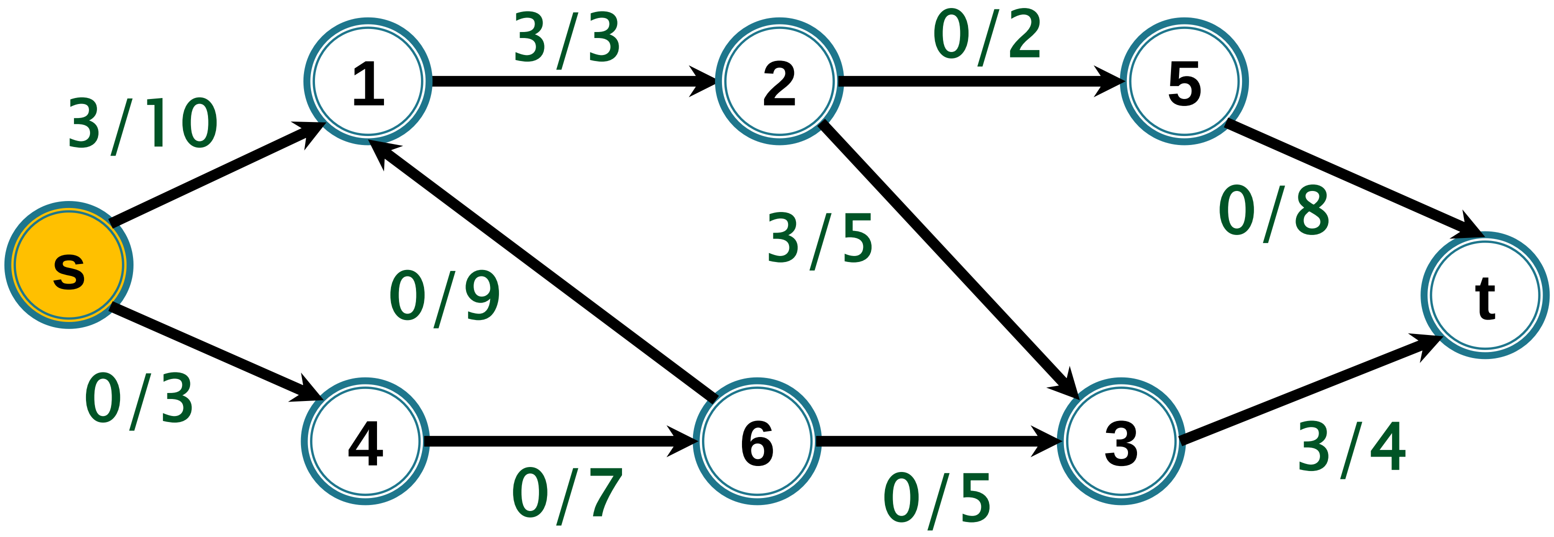




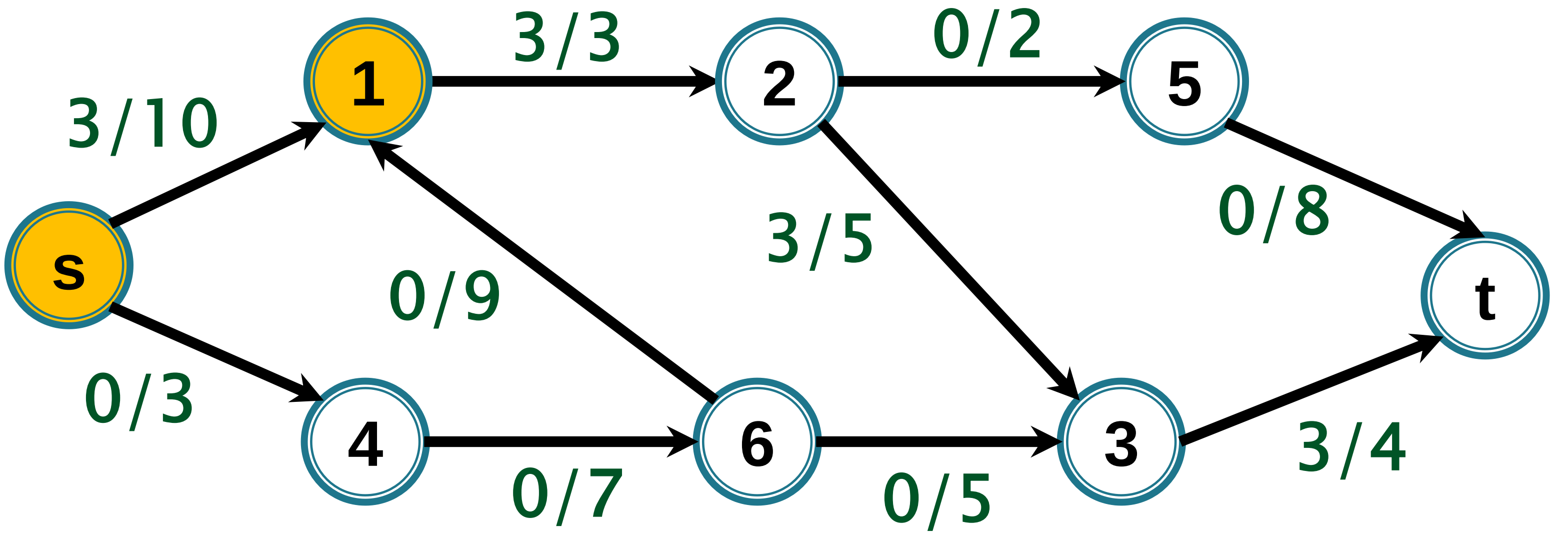
$$i(P) = \min\{10, 3, 5, 4\} = 3$$


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

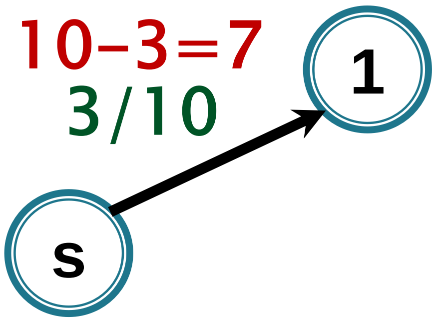
construieste_s-t_lant_nesat_BF

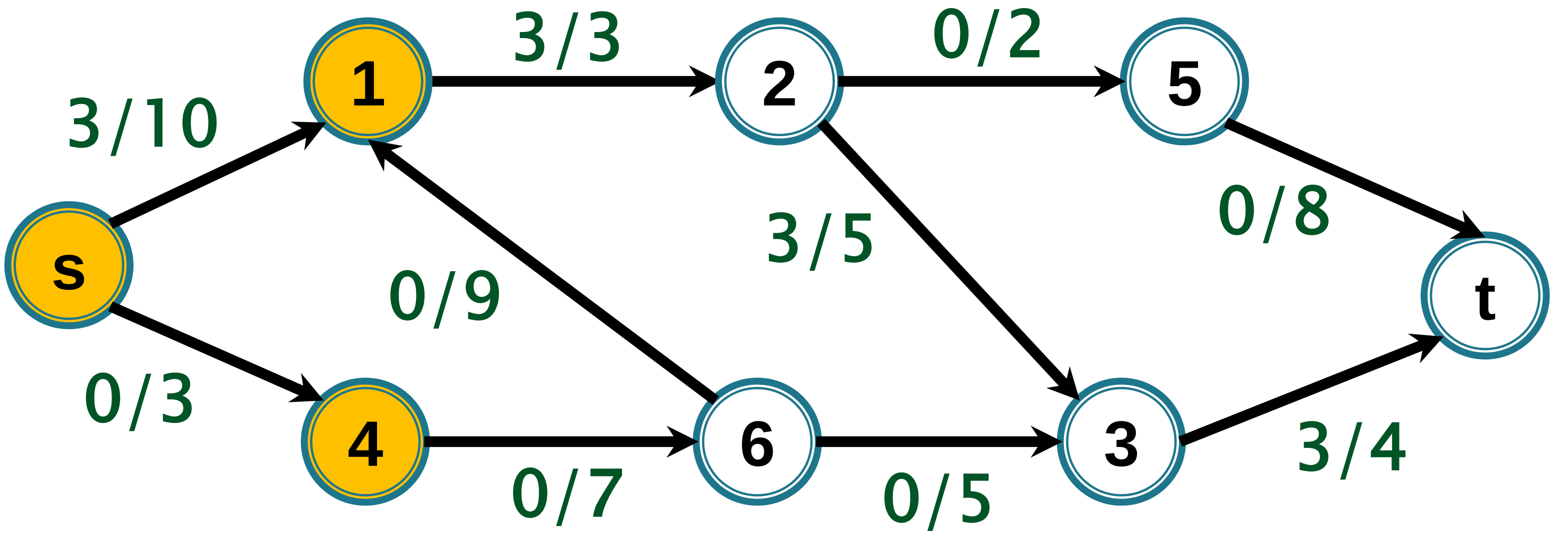

s
1
2
5
t
4
6
3
3/10
3/3
0/2
0/8
3/5
0/9
0/3
0/7
0/5
3/4

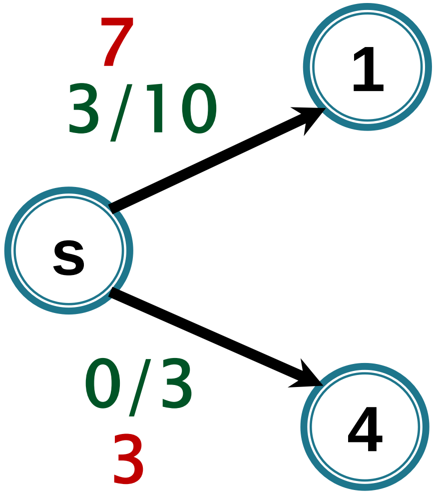

s

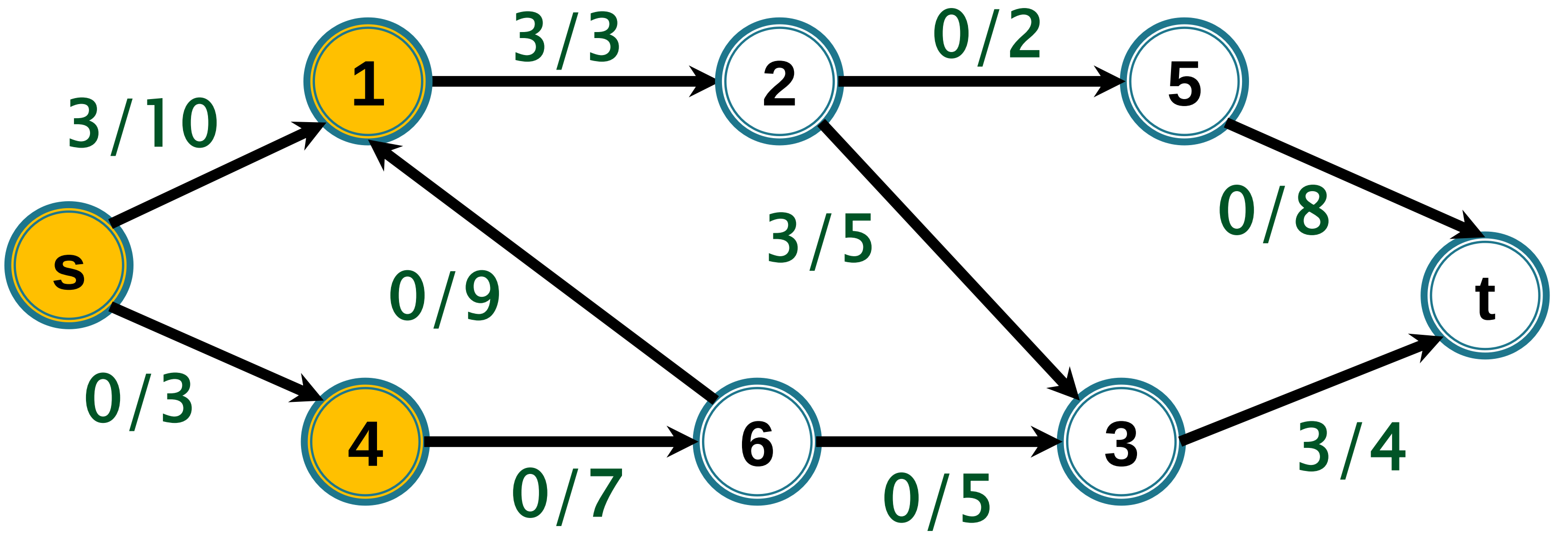


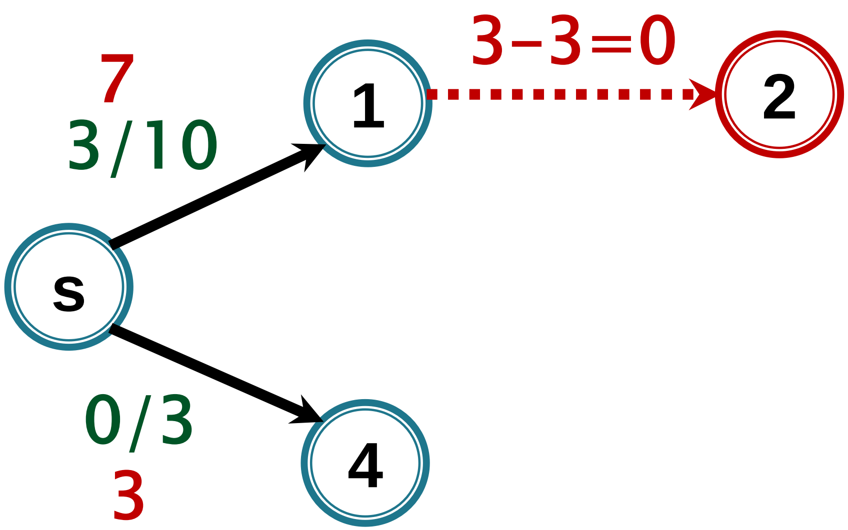
10−3=7

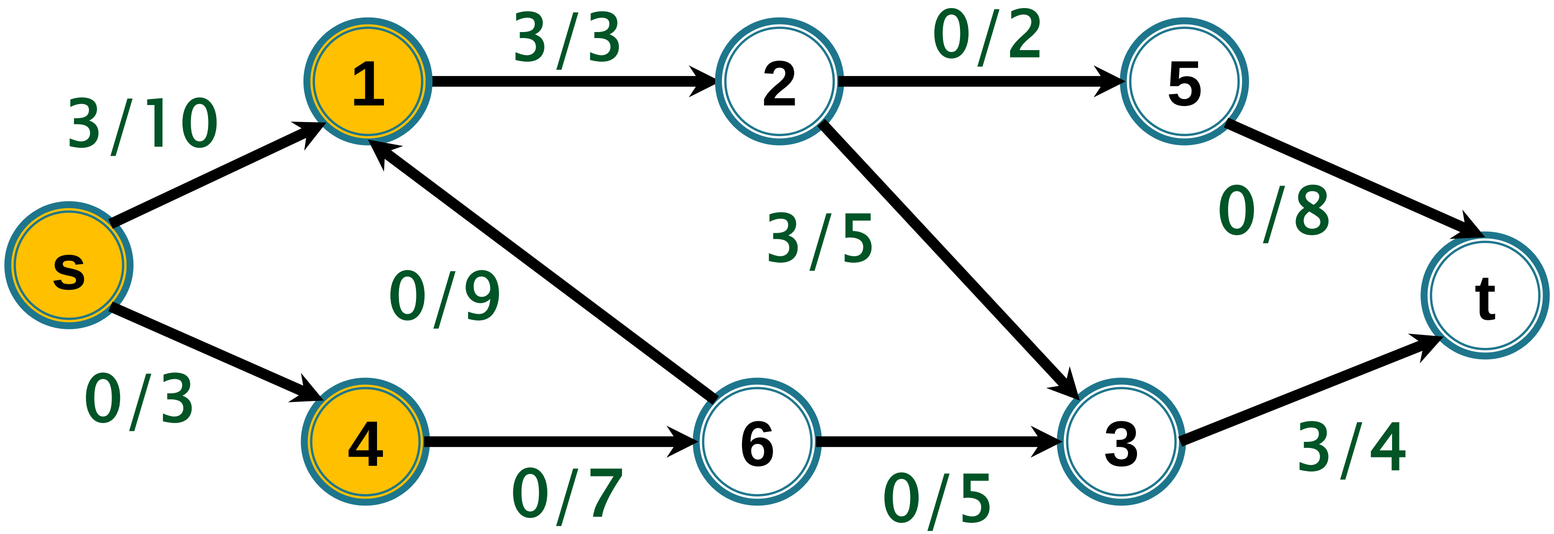


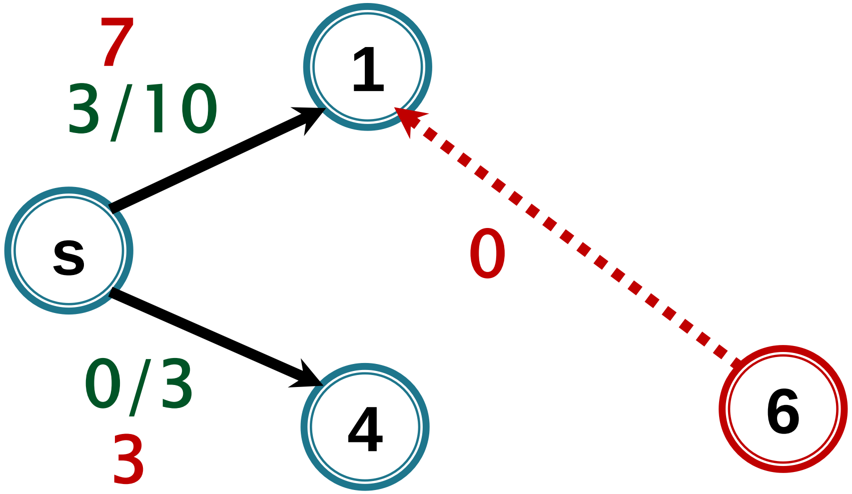

s
1
2
5
t
4
6
3
3/10
3/3
0/2
0/8
3/5
0/9
0/3
0/7
0/5
