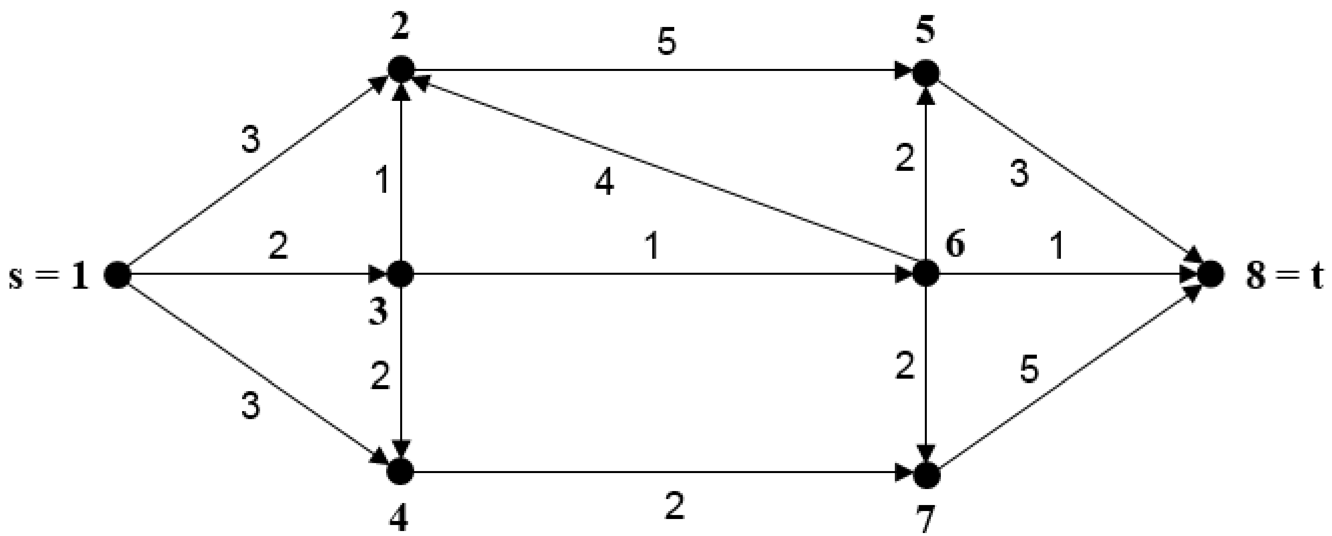
GRÁFELMÉLET

Hálózati folyamok

8. előadás

Legyen G(V, E) egy összefüggő súlyozott irányított gráf, amelynek van forráspontja (s) és nyelőpontja (t). Az élek súlyértékeit ez esetben **kapacitás**nak nevezzük, és feltételezzük, hogy nemnegatív valós számok $c : \mathrm{E} \rightarrow \mathrm{R}_0^+$. A (G, s, t, c) négyest **hálózati gráf**nak nevezzük.

Legyen $G(V, E)$ egy összefüggő súlyozott irányított gráf, amelynek van forráspontja (s) és nyelőpontja (t). Az élek súlyértékeit ez esetben **kapacitás**nak nevezzük, és feltételezzük, hogy nemnegatív valós számok $c : E \to R_0^+$. A (G, s, t, c) négyest **hálózati gráf**nak nevezzük.



Legyen $G(V, E)$ egy összefüggő súlyozott irányított gráf, amelynek van forráspontja (s) és nyelőpontja (t). Az élek súlyértékeit ez esetben **kapacitás**nak nevezzük, és feltételezzük, hogy nemnegatív valós számok $c : E \to R_0^+$. A (G, s, t, c) négyest **hálózati gráf**nak nevezzük.

Minden hálózat esetén beszélhetünk a hálózaton átfolyó **folyam**ról. Ez úgy képzelhető el, mintha a hálózati gráf egy vízvezetékrendszer lenne, amelyen átfolyik egy bizonyos mennyiségű víz. A kapacitásértékek a csövek vastagságát jelölik. A víz a forráspontból indul ki, és a nyelőpontban nyelődik el. A hálózaton átfolyó folyamot definiálni nem jelent mást, mint megmondani, hogy a hálózati gráf minden egyes élén mekkora lesz az átfolyó folyammennyiség.

Feladat: Határozzuk meg a hálózaton átfolyó maximális folyamot!

Feladat: Határozzuk meg a hálózaton átfolyó maximális folyamot!

Az $f : \mathrm{E} \to \mathrm{R}^{+}$ függvényt a (G, s, t, c) **hálózaton átfolyó folyam**nak nevezzük, ha teljesül az alábbi két feltétel:

- **minden** $(u, v) \in \mathrm{E}$ **él** esetén $f((u, v)) \leq c((u, v))$, azaz egyetlen élen sem lehet nagyobb a folyamérték, mint az adott él kapacitása,

Feladat: Határozzuk meg a hálózaton átfolyó maximális folyamot!

Az $f : \mathrm{E} \rightarrow \mathrm{R}^+$ függvényt a (G, s, t, c) **hálózaton átfolyó folyam**nak nevezzük, ha teljesül az alábbi két feltétel:

- **minden** $(u, v) \in \mathrm{E}$ **él** esetén $f((u, v)) \leq c((u, v))$, azaz egyetlen élen sem lehet nagyobb a folyamérték, mint az adott él kapacitása,
- **minden** $u \in \mathrm{V(G)} \setminus \{s, t\}$ **csúcspont** esetén a be-éleken befolyó folyamértékek összege egyenlő a ki-éleken kifolyó folyamértékek összegével.

Feladat: Határozzuk meg a hálózaton átfolyó maximális folyamot!

Az $f : \mathrm{E} \to \mathrm{R}^{+}$ függvényt a (G, s, t, c) **hálózaton átfolyó folyam**nak nevezzük, ha teljesül az alábbi két feltétel:

- **minden** $(u, v) \in \mathrm{E}$ **él** esetén $f((u, v)) \leq c((u, v))$, azaz egyetlen élen sem lehet nagyobb a folyamérték, mint az adott él kapacitása,
- **minden** $u \in \mathrm{V(G)} \setminus \{s, t\}$ **csúcspont** esetén a be-éleken befolyó folyamértékek összege egyenlő a ki-éleken kifolyó folyamértékek összegével.

A hálózaton átfolyó folyamérték egyenlő a forrásból annak ki-élein kifolyó folyammennyiséggel, azaz ezen ki-élek folyamértékeinek összegével. Ez az érték természetesen azonos a nyelőpont be-élei folyamértékeinek összegével. Ami minket érdekel, az egy adott hálózaton átfolyható maximális folyamérték. Ezt határozza meg a **Ford-Fulkerson algoritmus**

Lester Randolph Ford
1886 – 1967

Delbert Ray Fulkerson
1924 – 1976

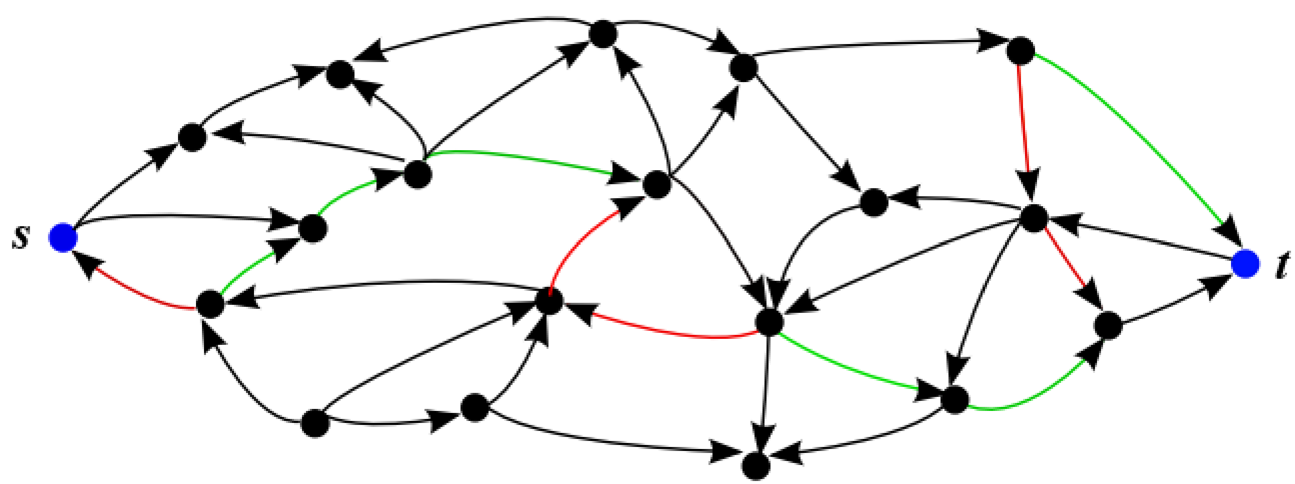
Egy hálózati gráf $s \to t$ **út**ján egy olyan s-ből t-be vezető utat értünk, amelyen az élek tetszőleges irányításúak lehetnek. Az út nem tartalmazhatja kétszer ugyanazt a csúcspontot, és az útmenti szomszédos éleknek illeszkedniük kell egymáshoz.

Egy hálózati gráf $s \to t$ **út**ján egy olyan s-ből t-be vezető utat értünk, amelyen az élek tetszőleges irányításúak lehetnek. Az út nem tartalmazhatja kétszer ugyanazt a csúcspontot, és az útmenti szomszédos éleknek illeszkedniük kell egymáshoz.

Haladjunk végig egy adott $s \to t$ úton ebben az irányban. Azokat az éleket, amelyeken irányításuknak megfelelően haladunk át, az illető útra nézve **előremutató**, a többieket pedig az illető útra nézve **visszamutató él**eknek nevezzük.

Egy hálózati gráf $s \to t$ **út**ján egy olyan s-ből t-be vezető utat értünk, amelyen az élek tetszőleges irányításúak lehetnek. Az út nem tartalmazhatja kétszer ugyanazt a csúcspontot, és az útmenti szomszédos éleknek illeszkedniük kell egymáshoz.

Haladjunk végig egy adott $s \to t$ úton ebben az irányban. Azokat az éleket, amelyeken irányításuknak megfelelően haladunk át, az illető útra nézve **előremutató**, a többieket pedig az illető útra nézve **visszamutató él**eknek nevezzük.



Egy hálózati gráf $s \to t$ **út**ján egy olyan s-ből t-be vezető utat értünk, amelyen az élek tetszőleges irányításúak lehetnek. Az út nem tartalmazhatja kétszer ugyanazt a csúcspontot, és az útmenti szomszédos éleknek illeszkedniük kell egymáshoz.

Haladjunk végig egy adott $s \to t$ úton ebben az irányban. Azokat az éleket, amelyeken irányításuknak megfelelően haladunk át, az illető útra nézve **előremutató**, a többieket pedig az illető útra nézve **visszamutató él**eknek nevezzük.

Az $s \to t$ út valamely élét **telített**nek nevezzük, ha *előremutató* és a kapacitásával megegyező értékű folyam halad át rajta, vagy *visszamutató* és nulla mennyiségű folyam folyik vissza rajta.

Egy hálózati gráf $s \to t$ **út**ján egy olyan s-ből t-be vezető utat értünk, amelyen az élek tetszőleges irányításúak lehetnek. Az út nem tartalmazhatja kétszer ugyanazt a csúcspontot, és az útmenti szomszédos éleknek illeszkedniük kell egymáshoz.

Haladjunk végig egy adott $s \to t$ úton ebben az irányban. Azokat az éleket, amelyeken irányításuknak megfelelően haladunk át, az illető útra nézve **előremutató**, a többieket pedig az illető útra nézve **visszamutató él**eknek nevezzük.

Az $s \to t$ út valamely élét **telített**nek nevezzük, ha *előremutató* és a kapacitásával megegyező értékű folyam halad át rajta, vagy *visszamutató* és nulla mennyiségű folyam folyik vissza rajta.

Az $s \to t$ út valamely élét **telítetlen**nek nevezzük, ha *előremutató* és a kapacitásánál kisebb értékű folyam halad át rajta, vagy *visszamutató* és nullánál nagyobb mennyiségű folyam folyik vissza rajta.

8.1. tétel: (MAX folyam MIN vágás tétel)

Egy hálózaton átfolyó maximális folyammennyiség értéke egyenlő a minimális kapacitású vágás kapacitásával.

8.1. tétel: (MAX folyam MIN vágás tétel)

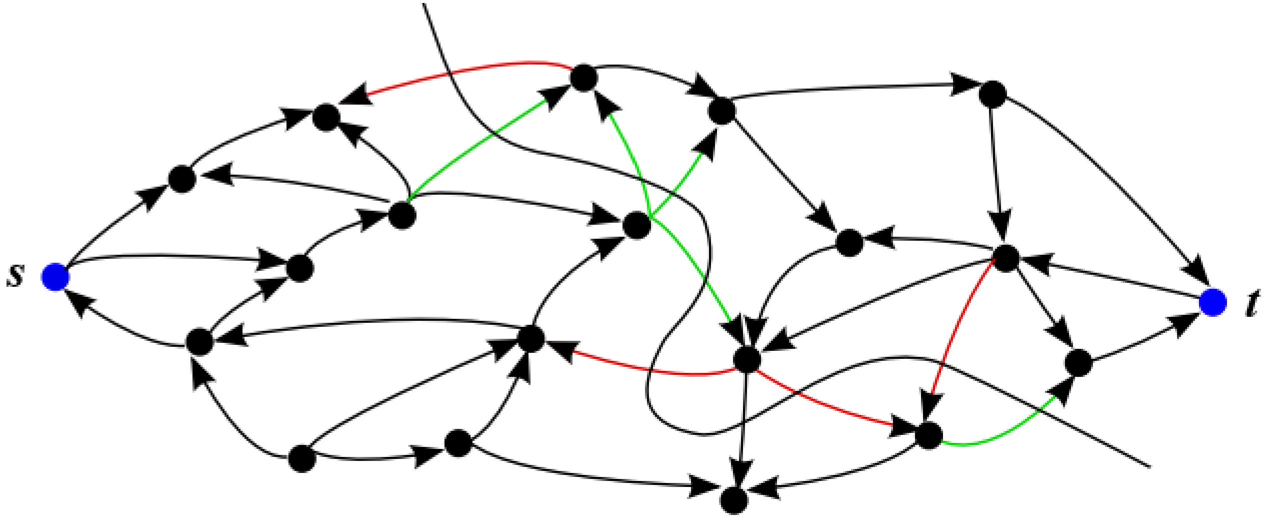
Egy hálózaton átfolyó maximális folyammennyiség értéke egyenlő a minimális kapacitású vágás kapacitásával.

A vágás fogalmán itt a következőt értjük: Legyen (X, Y) a hálózati gráf pontjainak egy olyan partíciója ($X \cup Y = V$, $X \cap Y = \varnothing$), amelyre $s \in X$ és $t \in Y$. Az (X, Y) vágást a gráf azon élei alkotják, amelyeknek egyik végpontja X-ben, a másik pedig Y-ban van. Az (X, Y) vágás kapacitása a definíció szerint egyenlő az X-től Y felé mutató élek (előremutató élek) kapacitásainak összegével.

8.1. tétel: (MAX folyam MIN vágás tétel)

Egy hálózaton átfolyó maximális folyammennyiség értéke egyenlő a minimális kapacitású vágás kapacitásával.

A vágás fogalmán itt a következőt értjük: Legyen (X, Y) a hálózati gráf pontjainak egy olyan partíciója ($X \cup Y = V$, $X \cap Y = \emptyset$), amelyre $s \in X$ és $t \in Y$. Az (X, Y) vágást a gráf azon élei alkotják, amelyeknek egyik végpontja X-ben, a másik pedig Y-ban van. Az (X, Y) vágás kapacitása a definíció szerint egyenlő az X-től Y felé mutató élek (előremutató élek) kapacitásainak összegével.



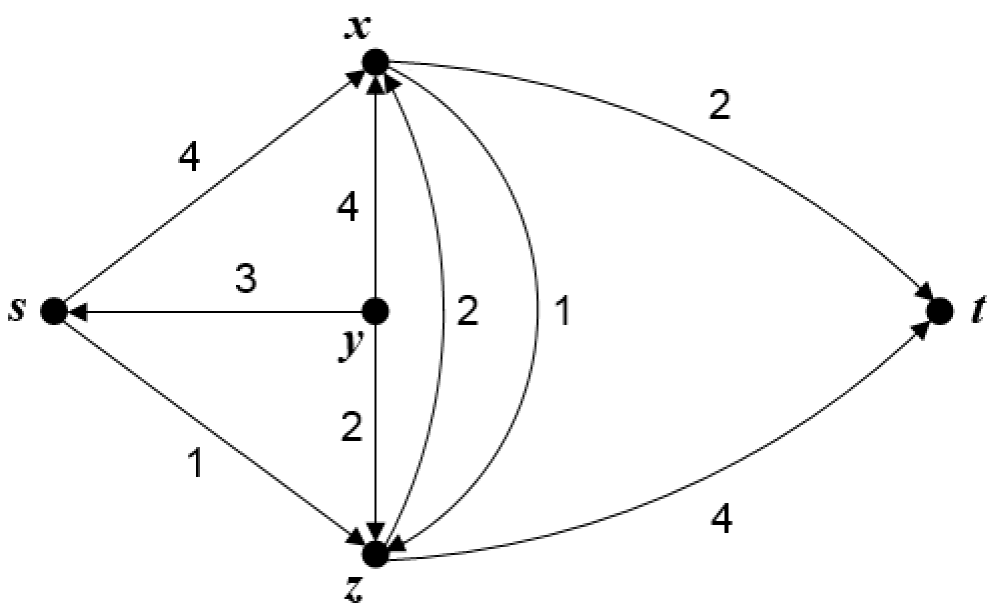
8.1. tétel: (MAX folyam MIN vágás tétel)

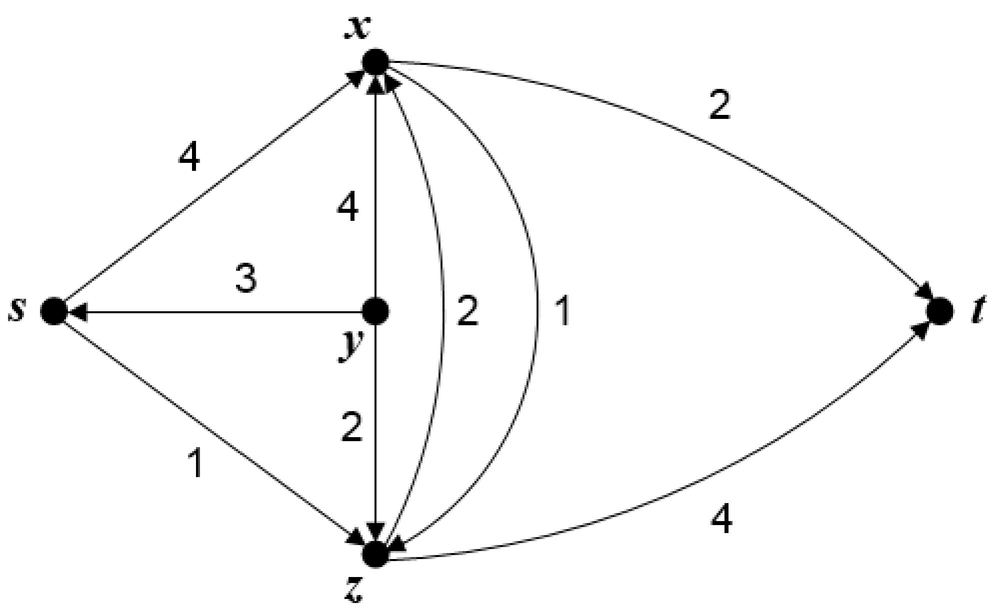
Egy hálózaton átfolyó maximális folyammennyiség értéke egyenlő a minimális kapacitású vágás kapacitásával.

Bizonyítás:

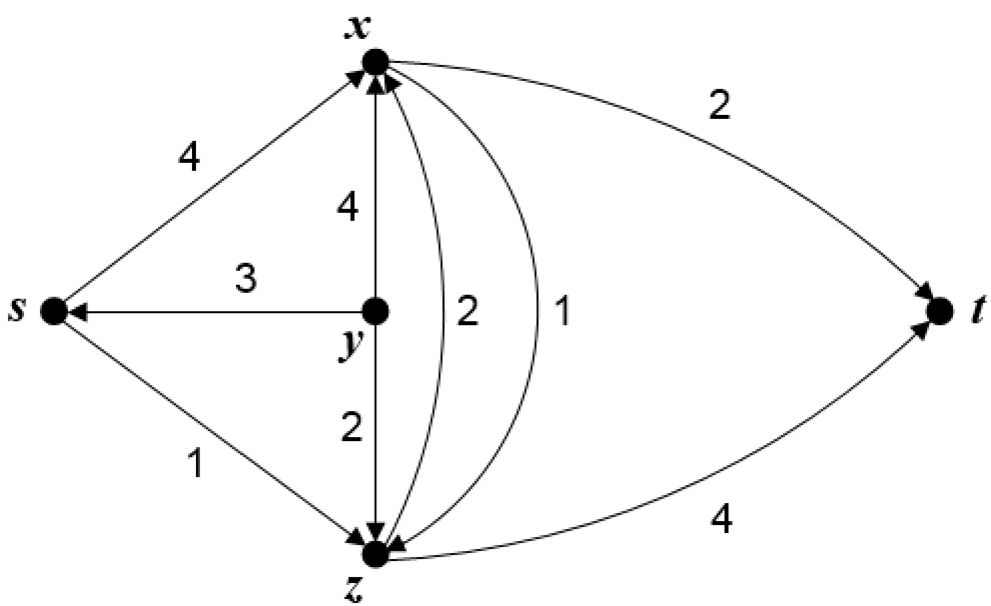
Egy vágáson természetesen akkor folyik át (előre irányban) a legnagyobb mennyiségű folyam, ha a vágás minden *előremutató éle* telített, és a *visszamutató éleken* nem folyik vissza semmi (azaz, az illető vágásra nézve ezek is telítettek). Ez az érték éppen a vágás kapacitásával lesz egyenlő. Mivel a hálózaton átfolyó folyamnak a gráf minden vágásán át kell haladnia, ezért a maximális folyamérték a „minimális vágás“ kapacitásával lesz egyenlő.

□


x
s
y
z
t
4
4
3
2
1
2
2
1
2
4



Vágás $(P, \overline{P})$	$c(P, \overline{P})$
$P = \{s\}, \overline{P} = \{x, y, z, t\}$	$4 + 1 = 5$
$P = \{s, x\}, \overline{P} = \{y, z, t\}$	$1 + 2 + 1 = 4$
$P = \{s, y\}, \overline{P} = \{x, z, t\}$	$4 + 4 + 2 + 1 = 11$
$P = \{s, z\}, \overline{P} = \{x, y, t\}$	$4 + 2 + 4 = 10$
$P = \{s, x, y\}, \overline{P} = \{z, t\}$	$1 + 1 + 2 + 2 = 6$
$P = \{s, x, z\}, \overline{P} = \{y, t\}$	$2 + 4 = 6$
$P = \{s, y, z\}, \overline{P} = \{x, t\}$	$4 + 4 + 2 + 4 = 14$
$P = \{s, x, y, z\}, \overline{P} = \{t\}$	$2 + 4 = 6$



Vágás $(P,\ \overline{P})$	$c(P,\ \overline{P})$
$P = \{s\},\ \overline{P} = \{x, y, z, t\}$	$4 + 1 = 5$
$P = \{s, x\},\ \overline{P} = \{y, z, t\}$	$1 + 2 + 1 = 4$
$P = \{s, y\},\ \overline{P} = \{x, z, t\}$	$4 + 4 + 2\ +\ 1 = 11$
$P = \{s, z\},\ \overline{P} = \{x, y, t\}$	$4 + 2 + 4 = 10$
$P = \{s, x, y\},\ \overline{P} = \{z, t\}$	$1 + 1 + 2 + 2 = 6$
$P = \{s, x, z\},\ \overline{P} = \{y, t\}$	$2 + 4 = 6$
$P = \{s, y, z\},\ \overline{P} = \{x, t\}$	$4 + 4 + 2 + 4 = 14$
$P = \{s, x, y, z\},\ \overline{P} = \{t\}$	$2 + 4 = 6$

A maximális folyam értéke: **4**

Javítóútnak nevezünk egy olyan $s \to t$ utat, amelynek minden éle telítetlen.

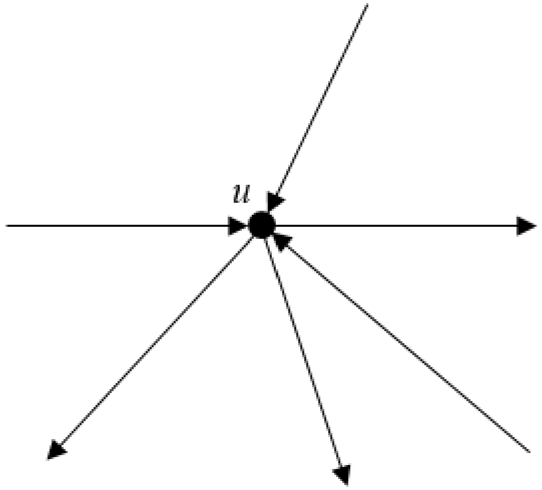
Minden javítóút mentén növelhető a hálózaton átfolyó folyamérték: ha egy javítóút minden előremutató élén növeljük, és minden visszamutató élén csökkentjük a folyamértéket x-el, akkor a hálózaton áthaladó folyammennyiség nő x-el.

Javítóútnak nevezünk egy olyan $s \to t$ utat, amelynek minden éle telítetlen.

Minden javítóút mentén növelhető a hálózaton átfolyó folyamérték: ha egy javítóút minden előremutató élén növeljük, és minden visszamutató élén csökkentjük a folyamértéket x-el, akkor a hálózaton áthaladó folyammennyiség nő x-el.

Legyen u a javítóút egy pontja. Az u pontot a javítóúton közvetlenül megelőző és követő élek irányítását tekintve, négy eset állhat fenn:

- <u>mindkét él az u csúcspont be-éle</u>

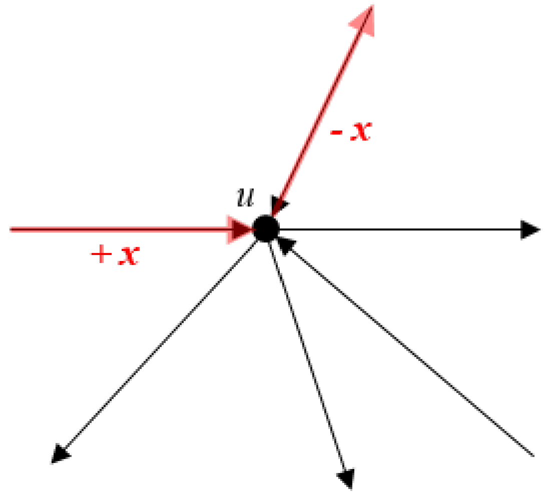


Javítóútnak nevezünk egy olyan $s \to t$ utat, amelynek minden éle telítetlen.

Minden javítóút mentén növelhető a hálózaton átfolyó folyamérték: ha egy javítóút minden előremutató élén növeljük, és minden visszamutató élén csökkentjük a folyamértéket x-el, akkor a hálózaton áthaladó folyammennyiség nő x-el.

Legyen u a javítóút egy pontja. Az u pontot a javítóúton közvetlenül megelőző és követő élek irányítását tekintve, négy eset állhat fenn:

- <u>mindkét él az u csúcspont be-éle</u>

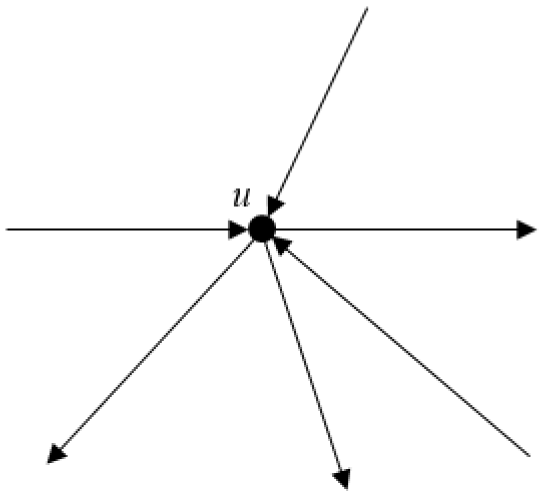


Javítóútnak nevezünk egy olyan $s \to t$ utat, amelynek minden éle telítetlen.

Minden javítóút mentén növelhető a hálózaton átfolyó folyamérték: ha egy javítóút minden előremutató élén növeljük, és minden visszamutató élén csökkentjük a folyamértéket x-el, akkor a hálózaton áthaladó folyammennyiség nő x-el.

Legyen u a javítóút egy pontja. Az u pontot a javítóúton közvetlenül megelőző és követő élek irányítását tekintve, négy eset állhat fenn:

- mindkét él az u csúcspont ki-éle

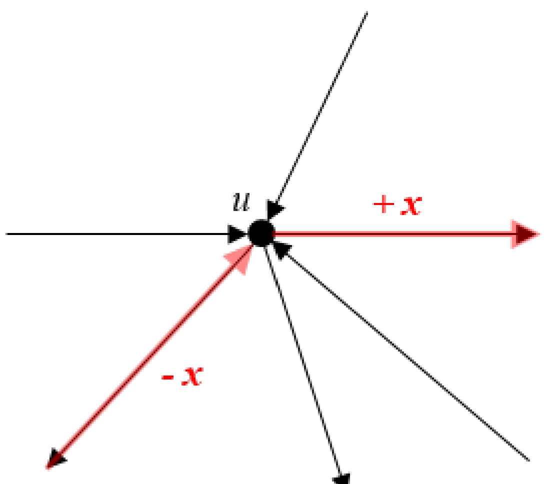


Javítóútnak nevezünk egy olyan $s \to t$ utat, amelynek minden éle telítetlen.

Minden javítóút mentén növelhető a hálózaton átfolyó folyamérték: ha egy javítóút minden előremutató élén növeljük, és minden visszamutató élén csökkentjük a folyamértéket x-el, akkor a hálózaton áthaladó folyammennyiség nő x-el.

Legyen u a javítóút egy pontja. Az u pontot a javítóúton közvetlenül megelőző és követő élek irányítását tekintve, négy eset állhat fenn:

- <u>mindkét él az u csúcspont ki-éle</u>

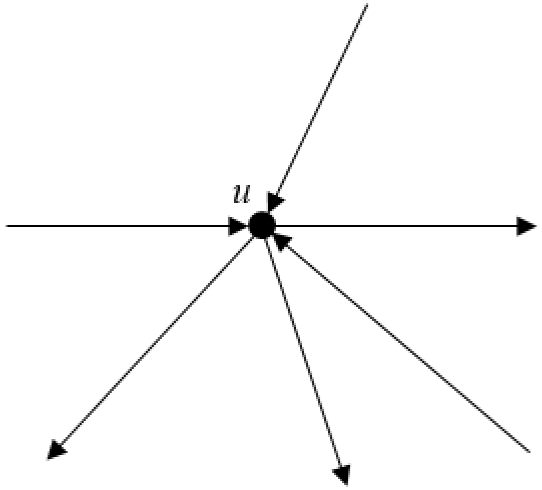


Javítóútnak nevezünk egy olyan $s \to t$ utat, amelynek minden éle telítetlen.

Minden javítóút mentén növelhető a hálózaton átfolyó folyamérték: ha egy javítóút minden előremutató élén növeljük, és minden visszamutató élén csökkentjük a folyamértéket x-el, akkor a hálózaton áthaladó folyammennyiség nő x-el.