3/4

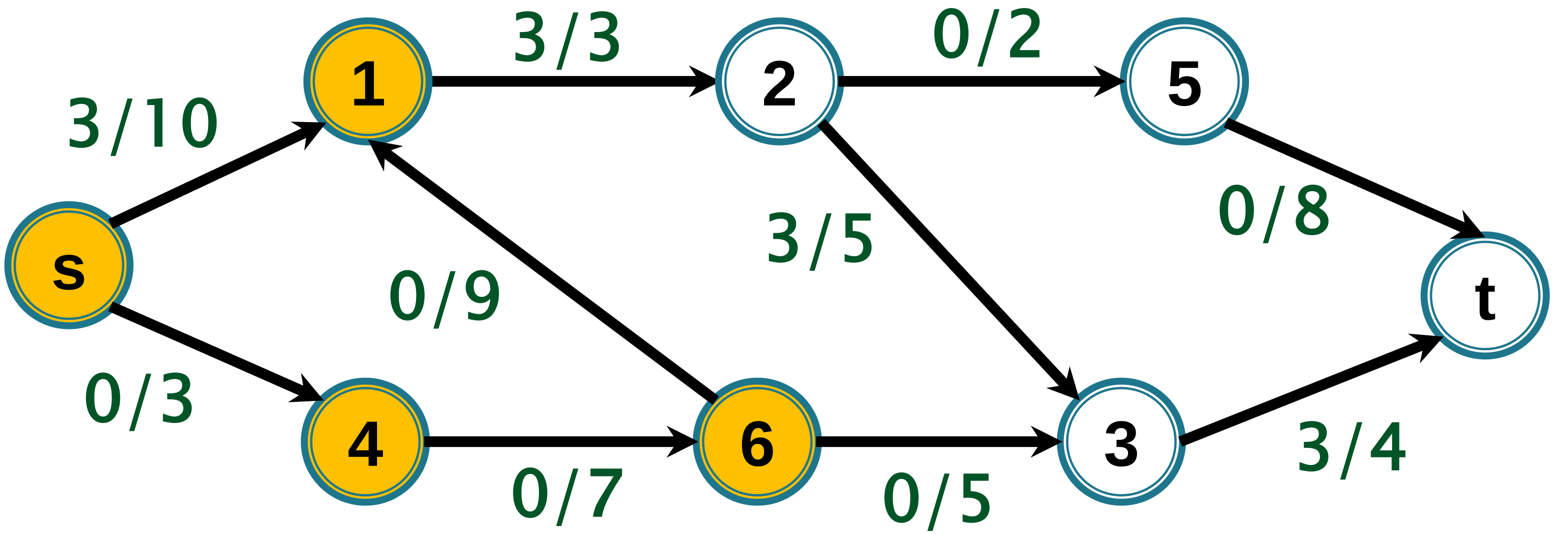

7
3/10
s
1
4
0/3
3

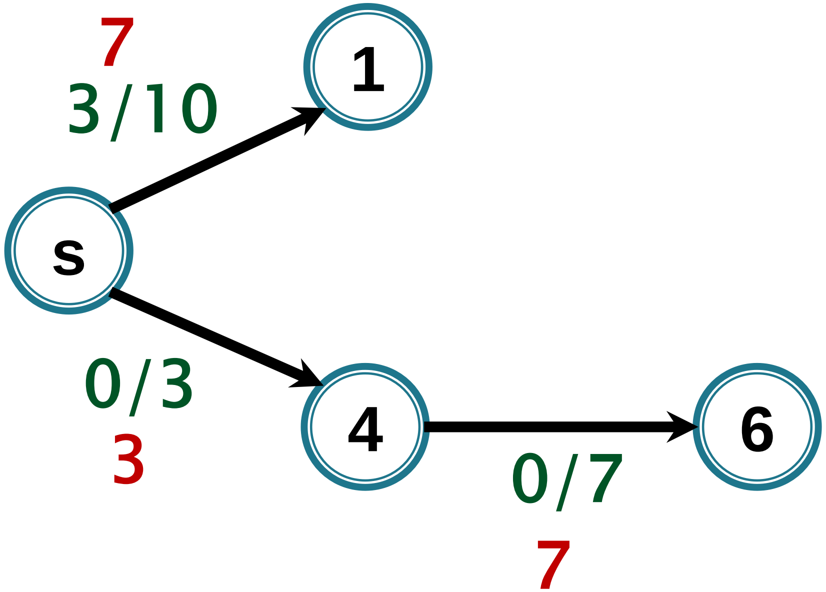

3/10
0/3
3/3
0/9
0/7
0/2
3/5
0/5
0/8
3/4
s
1
2
5
4
6
3
t

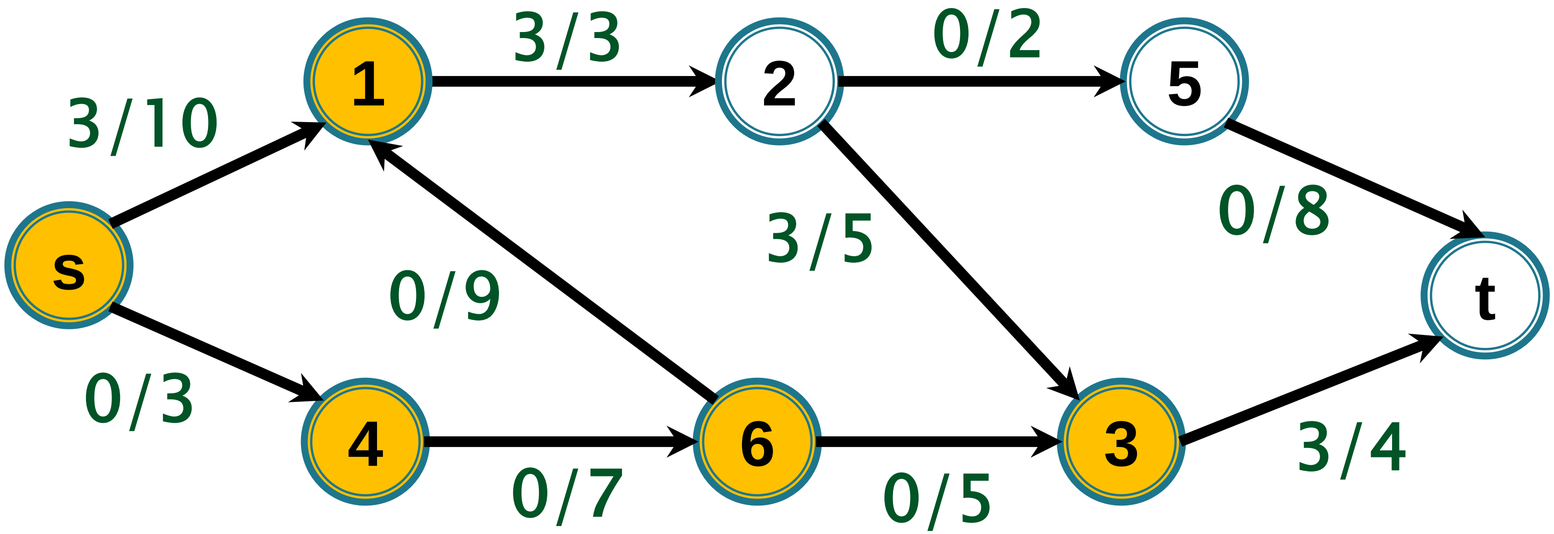

7
3/10
3−3=0
0/3
3
s
1
2
4


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

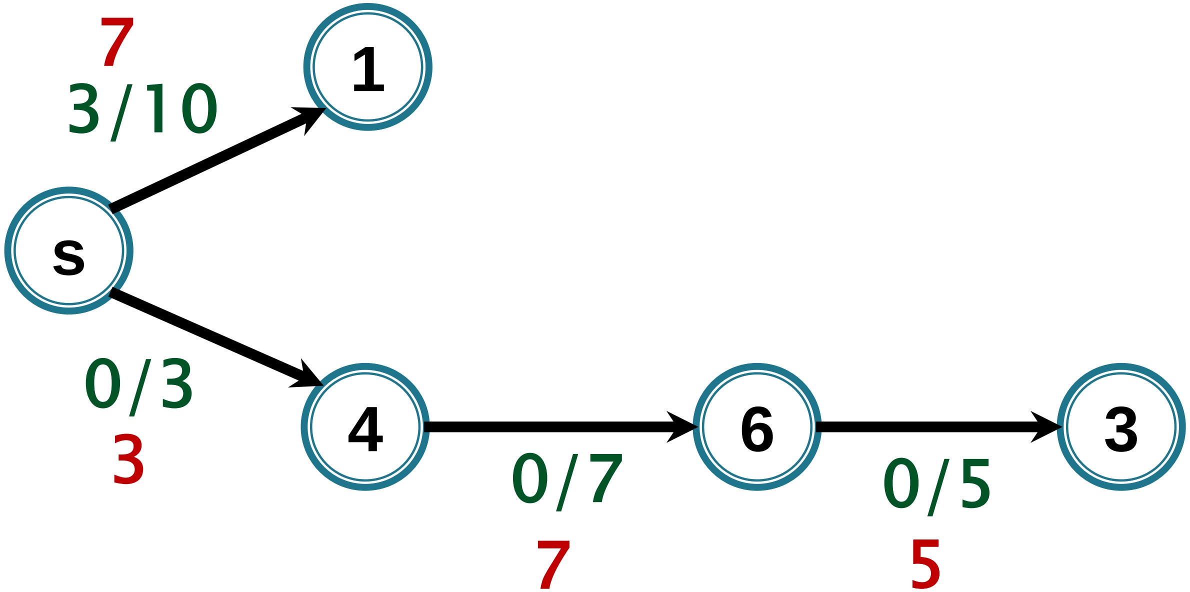

7
3/10
1
s
0
0/3
3
4
6


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

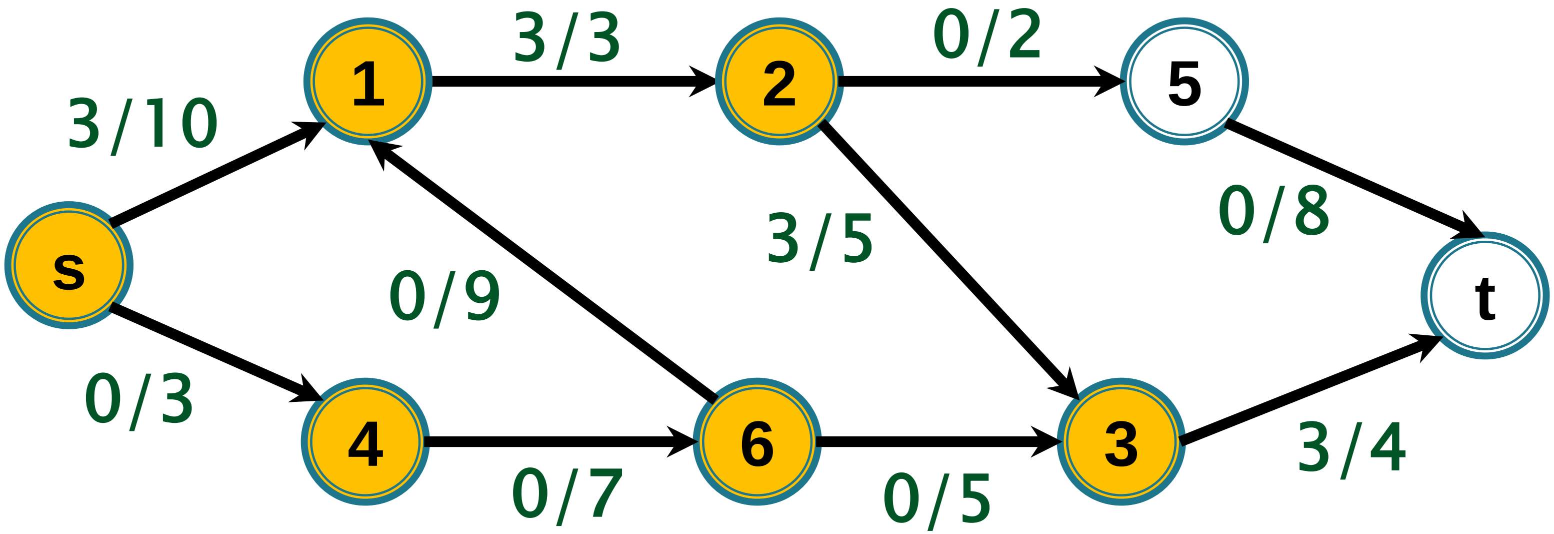

7
3/10
1
s
0/3
3
4
6
0/7
7

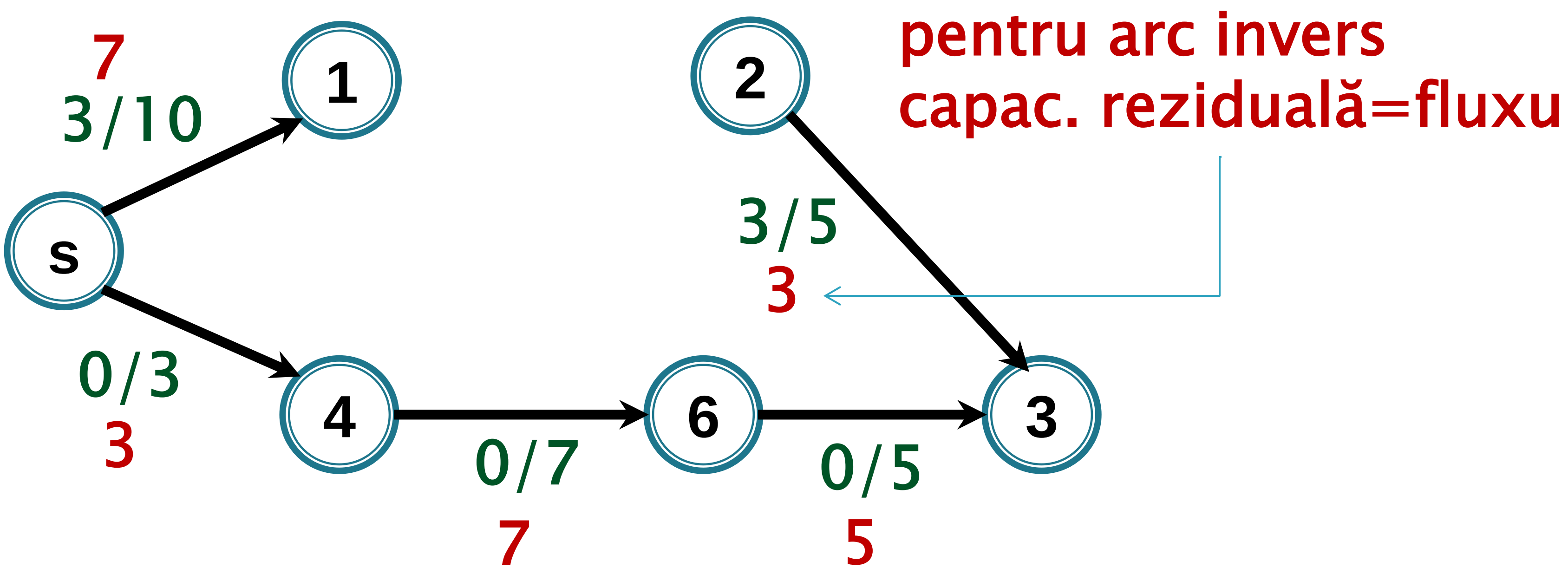


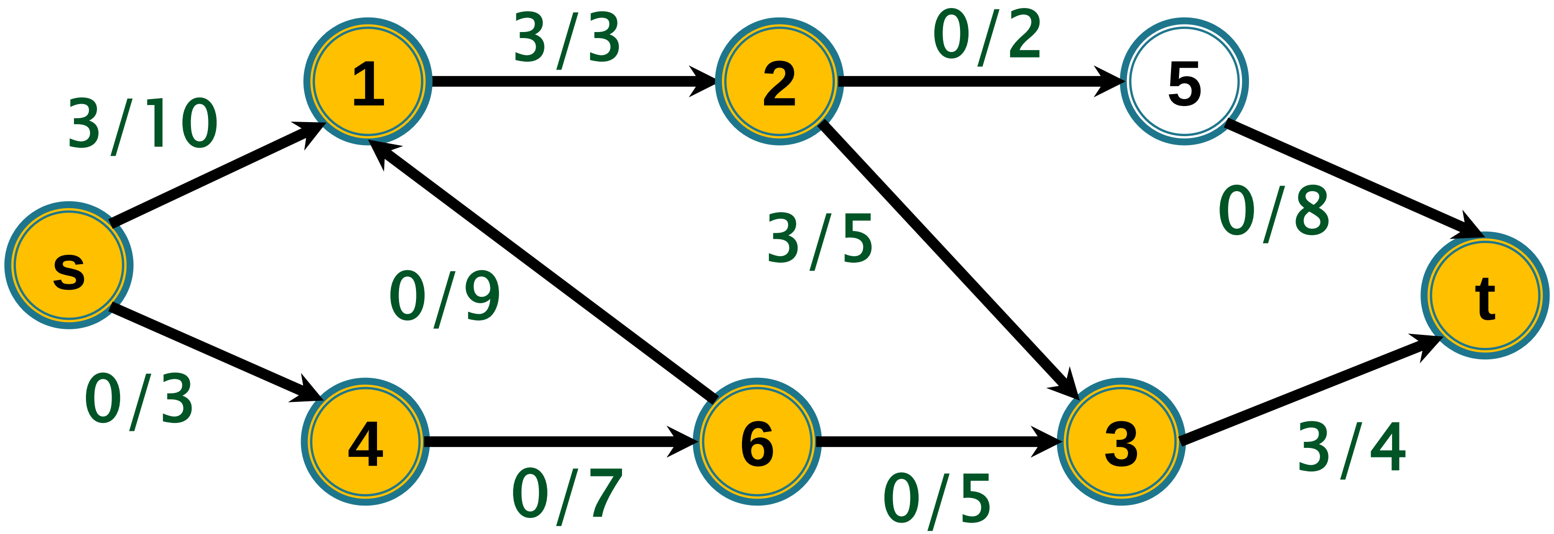
3/3
0/2
3/10
1
2
5
0/8
3/5
0/9
s
t
0/3
4
6
3
3/4
0/7
0/5



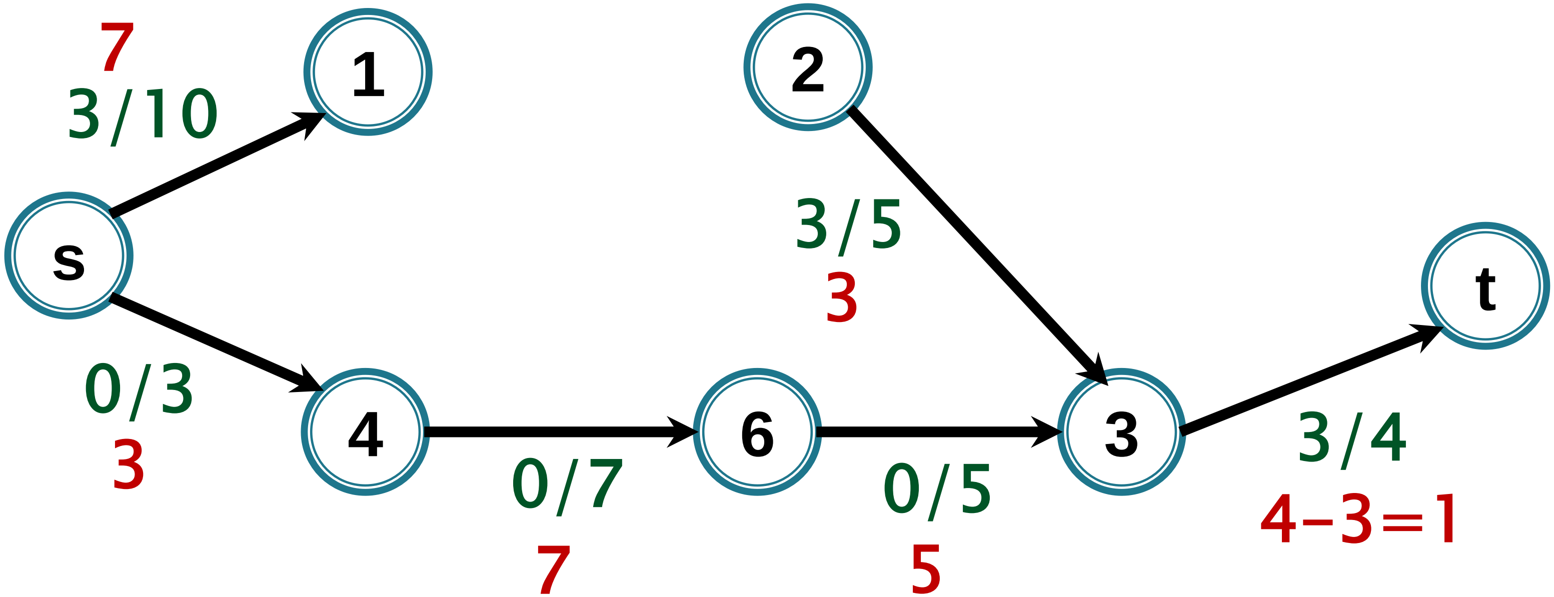
7
3/10
1
s
0/3
3
4
6
3
0/7
0/5
7
5


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

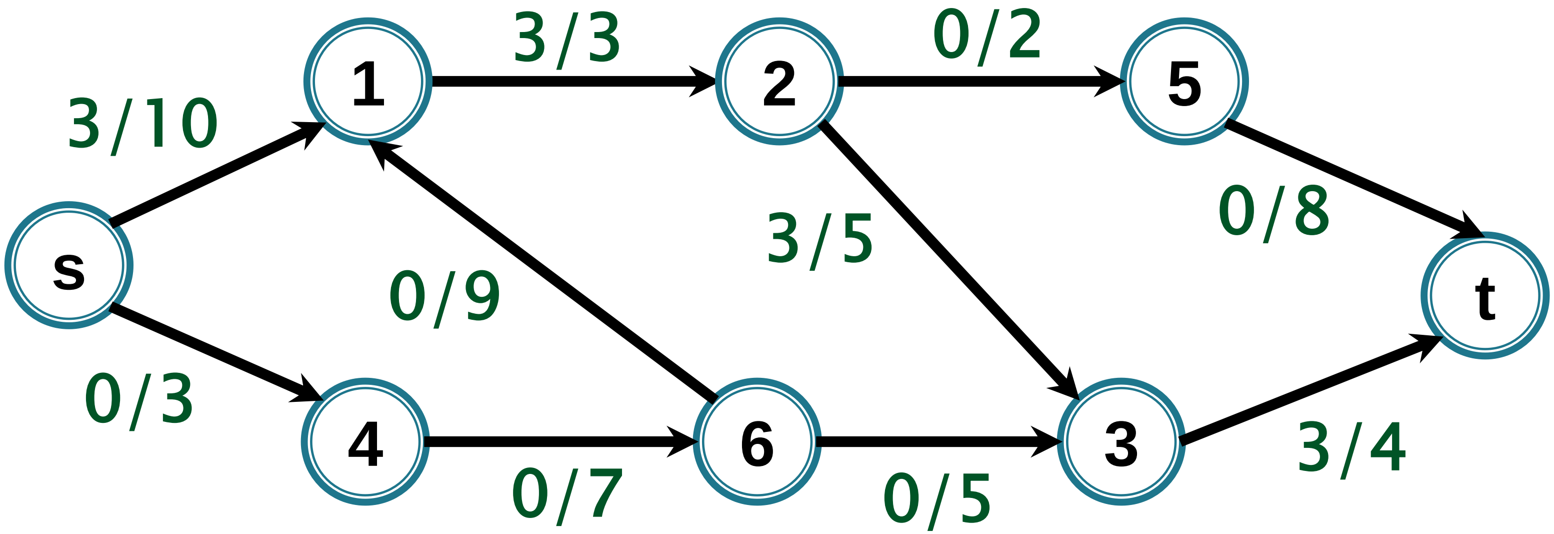


3/10
3/3
0/2
1
2
5
s
0/9
3/5
0/8
t
0/3
4
6
3
3/4
0/7
0/5

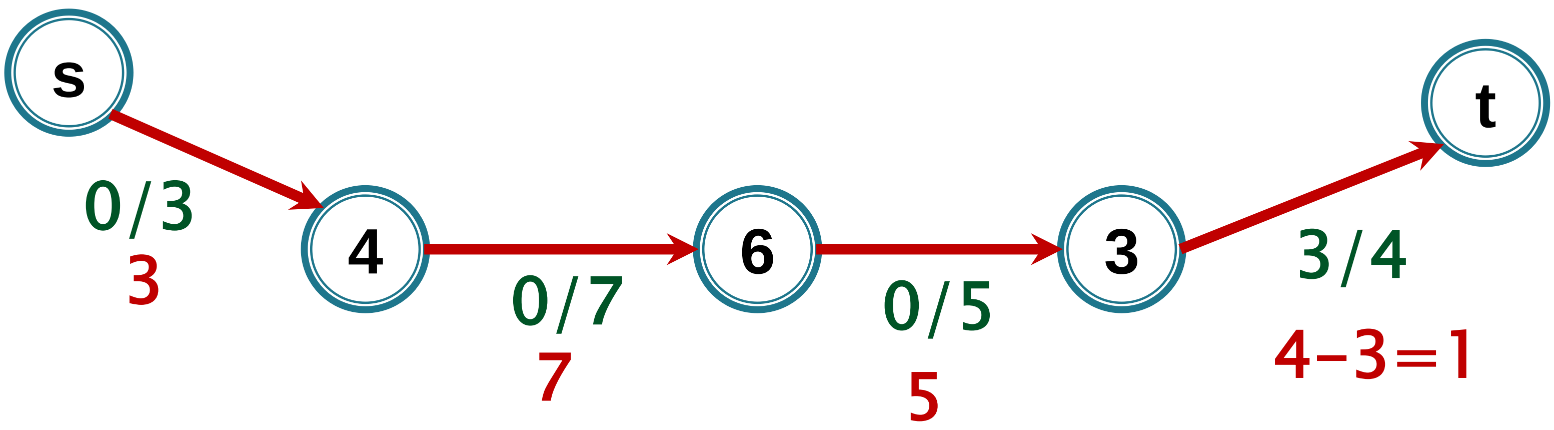


7
3/10
1
2
s
3/5
3
t
0/3
3
4
6
3
3/4
0/7
0/5
4-3=1
7
5

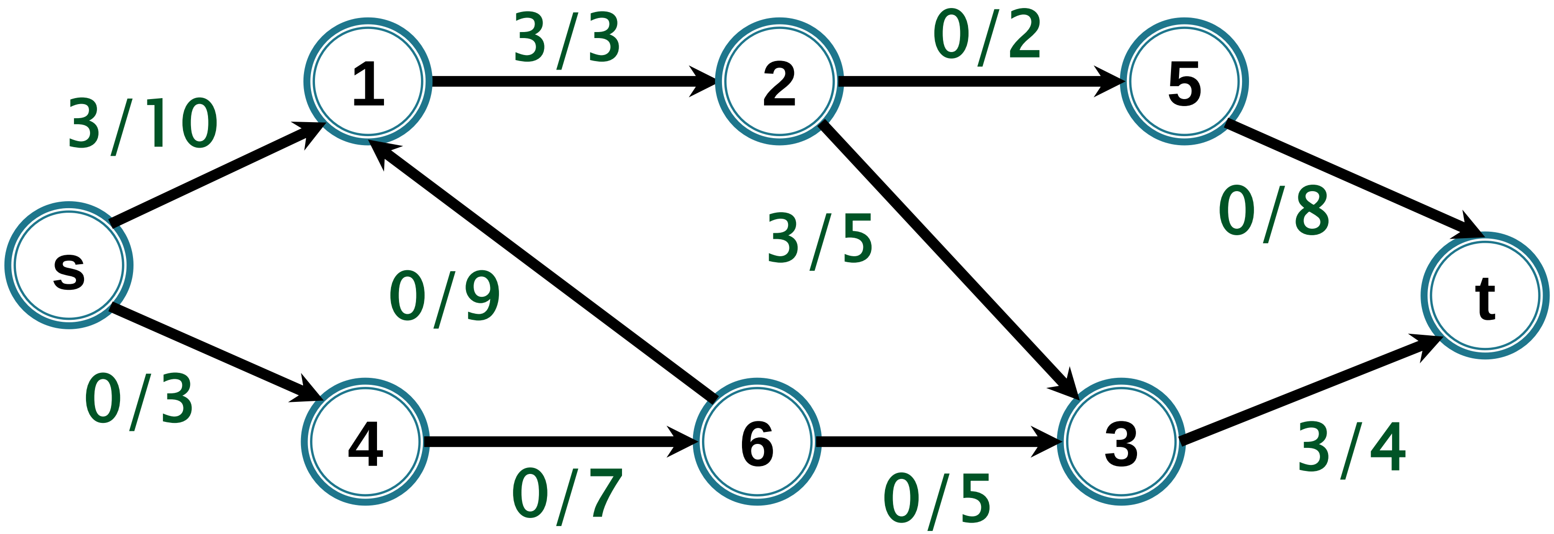
revizuieste_flux_lant

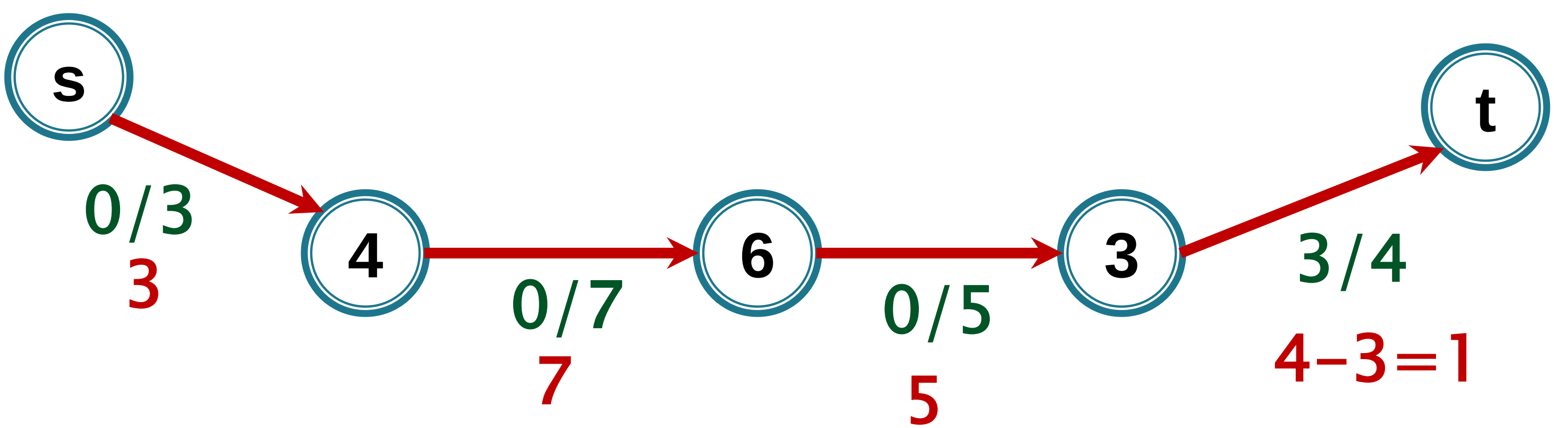


3/10
1
3/3
2
0/2
5
s
0/9
3/5
0/8
t
0/3
4
6
3
3/4
0/7
0/5

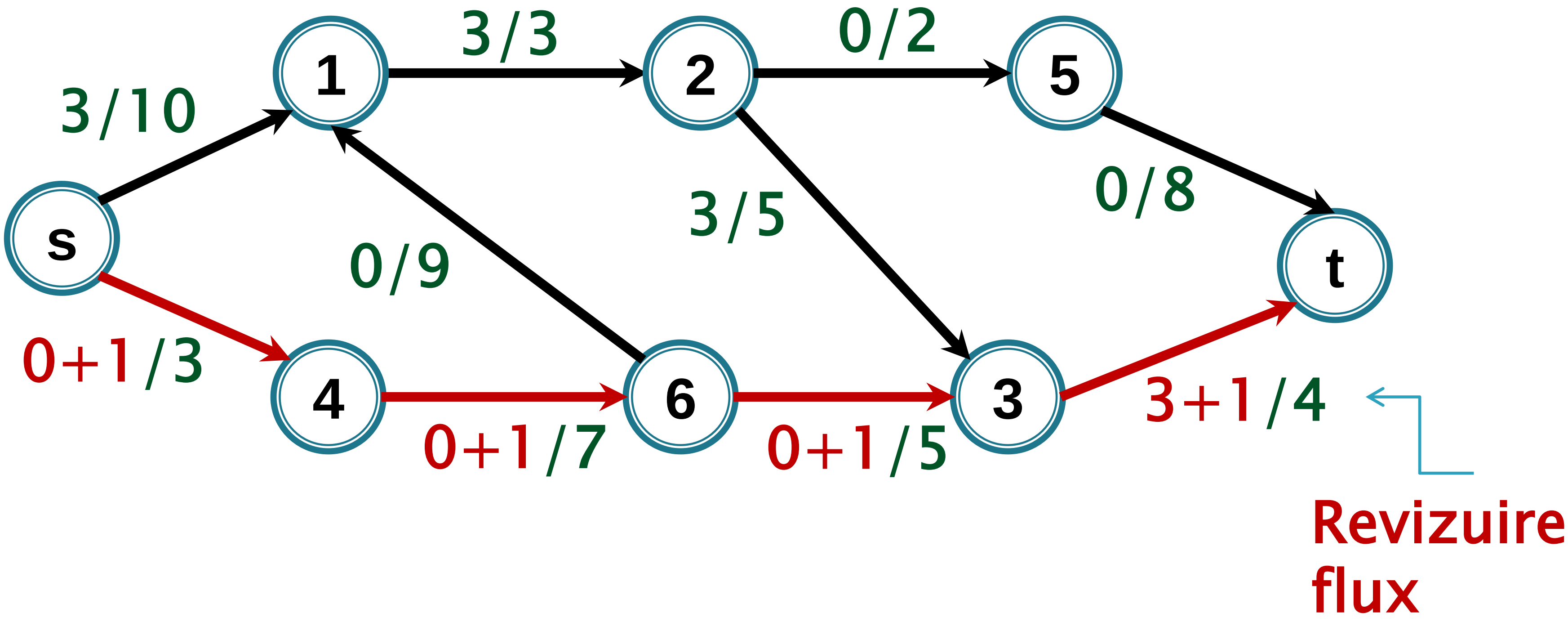


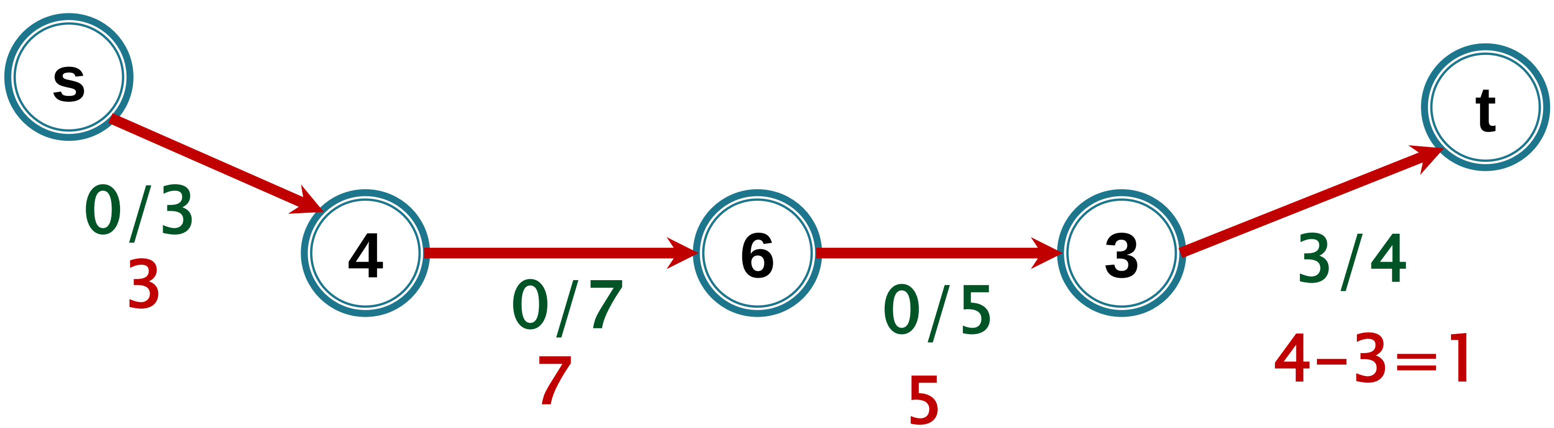
s
t
0/3
3
4
6
3
3/4
0/7
7
0/5
5
4−3=1



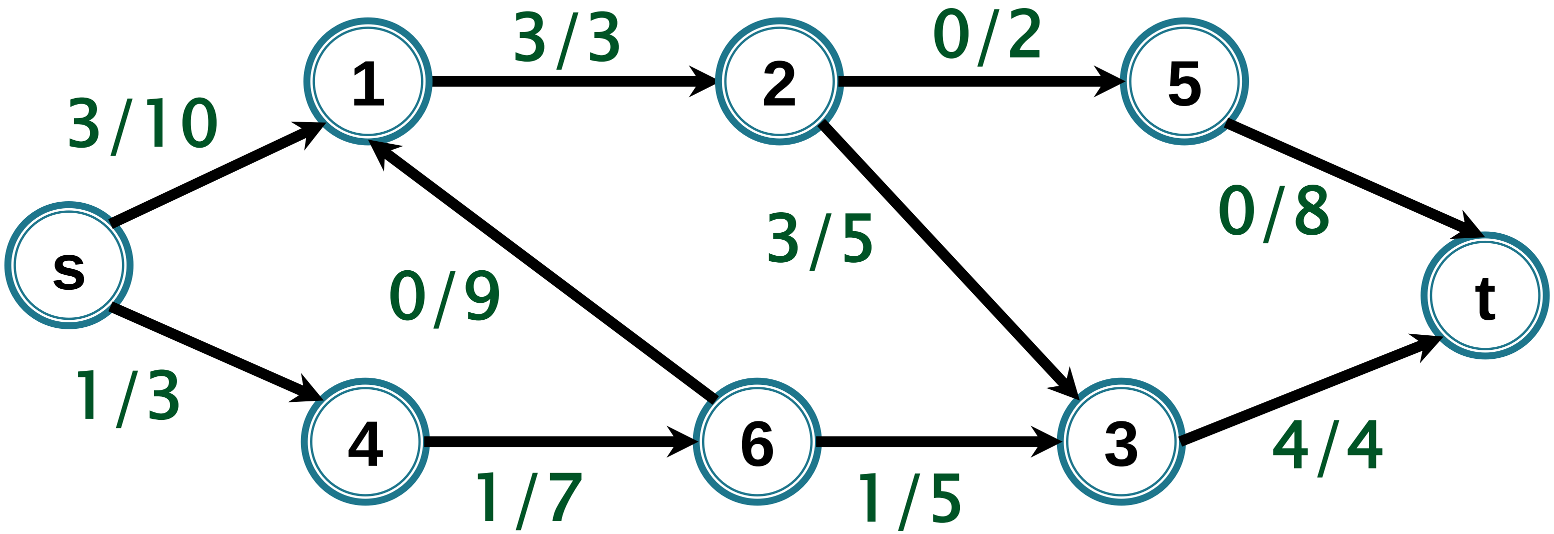


$$i(P) = \min\{3, 7, 5, 1\} = 1$$

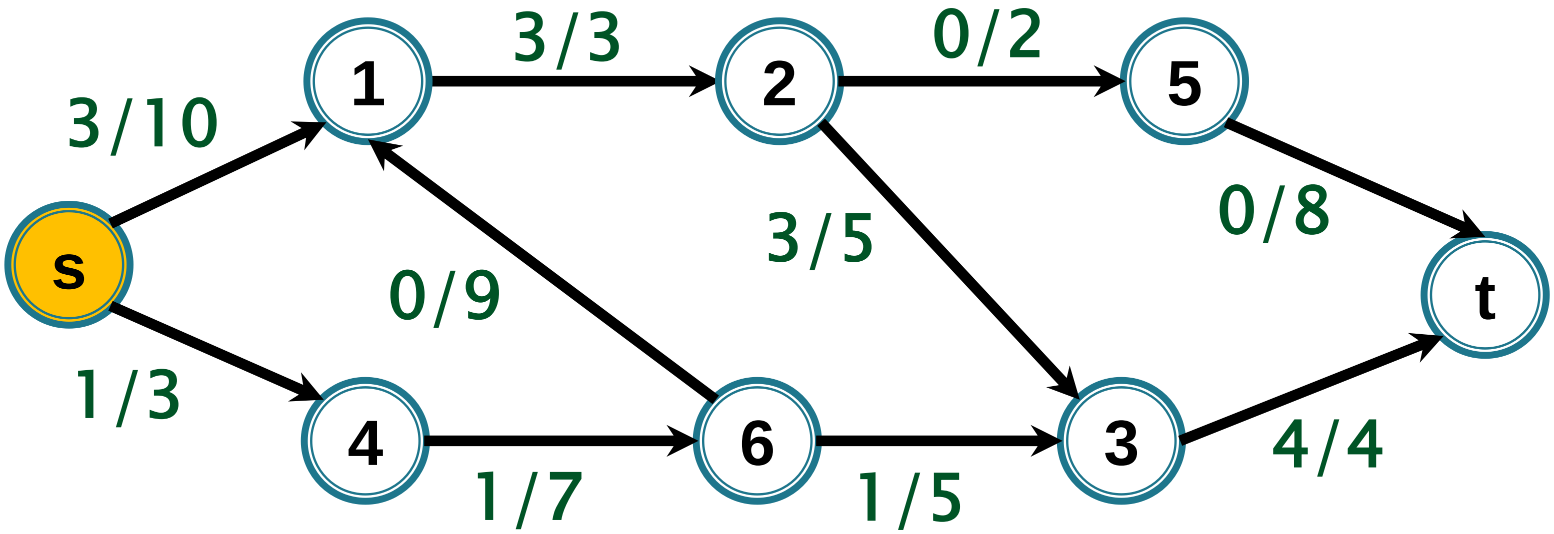

s