Legyen u a javítóút egy pontja. Az u pontot a javítóúton közvetlenül megelőző és követő élek irányítását tekintve, négy eset állhat fenn:

- <u>az első él az u csúcspont be-éle, a második pedig ki-éle</u>

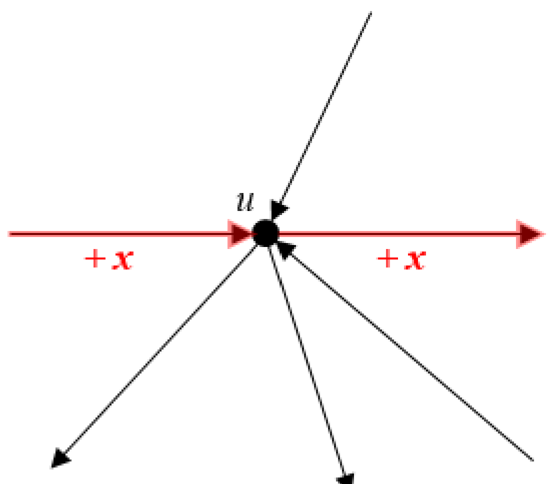


Javítóútnak nevezünk egy olyan $s \to t$ utat, amelynek minden éle telítetlen.

Minden javítóút mentén növelhető a hálózaton átfolyó folyamérték: ha egy javítóút minden előremutató élén növeljük, és minden visszamutató élén csökkentjük a folyamértéket x-el, akkor a hálózaton áthaladó folyammennyiség nő x-el.

Legyen u a javítóút egy pontja. Az u pontot a javítóúton közvetlenül megelőző és követő élek irányítását tekintve, négy eset állhat fenn:

- az első él az u csúcspont be-éle, a második pedig ki-éle

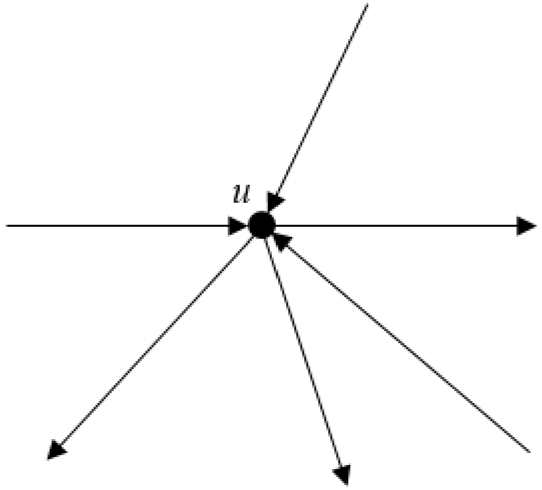


Javítóútnak nevezünk egy olyan $s \to t$ utat, amelynek minden éle telítetlen.

Minden javítóút mentén növelhető a hálózaton átfolyó folyamérték: ha egy javítóút minden előremutató élén növeljük, és minden visszamutató élén csökkentjük a folyamértéket x-el, akkor a hálózaton áthaladó folyammennyiség nő x-el.

Legyen u a javítóút egy pontja. Az u pontot a javítóúton közvetlenül megelőző és követő élek irányítását tekintve, négy eset állhat fenn:

- az első él az u csúcspont ki-éle, a második pedig be-éle

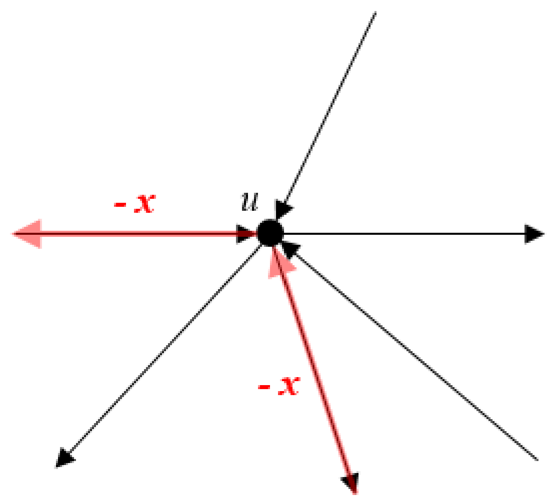


Javítóútnak nevezünk egy olyan $s \to t$ utat, amelynek minden éle telítetlen.

Minden javítóút mentén növelhető a hálózaton átfolyó folyamérték: ha egy javítóút minden előremutató élén növeljük, és minden visszamutató élén csökkentjük a folyamértéket x-el, akkor a hálózaton áthaladó folyammennyiség nő x-el.

Legyen u a javítóút egy pontja. Az u pontot a javítóúton közvetlenül megelőző és követő élek irányítását tekintve, négy eset állhat fenn:

- az első él az u csúcspont ki-éle, a második pedig be-éle



A legnagyobb érték, amellyel egy javítóút mentén növelhető a folyam:

- ha (u, v) előremutató él, akkor a rajta áthaladó folyam legfeljebb $c((u, v)) - f((u, v))$ értékkel növelhető,
- ha (u, v) visszamutató él, akkor a rajta áthaladó folyam legfeljebb $f((u, v))$ értékkel csökkenthető.

A legnagyobb érték, amellyel egy javítóút mentén növelhető a folyam:

- ha (u, v) előremutató él, akkor a rajta áthaladó folyam legfeljebb $c((u, v)) - f((u, v))$ értékkel növelhető,
- ha (u, v) visszamutató él, akkor a rajta áthaladó folyam legfeljebb $f((u, v))$ értékkel csökkenthető.

Nyilvánvaló, hogy ezen értékek minimuma lesz az a legnagyobb közös érték, amellyel a javító út minden élén javítható lesz a folyamérték (az előremutató éleken növelhető, a visszamutató éleken pedig csökkenthető).

A legnagyobb érték, amellyel egy javítóút mentén növelhető a folyam:

- ha (u, v) előremutató él, akkor a rajta áthaladó folyam legfeljebb $c((u, v)) - f((u, v))$ értékkel növelhető,
- ha (u, v) visszamutató él, akkor a rajta áthaladó folyam legfeljebb $f((u, v))$ értékkel csökkenthető.

Nyilvánvaló, hogy ezen értékek minimuma lesz az a legnagyobb közös érték, amellyel a javító út minden élén javítható lesz a folyamérték (az előremutató éleken növelhető, a visszamutató éleken pedig csökkenthető).

Mivel minden javítóút első éle előremutató ki-éle s-nek, ezért a rajta átfolyó folyammennyiség növeléséhez a forráspont termelését kell megnövelni. Más szóval, a leírt javítási művelet valóban növeli a hálózaton átfolyó folyammennyiséget.

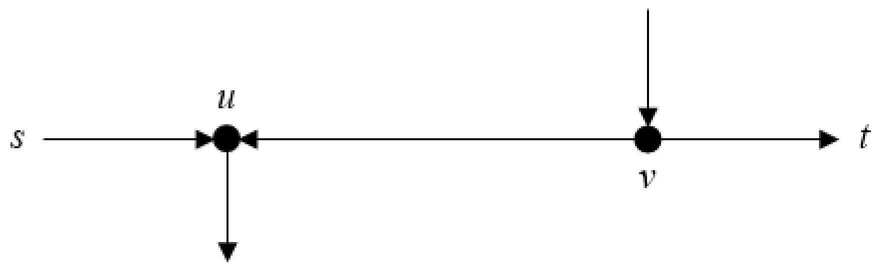
Nyilvánvaló, hogy az előremutató élek folyamértékeinek növelésével nő a hálózaton átfolyó folyammennyiség.

Nyilvánvaló, hogy az előremutató élek folyamértékeinek növelésével nő a hálózaton átfolyó folyammennyiség.

Az már nehezebben átlátható, hogy a visszamutató élek folyamértékeinek csökkentése szintén növeli a hálózaton átfolyó folyammennyiséget.

Nyilvánvaló, hogy az előremutató élek folyamértékeinek növelésével nő a hálózaton átfolyó folyammennyiség.

Az már nehezebben átlátható, hogy a visszamutató élek folyamértékeinek csökkentése szintén növeli a hálózaton átfolyó folyammennyiséget:

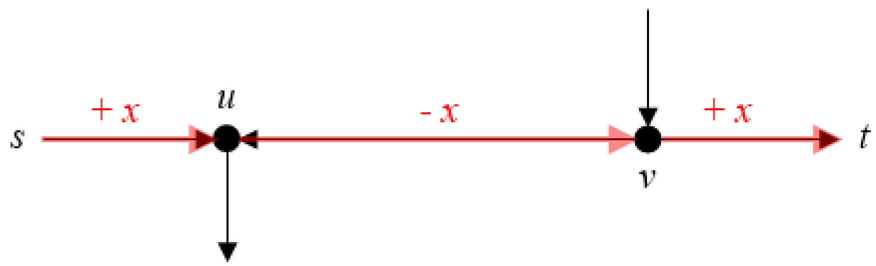


Nyilvánvaló, hogy az előremutató élek folyamértékeinek növelésével nő a hálózaton átfolyó folyammennyiség.

Az már nehezebben átlátható, hogy a visszamutató élek folyamértékeinek csökkentése szintén növeli a hálózaton átfolyó folyammennyiséget:

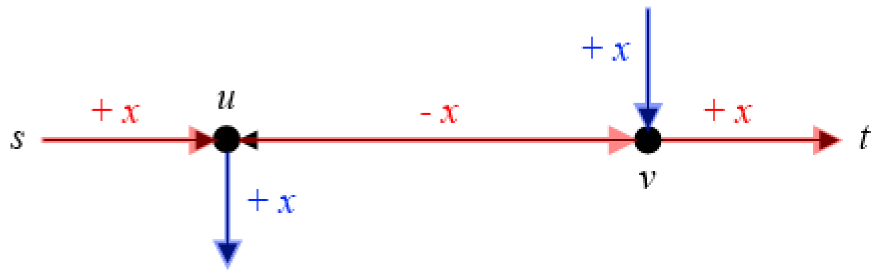
Nyilvánvaló, hogy az előremutató élek folyamértékeinek növelésével nő a hálózaton átfolyó folyammennyiség.

Az már nehezebben átlátható, hogy a visszamutató élek folyamértékeinek csökkentése szintén növeli a hálózaton átfolyó folyammennyiséget:



Nyilvánvaló, hogy az előremutató élek folyamértékeinek növelésével nő a hálózaton átfolyó folyammennyiség.

Az már nehezebben átlátható, hogy a visszamutató élek folyamértékeinek csökkentése szintén növeli a hálózaton átfolyó folyammennyiséget:



Ford-Fulkerson algoritmus

Stratégia: Az algoritmus az alábbi gondolatmenetet követi:

Kezdetben minden élen a folyamérték nulla.

Ford-Fulkerson algoritmus

Stratégia: Az algoritmus az alábbi gondolatmenetet követi:

Kezdetben minden élen a folyamérték nulla.

WHILE létezik $s \to t$ javítóút **DO**

1) keress egy javítóutat (KERES_JAVÍTÓ_ÚT eljárás)

Ford-Fulkerson algoritmus

Stratégia: Az algoritmus az alábbi gondolatmenetet követi:

Kezdetben minden élen a folyamérték nulla.

WHILE létezik $s \to t$ javítóút **DO**

1) keress egy javítóutat (KERES_JAVÍTÓ_ÚT eljárás)

2) határozd meg azt a maximumot, amellyel az illető javító út mentén növelhető a folyamérték (JAVÍTÁS eljárás)

Ford-Fulkerson algoritmus

Stratégia: Az algoritmus az alábbi gondolatmenetet követi:

Kezdetben minden élen a folyamérték nulla.

WHILE létezik $s \to t$ javítóút **DO**

1) keress egy javítóutat (KERES_JAVÍTÓ_ÚT eljárás)

2) határozd meg azt a maximumot, amellyel az illető javító út mentén növelhető a folyamérték (JAVÍTÁS eljárás)

3) végezd el a javítóút mentén a javítás által feltételezett folyamérték korrekciókat (JAVÍTÁS eljárás)

Ford-Fulkerson algoritmus

Stratégia: Az algoritmus az alábbi gondolatmenetet követi:

Kezdetben minden élen a folyamérték nulla.

WHILE létezik $s \to t$ javítóút **DO**

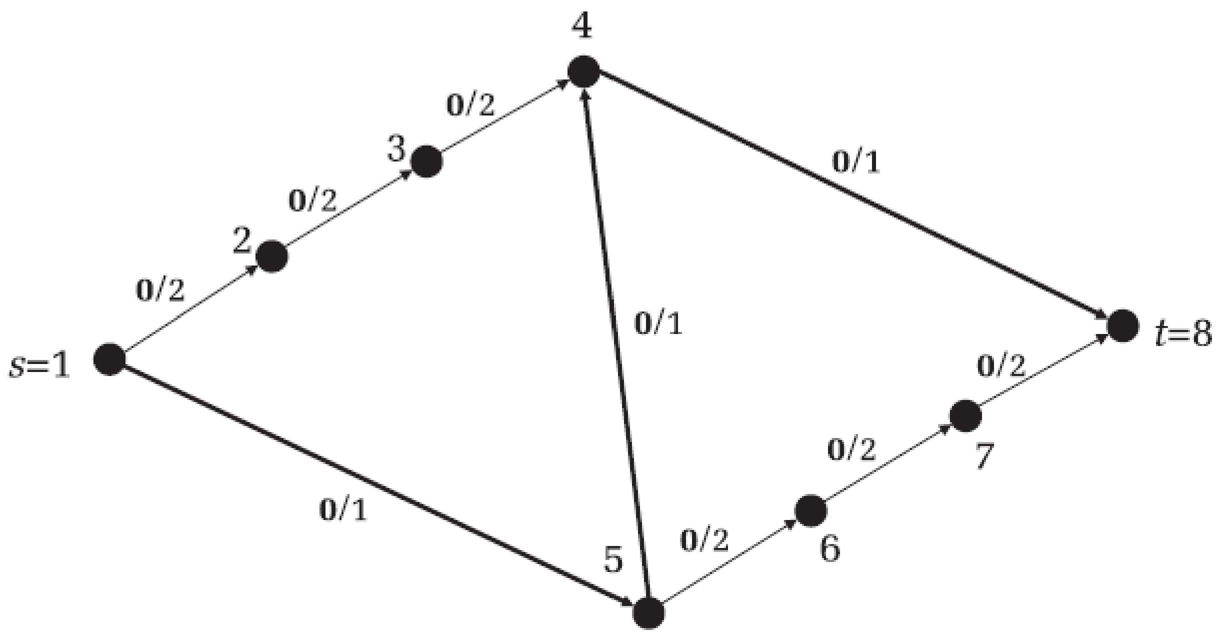
1) keress egy javítóutat (KERES_JAVÍTÓ_ÚT eljárás)

2) határozd meg azt a maximumot, amellyel az illető javító út mentén növelhető a folyamérték (JAVÍTÁS eljárás)

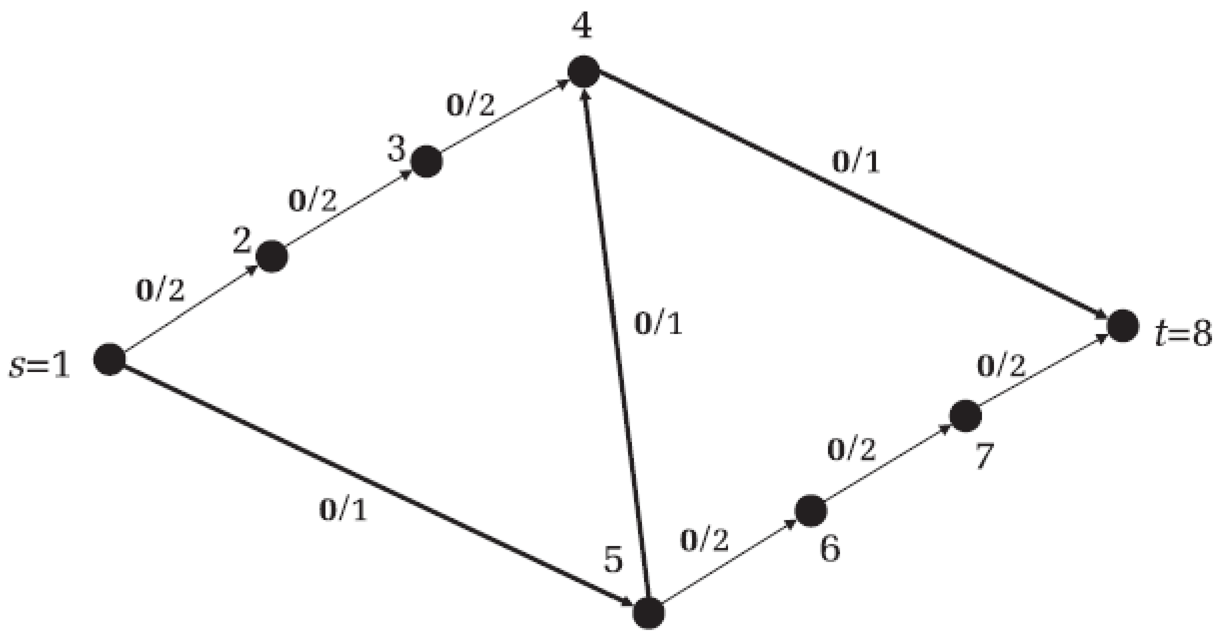
3) végezd el a javítóút mentén a javítás által feltételezett folyamérték korrekciókat (JAVÍTÁS eljárás)

A maximális folyamérték a javítóértékek összege lesz.

Példa arra, miért nem elég csak az irányított javítóutakat (amelyeken minden él előremutató) figyelembe venni:

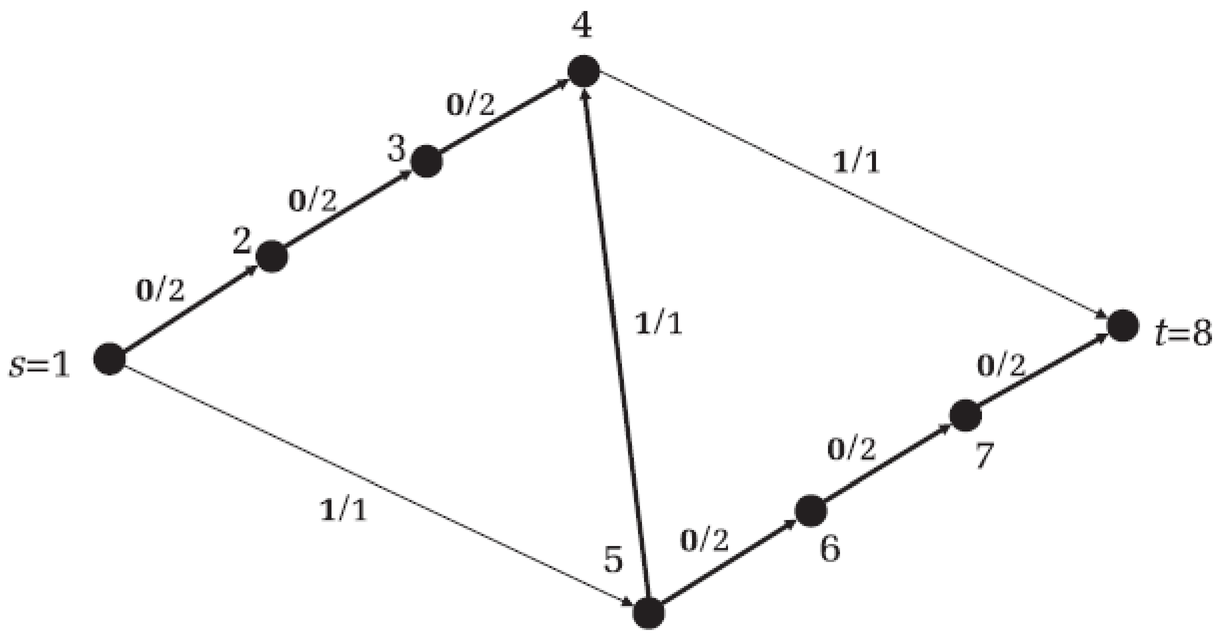


Példa arra, miért nem elég csak az irányított javítóutakat (amelyeken minden él előremutató) figyelembe venni:



A legrövidebb élszámú javítóút az $(1, 5, 4, 8)$ lesz, amelyen 1-gyel javítható a folyamérték. E javítást követően nincs több irányított javítóút.

Példa arra, miért nem elég csak az irányított javítóutakat (amelyeken minden él előremutató) figyelembe venni:



A hálózaton átfolyó folyamérték javítható az $(1, 2, 3, 4, 5, 6, 7, 8)$ úton, amelyen $(5, 4)$ visszamutató él.

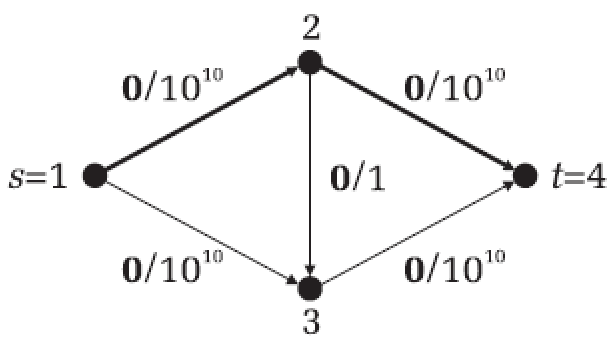
8.2 tétel:

Egy hálózaton átfolyó folyammennyiség akkor és csakis akkor maximális, ha a hálózatban nem létezik javítóút.

8.2 tétel:

Egy hálózaton átfolyó folyammennyiség akkor és csakis akkor maximális, ha a hálózatban nem létezik javítóút.

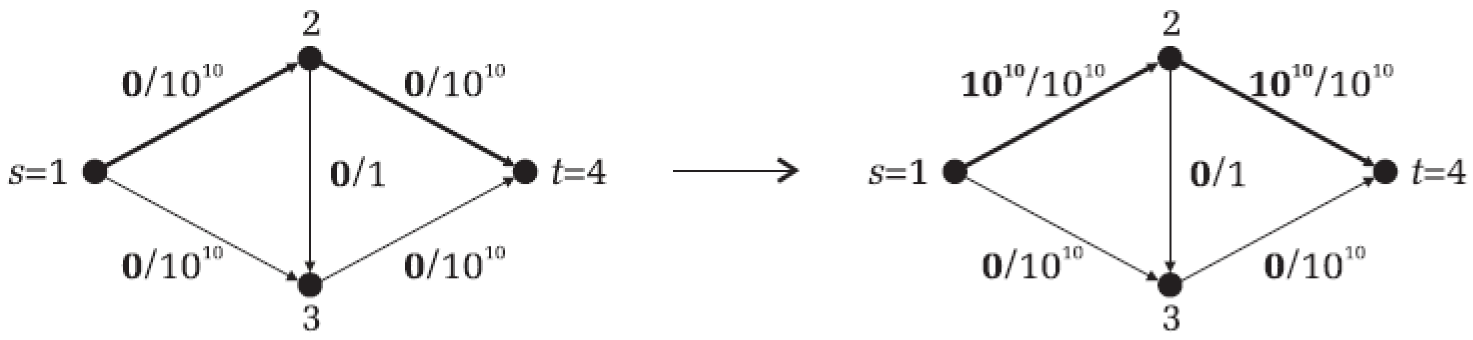
Az alábbi példa szemlélteti, hogy nem mindegy, milyen sorrendben találjuk meg a javítóutakat:



8.2 tétel:

Egy hálózaton átfolyó folyammennyiség akkor és csakis akkor maximális, ha a hálózatban nem létezik javítóút.

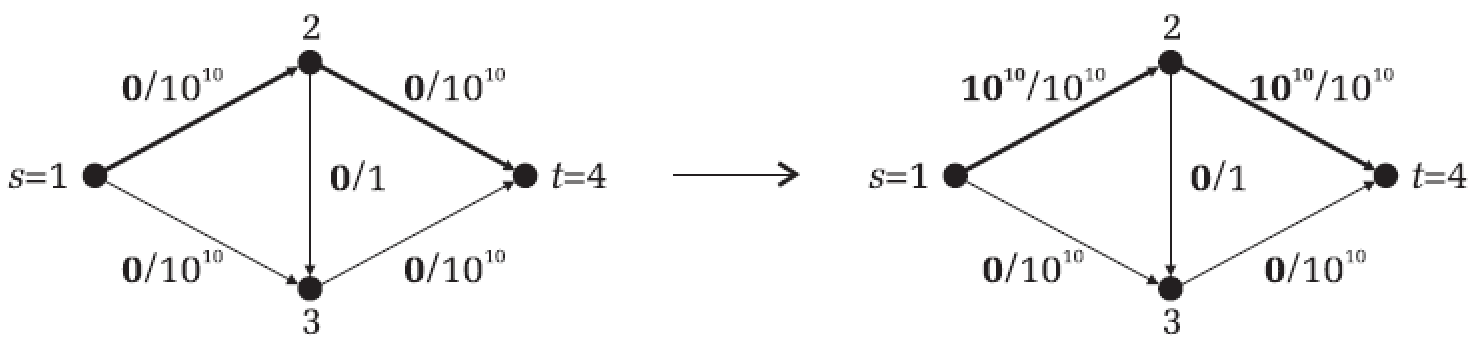
Az alábbi példa szemlélteti, hogy nem mindegy, milyen sorrendben találjuk meg a javítóutakat:

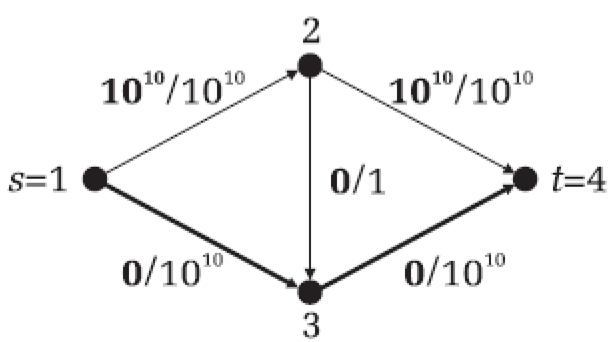


8.2 tétel:

Egy hálózaton átfolyó folyammennyiség akkor és csakis akkor maximális, ha a hálózatban nem létezik javítóút.

Az alábbi példa szemlélteti, hogy nem mindegy, milyen sorrendben találjuk meg a javítóutakat:

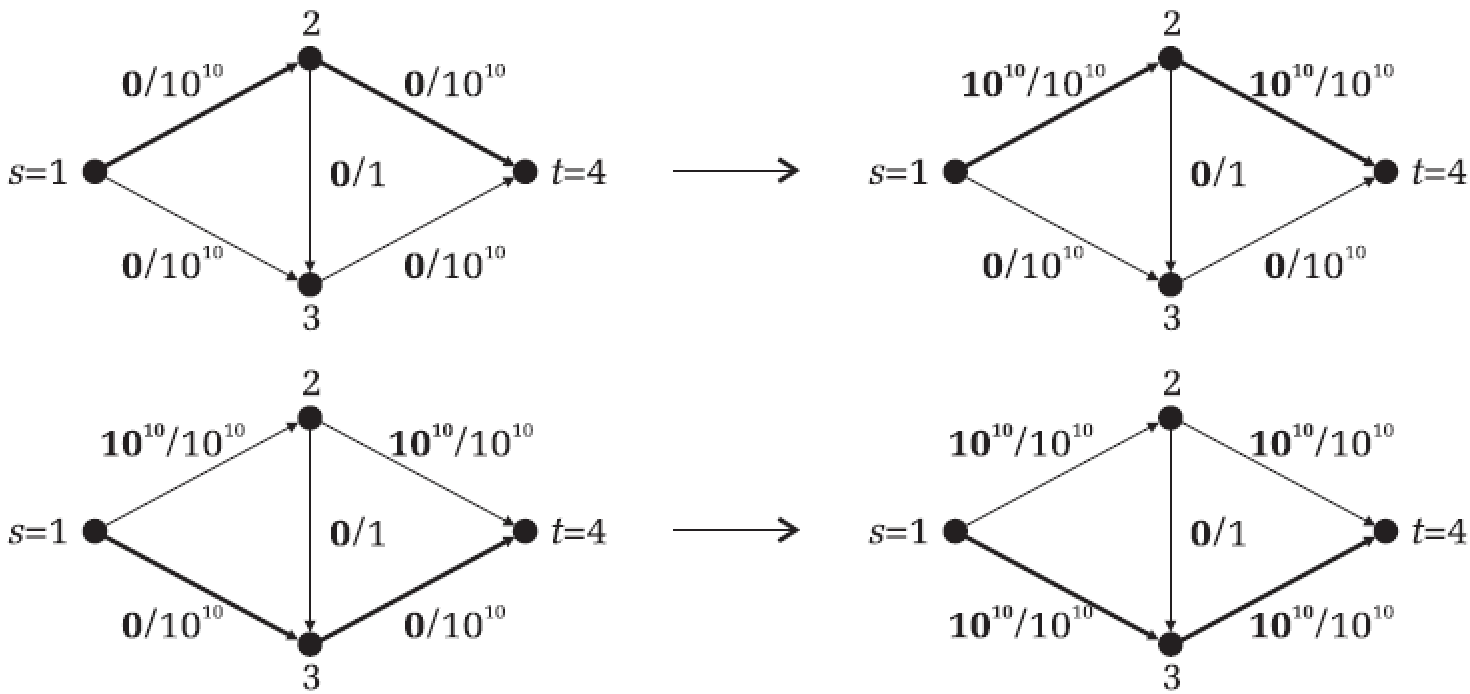




8.2 tétel:

Egy hálózaton átfolyó folyammennyiség akkor és csakis akkor maximális, ha a hálózatban nem létezik javítóút.

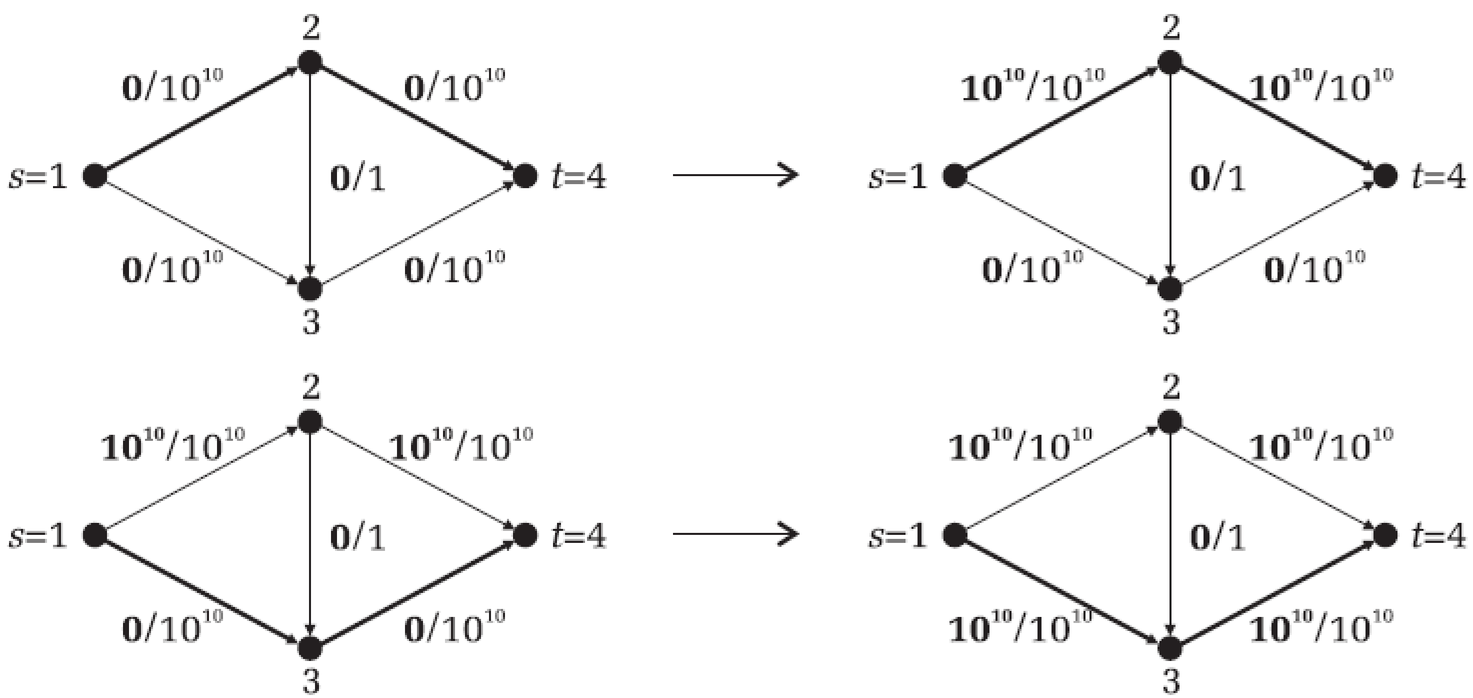
Az alábbi példa szemlélteti, hogy nem mindegy, milyen sorrendben találjuk meg a javítóutakat:



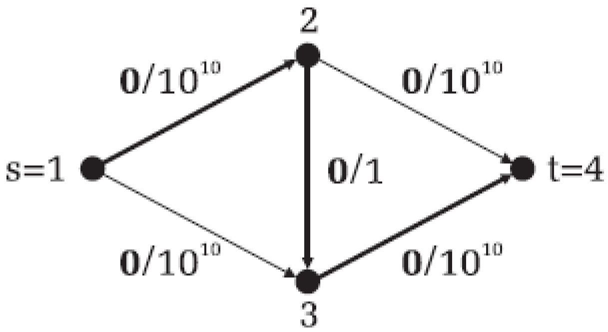
8.2 tétel:

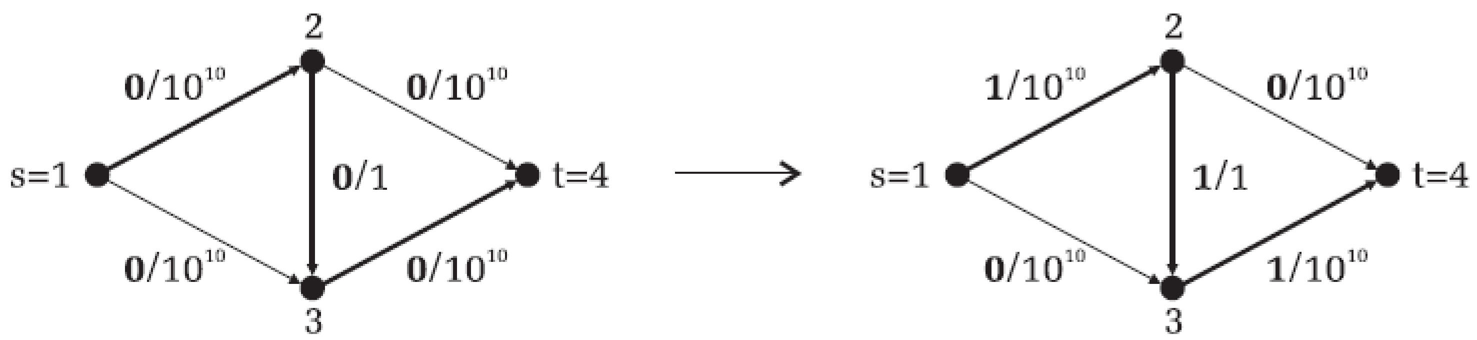
Egy hálózaton átfolyó folyammennyiség akkor és csakis akkor maximális, ha a hálózatban nem létezik javítóút.

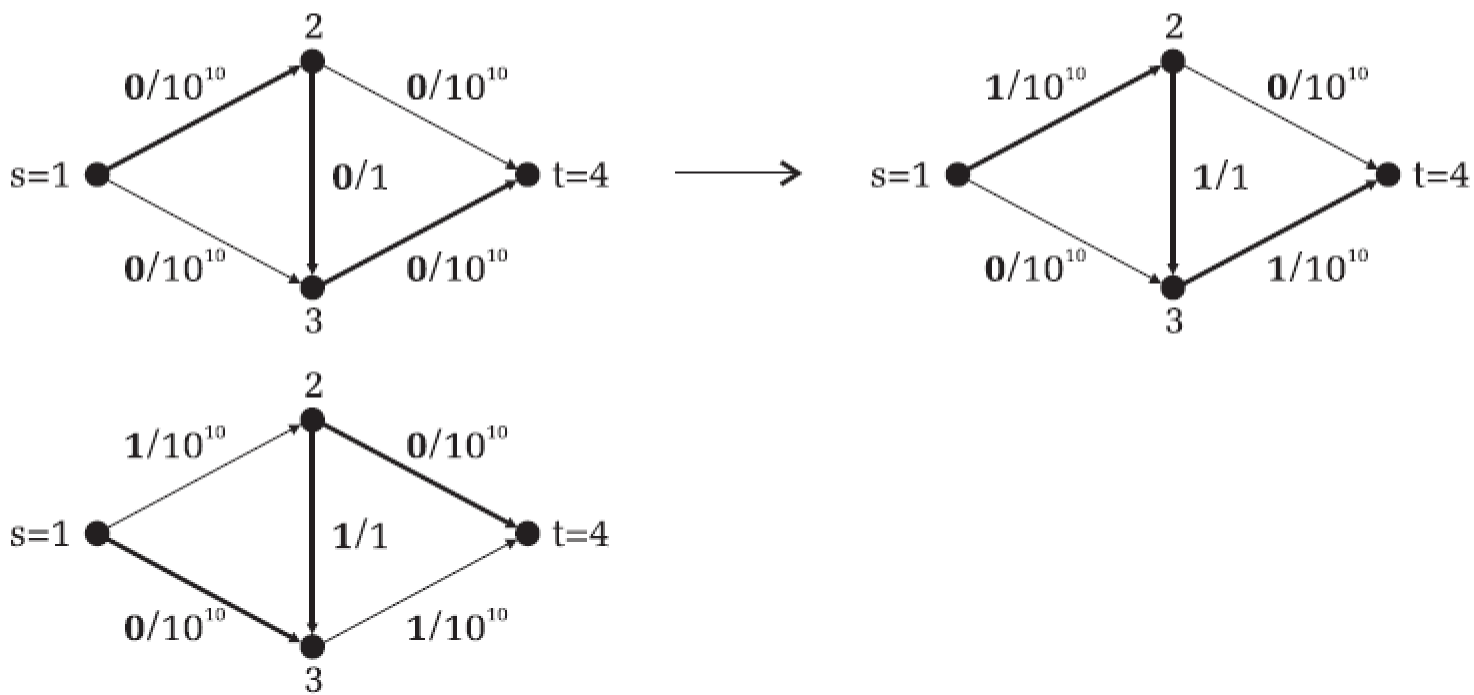
Az alábbi példa szemlélteti, hogy nem mindegy, milyen sorrendben találjuk meg a javítóutakat:

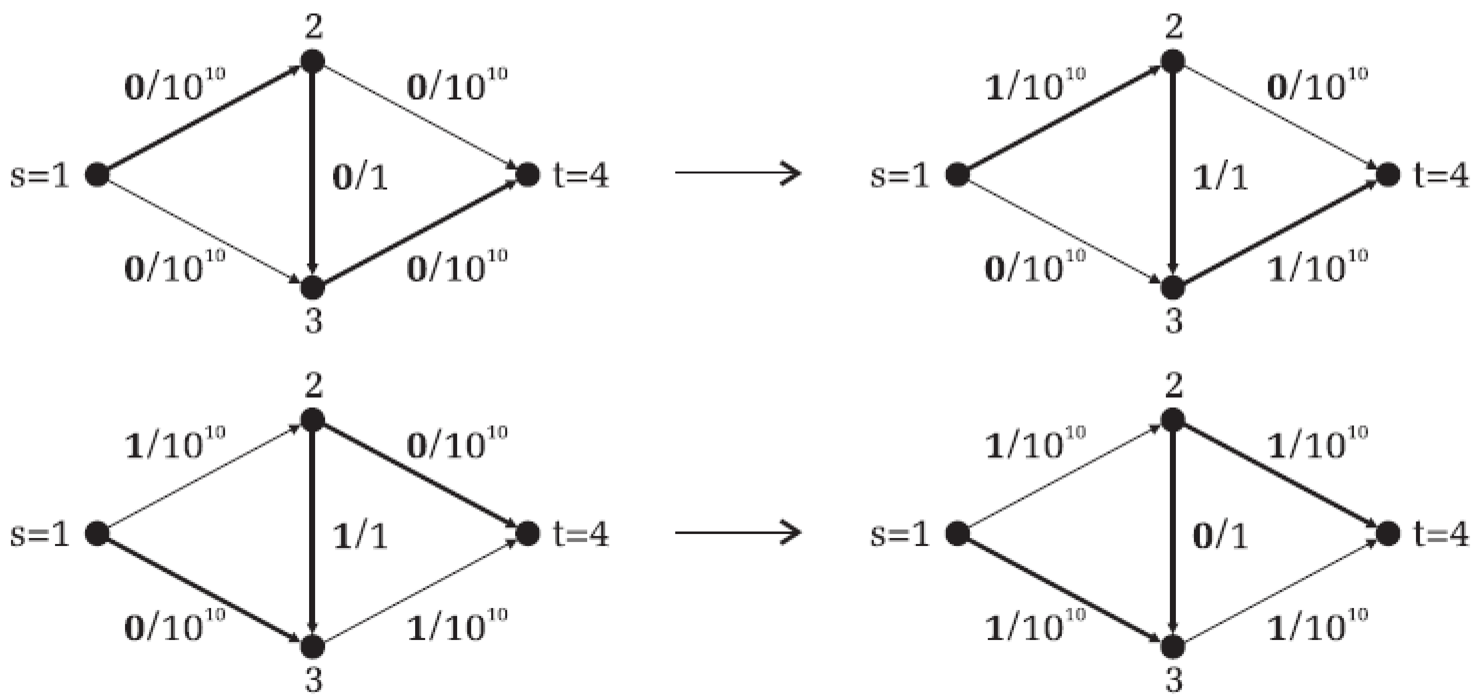


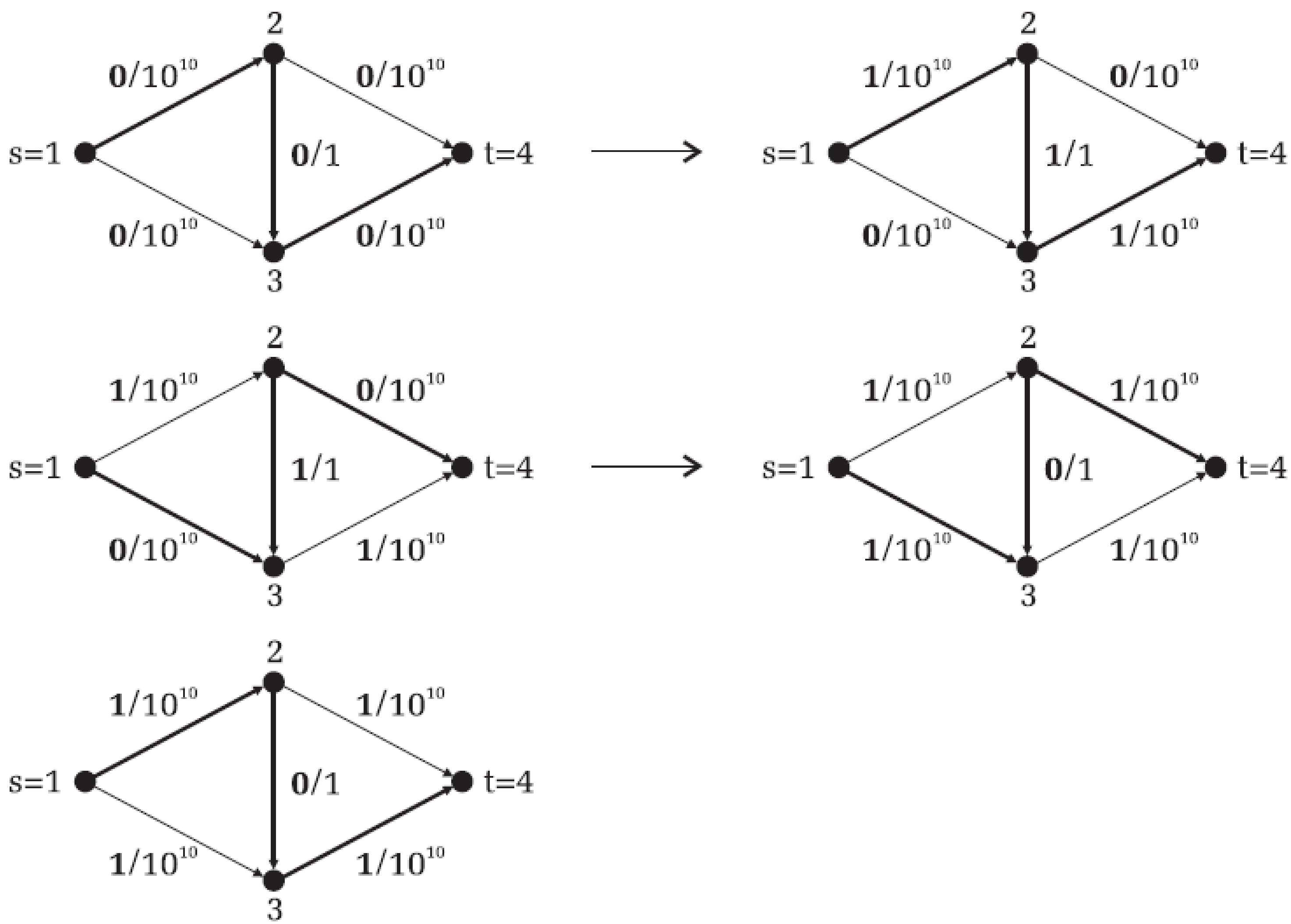
2 lépés

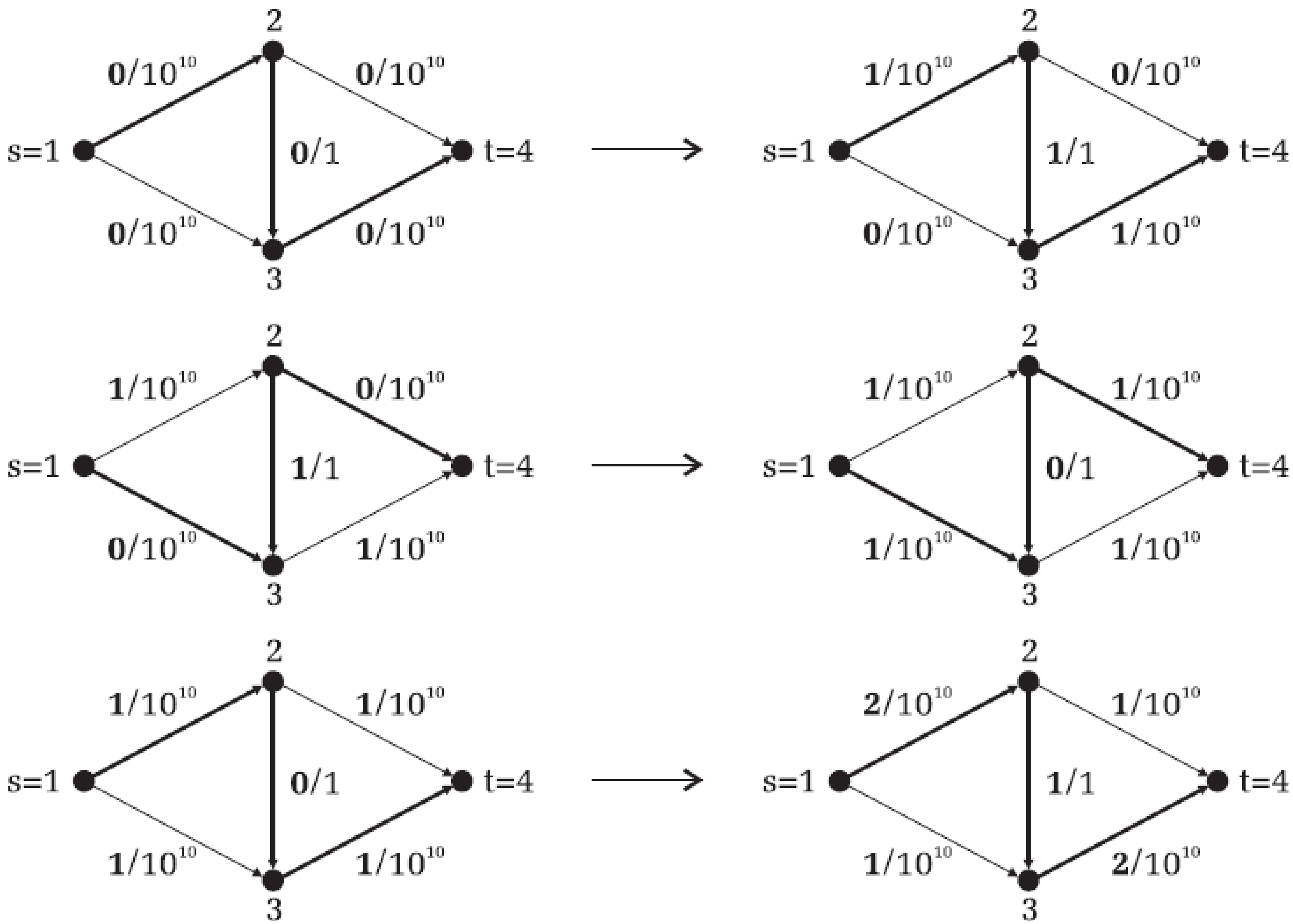

2
0/10^10
0/10^10
s=1
0/1
t=4
0/10^10
0/10^10
3

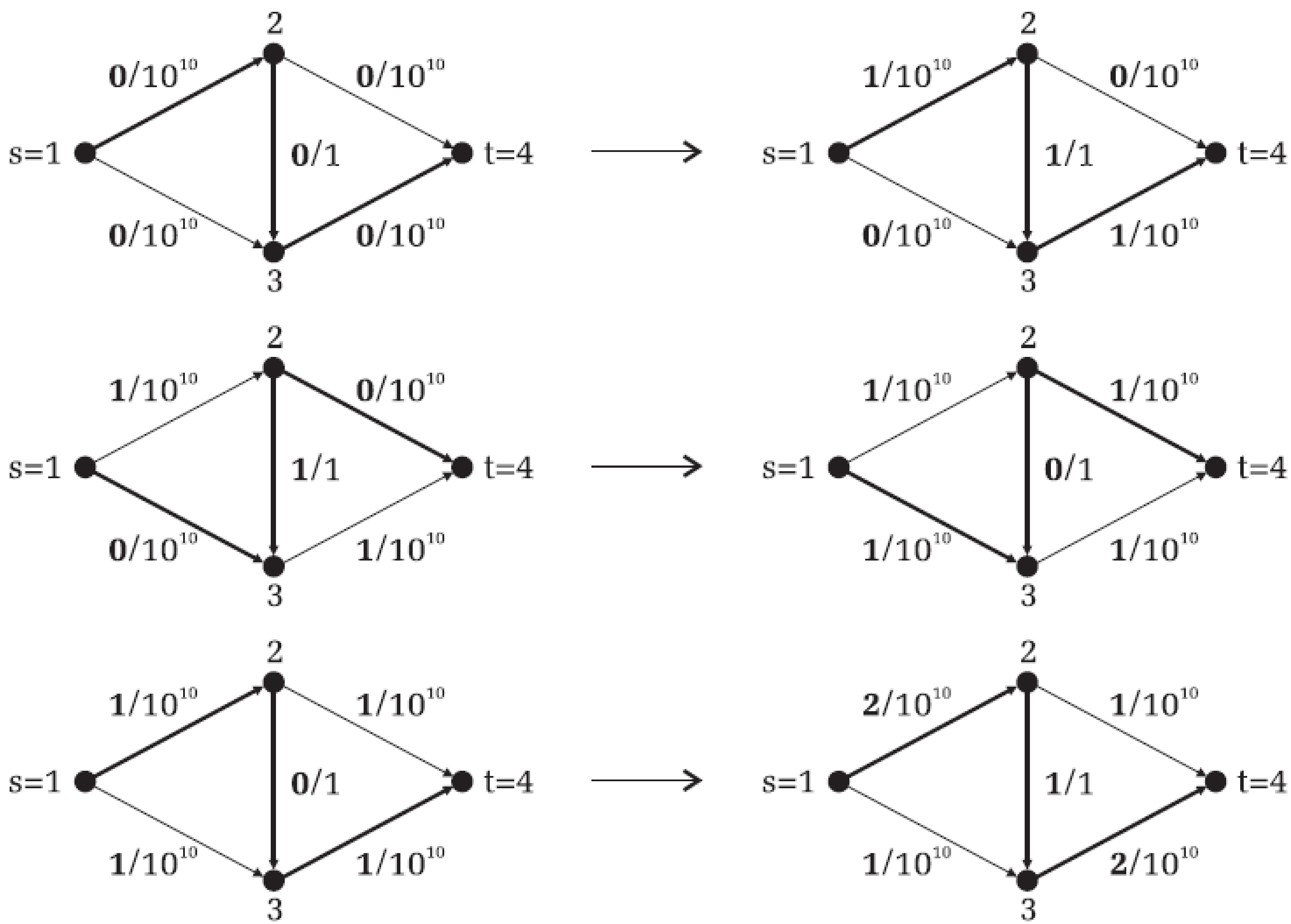

2
0/10^10
0/10^10
s=1
0/1
t=4
0/10^10
0/10^10
3
2
1/10^10
0/10^10
s=1
1/1
t=4
0/10^10
1/10^10
3


2
0/10^10
0/10^10
s=1
0/1
t=4
0/10^10
0/10^10
3
2
1/10^10
0/10^10
s=1
1/1
t=4
0/10^10
1/10^10
3
2
1/10^10
0/10^10
s=1
1/1
t=4
0/10^10
1/10^10
3


2
0/10^10
0/10^10
s=1
0/1
t=4
0/10^10
0/10^10
3
2
1/10^10
0/10^10
s=1
1/1
t=4
0/10^10
1/10^10
3
2
1/10^10
0/10^10
s=1
1/1
t=4
0/10^10
1/10^10
3
2
1/10^10
1/10^10
s=1
0/1
t=4
1/10^10
1/10^10
3


2
0/10^10
0/10^10
s=1
0/1
t=4
0/10^10
0/10^10
3
2
1/10^10
0/10^10
s=1
1/1
t=4
0/10^10
1/10^10
3
2
1/10^10
0/10^10
s=1
1/1
t=4
0/10^10
1/10^10
3
2
1/10^10
1/10^10
s=1
0/1
t=4
1/10^10
1/10^10
3
2
1/10^10
1/10^10
s=1
0/1
t=4
1/10^10
1/10^10
3


2
3
s=1
t=4
0/10^10
0/10^10
0/1
0/10^10
0/10^10
2
3
s=1
t=4
1/10^10
0/10^10
1/1
0/10^10
1/10^10
2
3
s=1
t=4
1/10^10
0/10^10
1/1
0/10^10
1/10^10
2
3
s=1
t=4
1/10^10
1/10^10
0/1
1/10^10
1/10^10
2
3
s=1
t=4
1/10^10
1/10^10
0/1
1/10^10
1/10^10
2
3
s=1
t=4
2/10^10
1/10^10
1/1
1/10^10
2/10^10


2
0/10^10
0/10^10
s=1
0/1
t=4
0/10^10
0/10^10
3
2
1/10^10
0/10^10
s=1
1/1
t=4
0/10^10
1/10^10
3
2
1/10^10
0/10^10
s=1
1/1
t=4
0/10^10
1/10^10
3
2
1/10^10
1/10^10
s=1
0/1
t=4
1/10^10
1/10^10
3
2
1/10^10
1/10^10
s=1
0/1
t=4
1/10^10
1/10^10
3
2
2/10^10
1/10^10
s=1
1/1
t=4
1/10^10
2/10^10
3

(...)

2^{10} lépés

Ford-Fulkerson algoritmus bonyolultsága

Az algoritmus bonyolultsága tehát függ a folyam értékétől.

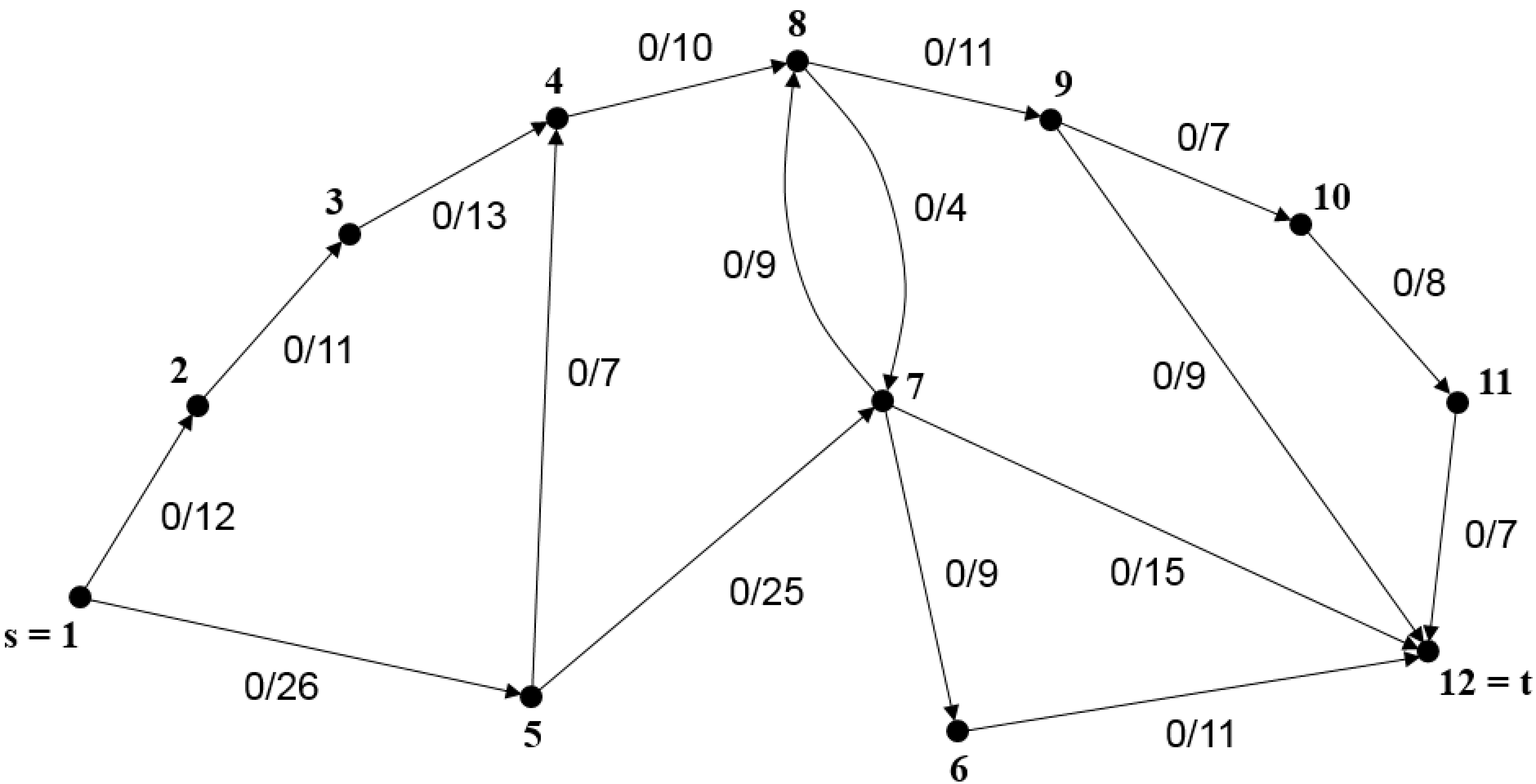
Legyen n a hálózati gráf csúcspontjainak, m éleinek száma, f pedig a folyam értéke. Ekkor az algoritmus bonyolultsága $O(mf)$.

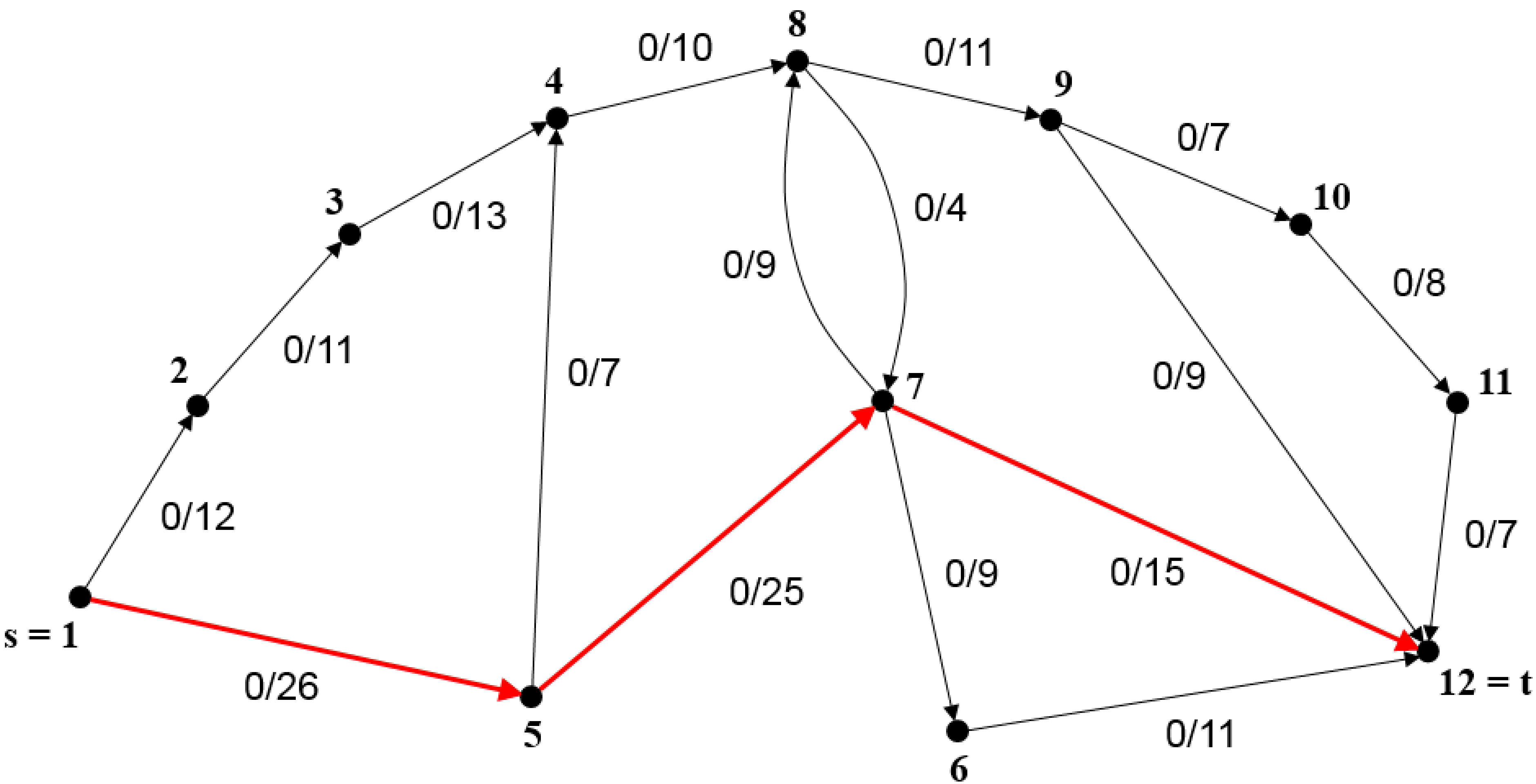
Ford-Fulkerson algoritmus bonyolultsága

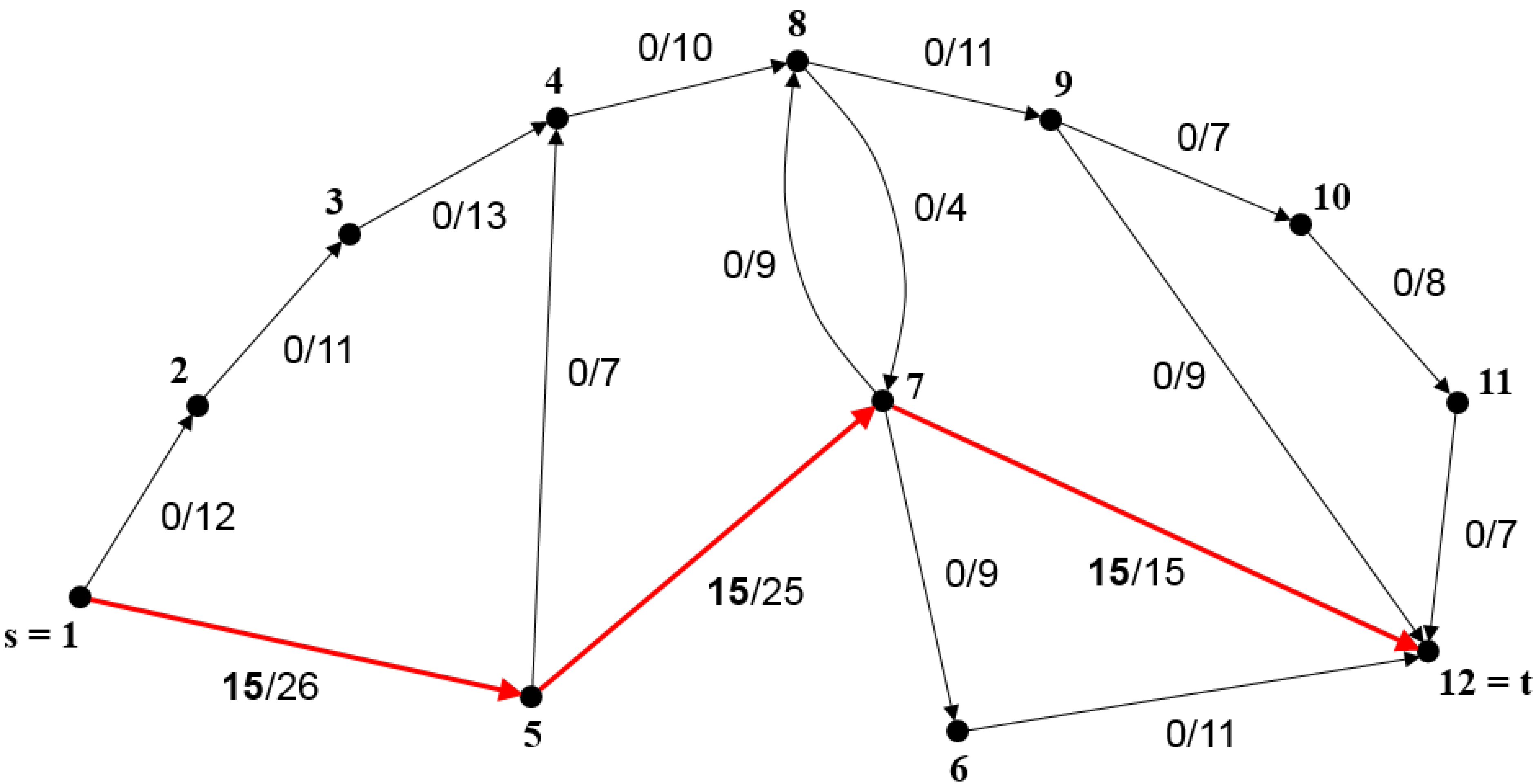
Az algoritmus bonyolultsága tehát függ a folyam értékétől.