1
2
5
t
4
6
3
3/10
3/3
0/2
0/8
3/5
0/9
0+1/3
0+1/7
0+1/5
3+1/4
Revizuire flux


s
4
6
3
t
0/3
3
0/7
7
0/5
5
3/4
4−3=1

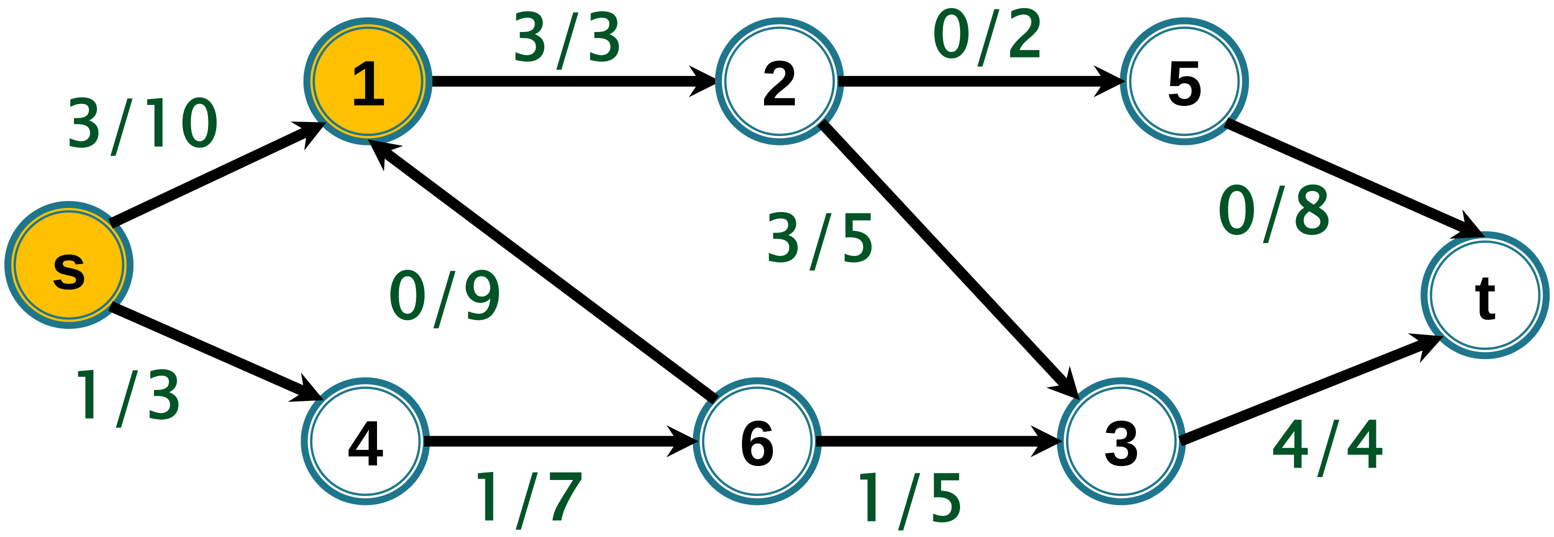
$i(P) = \min\{3, 7, 5, 1\} = 1$

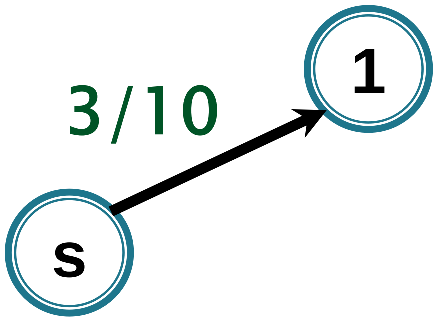

s
1
2
5
t
4
6
3
3/10
3/3
0/2
0/8
3/5
0/9
1/3
1/7
1/5
4/4

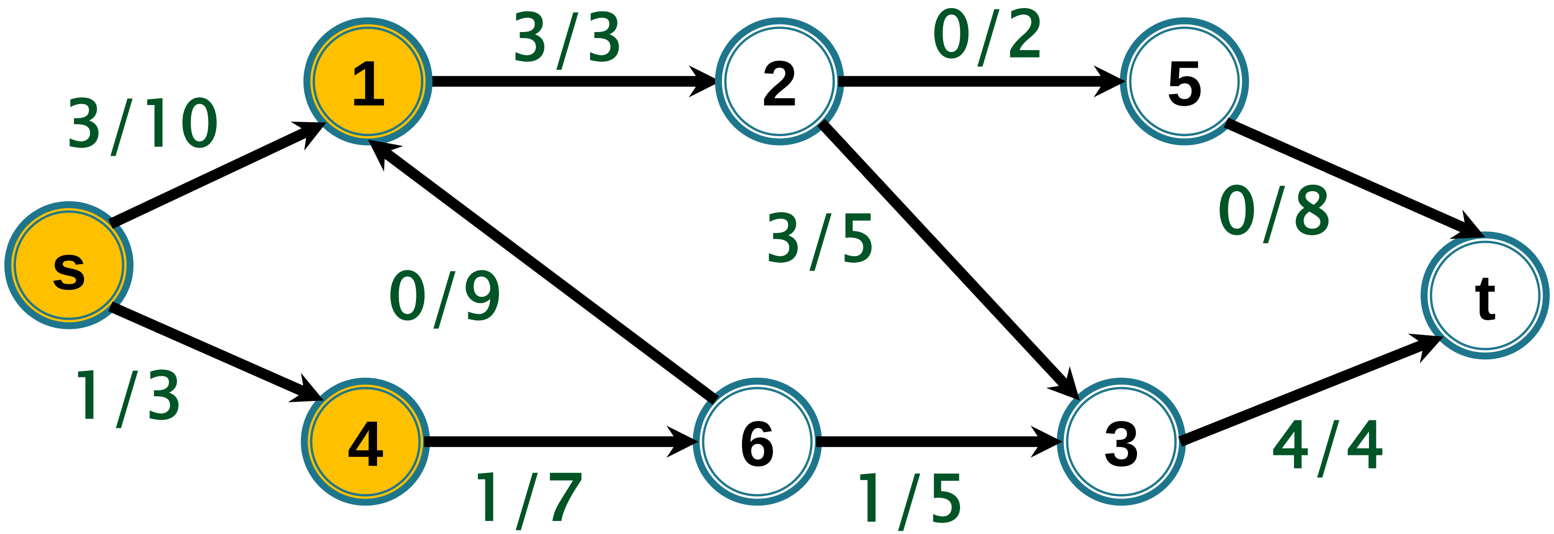
construieste_s-t_lant_nesat_BF

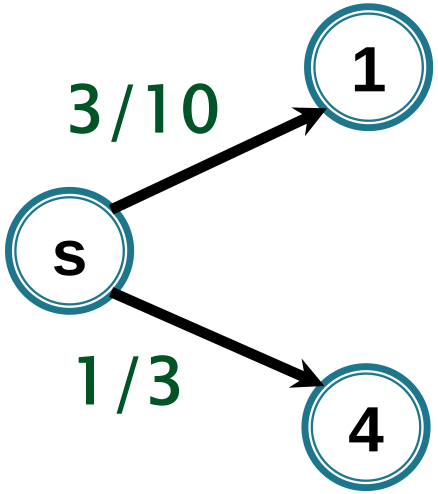

s
1
2
5
t
4
6
3
3/10
3/3
0/2
0/8
3/5
0/9
1/3
1/7
1/5
4/4

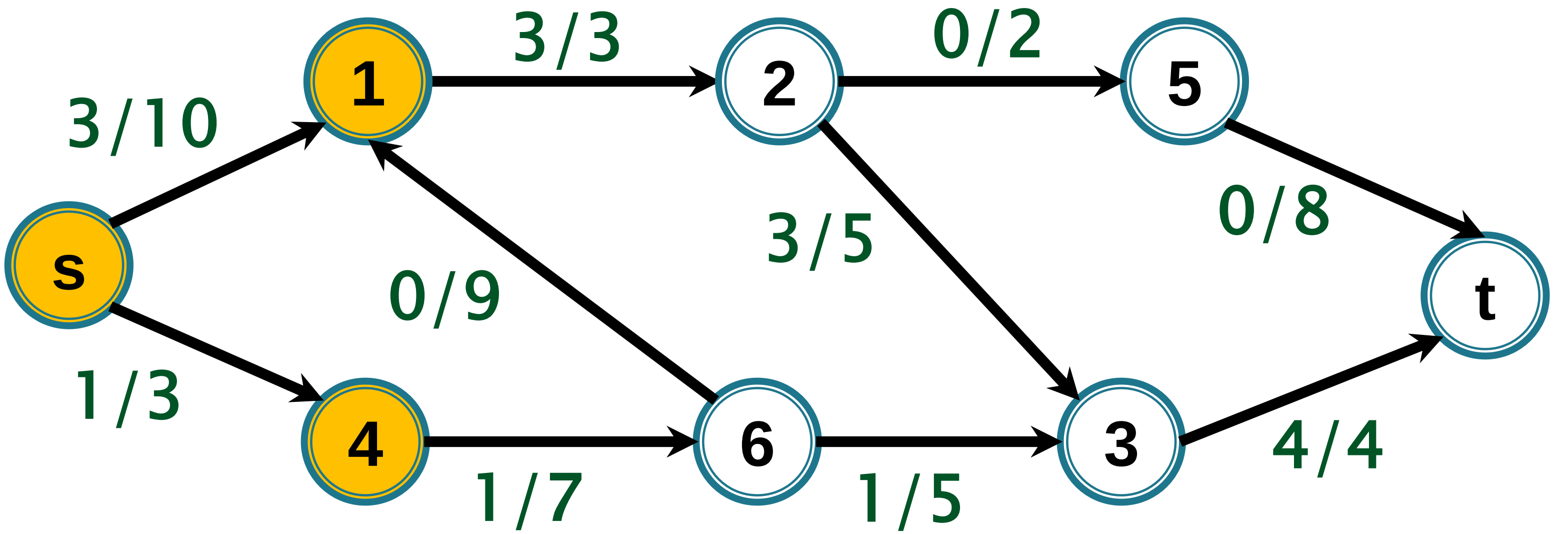

s

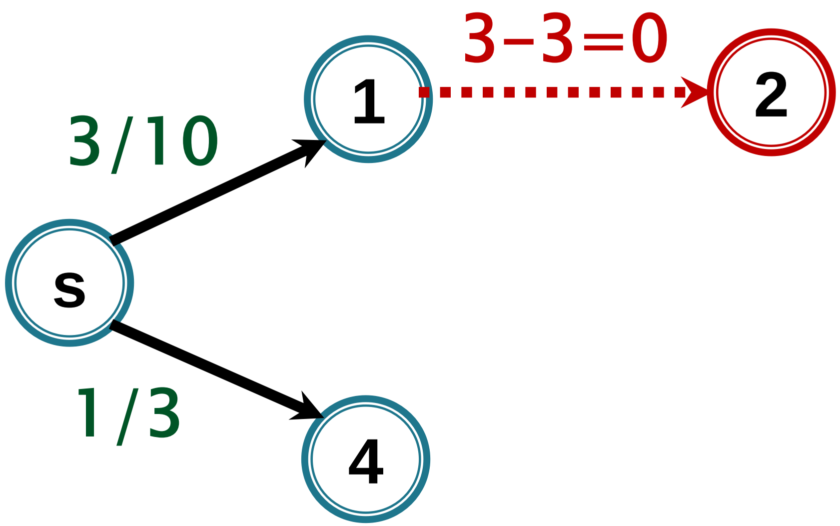

s
1
2
5
t
4
6
3
3/10
3/3
0/2
0/8
3/5
0/9
1/3
1/7
1/5
4/4

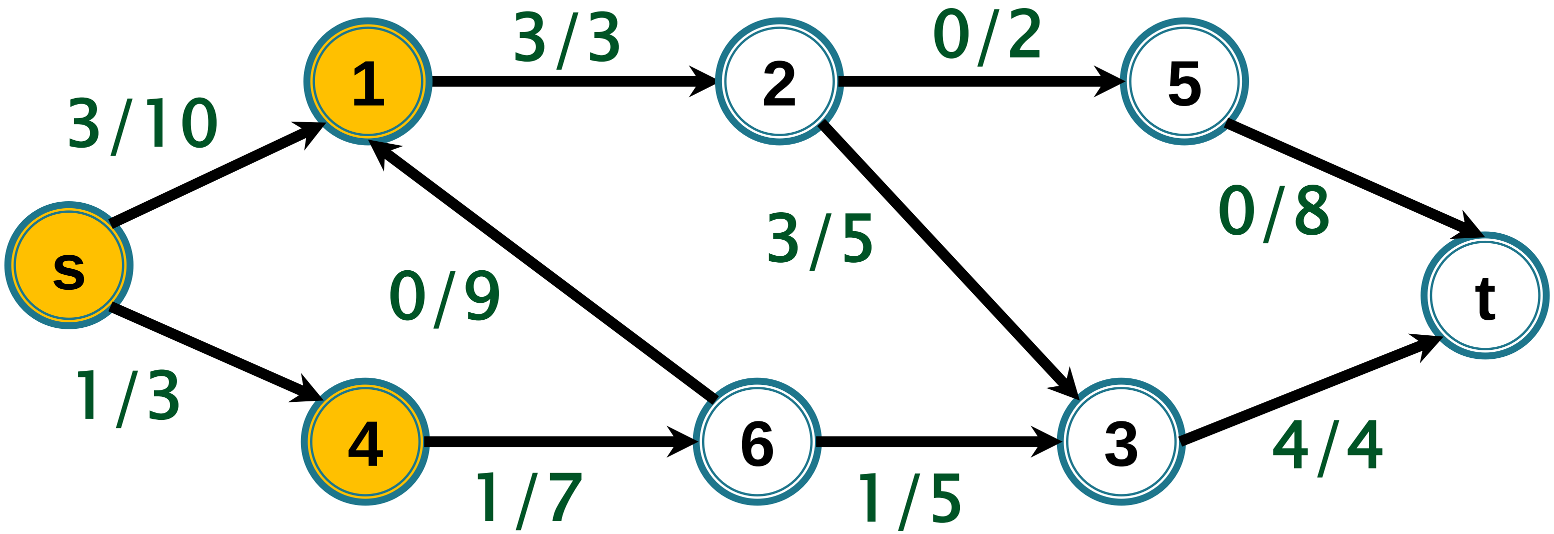

s
1
3/10


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

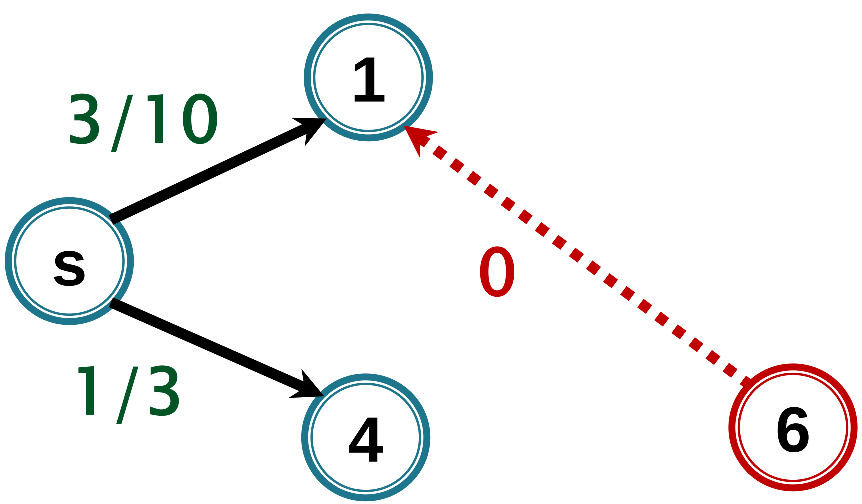