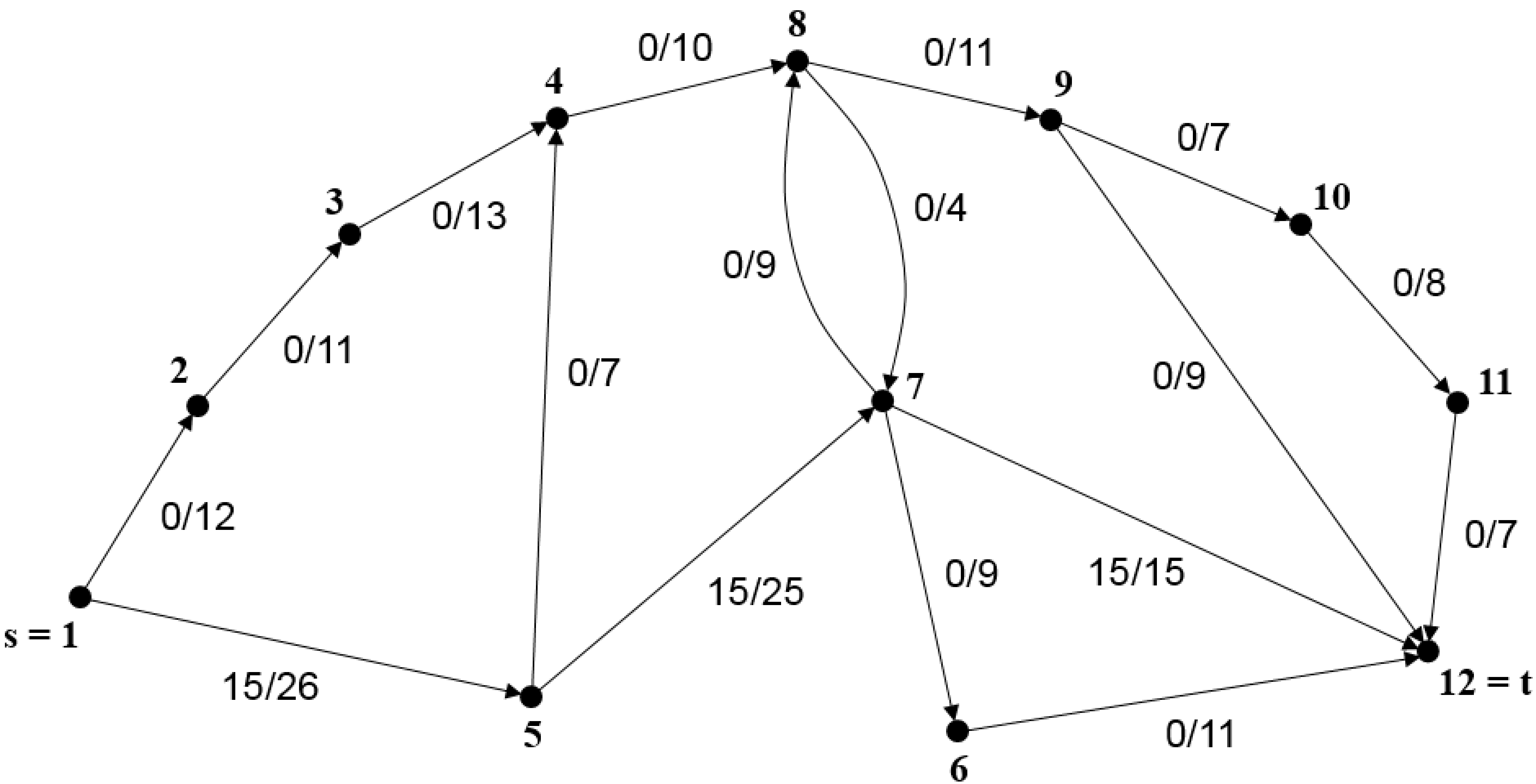
Legyen n a hálózati gráf csúcspontjainak, m éleinek száma, f pedig a folyam értéke. Ekkor az algoritmus bonyolultsága $O(mf)$.

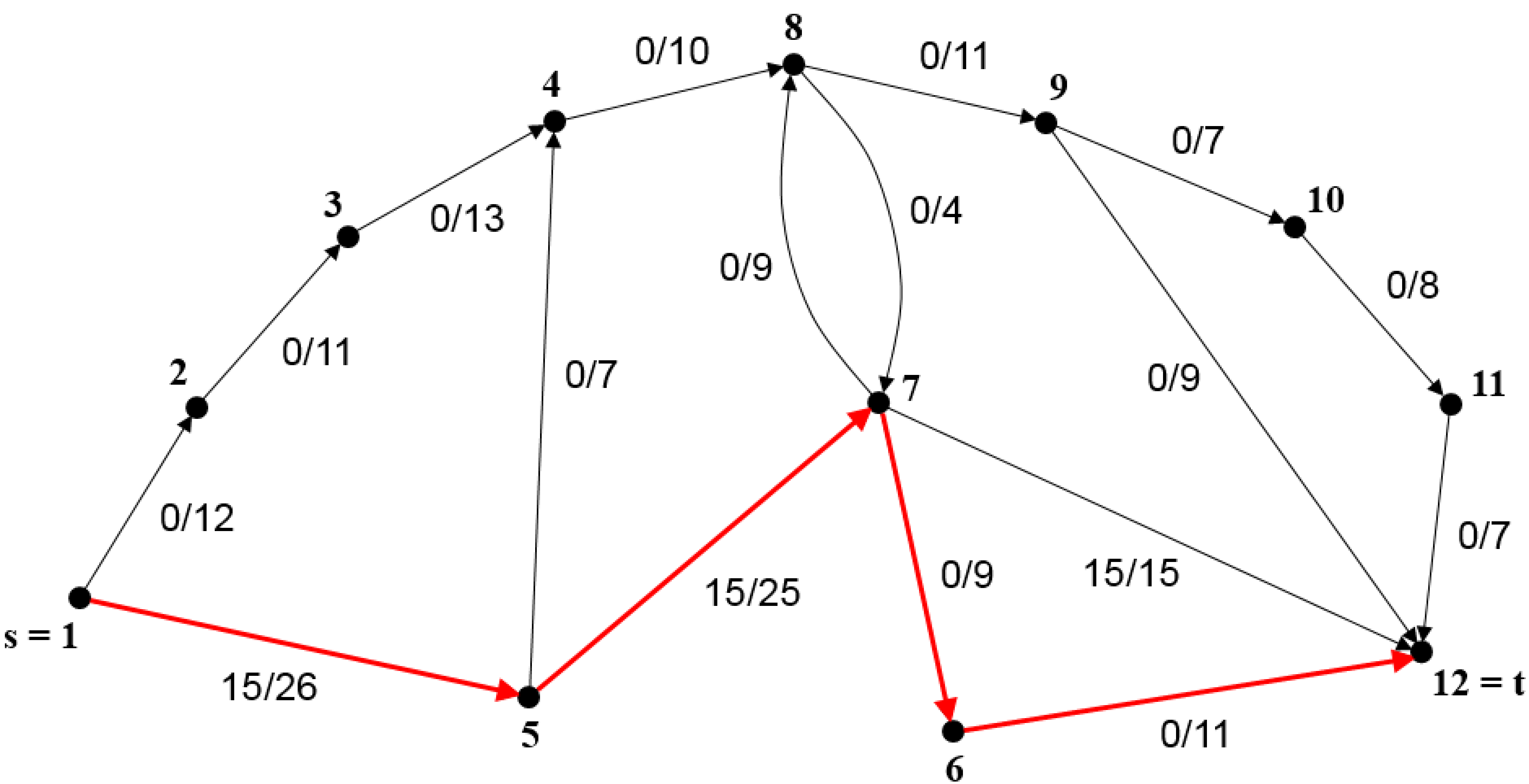
Az algoritmus futási ideje javítható, ha minden lépésben a legkevesebb élt tartalmazó javítóutat választjuk. Ekkor az algoritmus bonyolultsága $O(nm^2)$.

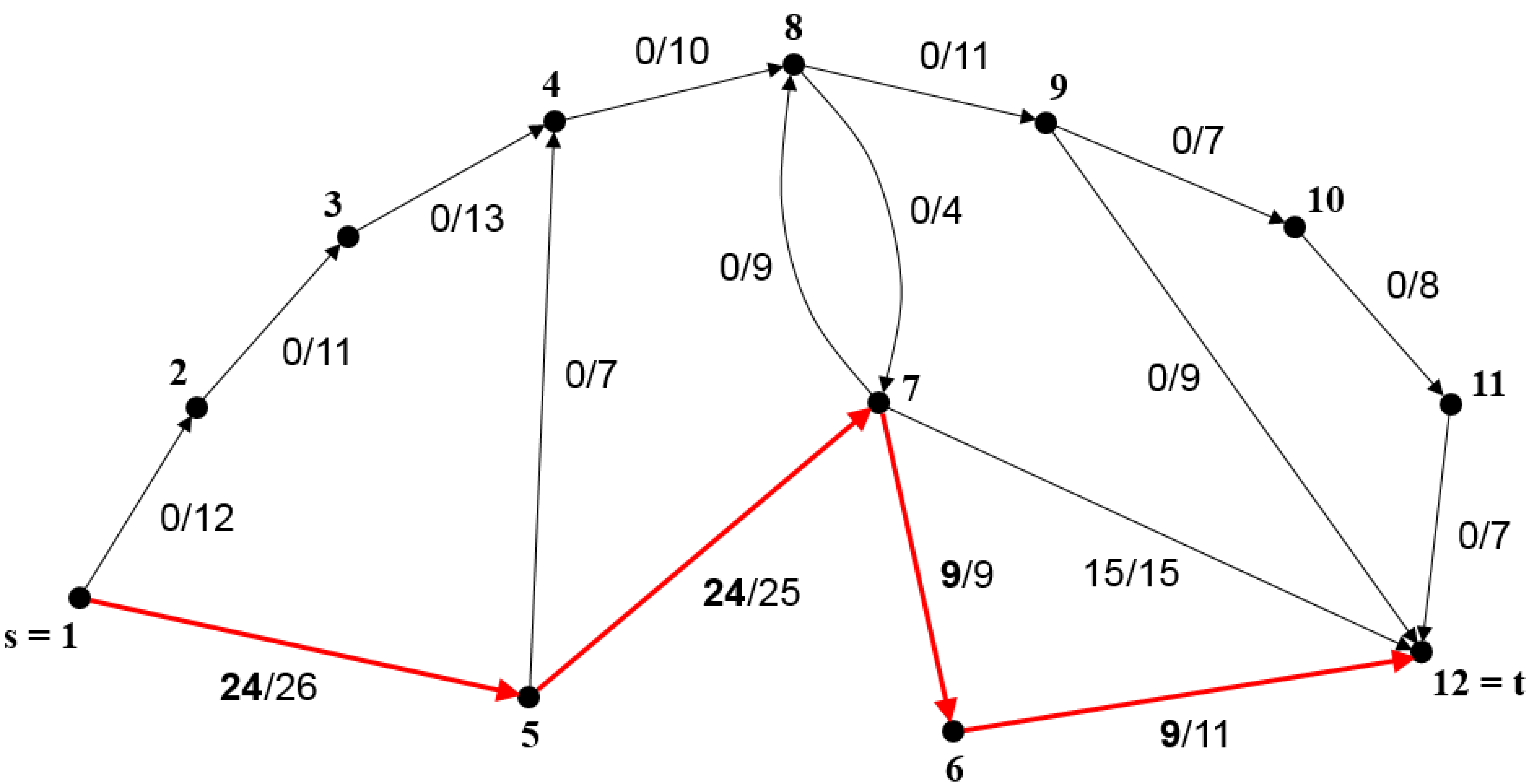

8
4
9
3
10
2
7
11
s = 1
12 = t
5
6
0/10
0/11
0/7
0/13
0/4
0/9
0/8
0/11
0/7
0/9
0/12
0/7
0/9
0/15
0/25
0/26
0/11

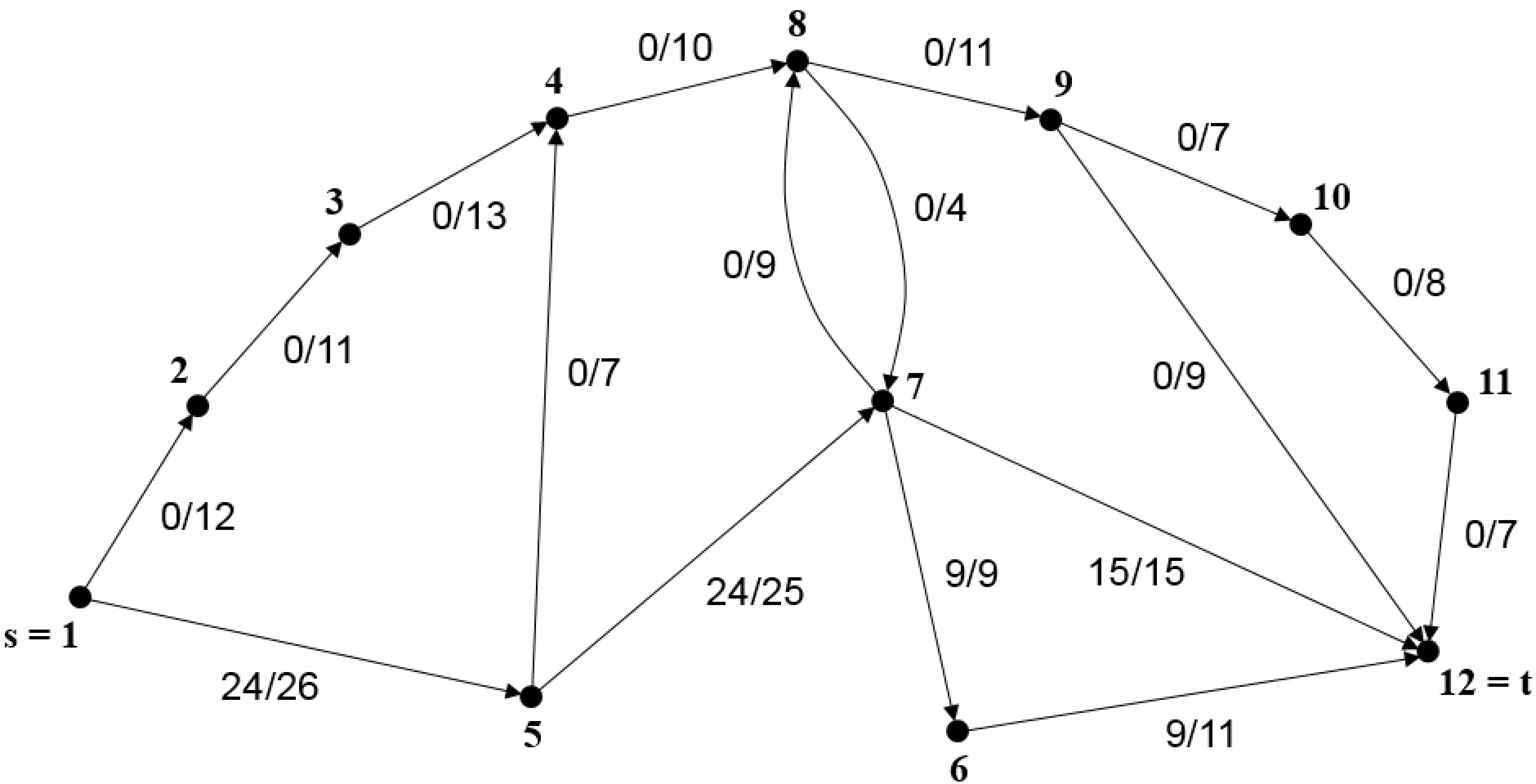

8
0/10
0/11
4
9
0/7
3
0/13
0/4
10
0/9
0/8
2
0/11
0/7
7
0/9
11
0/12
0/7
0/9
0/15
0/25
s = 1
0/26
12 = t
5
0/11
6

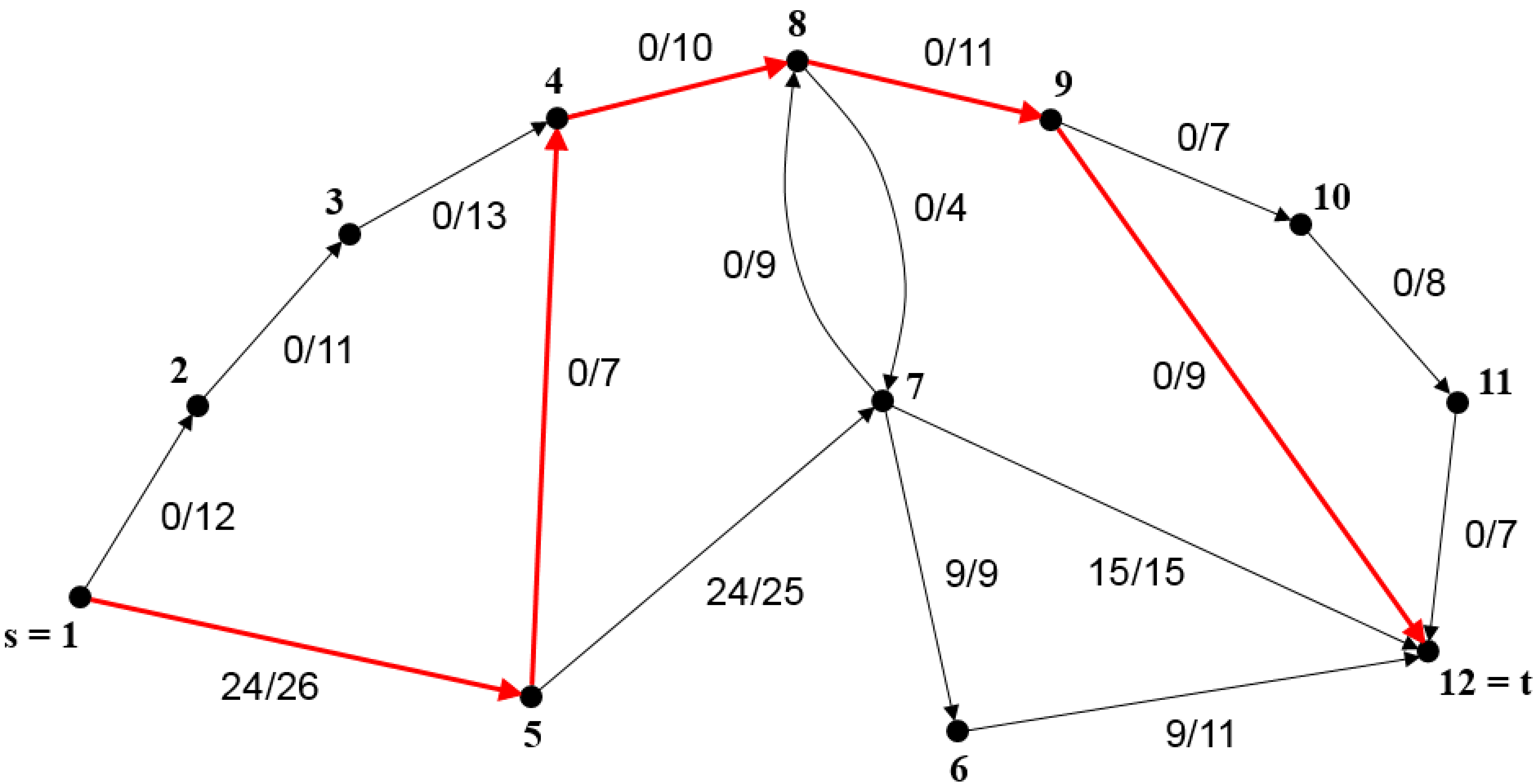

8
0/10
0/11
4
9
0/7
3
0/13
0/4
10
0/9
0/8
2
0/11
0/7
7
0/9
11
0/12
0/7
0/9
15/15
15/25
s = 1
12 = t
15/26
5
0/11
6

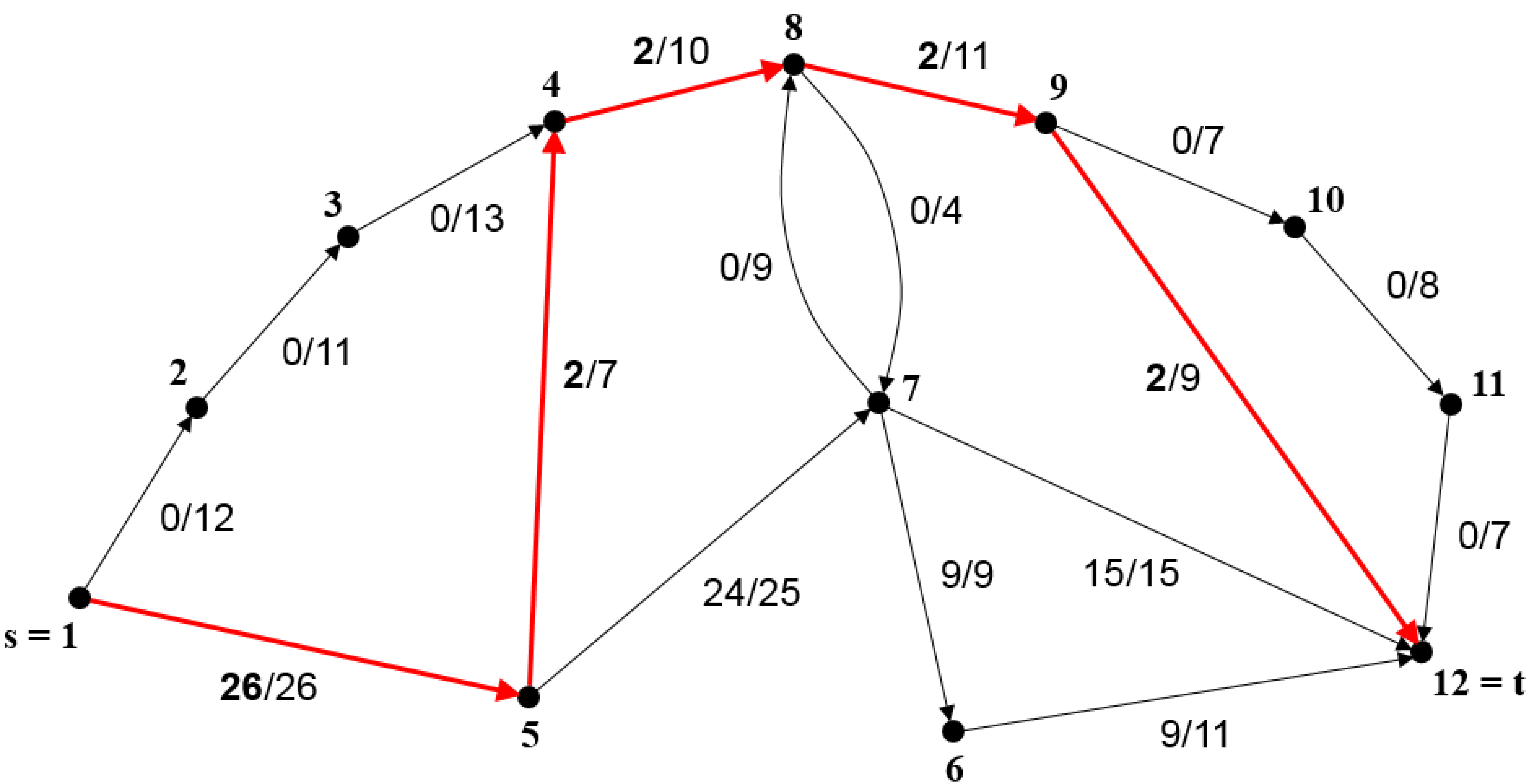

8
0/10
0/11
4
9
0/7
3
0/13
0/4
10
0/9
0/8
2
0/11
0/7
7
0/9
11
0/12
0/7
0/9
15/15
15/25
s = 1
15/26
12 = t
5
0/11
6

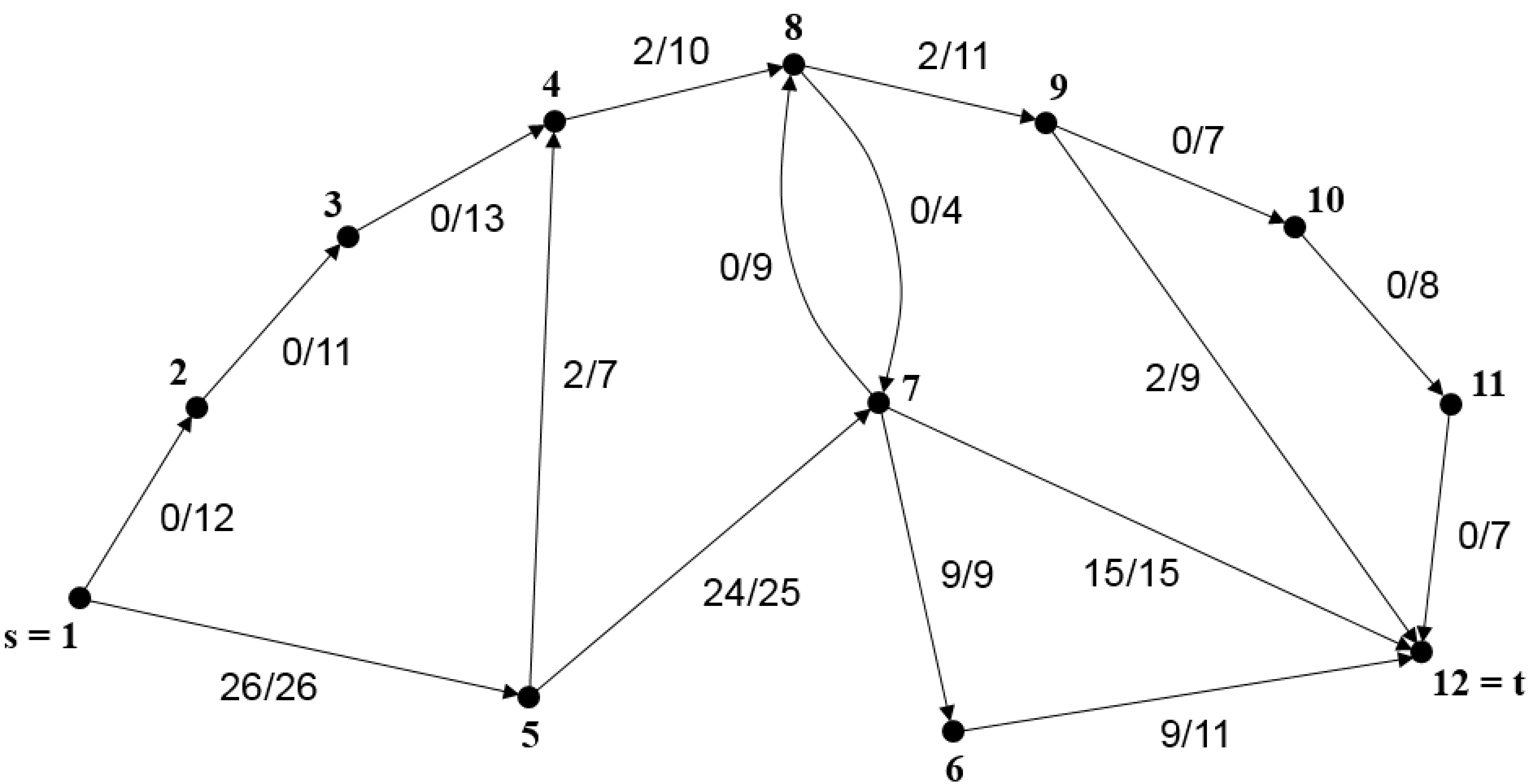

8
4
9
3
10
2
7
11
s = 1
5
6
12 = t
0/10
0/11
0/7
0/13
0/4
0/9
0/8
0/11
0/7
0/9
0/12
0/7
15/15
0/9
15/25
15/26
0/11

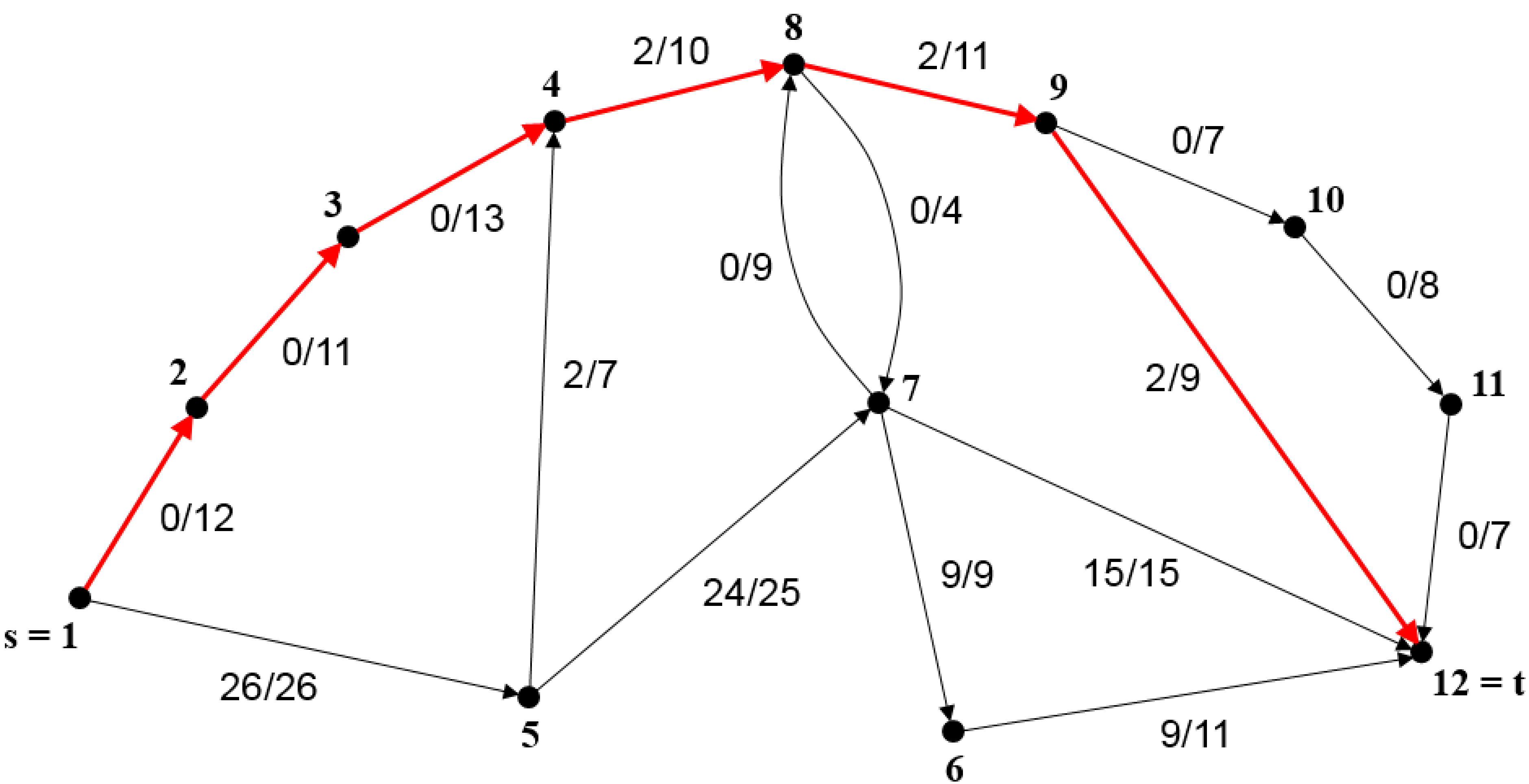

8
0/10
0/11
4
9
0/7
3
0/13
0/4
10
0/9
0/8
2
0/11
0/7
7
0/9
11
0/12
9/9
15/15
0/7
24/25
s = 1
24/26
12 = t
5
9/11
6

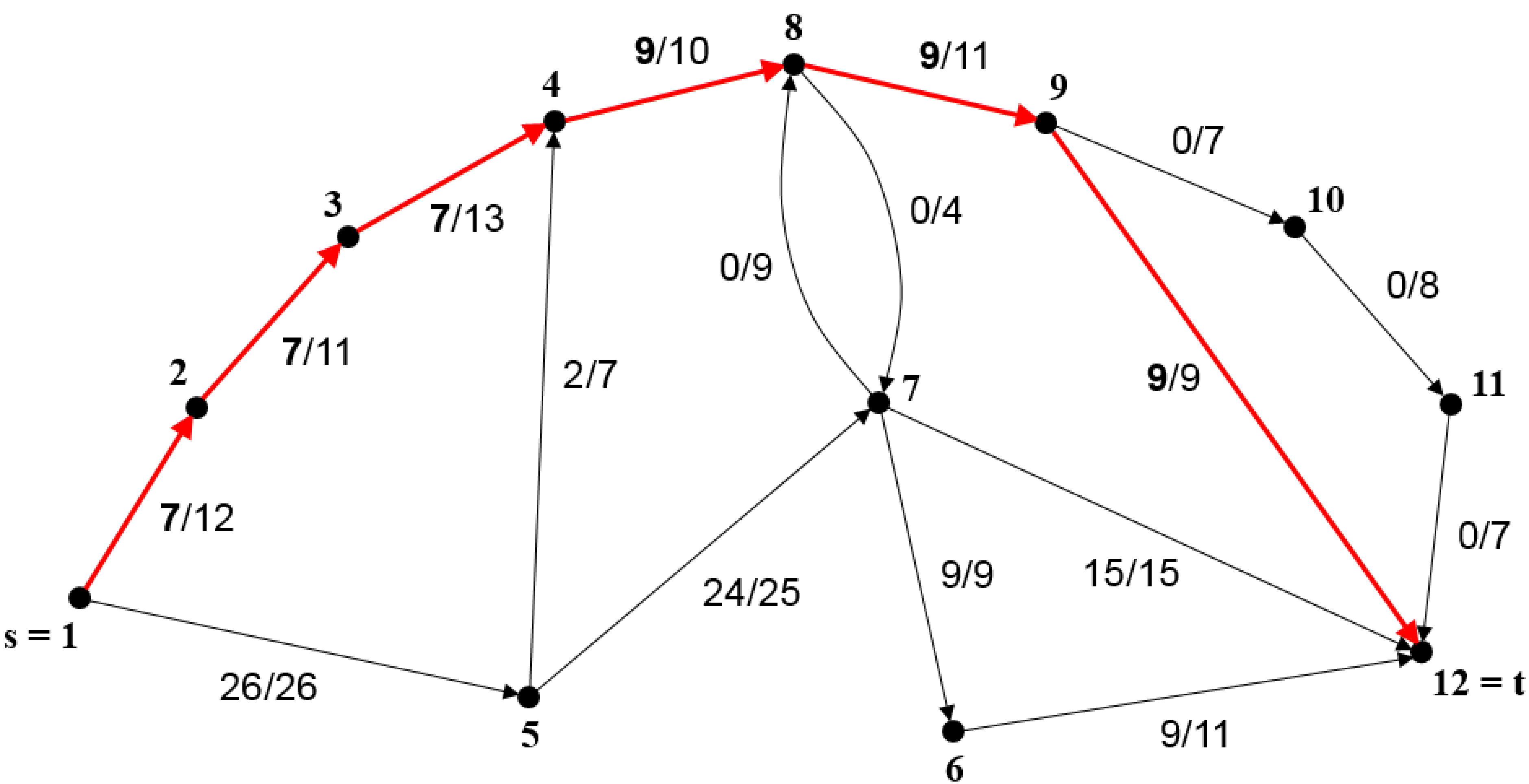

s = 1
2
3
4
5
6
7
8
9
10
11
12 = t
0/12
24/26
0/11
0/13
0/7
0/10
24/25
0/9
0/4
0/11
9/9
15/15
9/11
0/9
0/7
0/8
0/7

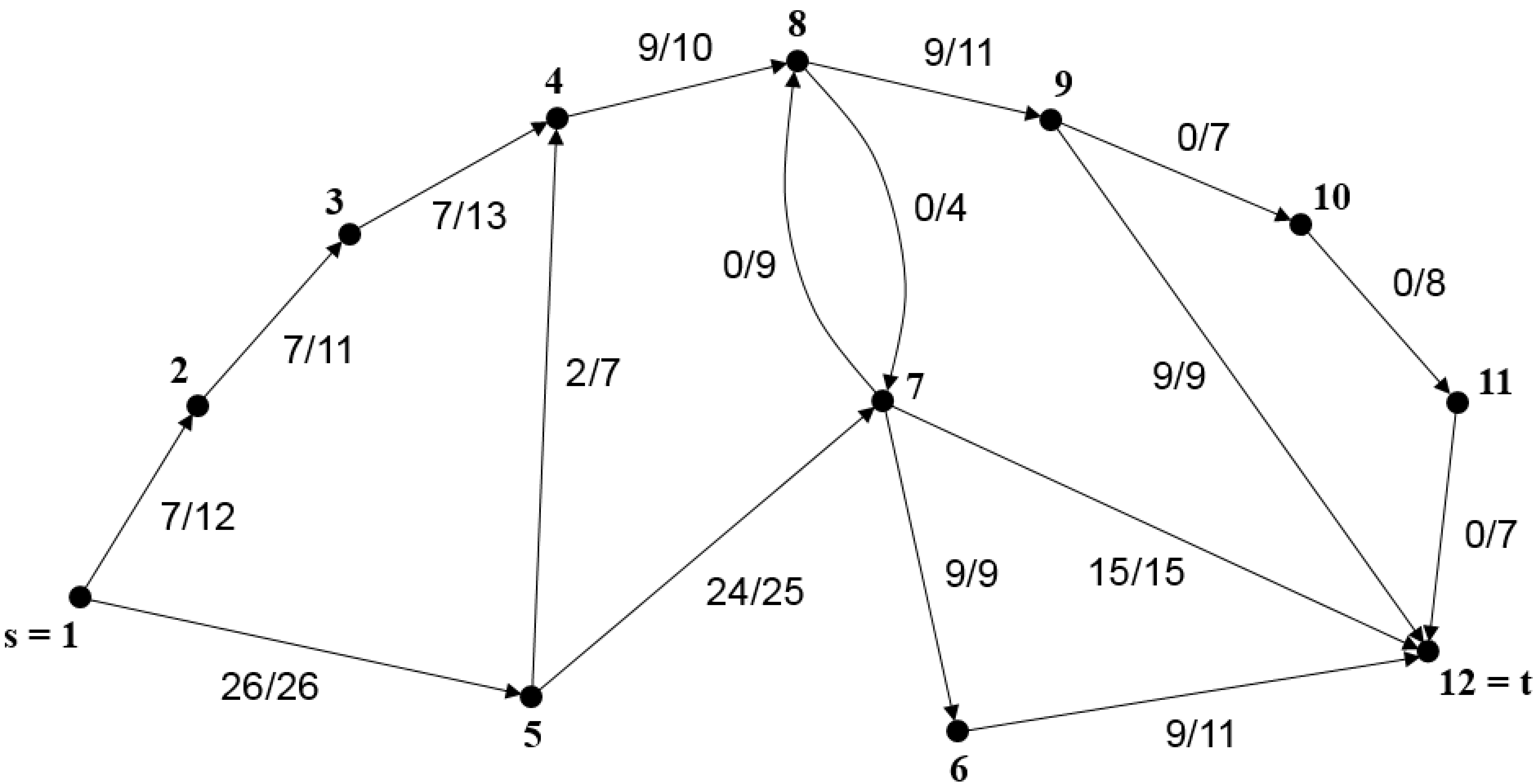

8
0/10
0/11
4
9
0/7
3
0/13
0/4
10
0/9
0/8
2
0/11
0/7
7
0/9
11
0/12
0/7
9/9
15/15
24/25
s = 1
24/26
12 = t
5
9/11
6

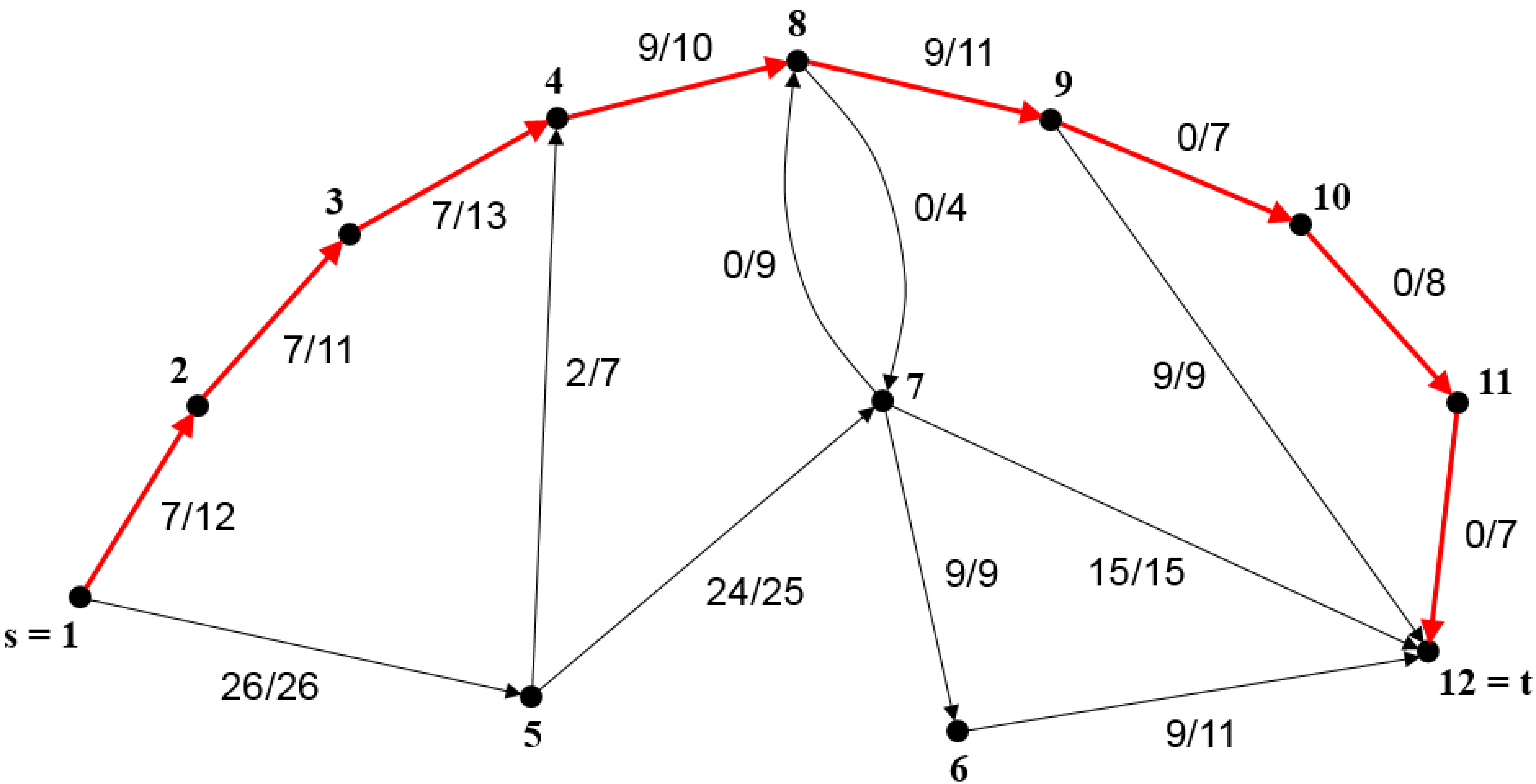

8
2/10
2/11
4
9
0/7
3
0/13
0/4
10
0/9
0/8
2
0/11
2/7
2/9
7
11
0/12
0/7
9/9
15/15
24/25
s = 1
26/26
12 = t
5
9/11
6

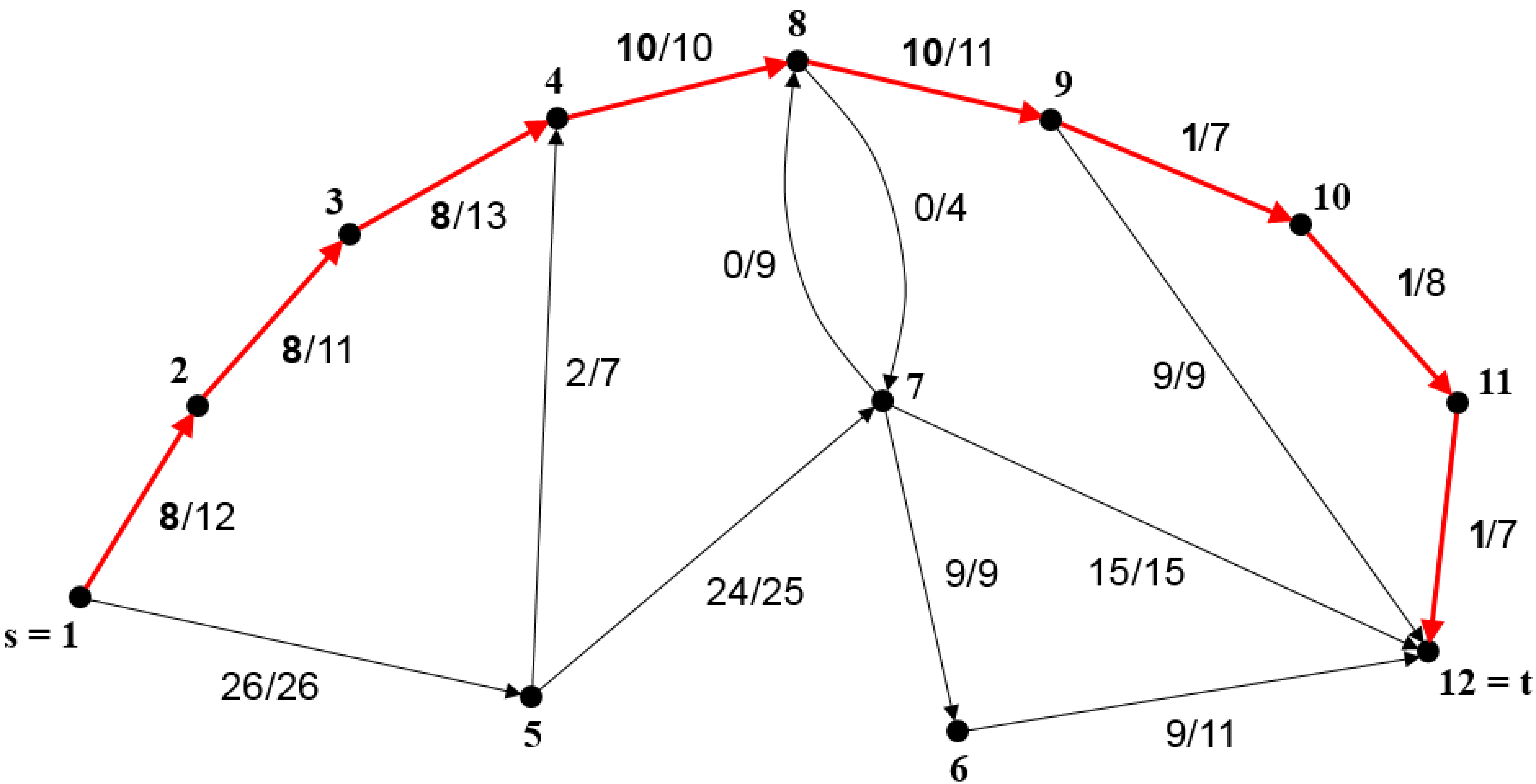

8
2/10
2/11
4
9
0/7
3
0/13
0/4
10
0/9
0/8
2
0/11
2/7
7
2/9
11
0/12
0/7
9/9
15/15
24/25
s = 1
26/26
12 = t
5
9/11
6

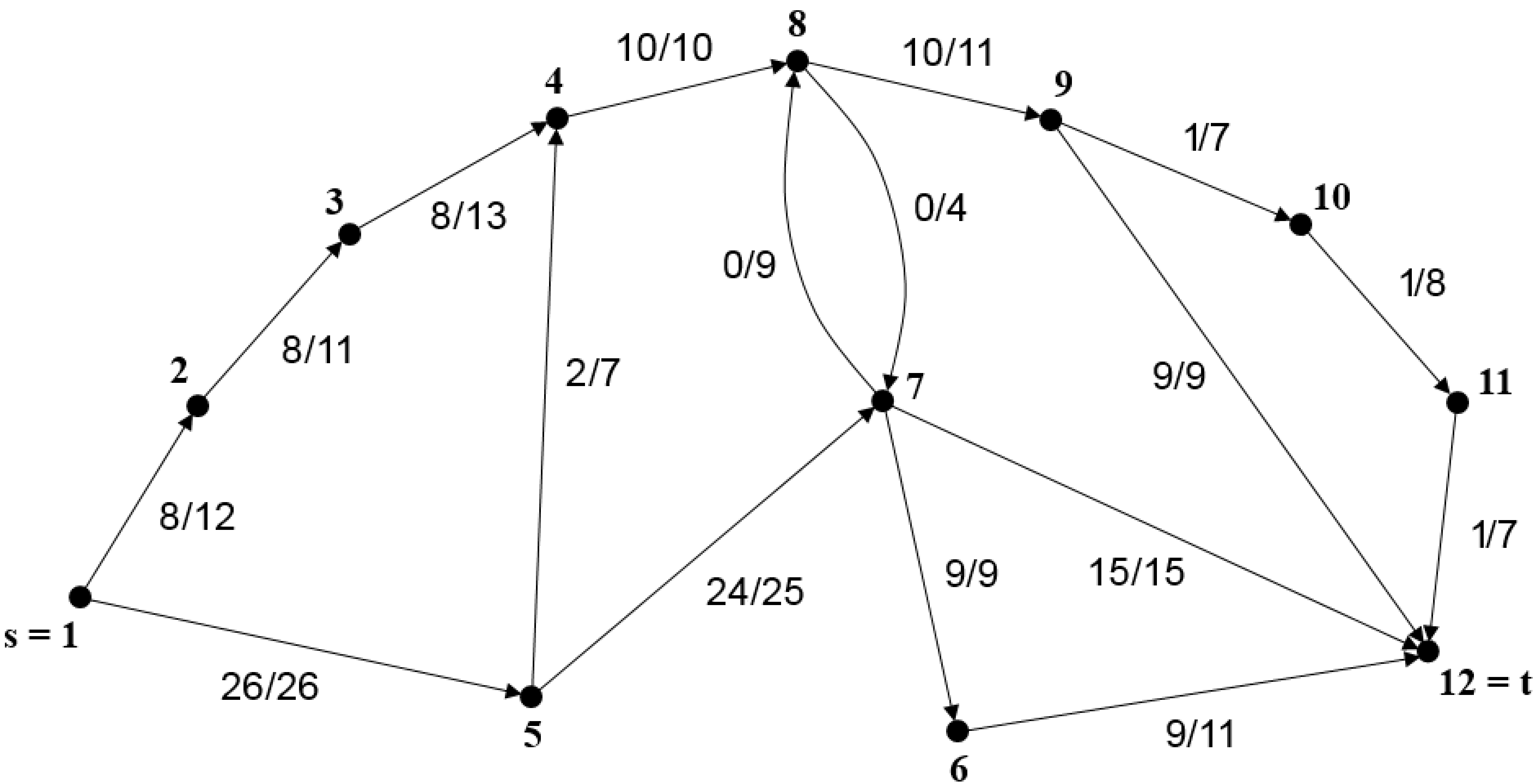

8
2/10
2/11
4
9
0/7
3
0/13
0/4
10
0/9
0/8
2
0/11
2/7
7
2/9
11
0/12
0/7
9/9
15/15
24/25
s = 1
26/26
12 = t
9/11
5
6

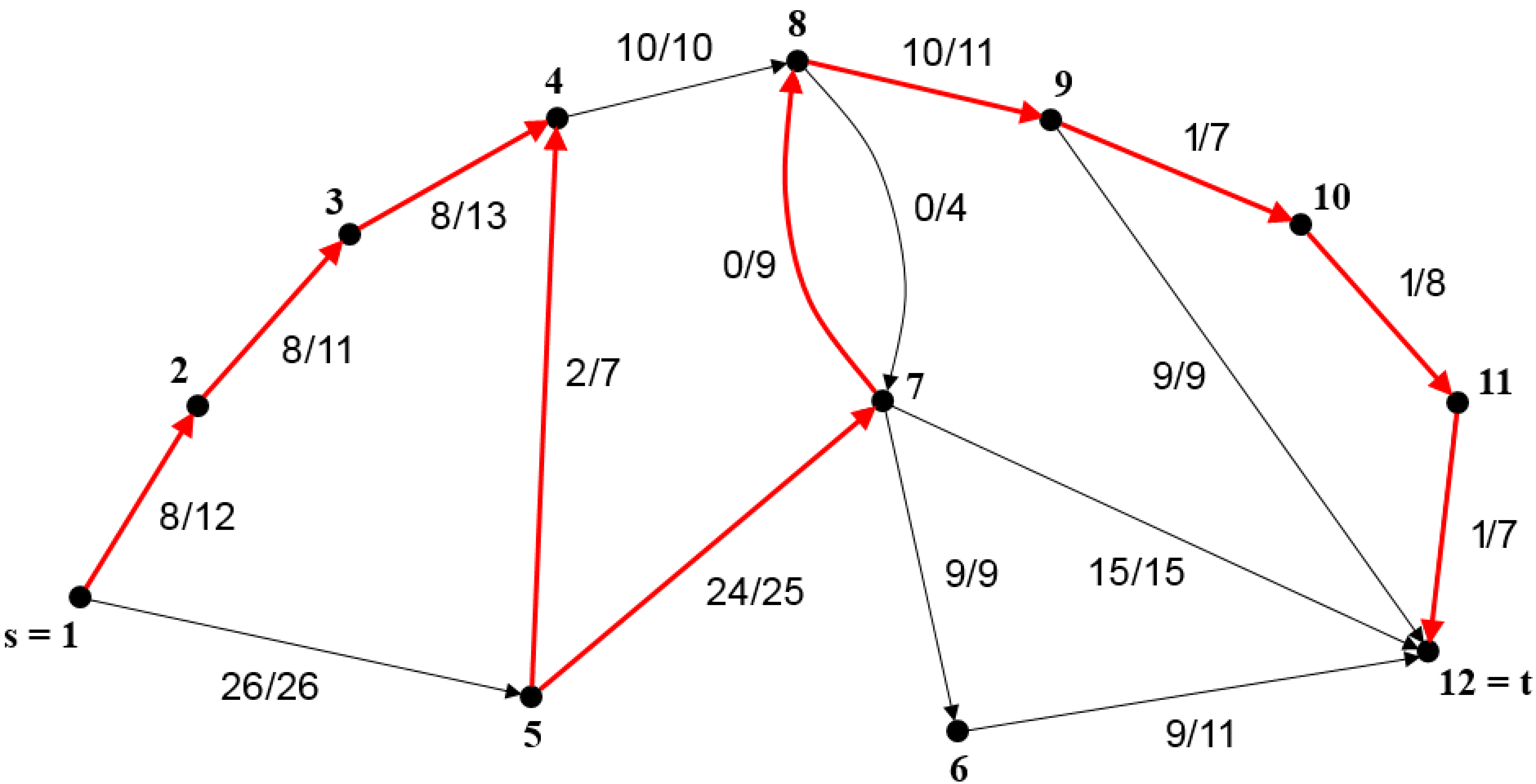

8
9/10
9/11
4
9
0/7
3
7/13
0/4
10
0/9
0/8
2
7/11
2/7
9/9
7
11
7/12
0/7
9/9
15/15
24/25
s = 1
12 = t
26/26
5
9/11
6

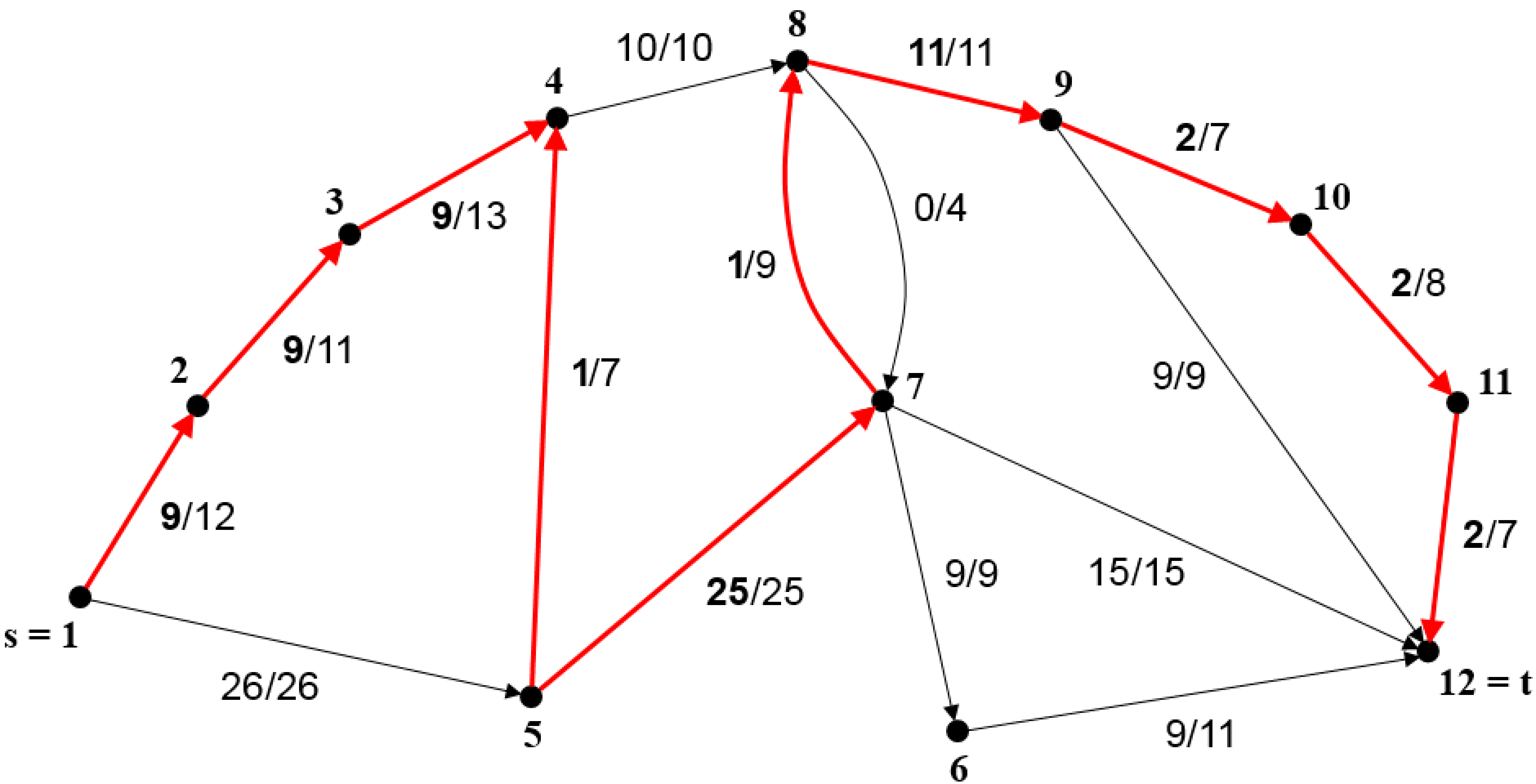

8
9/10
9/11
4
9
0/7
3
7/13
0/4
10
0/9
0/8
7/11
2
2/7
7
9/9
11
7/12
0/7
9/9
15/15
24/25
s = 1
12 = t
26/26
5
9/11
6