s
1
4
3/10
1/3


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

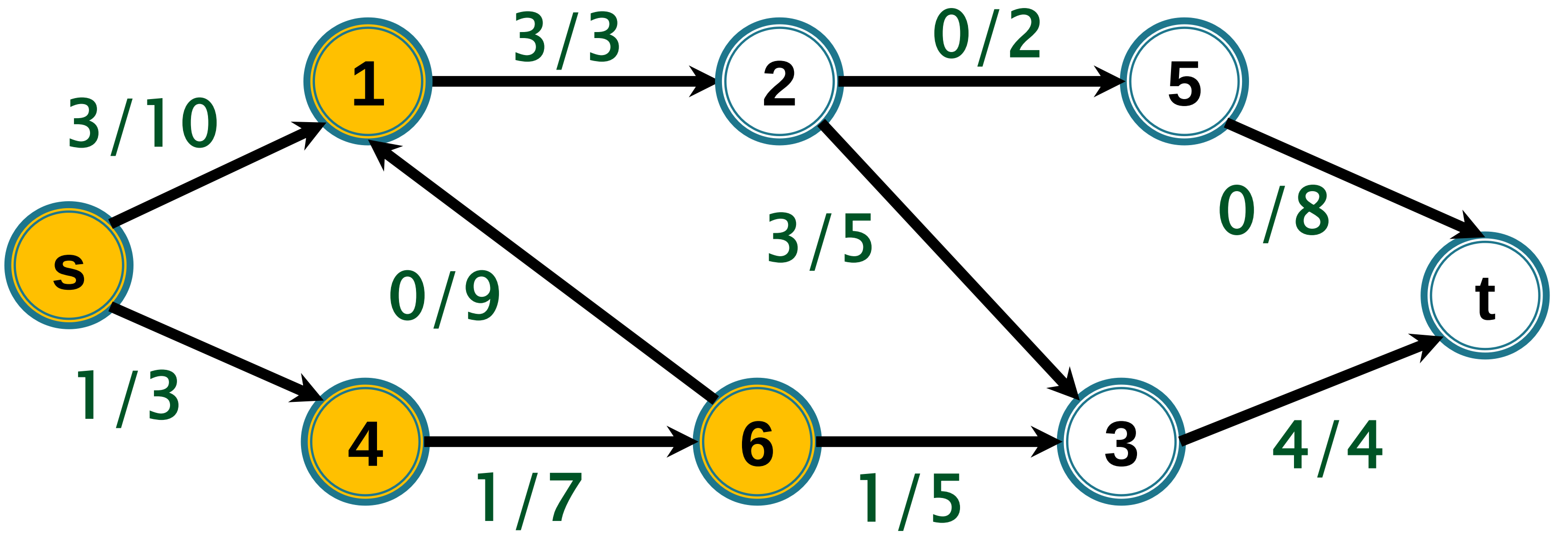

3–3=0
1
2
3/10
s
1/3
4

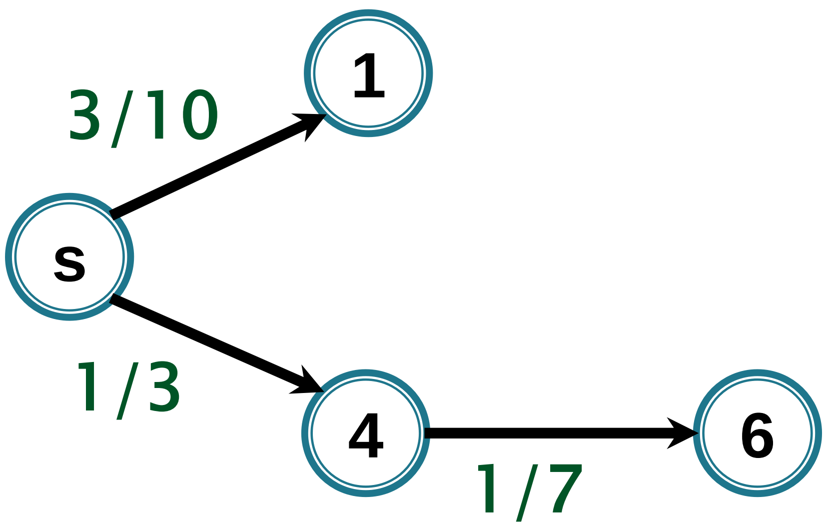


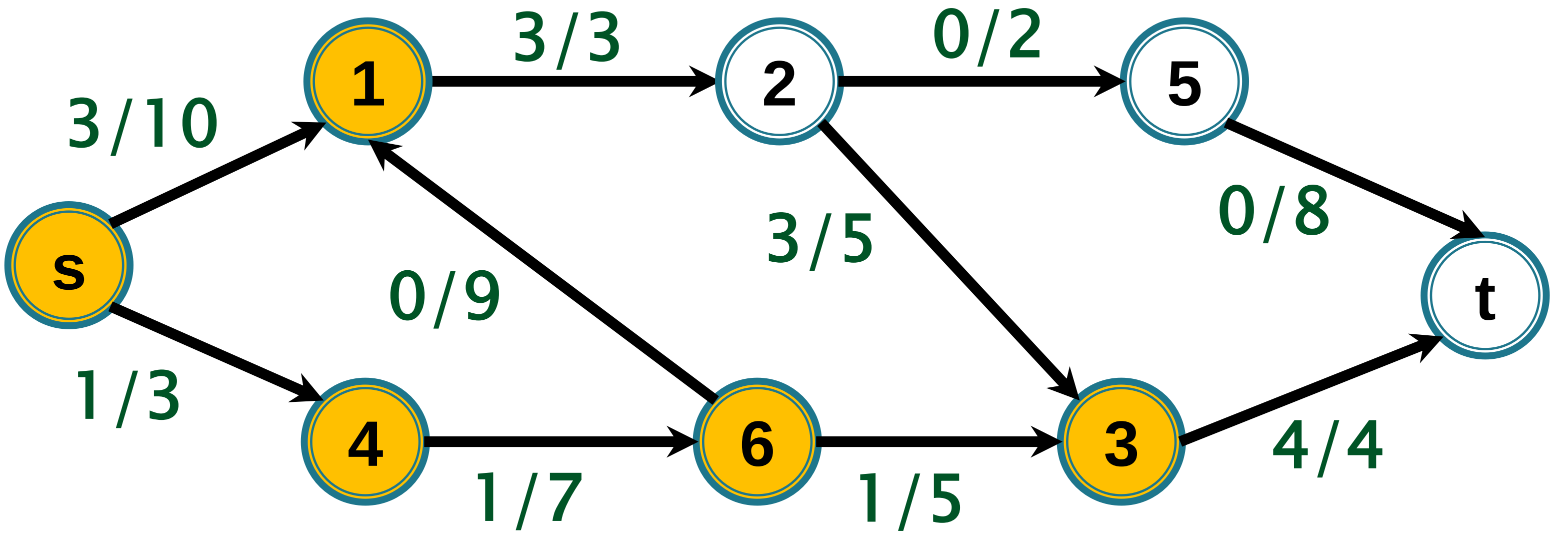
s
1
2
5
t
4
6
3
3/10
1/3
3/3
0/2
0/9
3/5
0/8
1/7
1/5
4/4

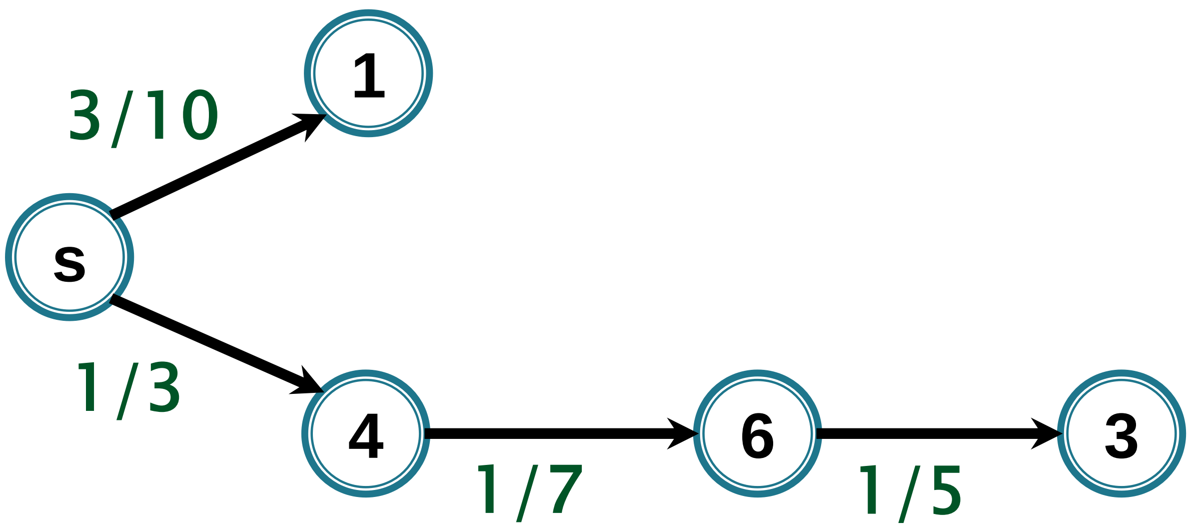


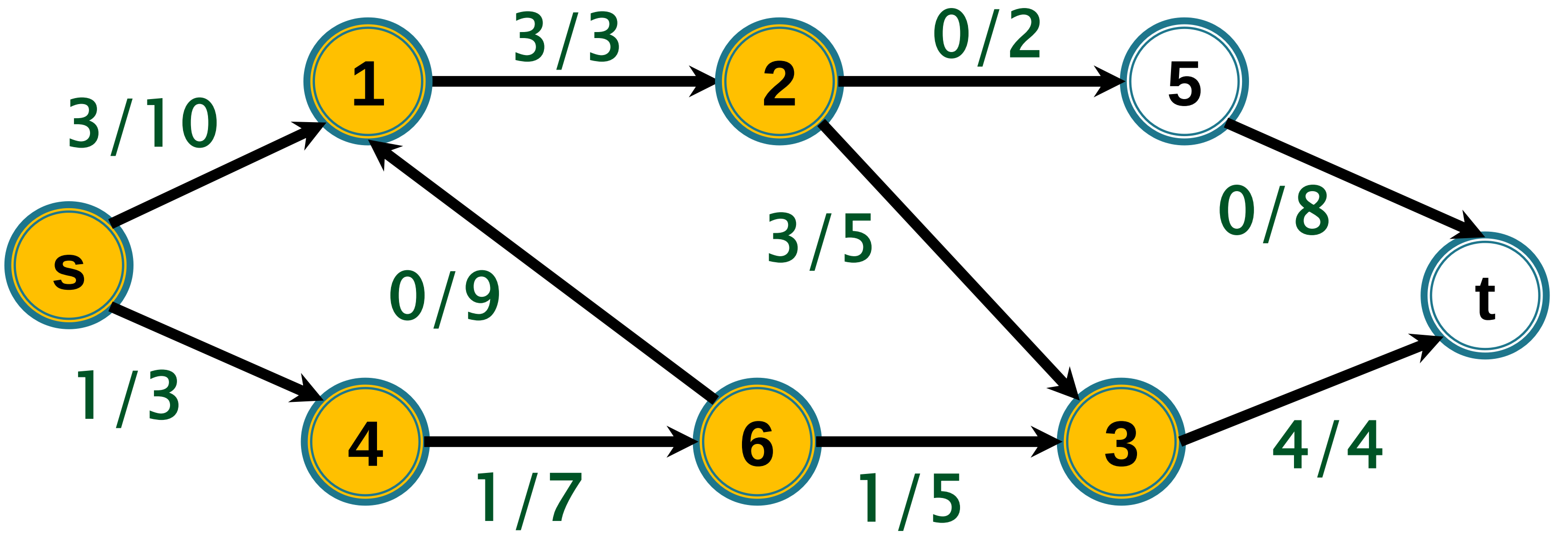
s
1
4
6
3/10
1/3
0


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

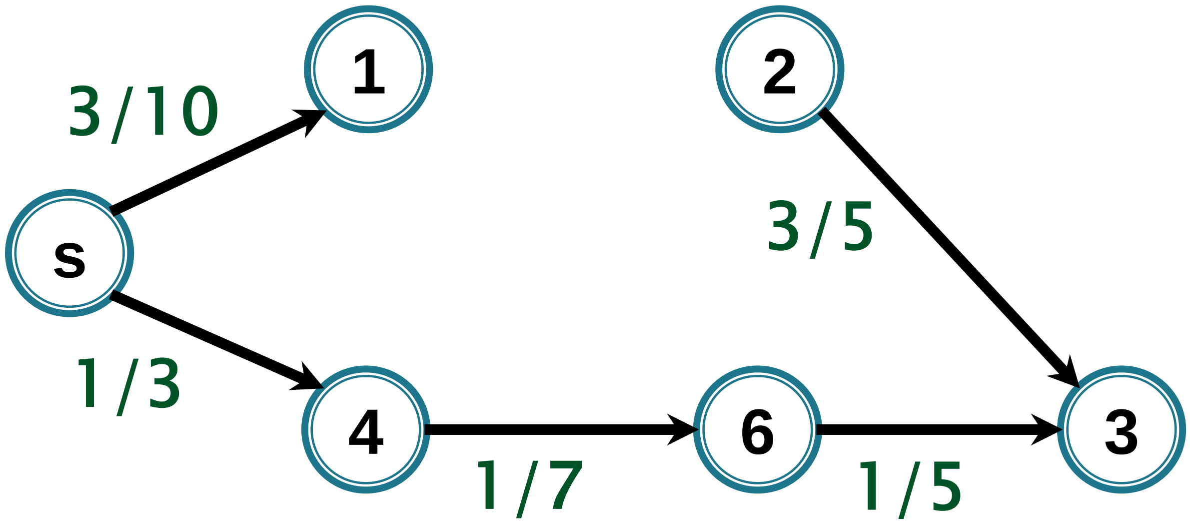

3/10
1
s
1/3
4
6
1/7


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

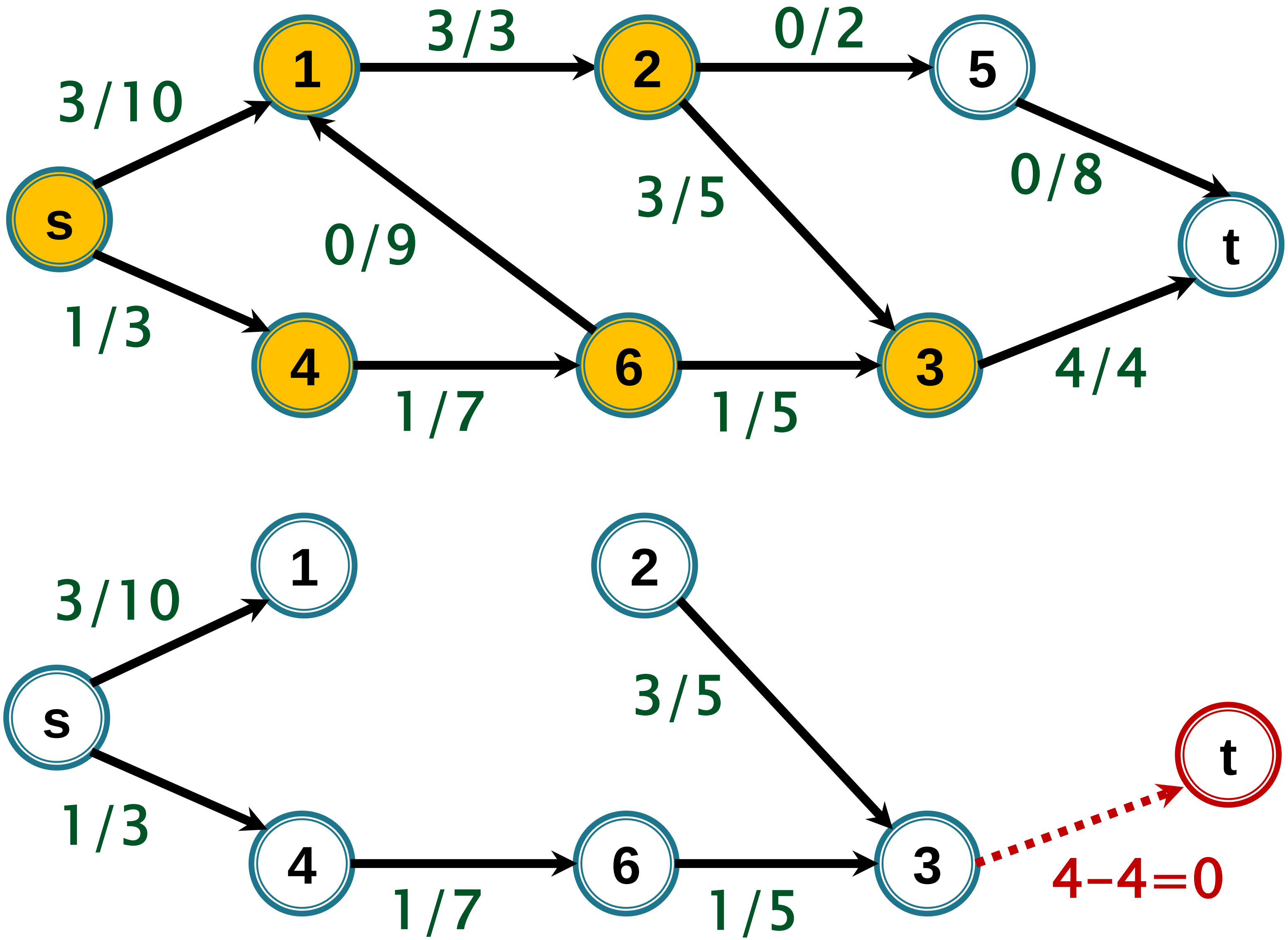

3/10
1/3
1/7
1/5
s
1
4
6
3



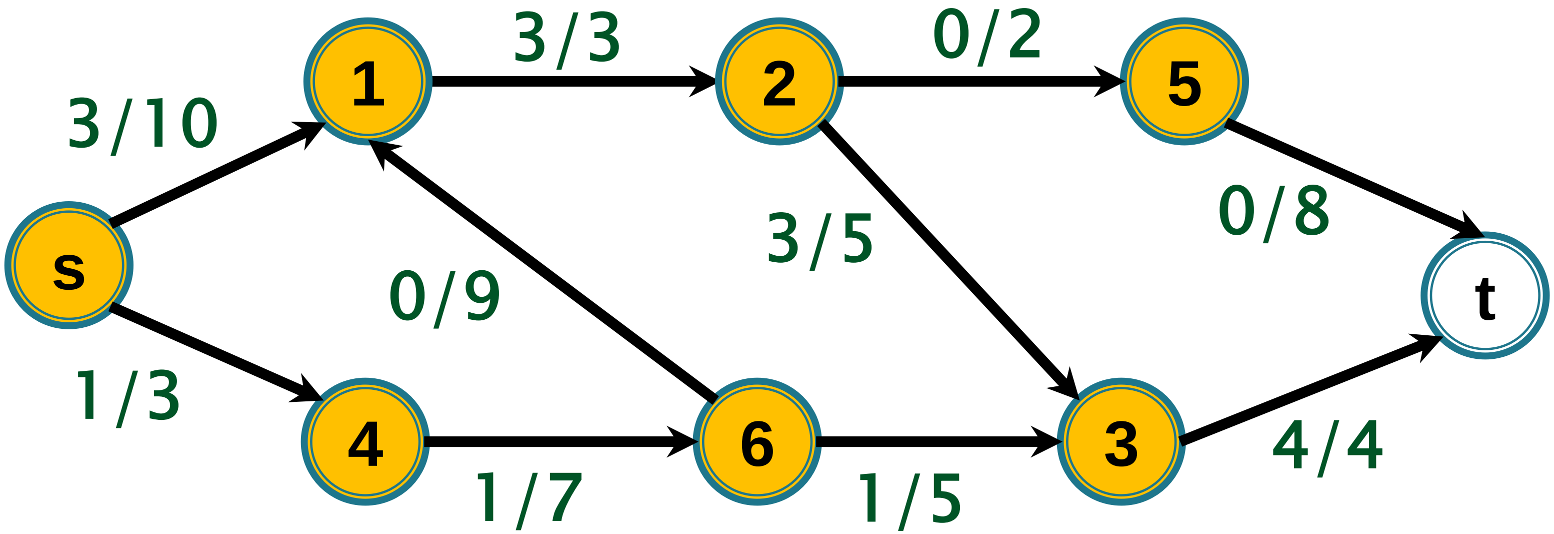
s
1
2
5
t
4
6
3
3/10
3/3
0/2
0/8
3/5
0/9
1/3
1/7
1/5
4/4