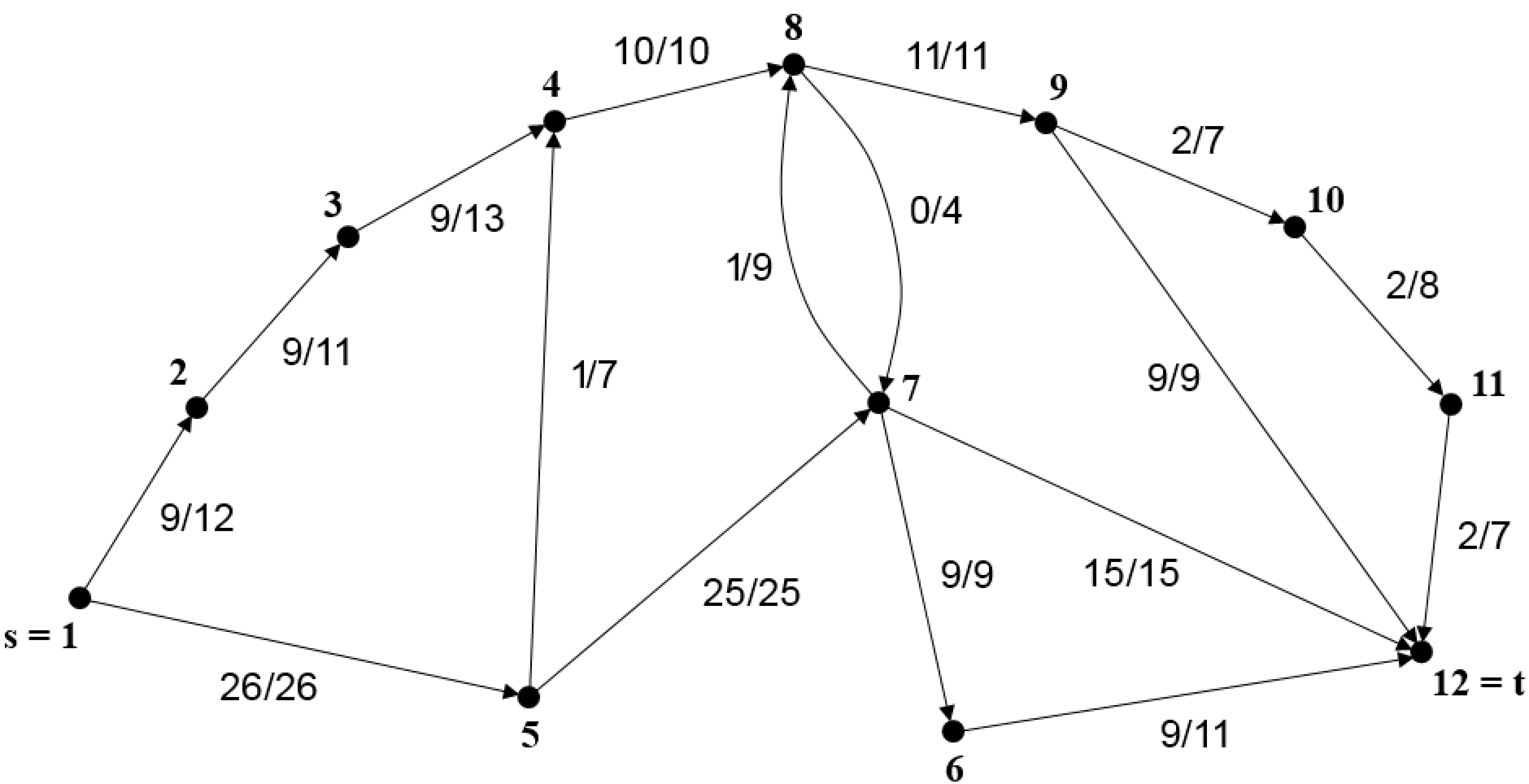

8
9/10
9/11
4
9
0/7
3
7/13
0/4
10
0/9
0/8
2
7/11
2/7
7
9/9
11
7/12
9/9
15/15
0/7
24/25
s = 1
12 = t
26/26
5
9/11
6

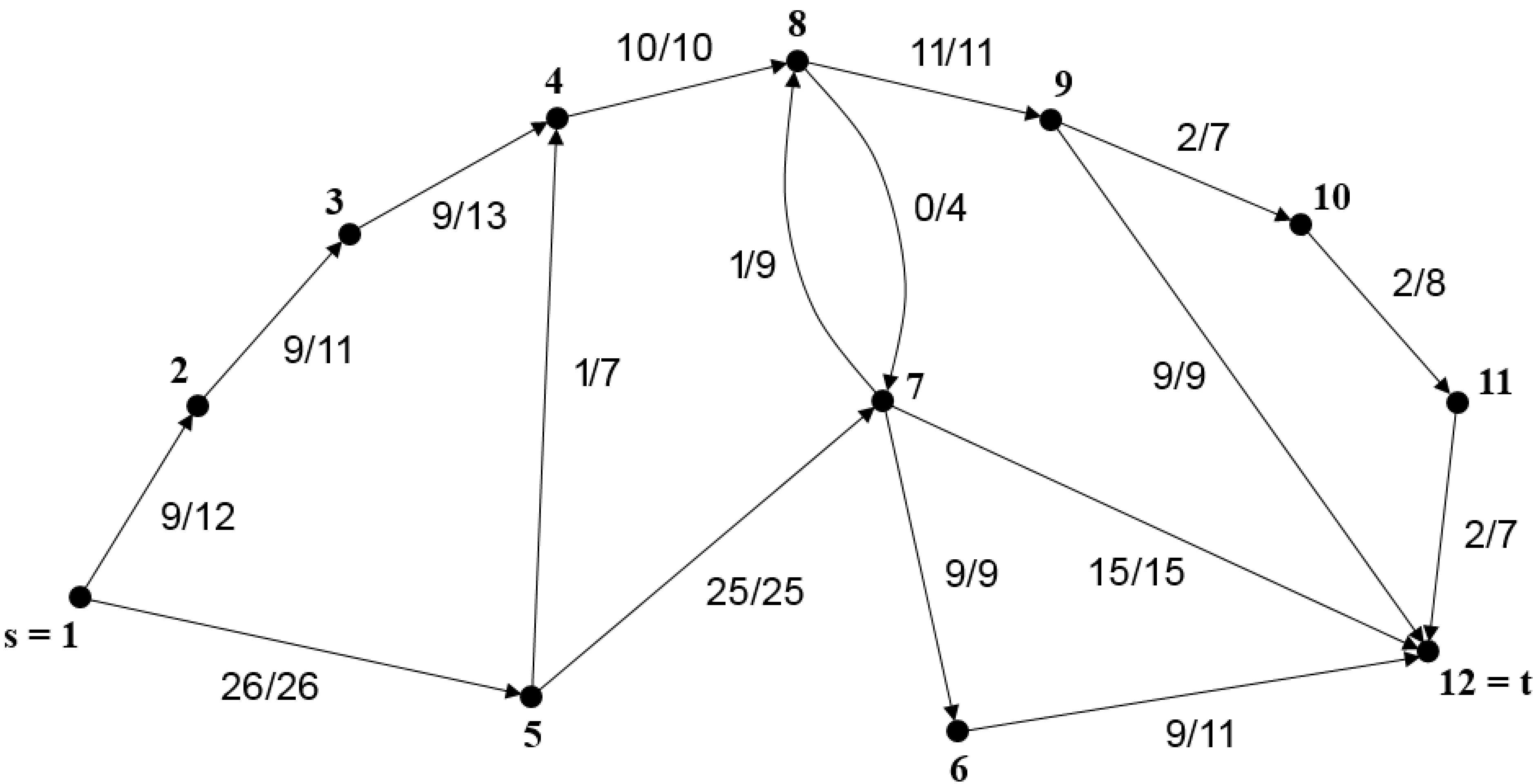

8
10/10
10/11
4
9
1/7
3
8/13
0/4
10
0/9
1/8
2
8/11
2/7
7
9/9
11
8/12
1/7
9/9
15/15
24/25
s = 1
12 = t
26/26
5
9/11
6


8
10/10
10/11
4
9
1/7
3
8/13
0/4
10
0/9
1/8
8/11
2
2/7
7
9/9
11
8/12
1/7
9/9
15/15
s = 1
24/25
12 = t
26/26
5
9/11
6


8
10/10
10/11
4
9
1/7
3
8/13
0/4
10
0/9
1/8
2
8/11
2/7
7
9/9
11
8/12
1/7
9/9
15/15
24/25
s = 1
26/26
12 = t
5
9/11
6


8
4
9
3
10
2
11
7
s = 1
5
6
12 = t
10/10
11/11
2/7
9/13
0/4
1/9
2/8
9/11
1/7
9/9
9/12
2/7
9/9
15/15
25/25
26/26
9/11


8
10/10
11/11
4
9
2/7
3
9/13
0/4
10
1/9
2/8
2
9/11
1/7
7
9/9
11
9/12
2/7
9/9
15/15
25/25
s = 1
12 = t
26/26
5
9/11
6



A maximális folyam értéke: **35**

Ford-Fulkerson algoritmus

A hálózati gráfot kétféleképpen is eltároljuk: szomszédsági mátrix és szomszédsági lista segítségével.

Ford-Fulkerson algoritmus

A hálózati gráfot kétféleképpen is eltároljuk: szomszédsági mátrix és szomszédsági lista segítségével.

Az `SZ_M[1..n,1..n]` szomszédsági mátrix elemei bejegyzés típusúak. Ha létezik az (u, v) él, akkor az `SZ_M[u,v].c` mező az illető él kapacitását, az `SZ_M[u,v].f` mező pedig az illető él folyamértékét tárolja. Ha az (u, v) él nem létezik, akkor az `SZ_M[u,v].c` értéke (-1).

Ford-Fulkerson algoritmus

A hálózati gráfot kétféleképpen is eltároljuk: szomszédsági mátrix és szomszédsági lista segítségével.

Az `SZ_M[1..n,1..n]` szomszédsági mátrix elemei bejegyzés típusúak. Ha létezik az (u, v) él, akkor az `SZ_M[u,v].c` mező az illető él kapacitását, az `SZ_M[u,v].f` mező pedig az illető él folyamértékét tárolja. Ha az (u, v) él nem létezik, akkor az `SZ_M[u,v].c` értéke (-1).

Az `SZ_L[1..n]` szomszédsági lista elemei szintén bejegyzés típusúak. Az `SZ_L[u].fokszám` mező az u csúcspont ki- és be-szomszédainak az összszámát tartalmazza, az `SZ_L[u].szomszédok[]` tömbmező pedig az u csúcspont ki- és be-szomszédait tárolja.

Ford-Fulkerson algoritmus

Algoritmus: A legrövidebb élszámú javító utat szélességi bejárással keressük meg (KERES_JAVÍTÓ_ÚT eljárás). A szélességi bejárás az irányítatlan szomszédsági listát használja, hogy az éleken mindkét irányban végig tudjon menni. Fontos megjegyezni, hogy olyan szélességi bejárásra van szükség, amelyik csak telítetlen éleken halad át. Így biztosítjuk be, hogy valóban javítóutat találjon.

Ford-Fulkerson algoritmus

Algoritmus: A legrövidebb élszámú javító utat szélességi bejárással keressük meg (KERES_JAVÍTÓ_ÚT eljárás). A szélességi bejárás az irányítatlan szomszédsági listát használja, hogy az éleken mindkét irányban végig tudjon menni. Fontos megjegyezni, hogy olyan szélességi bejárásra van szükség, amelyik csak telítetlen éleken halad át. Így biztosítjuk be, hogy valóban javítóutat találjon.

Ennek érdekében a KERES_JAVÍTÓ_ÚT eljárás a hálózati gráf szomszédsági mátrixát is használja. Az eljárás ebből a mátrixból nézi meg, hogy az aktuális u csúcspontnak a v csúcspont be- vagy ki-szomszédja.

Ford-Fulkerson algoritmus

Algoritmus: A legrövidebb élszámú javító utat szélességi bejárással keressük meg (KERES_JAVÍTÓ_ÚT eljárás). A szélességi bejárás az irányítatlan szomszédsági listát használja, hogy az éleken mindkét irányban végig tudjon menni. Fontos megjegyezni, hogy olyan szélességi bejárásra van szükség, amelyik csak telítetlen éleken halad át. Így biztosítjuk be, hogy valóban javítóutat találjon.

Ennek érdekében a KERES_JAVÍTÓ_ÚT eljárás a hálózati gráf szomszédsági mátrixát is használja. Az eljárás ebből a mátrixból nézi meg, hogy az aktuális u csúcspontnak a v csúcspont be- vagy ki-szomszédja.

Ha v ki-szomszéd (`SZ_M[u,v].c` $\neq -1$), akkor létezik az (u, v) él, amely előremutató élként kerül a keresett javítóútra. Ez esetben a telítetlenség feltétele: `SZ_M[u,v].f` $<$ `SZ_M[u,v].c` .

Ford-Fulkerson algoritmus

Ha v be-szomszéd ($\texttt{SZ\_M[v,u].c} \neq -1$), akkor a (v, u) él vissza-mutató élként kerül a keresett javítóútra. Ez esetben a telítetlenség feltétele: $\texttt{SZ\_M[v,u].f} > 0$.

Ford-Fulkerson algoritmus

Ha v be-szomszéd ($\texttt{SZ\_M[v,u].c} \neq -1$), akkor a (v, u) él vissza-mutató élként kerül a keresett javítóútra. Ez esetben a telítetlenség feltétele: $\texttt{SZ\_M[v,u].f} > 0$.

Ha u és v csúcspontok között (u, v) és (v, u) él egyaránt létezik, akkor az eljárás mindkettőt figyelembe veszi. Előbb az (u, v) élt ellenőrzi (mint előremutatót), és ha ez már telített, akkor megnézi a (v, u) élt is (mint visszamutatót). Fontos megjegyezni, hogy ha egy él telített előremutatóként, akkor visszamutatóként telítetlen (és fordítva). Kivételt képeznek a nulla kapacitású élek.

Ford-Fulkerson algoritmus

A KERES_JAVÍTÓ_ÚT eljárás a szélességi fát az `apa[1..n]` tömbben kódolja. Ha az eljárás talál $s \to t$ javítóutat, akkor fordított irányban ez a következő lesz:

`t, apa[t], apa[apa[t]], …, s`

Ford-Fulkerson algoritmus

A KERES_JAVÍTÓ_ÚT eljárás a szélességi fát az `apa[1..n]` tömbben kódolja. Ha az eljárás talál $s \to t$ javítóutat, akkor fordított irányban ez a következő lesz:

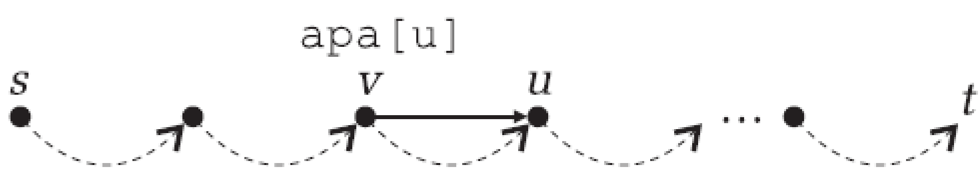
`t, apa[t], apa[apa[t]], ..., s`

Az `apa[u]` tömbelem attól függően pozitív vagy negatív előjelű, hogy az `apa[u]` csúcspont ki- vagy be-szomszédja az u csúcspontnak. Ha $v =$ `apa[u]` „pozitív apja” u-nak, akkor a (v, u) él szerepel a javítóúton (mint előremutató él).

Ford-Fulkerson algoritmus

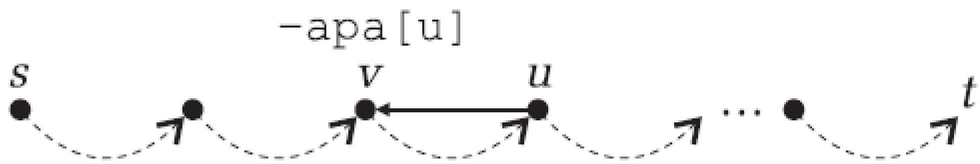
A KERES_JAVÍTÓ_ÚT eljárás a szélességi fát az `apa[1..n]` tömbben kódolja. Ha az eljárás talál $s \to t$ javítóutat, akkor fordított irányban ez a következő lesz:

Az `apa[u]` tömbelem attól függően pozitív vagy negatív előjelű, hogy az `apa[u]` csúcspont ki- vagy be-szomszédja az u csúcspontnak. Ha $v =$ `apa[u]` „pozitív apja” u-nak, akkor a (v, u) él szerepel a javítóúton (mint előremutató él).



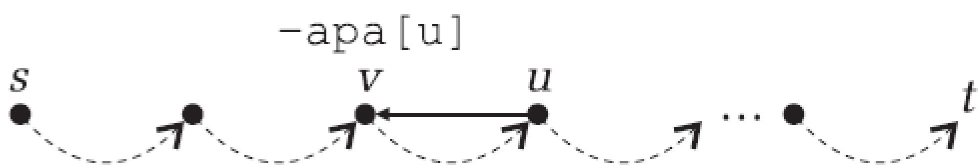
Ford-Fulkerson algoritmus

Ha $v =$ `apa[u]` „negatív apja” u-nak, akkor az (u, v) él szerepel a javítóúton (mint visszamutató él).



Ford-Fulkerson algoritmus

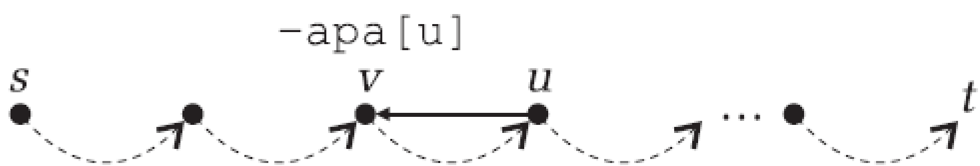
Ha $v =$ `apa[u]` „negatív apja” u-nak, akkor az (u, v) él szerepel a javítóúton (mint visszamutató él).



Ha a KERES_JAVÍTÓ_ÚT eljárás eljut a t csúcspontba, akkor meghívja a JAVÍTÁS eljárást. Ekkor az `apa[1..n]` tömbből kiolvasható a megtalált javítóút csúcspontjainak fordított sorrendje.

Ford-Fulkerson algoritmus

Ha $v = \texttt{apa[u]}$ „negatív apja” u-nak, akkor az (u, v) él szerepel a javítóúton (mint visszamutató él).



Ha a KERES_JAVÍTÓ_ÚT eljárás eljut a t csúcspontba, akkor meghívja a JAVÍTÁS eljárást. Ekkor az `apa[1..n]` tömbből kiolvasható a megtalált javítóút csúcspontjainak fordított sorrendje.

A JAVÍTÁS rekurzív eljárás, amely a t csúcspontból indul és apáról apára haladva meghatározza, hogy melyik az a legnagyobb érték, amellyel az adott javítóút mentén megnövelhető a folyamérték.

Ford-Fulkerson algoritmus

Minden u, v szomszédos csúcspontpárra $(v = \texttt{apa[u]})$

- ellenőrzi, hogy előremutató $(\texttt{apa[u]} > 0)$ vagy visszamutató él $(\texttt{apa[u]} < 0)$ van-e közöttük,

Ford-Fulkerson algoritmus

Minden u, v szomszédos csúcspontpárra ($v =$ `apa[u]`)

- ellenőrzi, hogy előremutató (`apa[u]` > 0) vagy visszamutató él (`apa[u]` < 0) van-e közöttük,
- kiszámítja mennyivel növelhető (`SZ_M[v,u].c - SZ_M[u,v].f`) illetve csökkenthető (`SZ_M[u,v].f`) az illető élen a folyamérték.

Ford-Fulkerson algoritmus

Minden u, v szomszédos csúcspontpárra ($v =$ `apa[u]`)

- ellenőrzi, hogy előremutató (`apa[u]` > 0) vagy visszamutató él (`apa[u]` < 0) van-e közöttük,
- kiszámítja mennyivel növelhető (`SZ_M[v,u].c - SZ_M[u,v].f`) illetve csökkenthető (`SZ_M[u,v].f`) az illető élen a folyamérték.

Ezen értékek közül a legkisebb lesz a keresett javítóérték. Az eljárás a minimum kereséséhez a **min** *cím szerint átadott* paramétert használja.

Ford-Fulkerson algoritmus

Minden u, v szomszédos csúcspontpárra ($v =$ `apa[u]`)

- ellenőrzi, hogy előremutató (`apa[u]` > 0) vagy visszamutató él (`apa[u]` < 0) van-e közöttük,
- kiszámítja mennyivel növelhető (`SZ_M[v,u].c - SZ_M[u,v].f`) illetve csökkenthető (`SZ_M[u,v].f`) az illető élen a folyamérték.

Ezen értékek közül a legkisebb lesz a keresett javítóérték. Az eljárás a minimum kereséséhez a **min** *cím szerint átadott* paramétert használja.

A JAVÍTÁS eljárás a rekurzió visszaútján végzi el a korrekciókat:

- minden előremutató élen növeli a folyamértéket **min**-nel,
- minden visszamutató élen csökkenti a folyamértéket **min**-nel.

Ford-Fulkerson algoritmus

Ha a KERES_JAVÍTÓ_ÚT eljárás nem jut el a *t* csúcspontba, akkor a **min** paraméterben nullát ad vissza a FORD_FULKERSON eljárásnak. Ez ennek alapján felismeri, hogy nincs több javítóút, a hálózati gráfban. Az **fe** változó a maximális folyam értékét fogja tartalmazni.

```
eljárás JAVÍTÁS(Sz_M[1..n,1..n],u,apa[1..n],min)
   ha apa[u] < 0 akkor
      v ← -apa[u]
      ha min > Sz_M[u,v].f akkor
         min ← Sz_M[u,v].f
      vége ha
      JAVÍTÁS(Sz_M,v,apa,min)
      Sz_M[u,v].f ← Sz_M[u,v].f - min
   különben
      ha apa[u] > 0 akkor
         v ← apa[u]
         ha min > - Sz_M[v,u].f akkor
            min ← Sz_M[v,u].c - Sz_M[v,u].f
         vége ha
         JAVÍTÁS(Sz_M,v,apa,min)
         Sz_M[v,u].f ← Sz_M[v,u].f + min
      vége ha
   vége ha
vége JAVÍTÁS
```

```
eljárás KERES_JAVÍTÓ_ÚT(Sz_M[1..n,1..n],Sz_L[1..n],s,t,min)
   minden u ∈ V(G)\{s} végezd
      szín[u] ← FEHÉR
      apa[u] ← 0
   vége minden
   szín[s] ← SZÜRKE
   apa[s] ← 0
   Q ← {s}
   amíg Q ≠ ∅ végezd
      u ← MÁSOL_SORELSŐ(Q)
      minden i ← 1, Sz_L[u].fokszám végezd
         v ← Sz_L[u].[i]
         ha szín[v] = FEHÉR akkor
            ha Sz_M[u,v].c ≠ -1 ÉS
                          Sz_M[u,v].f < Sz_M[u,v].c akkor
               szín[v] ← SZÜRKE
               apa[v] ← u
               ha v = t akkor
                  JAVÍTÁS(Sz_M,t,apa,min)
                  vissza
               vége ha
               BETESZ_SORVÉGÉRE(Q,v)
```

```
            különben
               ha Sz_M[v,u].c ≠ -1 ÉS Sz_M[v,u].f > 0 akkor
                  szín[v] ← SZÜRKE
                  apa[v] ← -u
                  ha v = t akkor
                     JAVÍTÁS(Sz_M,t,apa,min)
                     vissza
                  vége ha
                  BETESZ_SORVÉGÉRE(Q,v)
               vége ha
            vége ha
         vége ha
      vége minden
      TÖRÖL_SORELEJÉRŐL(Q)
      szín[u] ← FEKETE
   vége amíg
   min ← 0
vége KERES_JAVÍTÓ_ÚT
```

```
függvény FORD_FULKERSON(Sz_M[1..n,1..n],Sz_L[1..n],s,t)
   fe ← 0
   végezd
      min ← ∞
      KERES_JAVÍTÓ_ÚT(Sz_M,Sz_L,s,t,min)
      fe ← fe + min
   amíg min ≠ 0
   vissza fe
vége FORD_FULKERSON
```

```
függvény FORD_FULKERSON(Sz_M[1..n,1..n],Sz_L[1..n],s,t)
   fe ← 0
   végezd
      min ← ∞
      KERES_JAVÍTÓ_ÚT(Sz_M,Sz_L,s,t,min)
      fe ← fe + min
   amíg min ≠ 0
   vissza fe
vége FORD_FULKERSON
```

A FORD_FULKERSON algoritmus bonyolultsága $O(nm^2)$.

Ford-Fulkerson algoritmus reziduális hálózatokkal

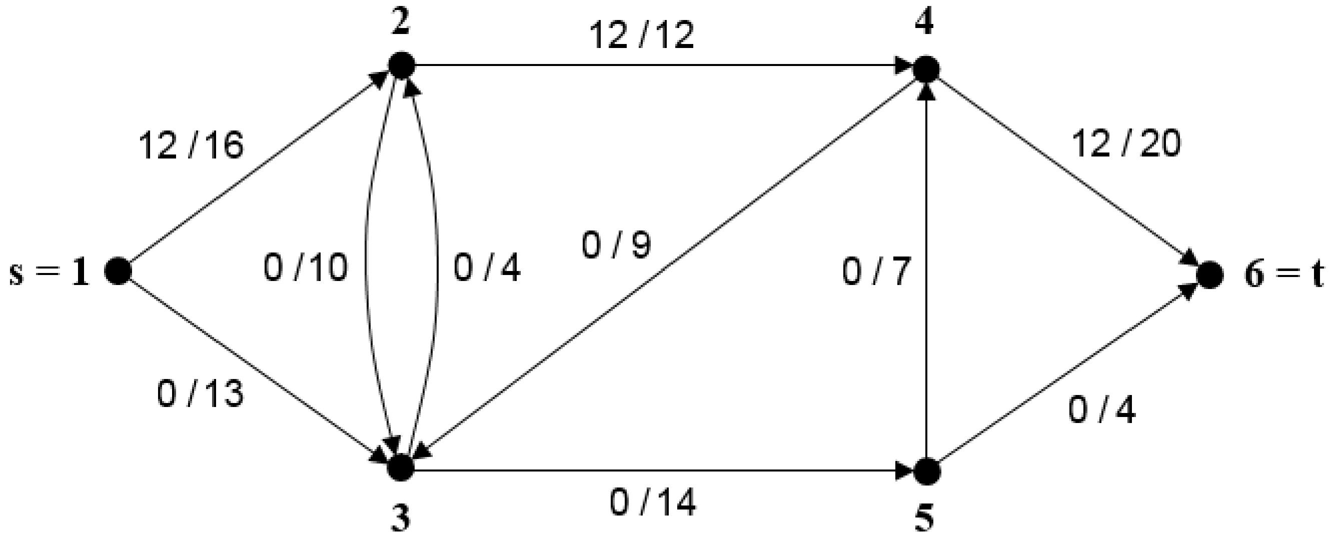
Legyen $(\mathrm{G}(\mathrm{V}, \mathrm{E}), s, t, c)$ egy hálózati gráf, f pedig egy folyam ebben a hálózatban. A hálózati gráf **reziduális hálózat**át úgy kapjuk meg, hogy az E élhalmazból elhagyjuk a 0 kapacitású éleket, az összes többi (u, v) élen pedig meghatározzuk az ún. *reziduális kapacitásokat* a következőképpen:

$$c_f(u, v) = c(u, v) - f(u, v)$$

Ford-Fulkerson algoritmus reziduális hálózatokkal

Legyen $(G(V, E), s, t, c)$ egy hálózati gráf, f pedig egy folyam ebben a hálózatban. A hálózati gráf **reziduális hálózat**át úgy kapjuk meg, hogy az E élhalmazból elhagyjuk a 0 kapacitású éleket, az összes többi (u, v) élen pedig meghatározzuk az ún. *reziduális kapacitások*at a következőképpen:

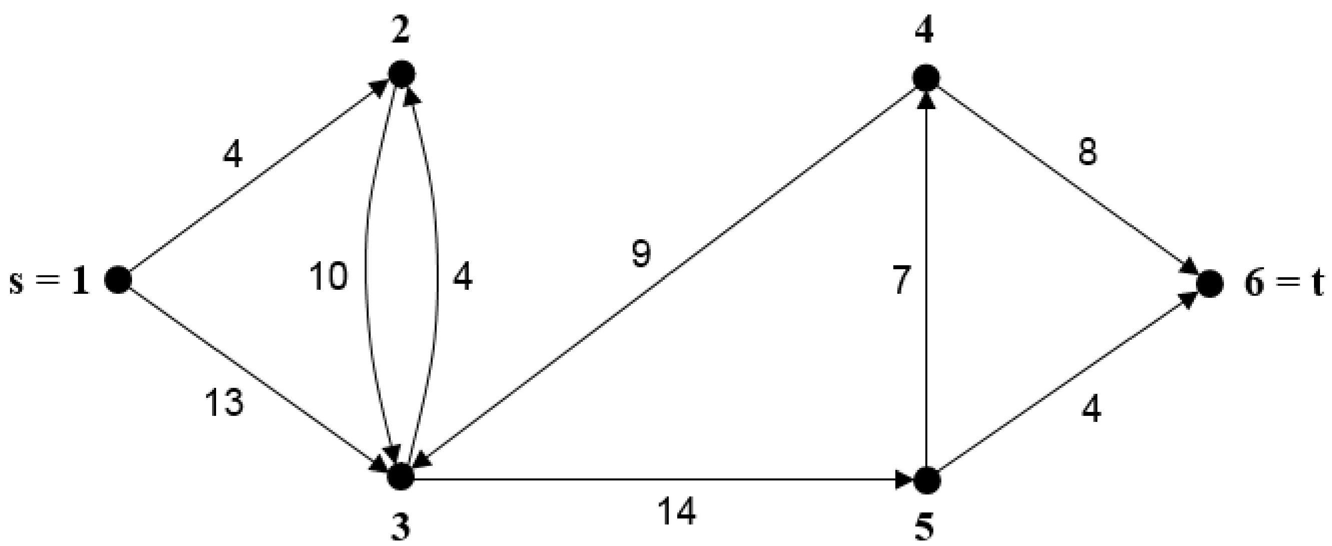
$$c_f(u, v) = c(u, v) - f(u, v)$$



Ford-Fulkerson algoritmus reziduális hálózatokkal

Legyen $(G(V, E), s, t, c)$ egy hálózati gráf, f pedig egy folyam ebben a hálózatban. A hálózati gráf **reziduális hálózat**át úgy kapjuk meg, hogy az E élhalmazból elhagyjuk a 0 kapacitású éleket, az összes többi (u, v) élen pedig meghatározzuk az ún. *reziduális kapacitásokat* a következőképpen:

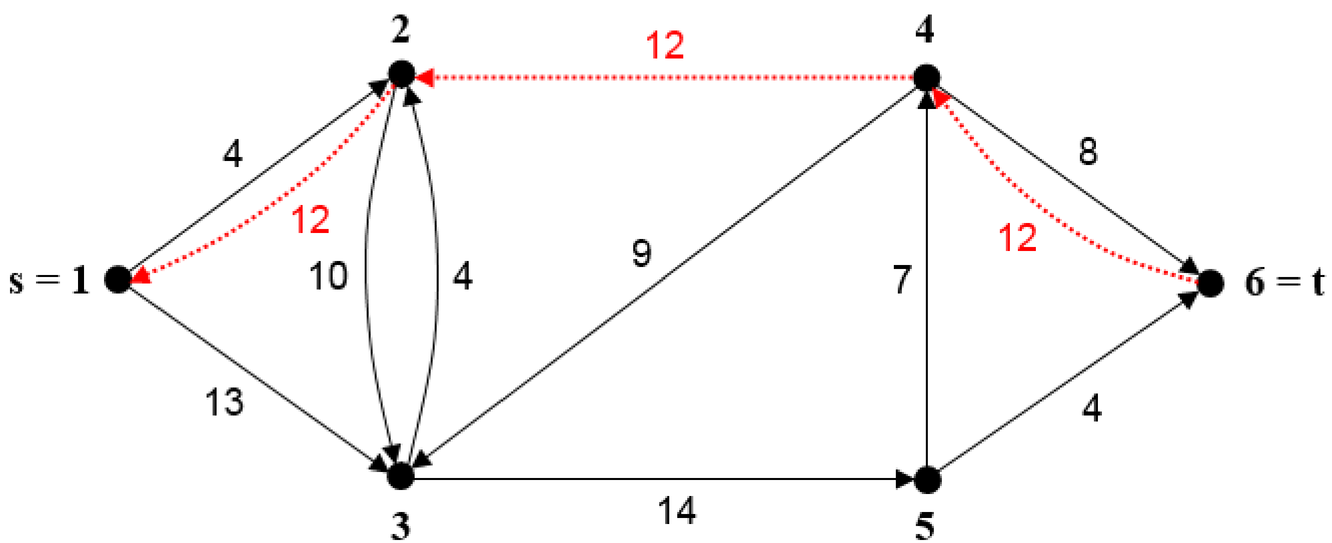
$$c_f(u, v) = c(u, v) - f(u, v)$$



Ford-Fulkerson algoritmus reziduális hálózatokkal

Legyen $(G(V, E), s, t, c)$ egy hálózati gráf, f pedig egy folyam ebben a hálózatban. A hálózati gráf **reziduális hálózat**át úgy kapjuk meg, hogy az E élhalmazból elhagyjuk a 0 kapacitású éleket, az összes többi (u, v) élen pedig meghatározzuk az ún. *reziduális kapacitásoka*t a következőképpen:

$$c_f(u, v) = c(u, v) - f(u, v)$$



Ford-Fulkerson algoritmus reziduális hálózatokkal

Stratégia: Az algoritmus az alábbi gondolatmenetet követi:

Kezdetben minden élen a folyamérték nulla.

WHILE a reziduális hálózatban létezik $s \to t$ javítóút **DO**

1) keress egy javítóutat

2) határozd meg azt a maximumot, amellyel az illető javító út mentén növelhető a folyamérték

3) végezd el a hálózatban a javítóút mentén a javítás által feltételezett folyamérték korrekciókat

Edmonds-Karp algoritmus reziduális hálózatokkal

Stratégia: Az algoritmus az alábbi gondolatmenetet követi:

Kezdetben minden élen a folyamérték nulla.

WHILE a reziduális hálózatban létezik $s \to t$ javítóút **DO**

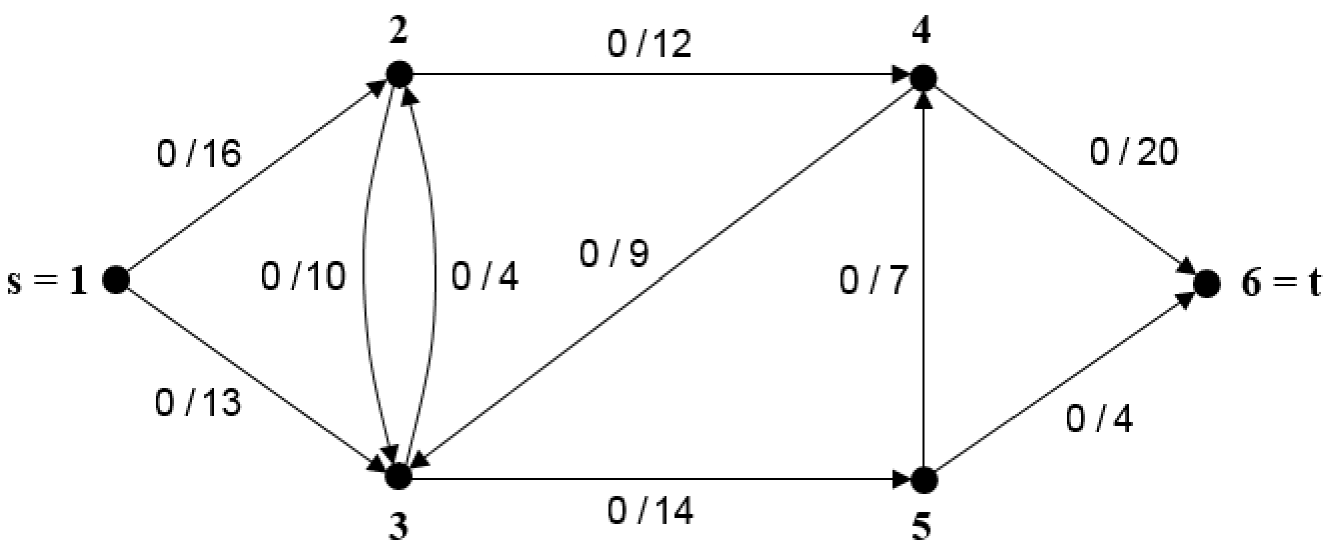
1) BFS kereséssel keresd meg a legrövidebb $s \to t$ javítóutat

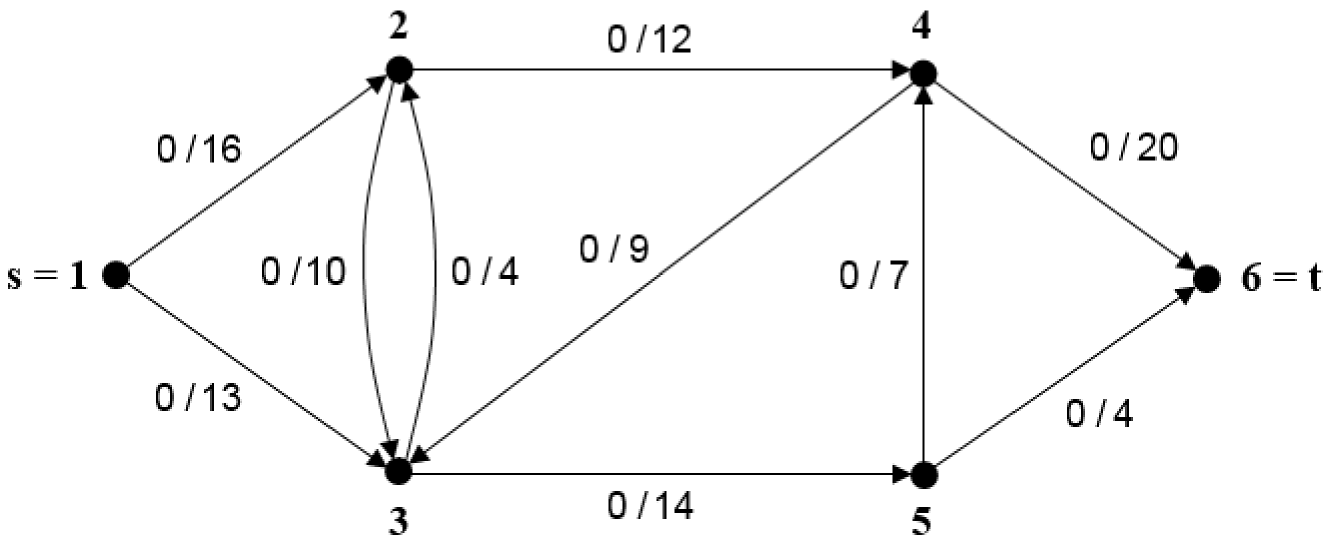
2) határozd meg azt a maximumot, amellyel az illető javító út mentén növelhető a folyamérték

3) végezd el a hálózatban a javítóút mentén a javítás által feltételezett folyamérték korrekciókat

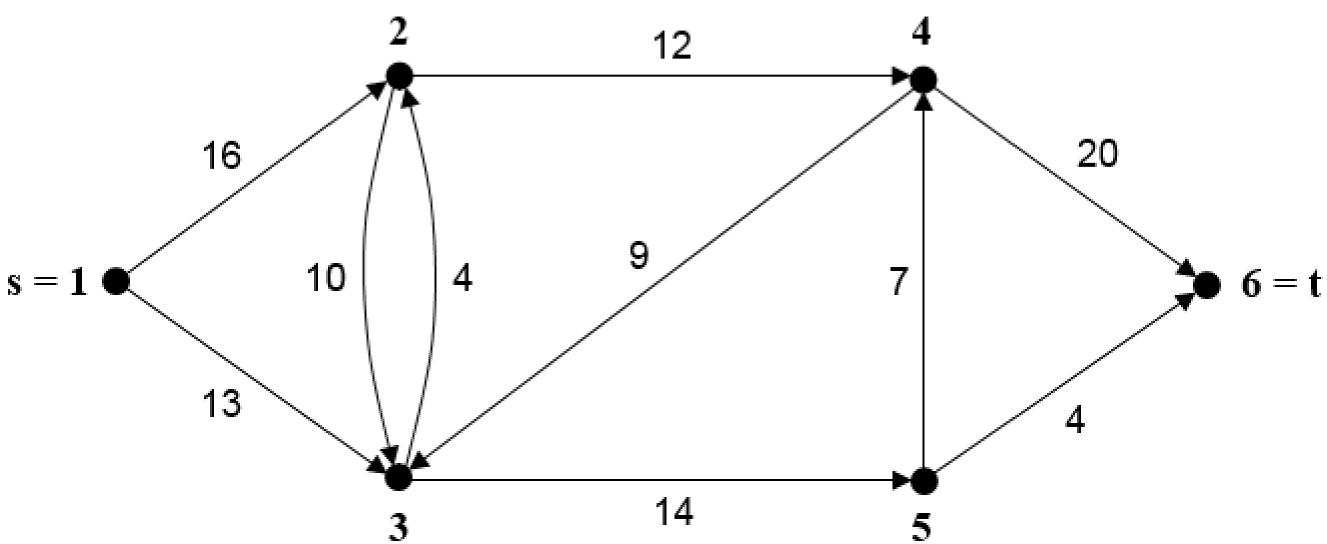
Jack Edmonds
1934 –

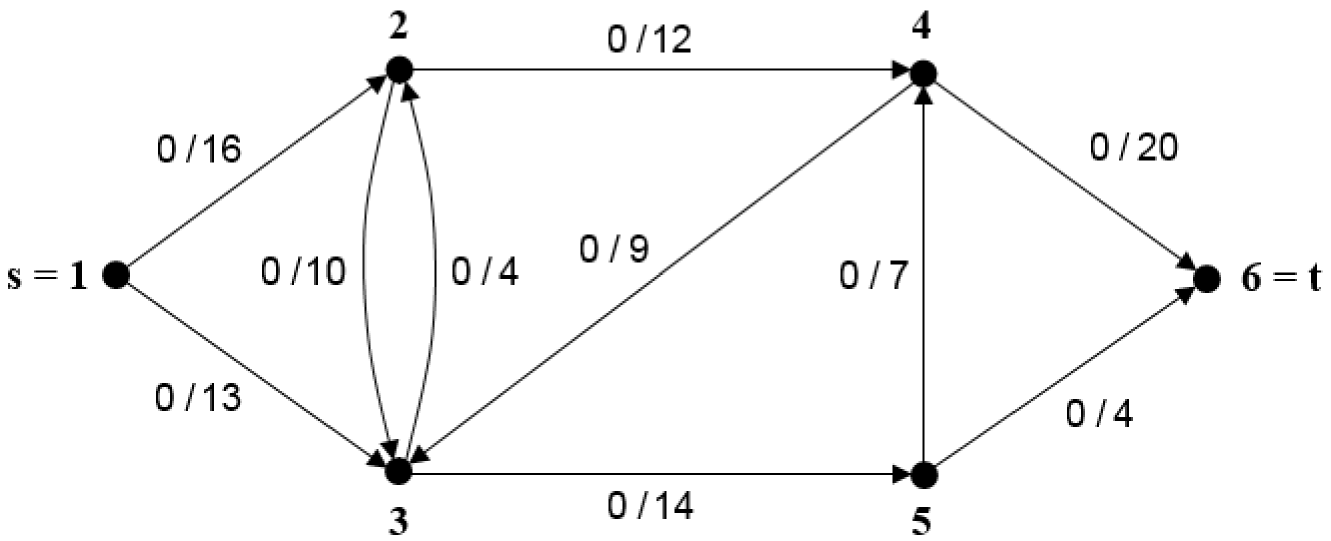
Richard Manning Karp
1935 –


2
4
0 / 12
0 / 16
0 / 20
s = 1
0 / 10
0 / 4
0 / 9
0 / 7
6 = t
0 / 13
0 / 4
0 / 14
3
5

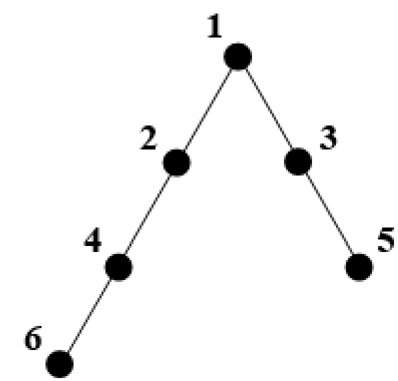


reziduális hálózat:

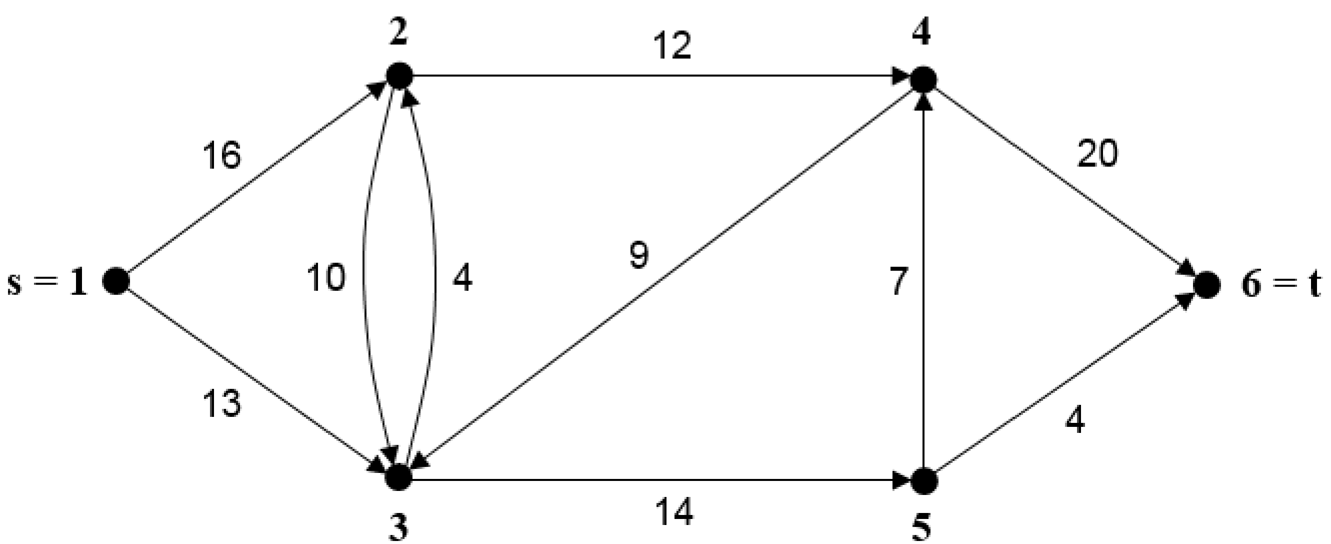


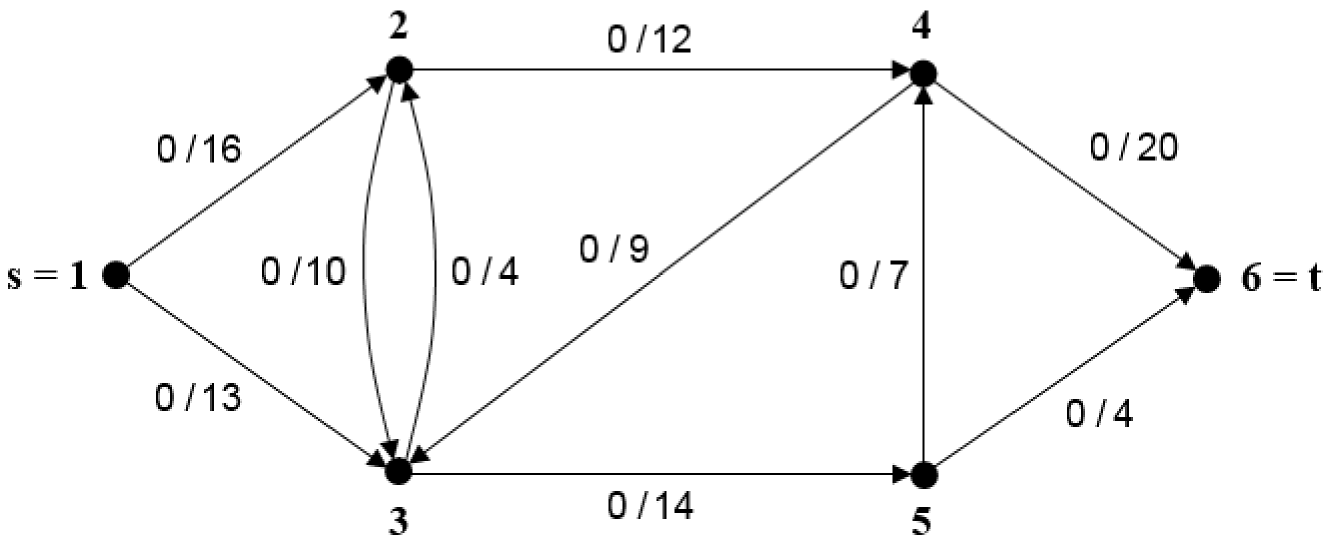


széleségi fa:

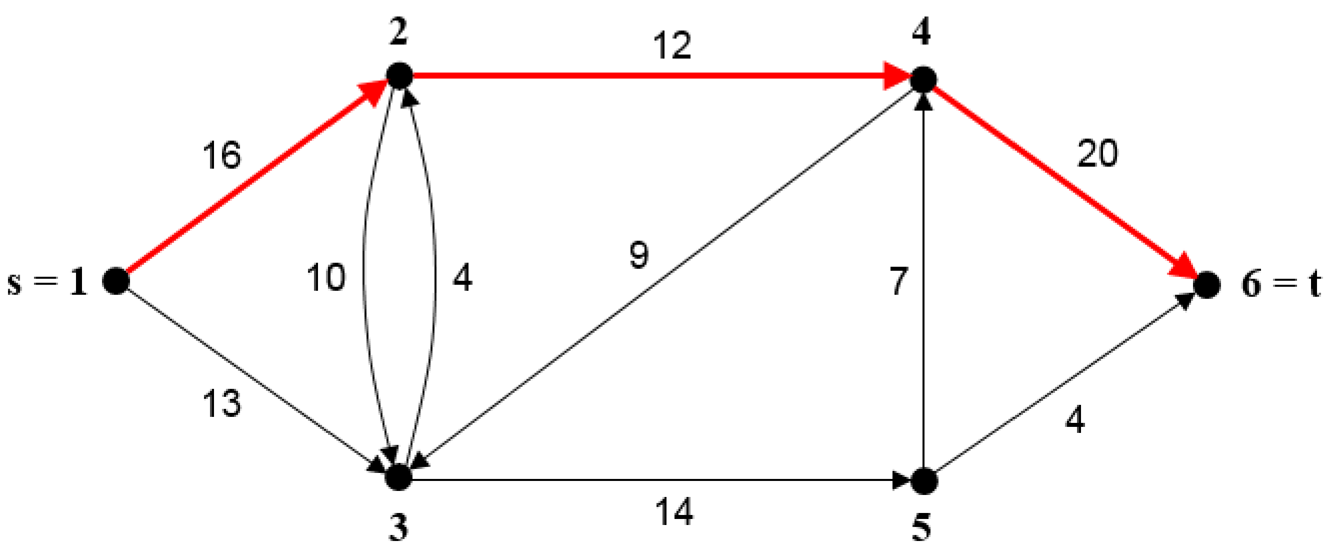


reziduális hálózat:

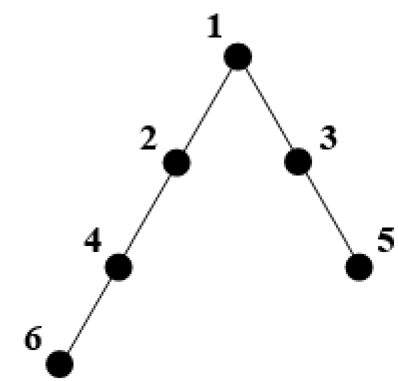


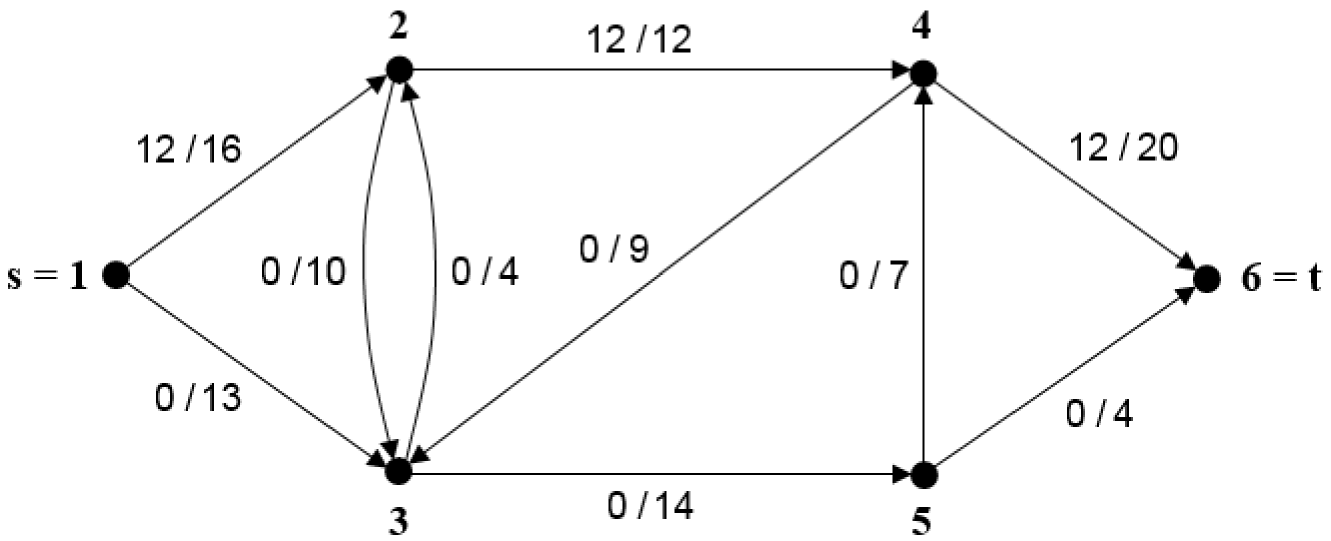


reziduális hálózat:

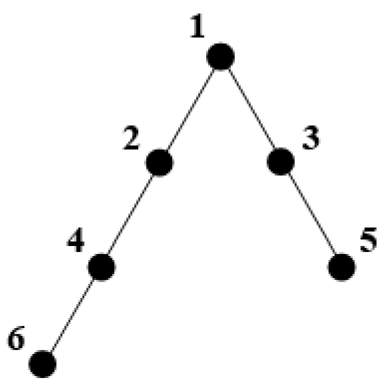


szélességi fa:

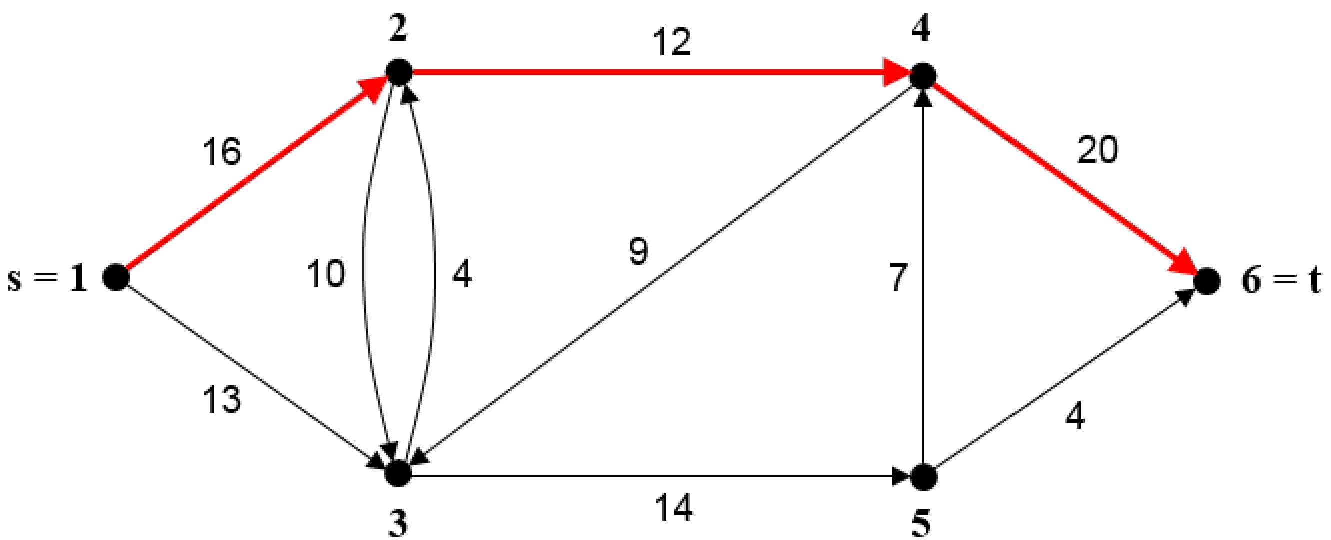


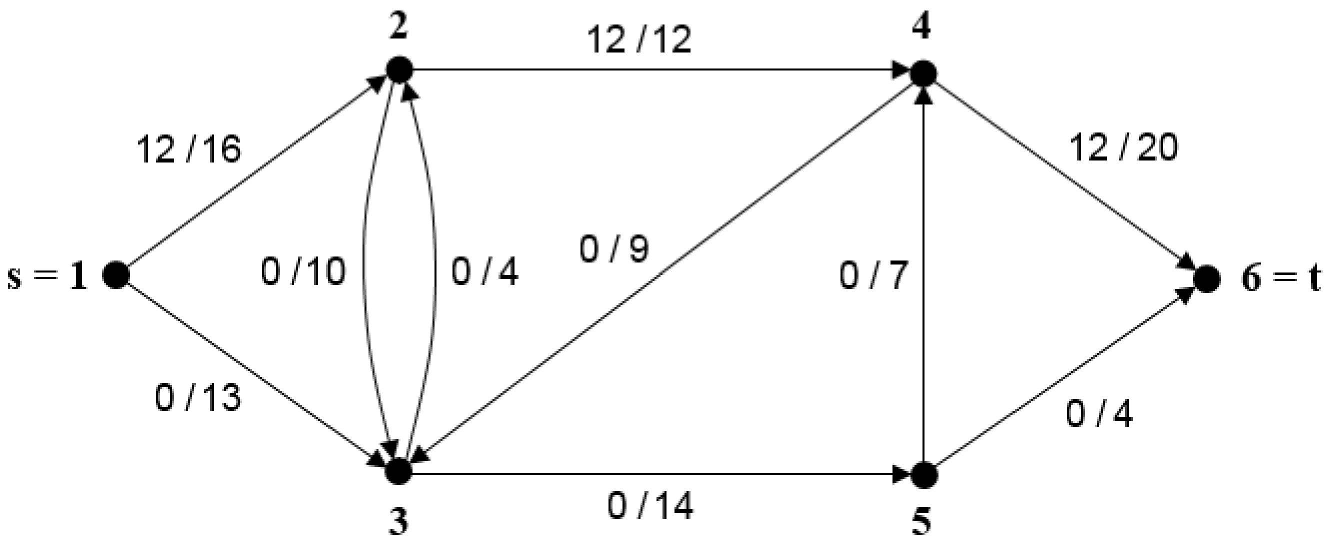


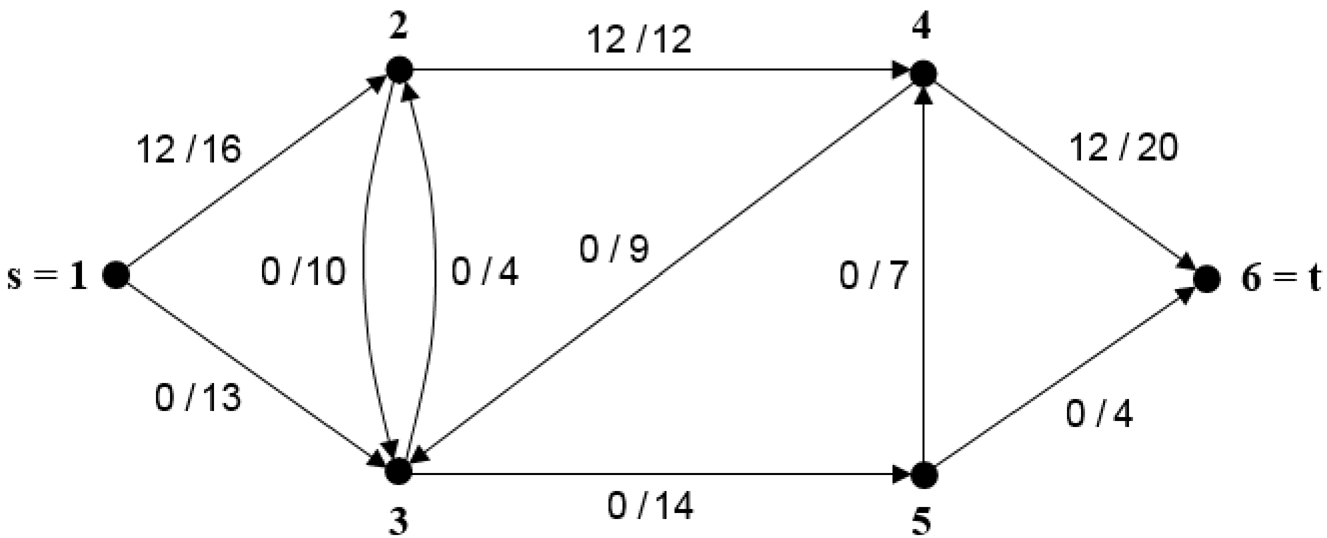
széleségi fa:



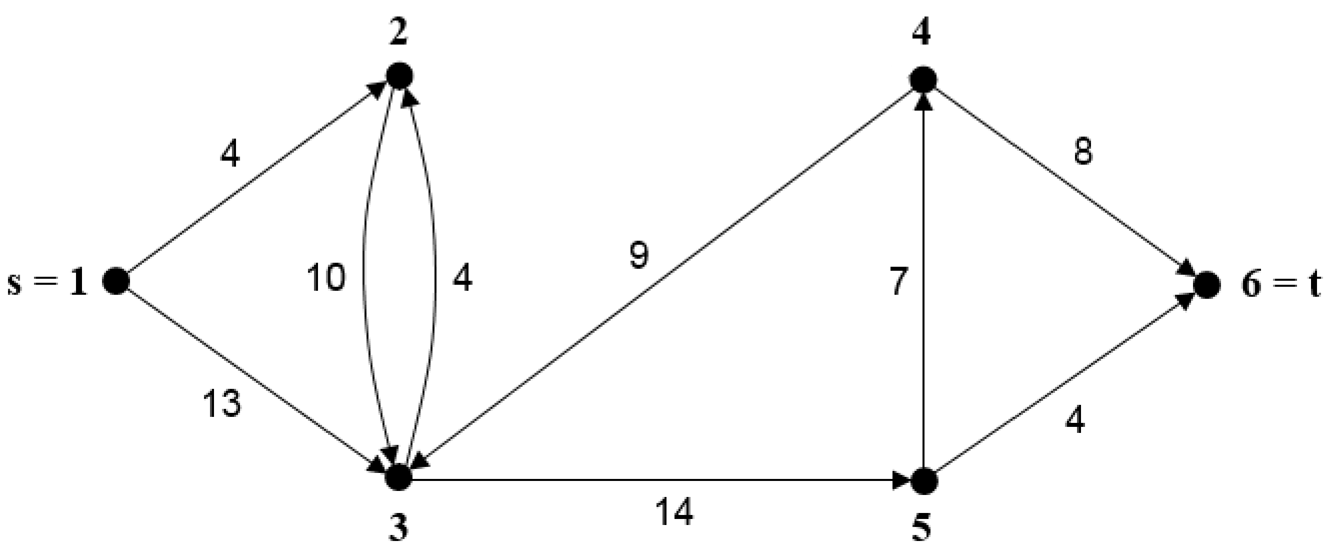
reziduális hálózat:

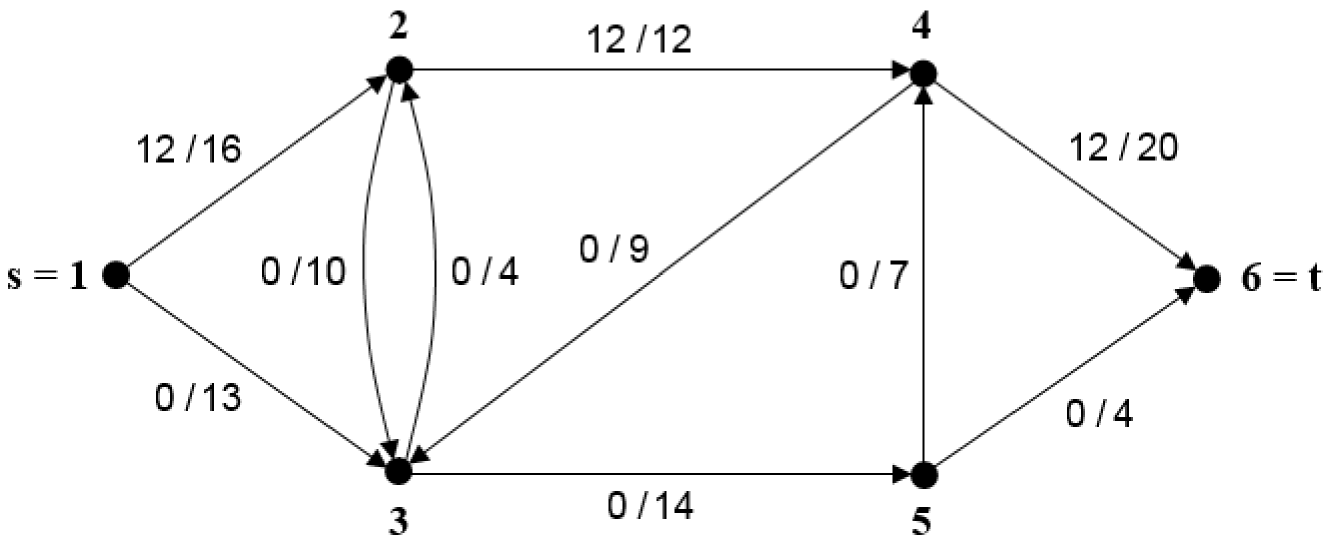



2
12 / 12
4
12 / 16
12 / 20
s = 1
0 / 10
0 / 4
0 / 9
0 / 7
6 = t
0 / 13
0 / 4
3
0 / 14
5