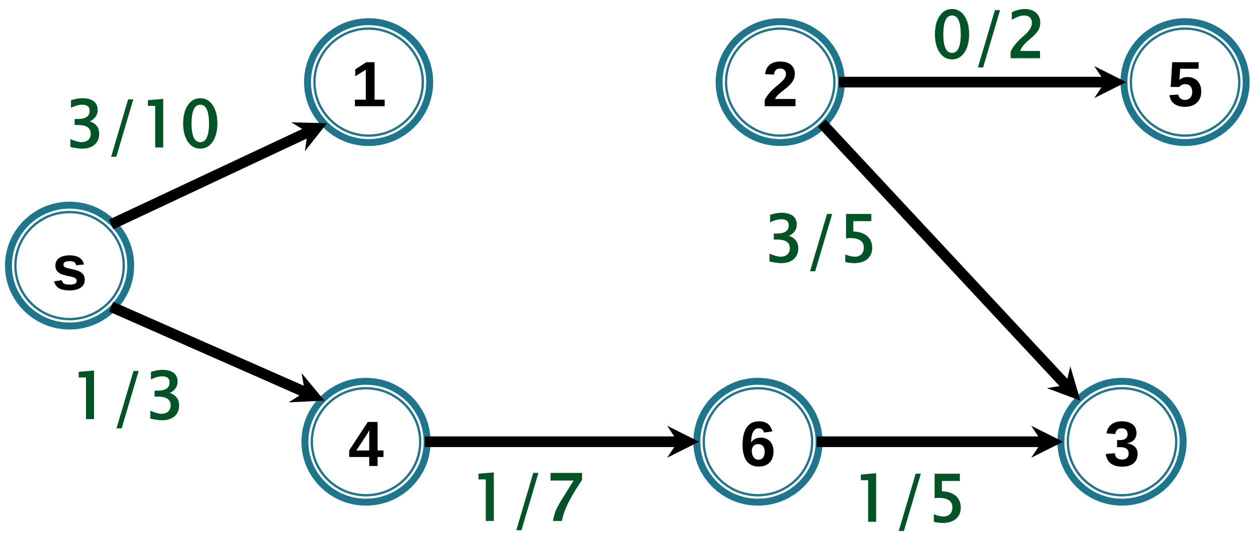


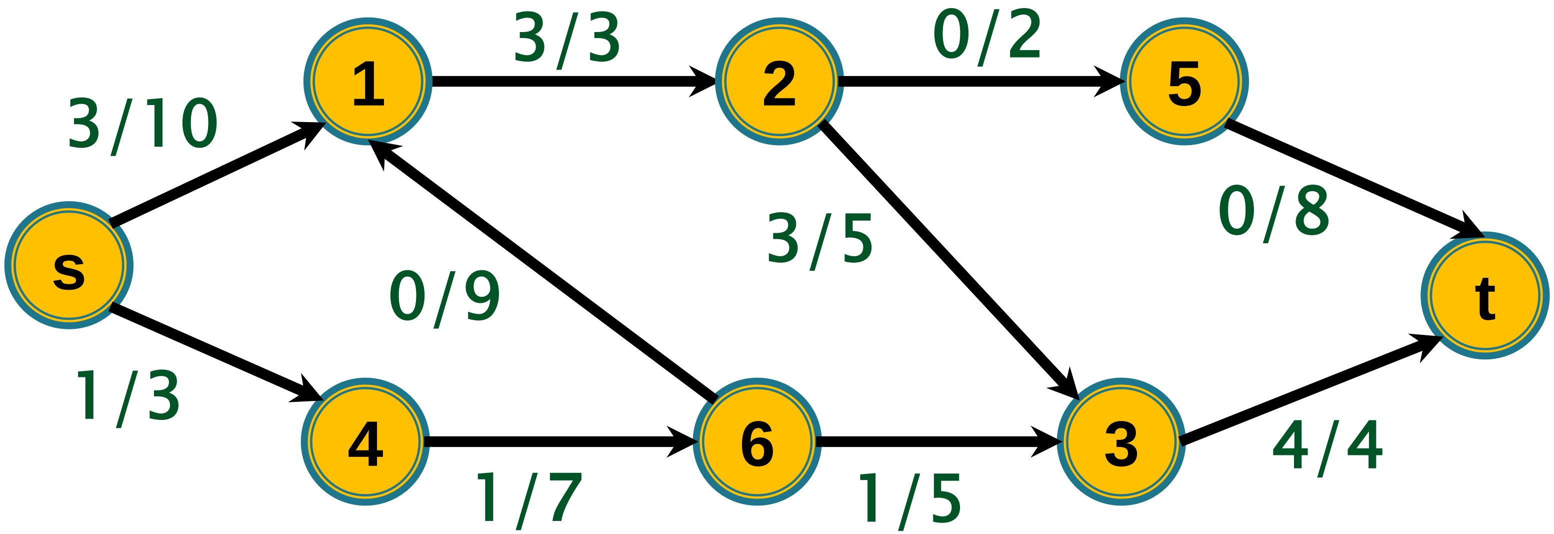
s
1
2
4
6
3
3/10
3/5
1/3
1/7
1/5



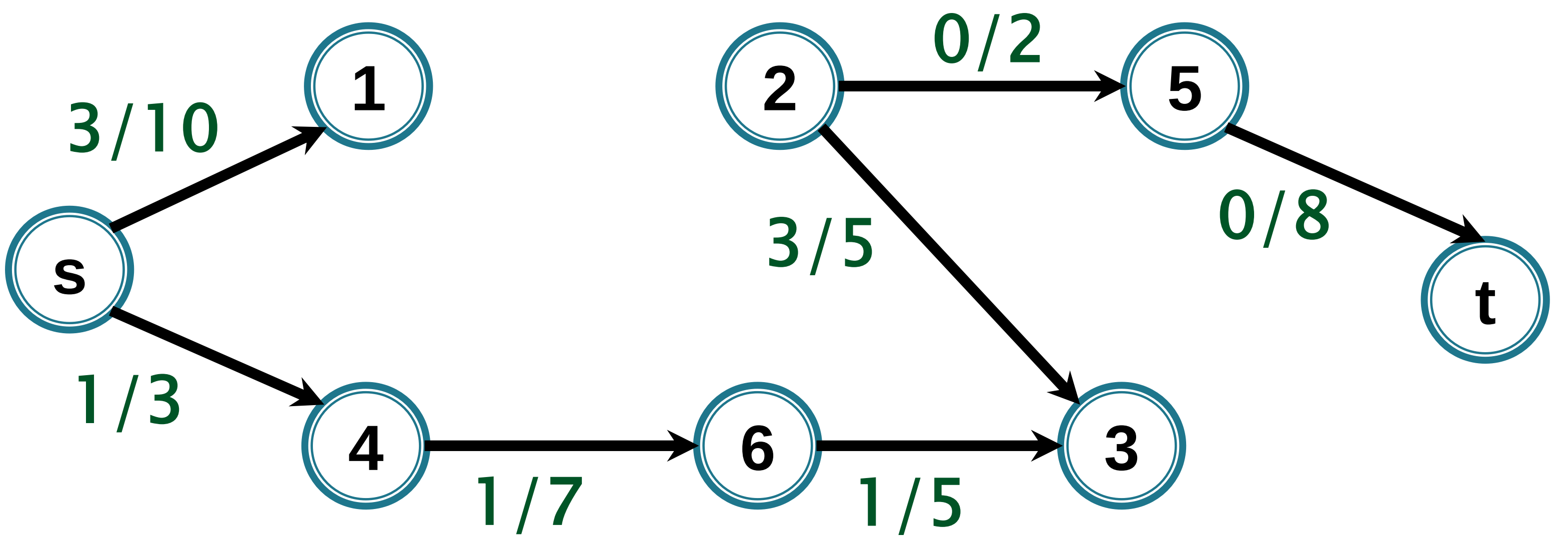
3/3
0/2
3/10
0/8
3/5
0/9
1/3
4/4
1/7
1/5
s
1
2
5
t
4
6
3
3/10
1
2
3/5
s
t
1/3
4
6
3
4−4=0
1/7
1/5


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

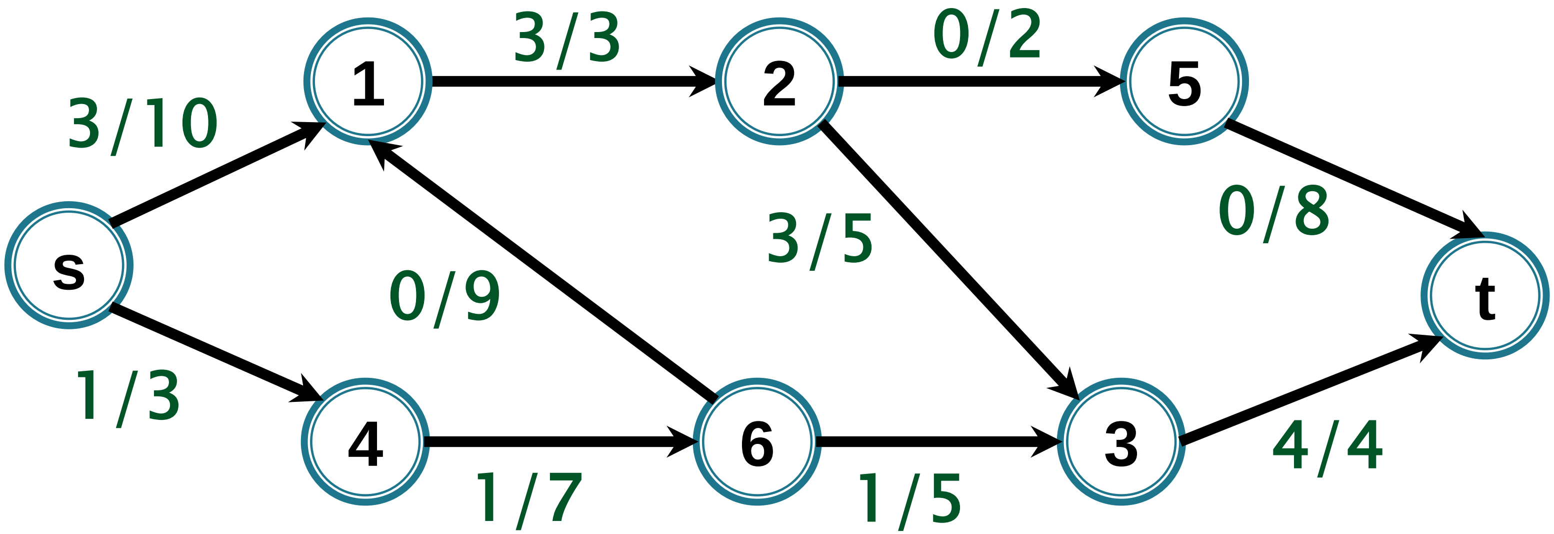


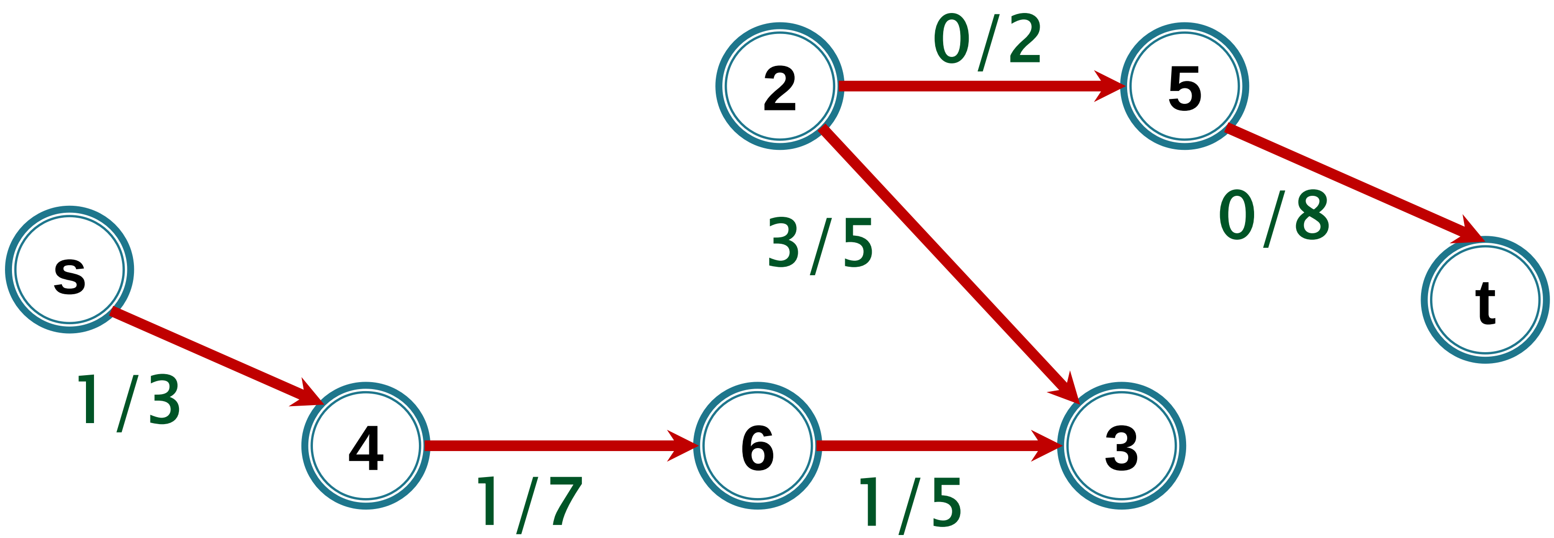
3/10
3/3
0/2
1
2
5
s
0/9
3/5
0/8
t
1/3
4
6
3
4/4
1/7
1/5

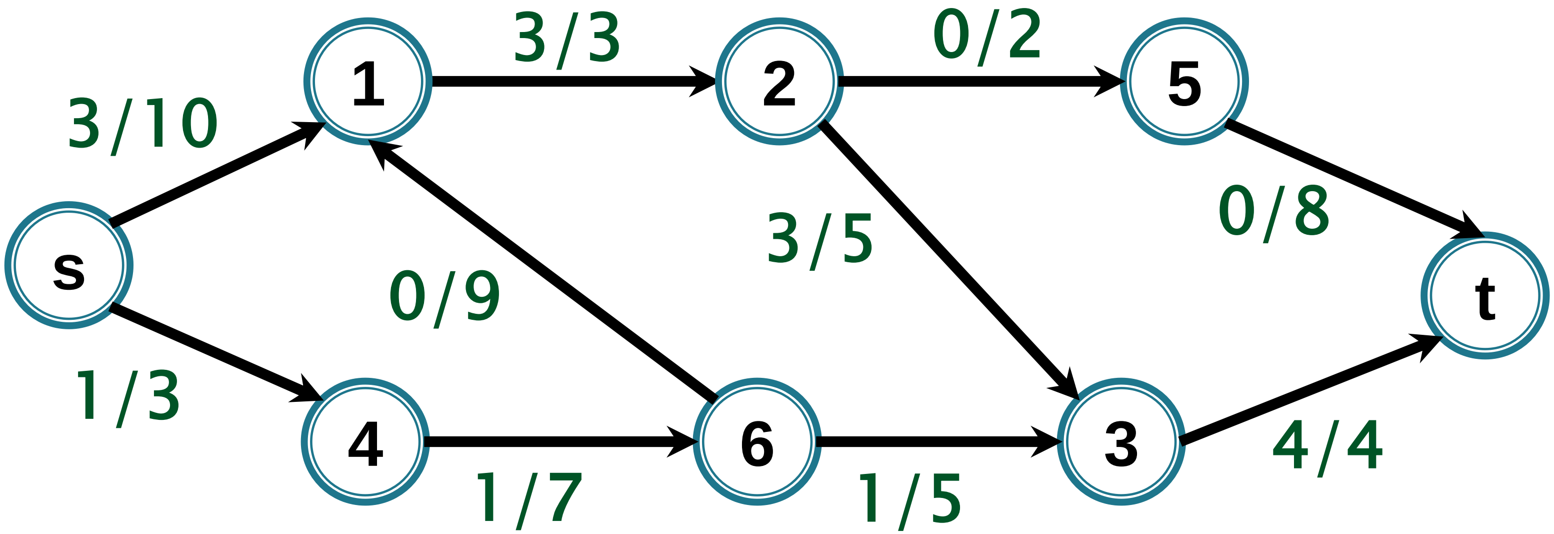


0/2
1
2
5
3/10
3/5
0/8
s
t
1/3
4
6
3
1/7
1/5

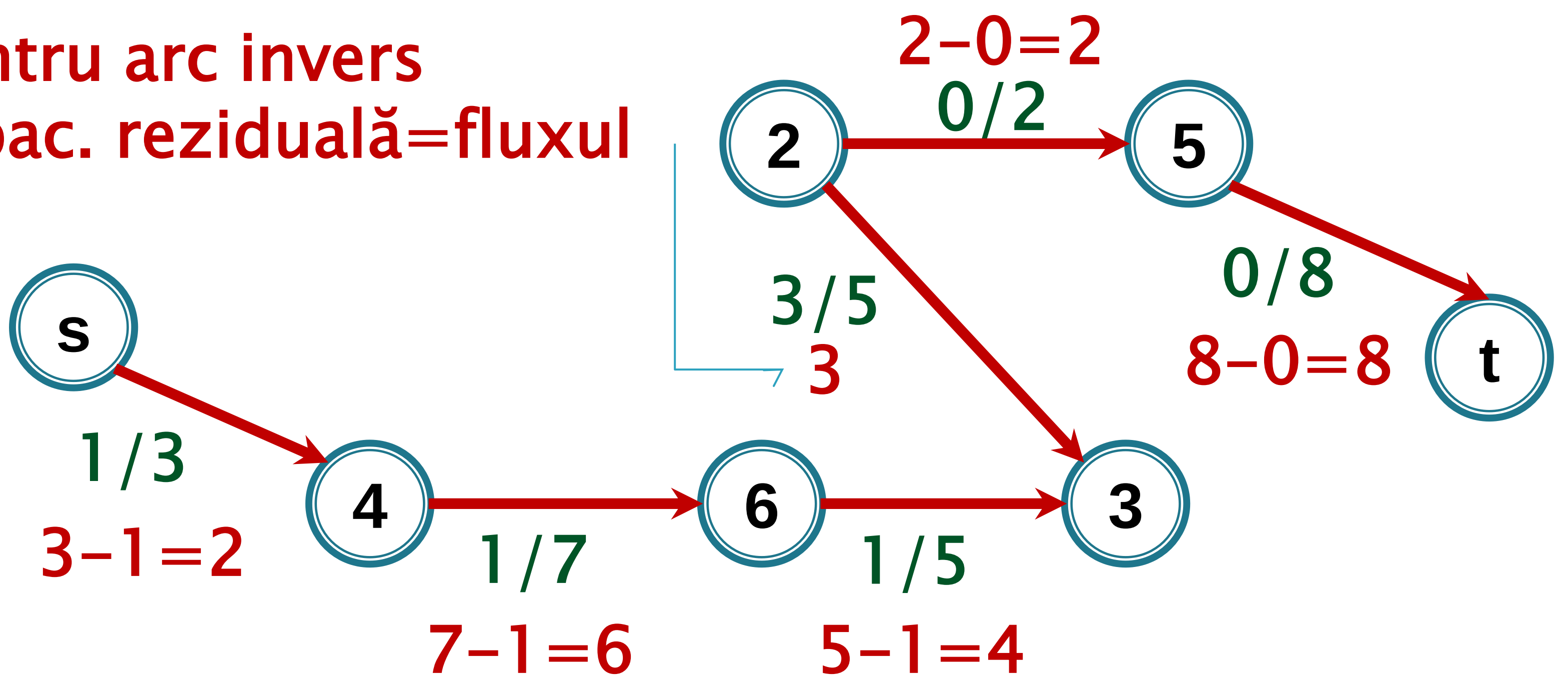
revizuieste_flux_lant

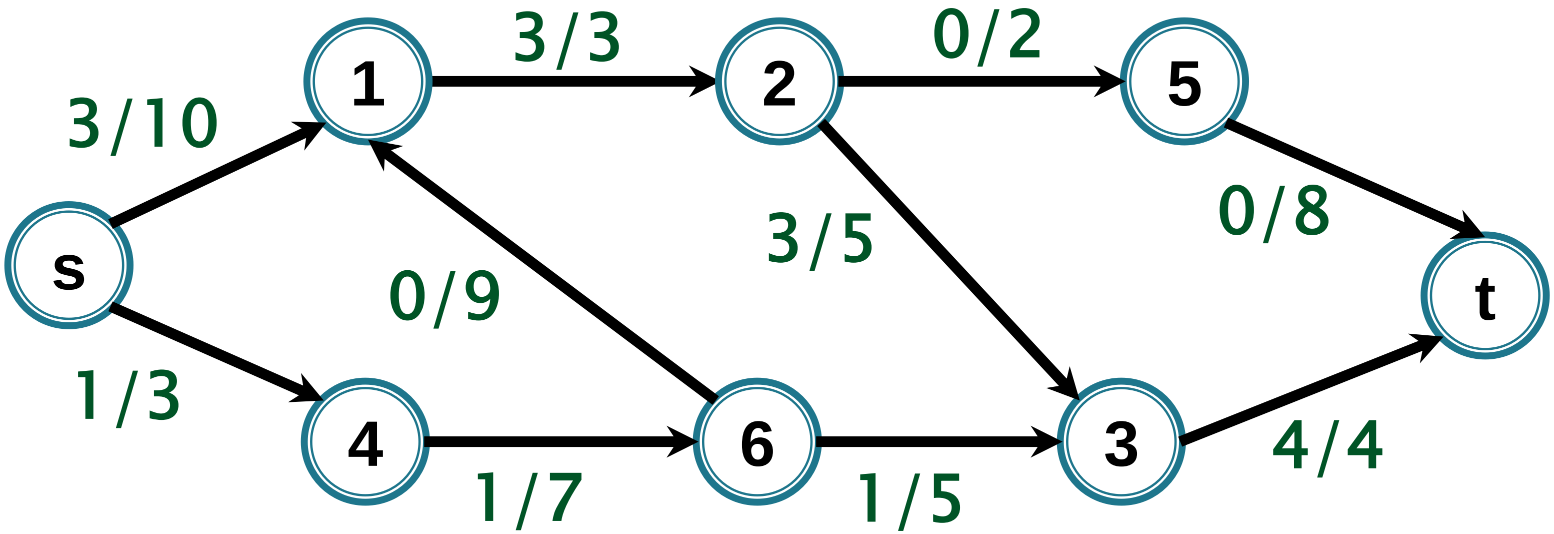

3/3
0/2
1
2
5
3/10
s
0/8
3/5
t
0/9
1/3
4
6
3
4/4
1/7
1/5

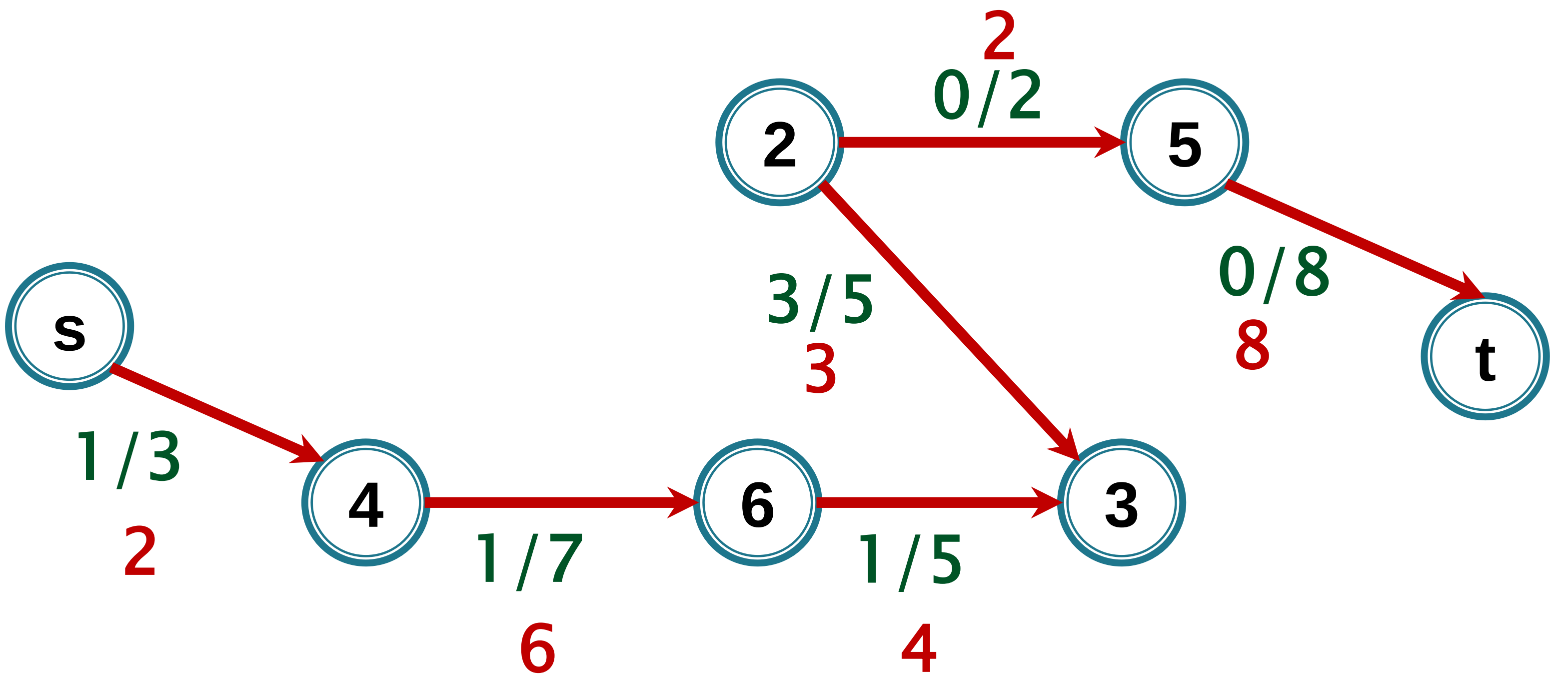

0/2
2
5
0/8
3/5
s
t
1/3
4
6
3
1/7
1/5



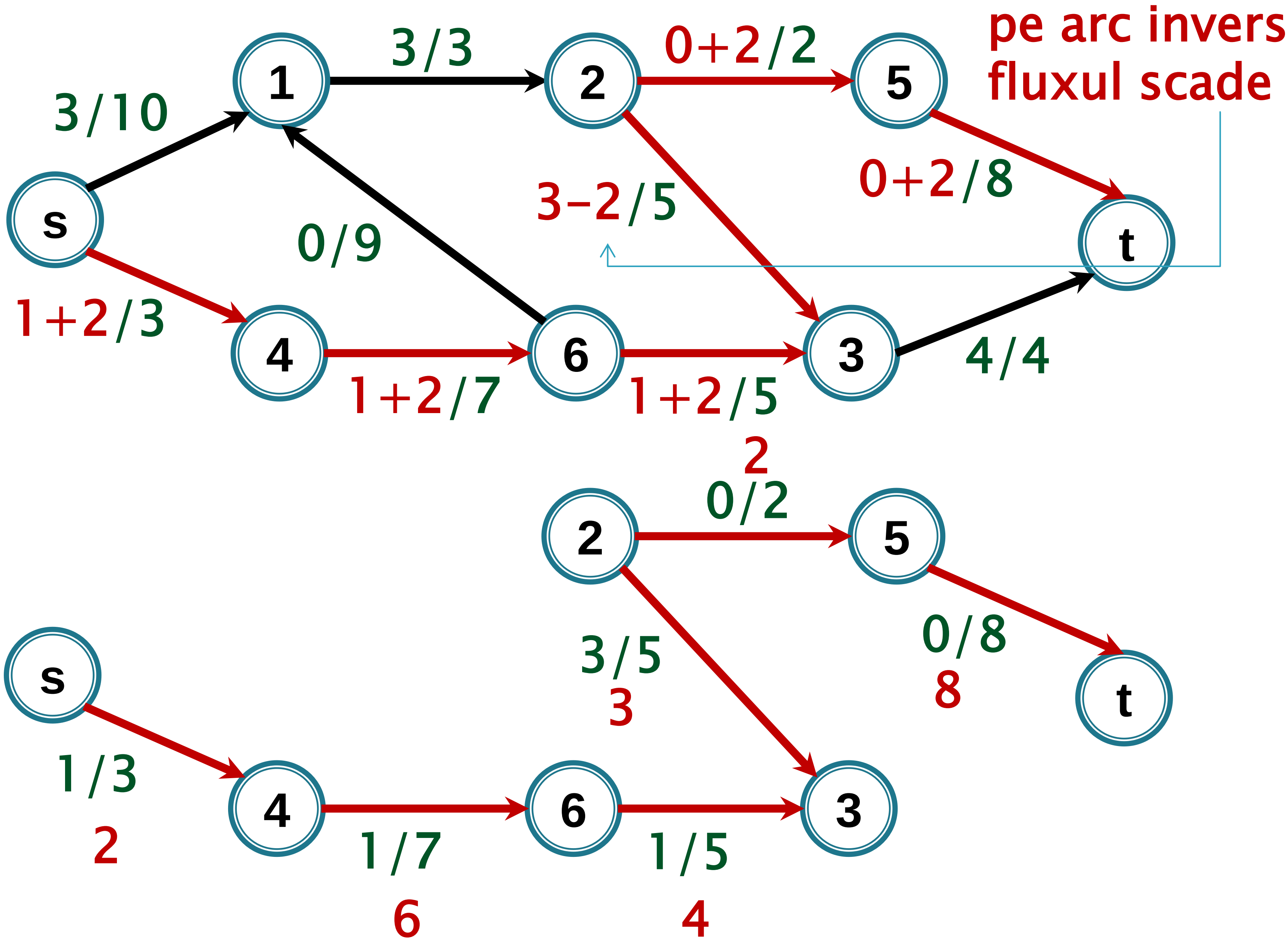
pentru arc invers
capac. reziduală=fluxul



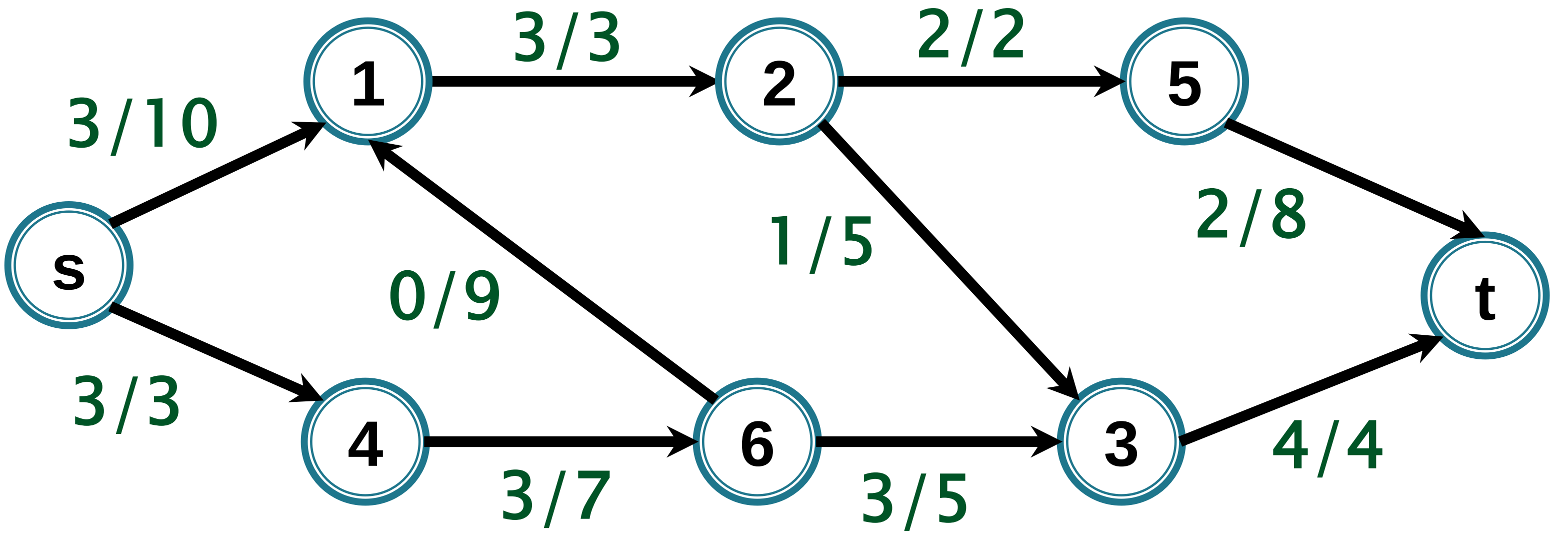




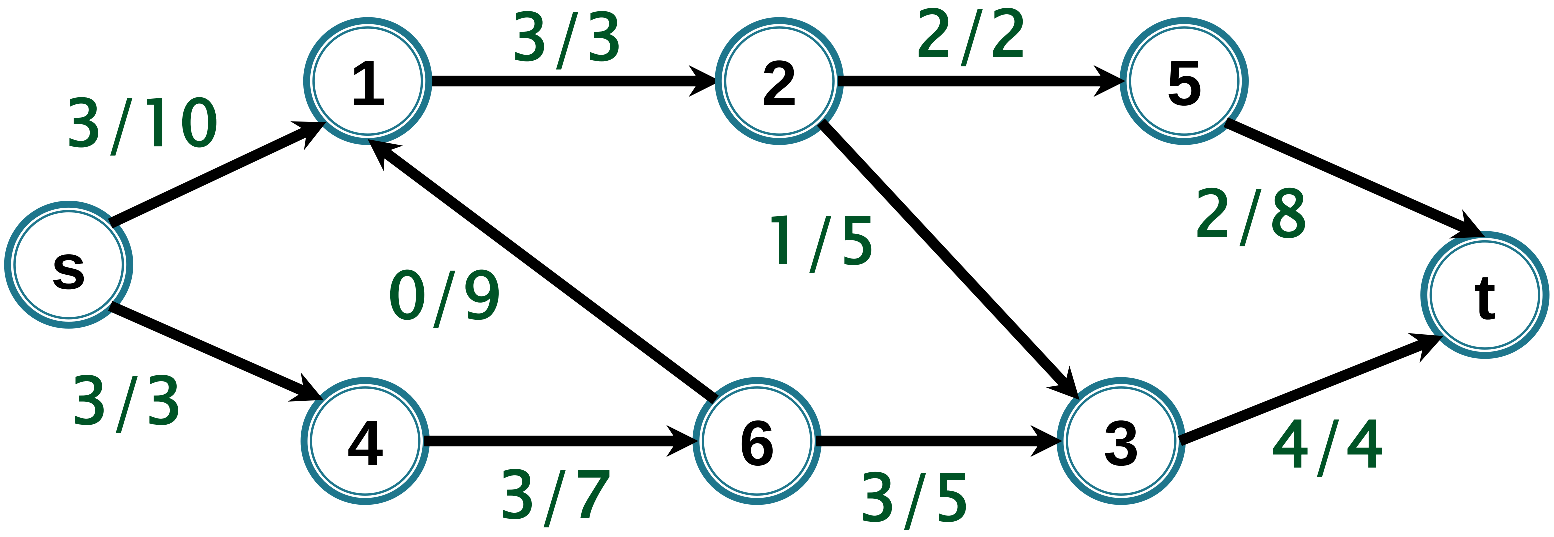
$$i(P) = \min\{ 2, 6, 4, 3, 2, 8\} = 2$$



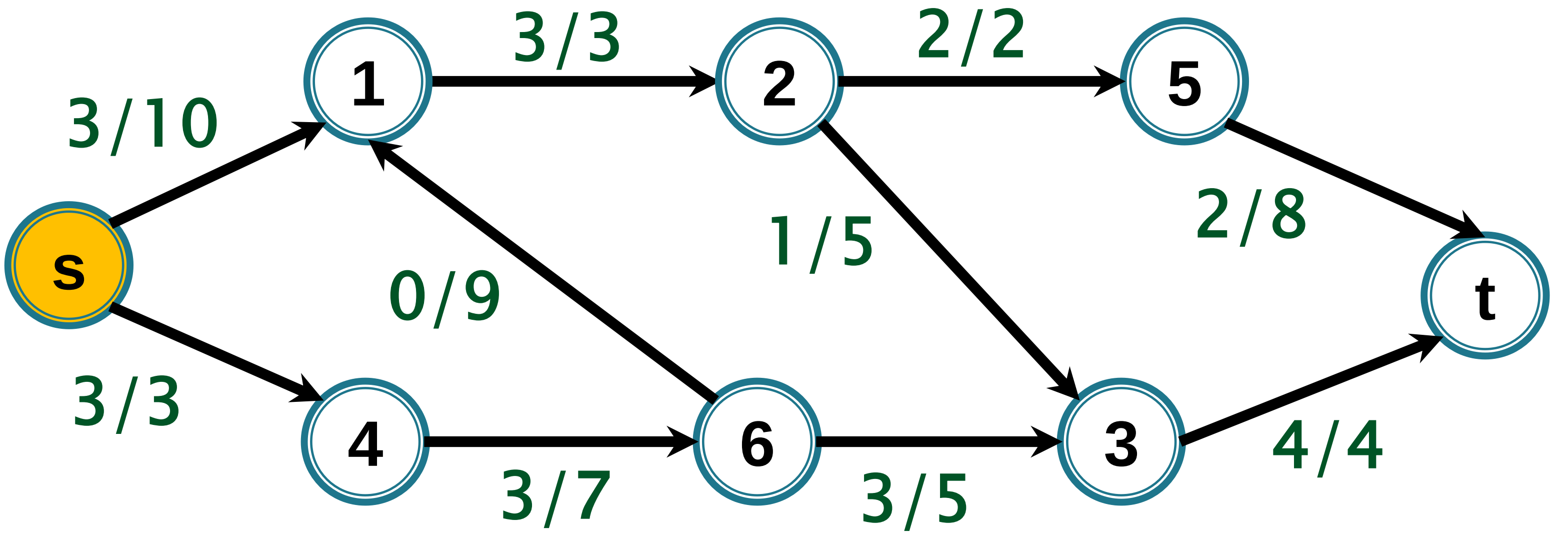
$$i(P) = \min\{\,2,\ 6,\ 4,\ 3,\ 2,\ 8\} = 2$$


s
1
2
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

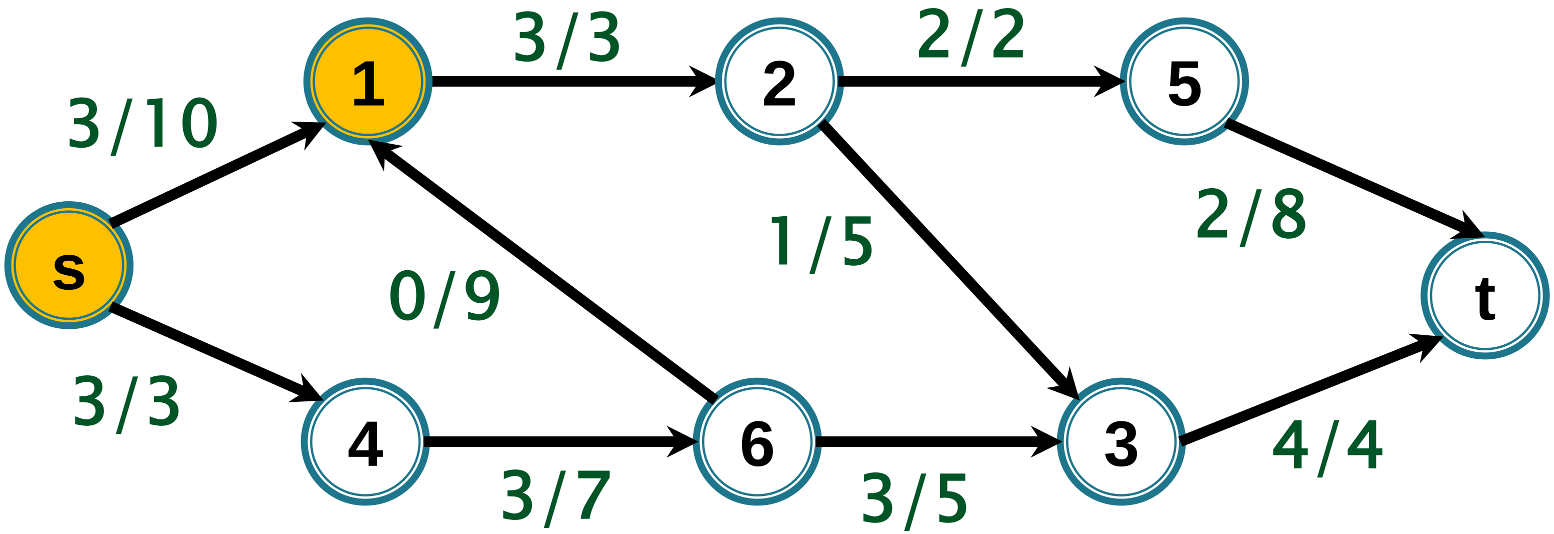
construieste_s-t_lant_nesat_BF



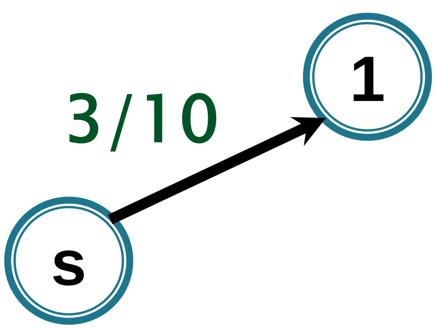
s
1
2
5
t
4
6
3
3/10
3/3
3/3
2/2
2/8
1/5
0/9
3/7
3/5
4/4


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

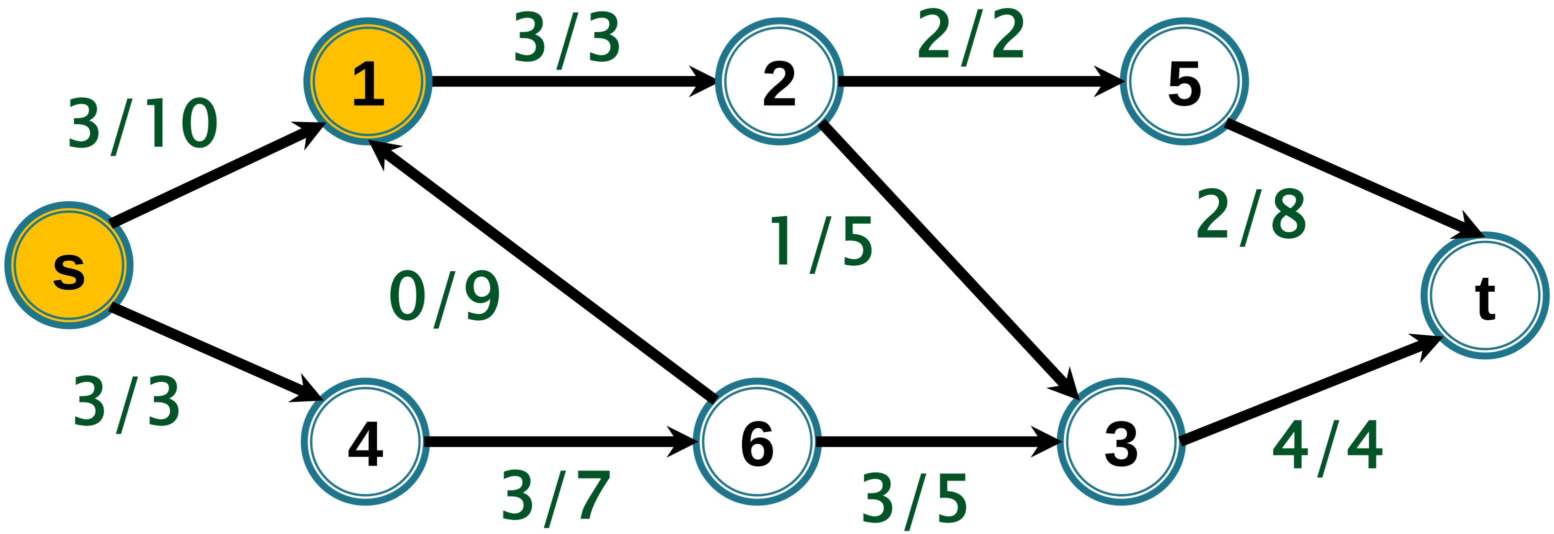

s

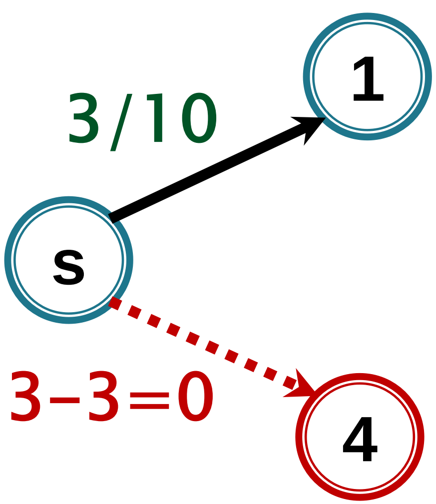


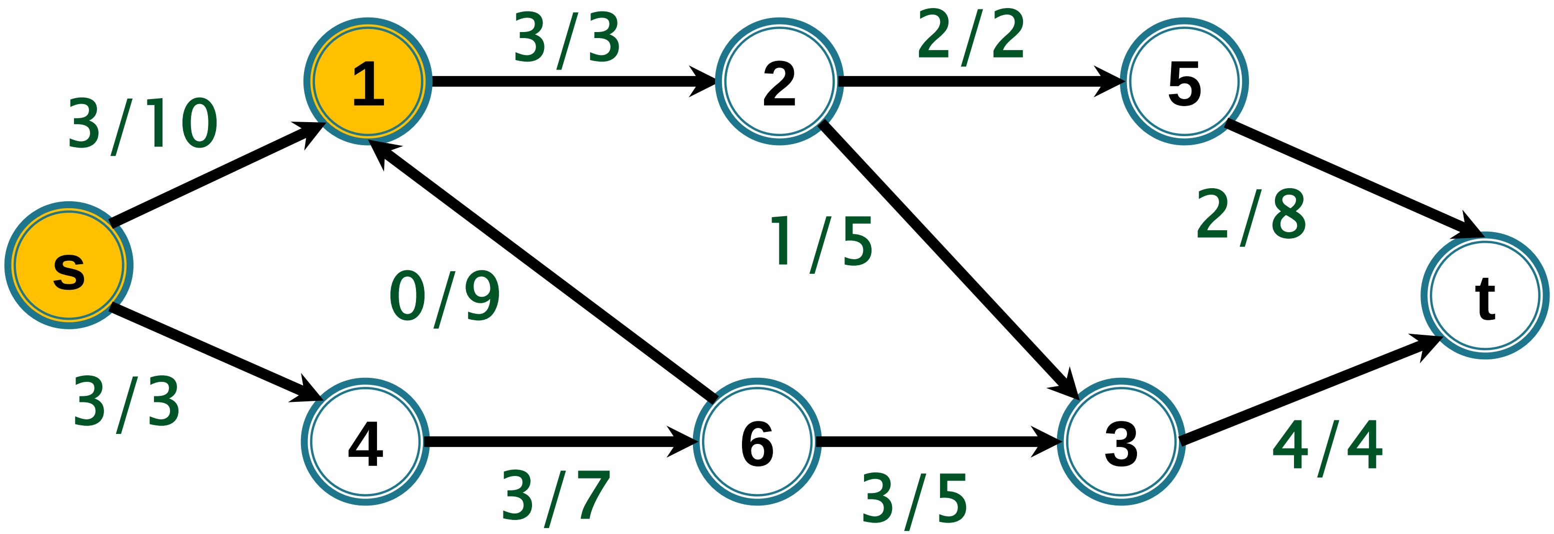
3/10
3/3
2/2
1/5
2/8
0/9
3/3
3/7
3/5
4/4
s
1
2
5
t
4
6
3



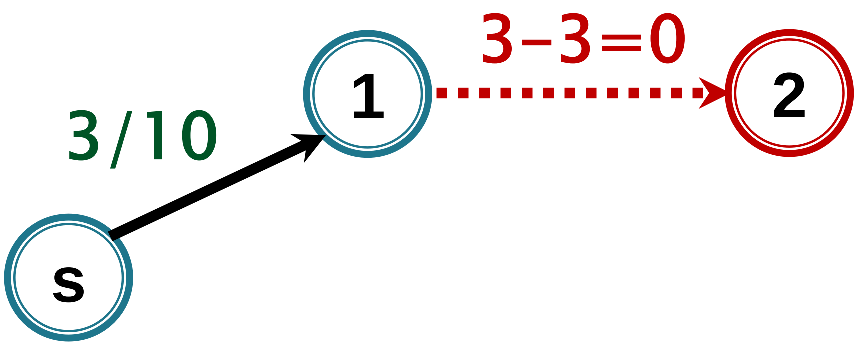
3/10
1
s


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

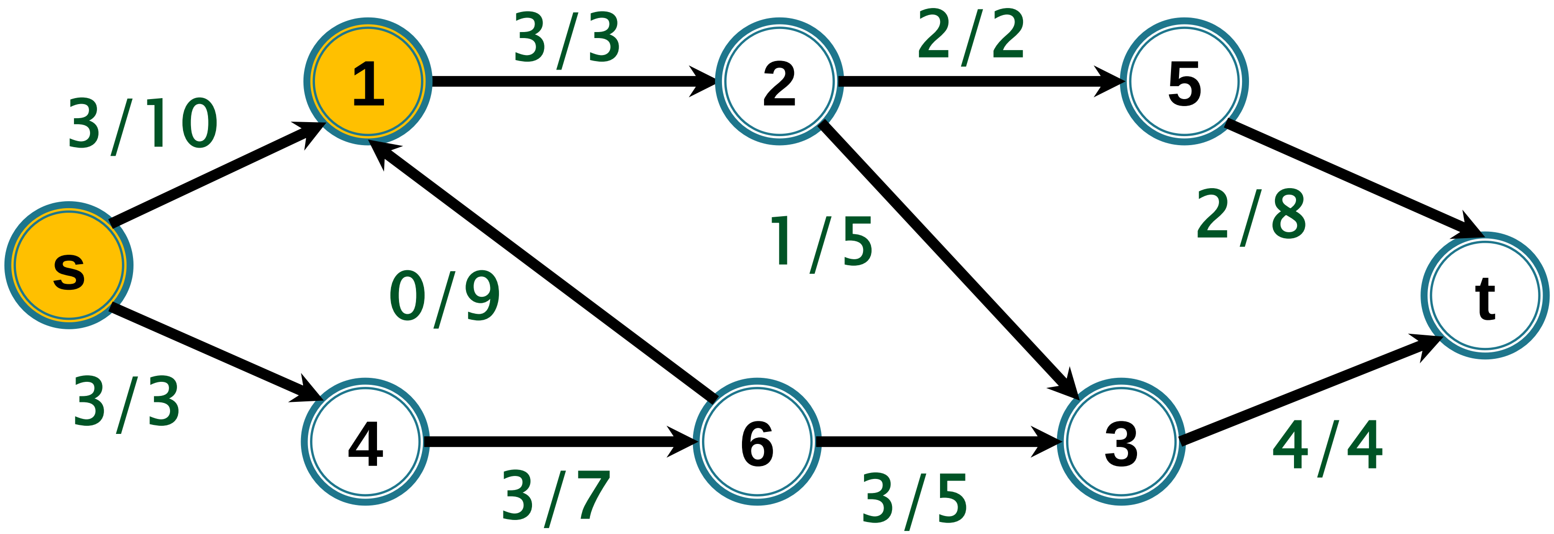

1
3/10
s
3–3=0
4

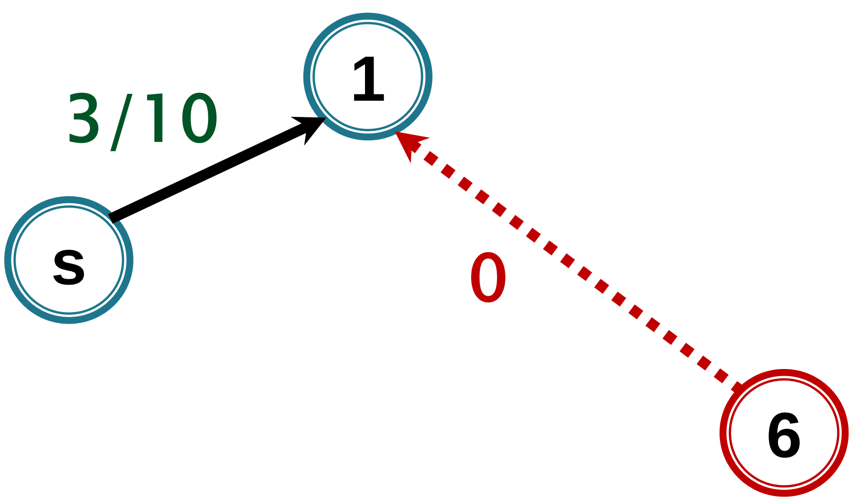


3/3
2/2
3/10
1
2
5
2/8
1/5
s
0/9
t
3/3
4
6
3
4/4
3/7
3/5



3−3=0
3/10
1
2
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
6
3/10
0

t nu este accesibil din s $\Rightarrow$ STOP

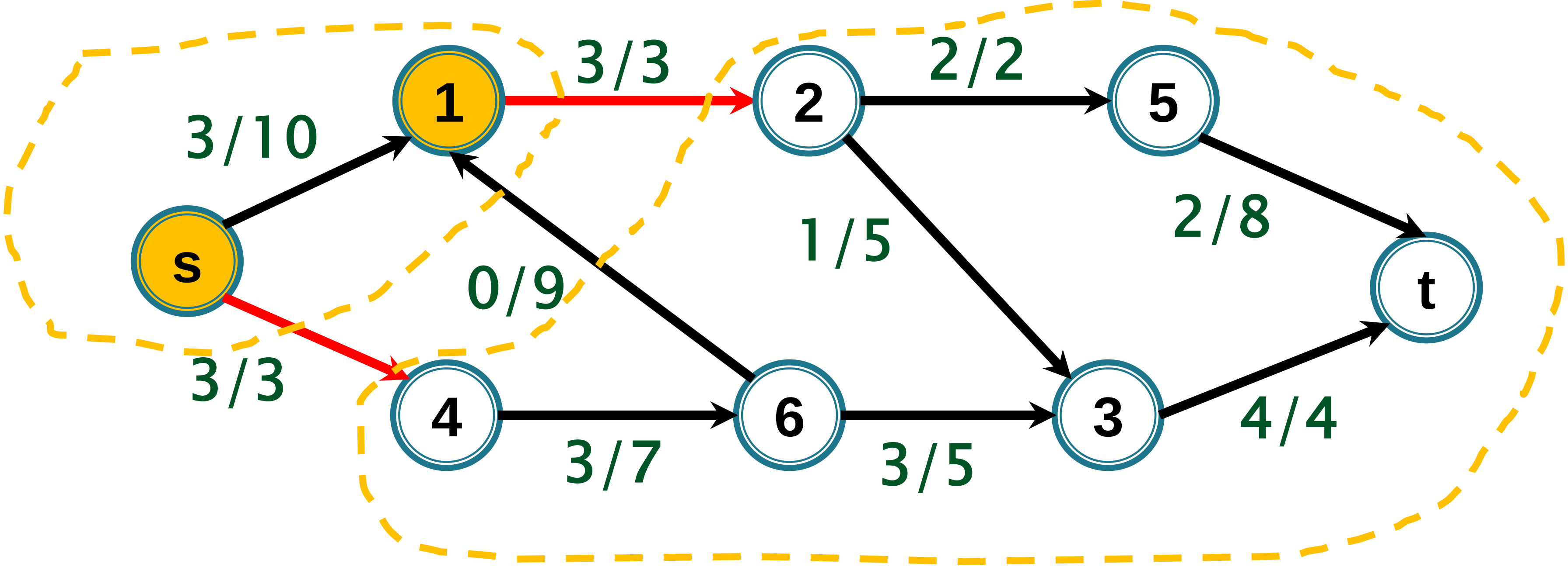
t nu este accesibil din s $\Rightarrow$ STOP

- f este flux maxim

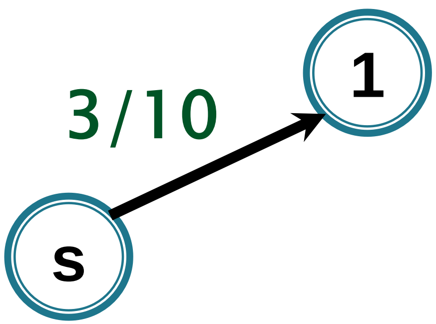
t nu este accesibil din s $\Rightarrow$ STOP

- 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(mC)
  - Edmonds Karp $O(nm^2)$