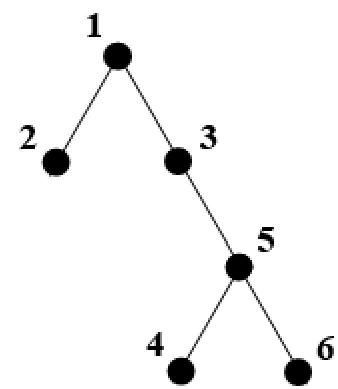


reziduális hálózat:

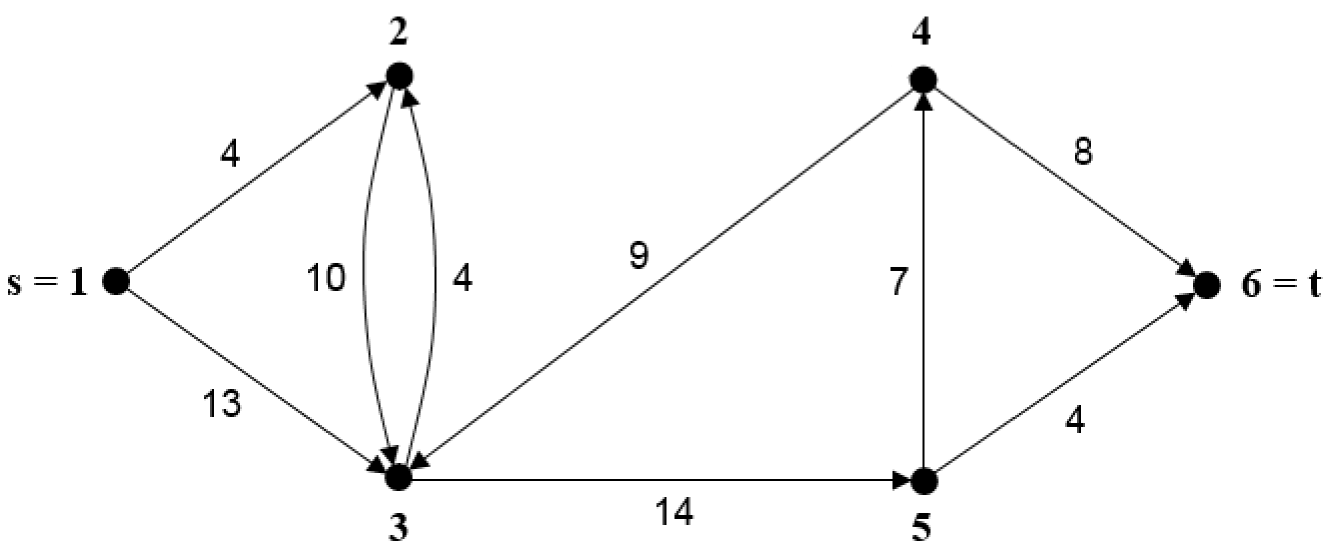


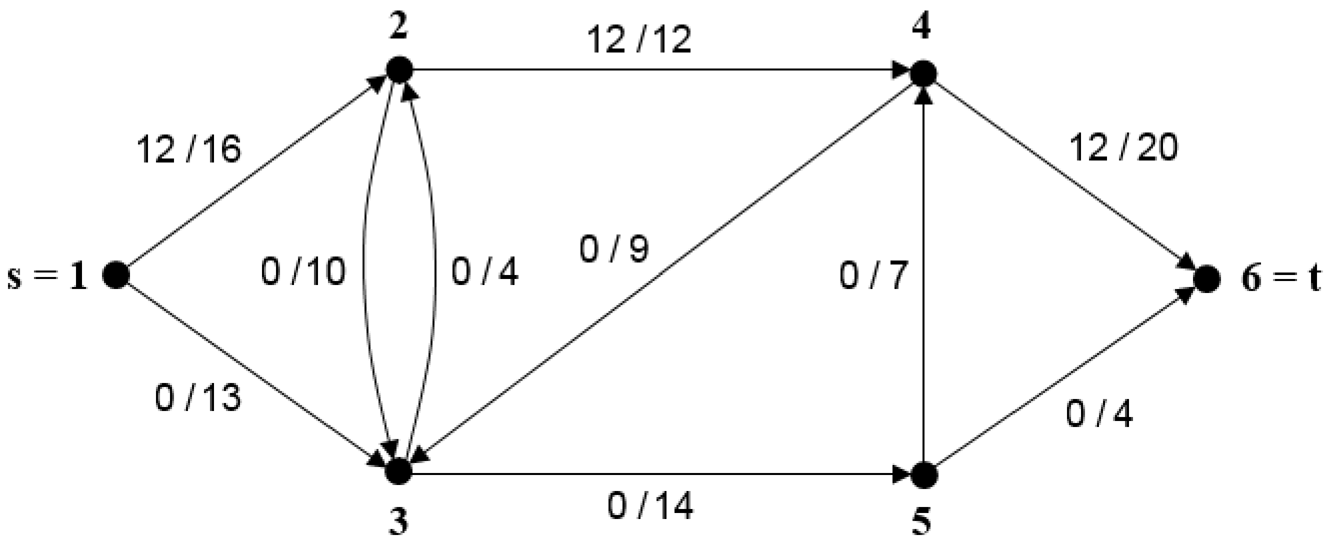


szélességi fa:

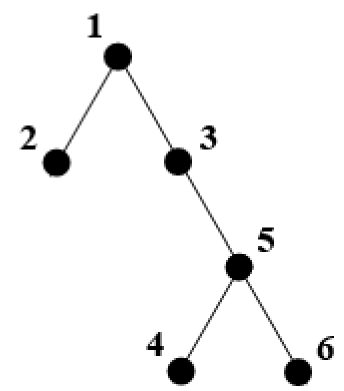


reziduális hálózat:

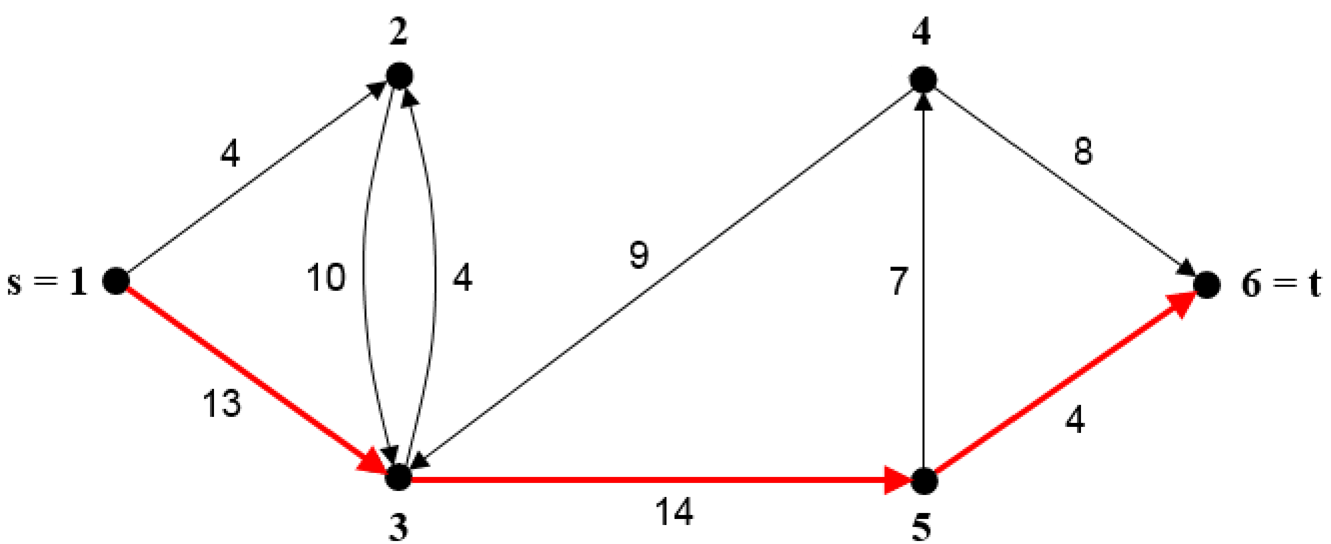


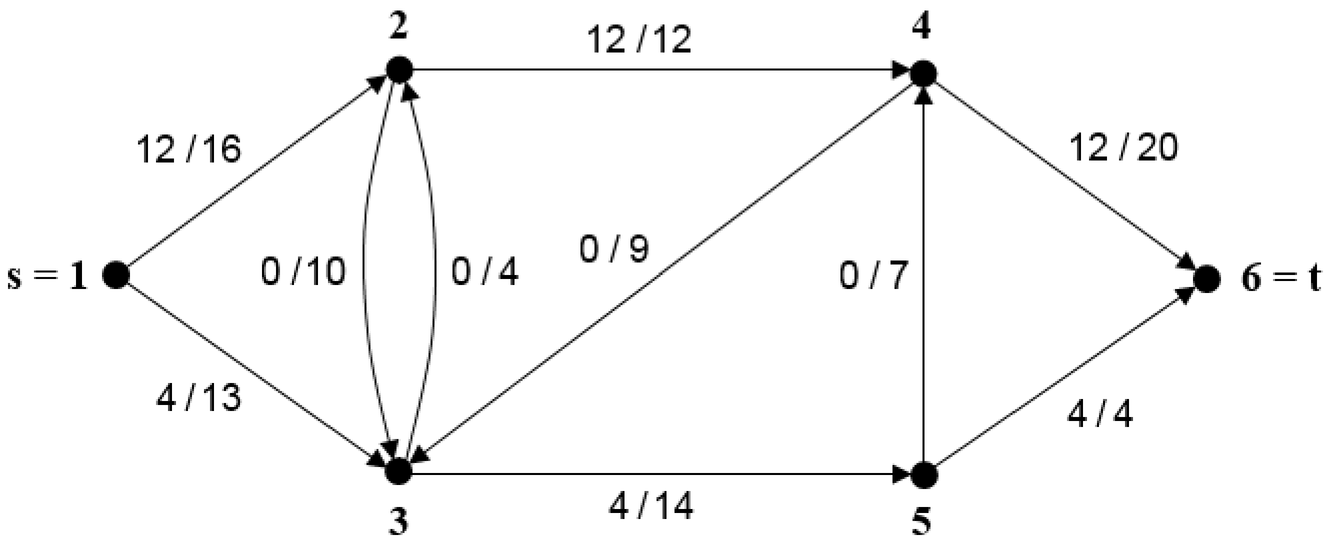


szélességi fa:

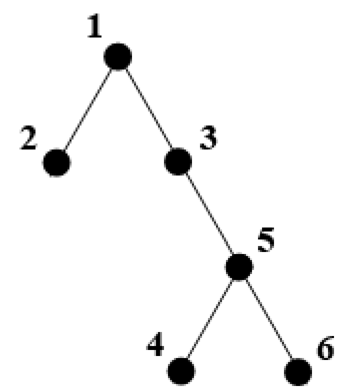


reziduális hálózat:

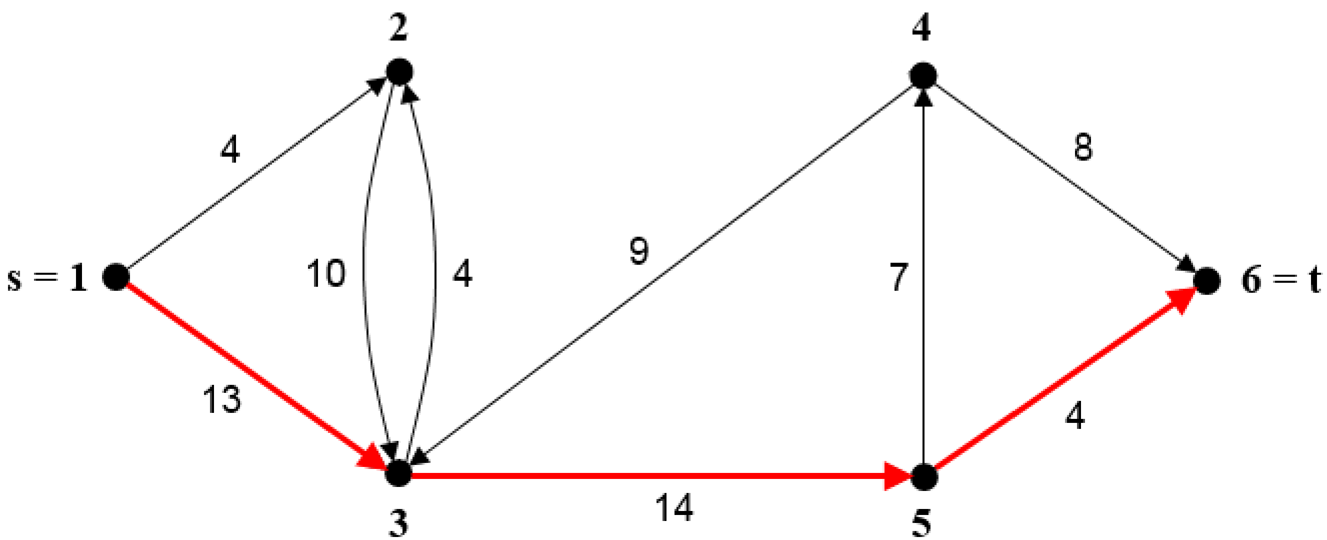


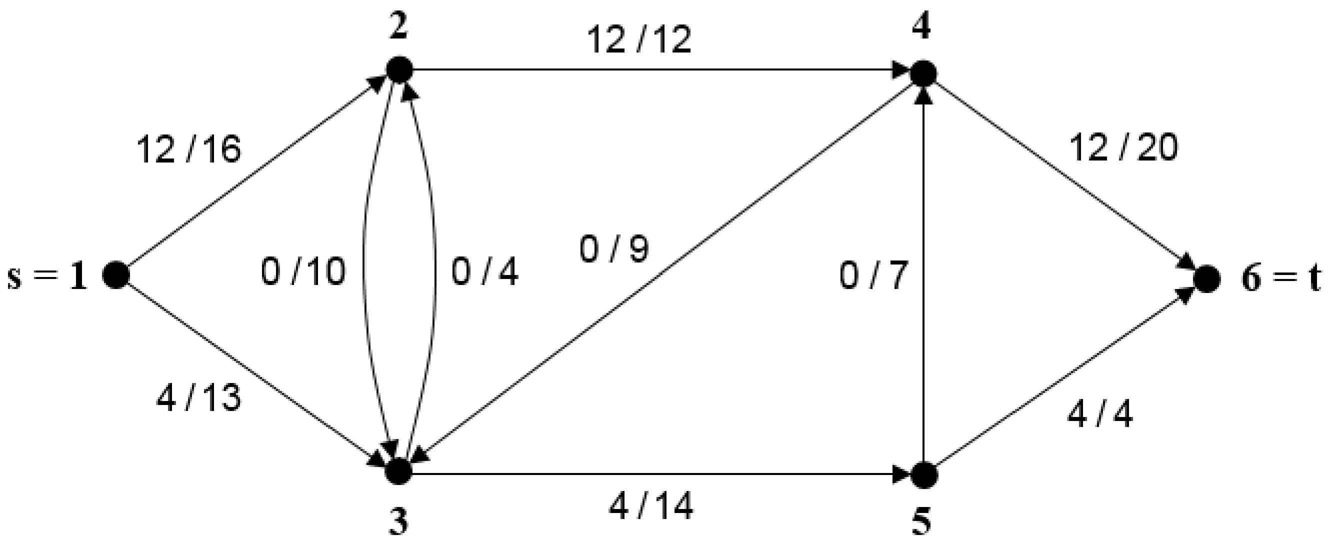


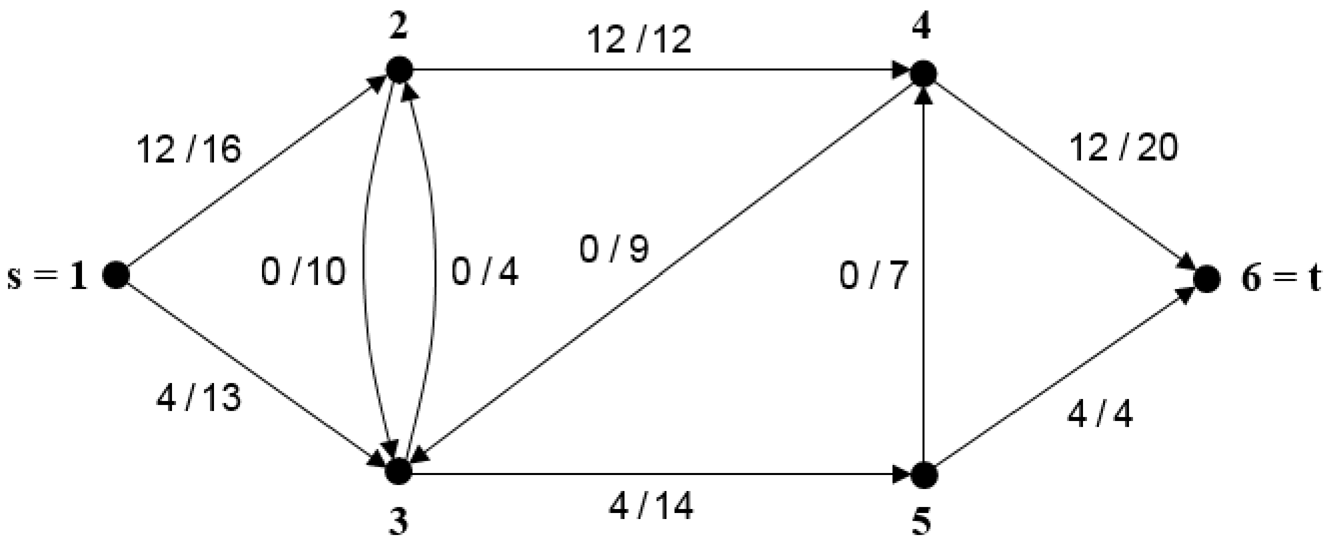
szélességi fa:



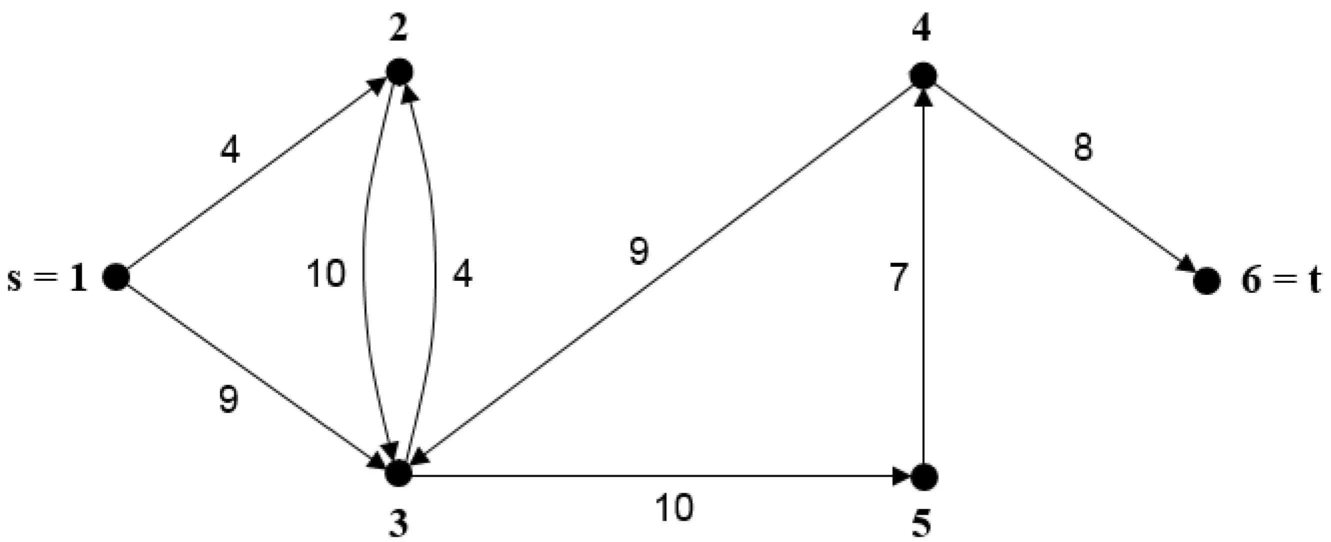
reziduális hálózat:

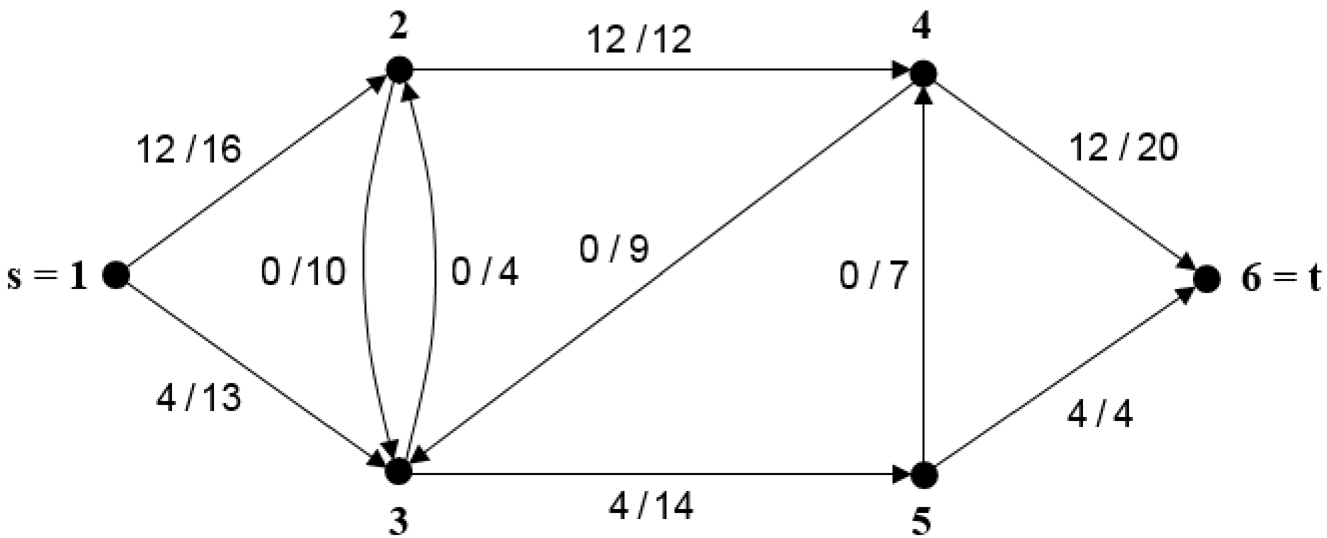



2
12 / 12
4
12 / 16
12 / 20
s = 1
0 / 10
0 / 4
0 / 9
0 / 7
6 = t
4 / 13
4 / 4
4 / 14
3
5

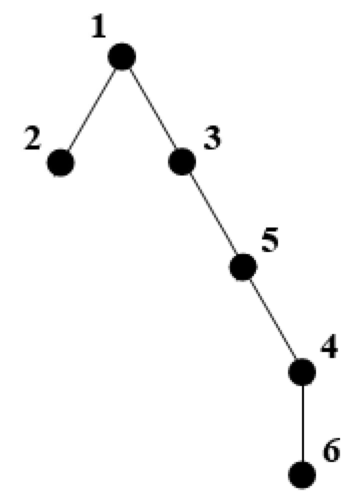


reziduális hálózat:

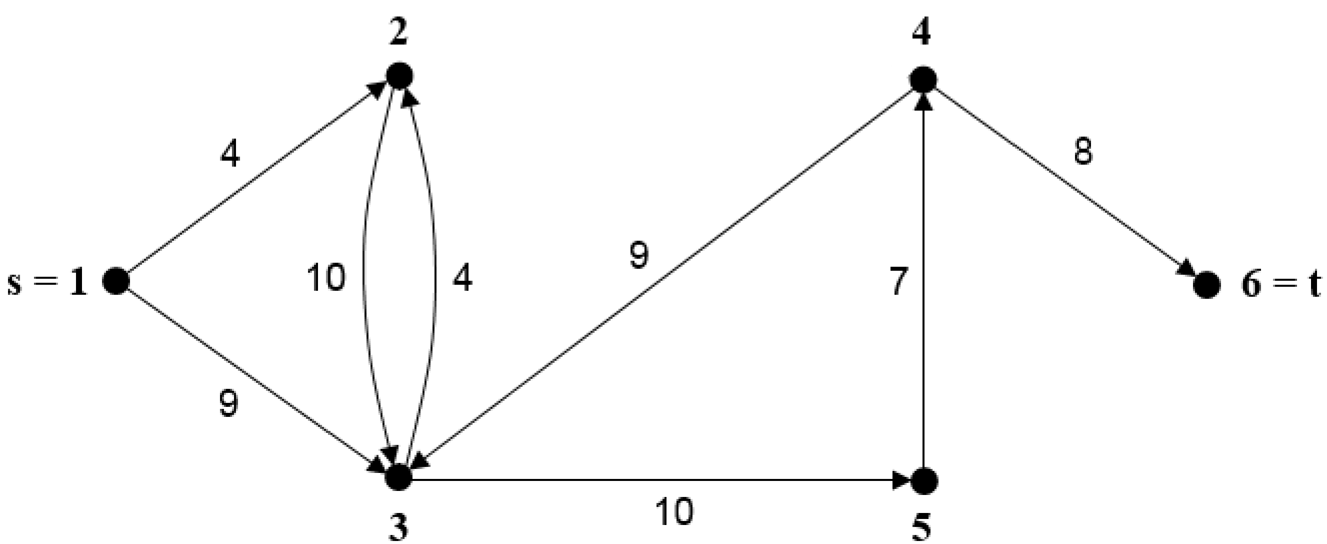


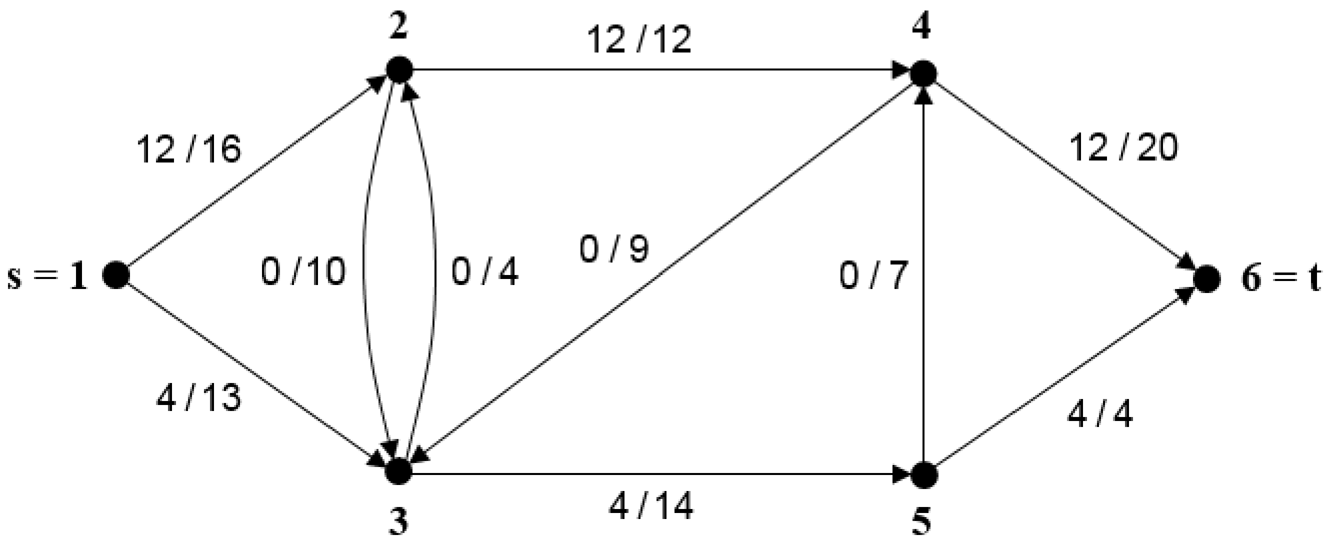


szélességi fa:

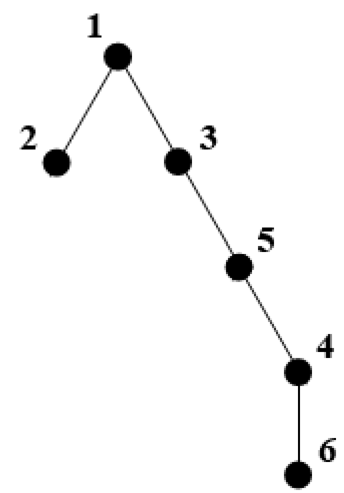


reziduális hálózat:

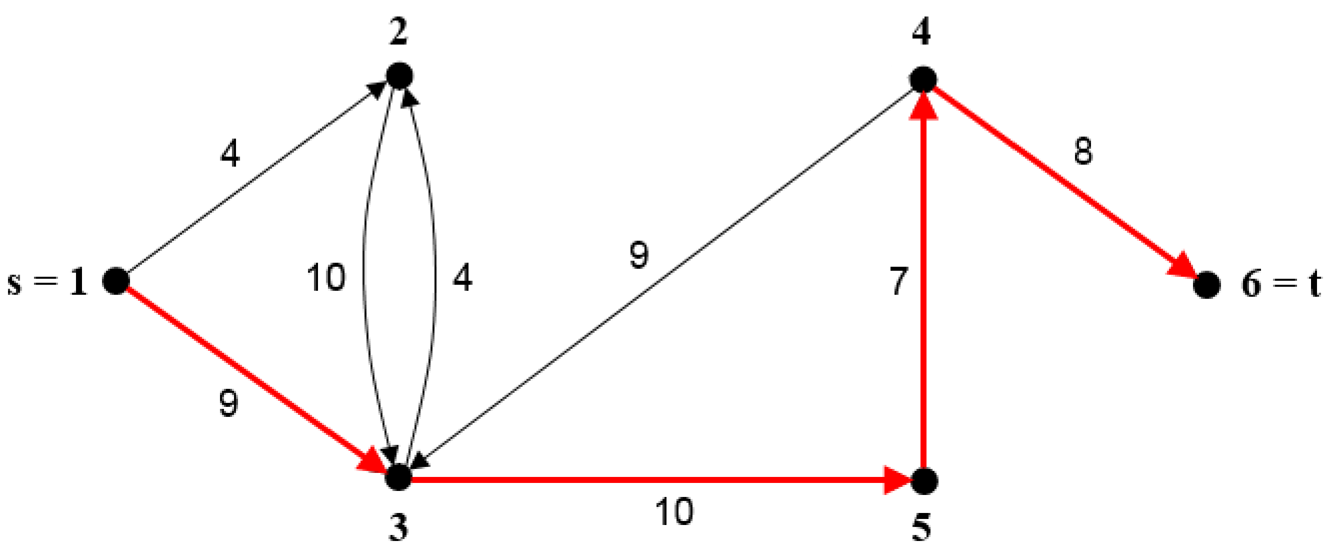


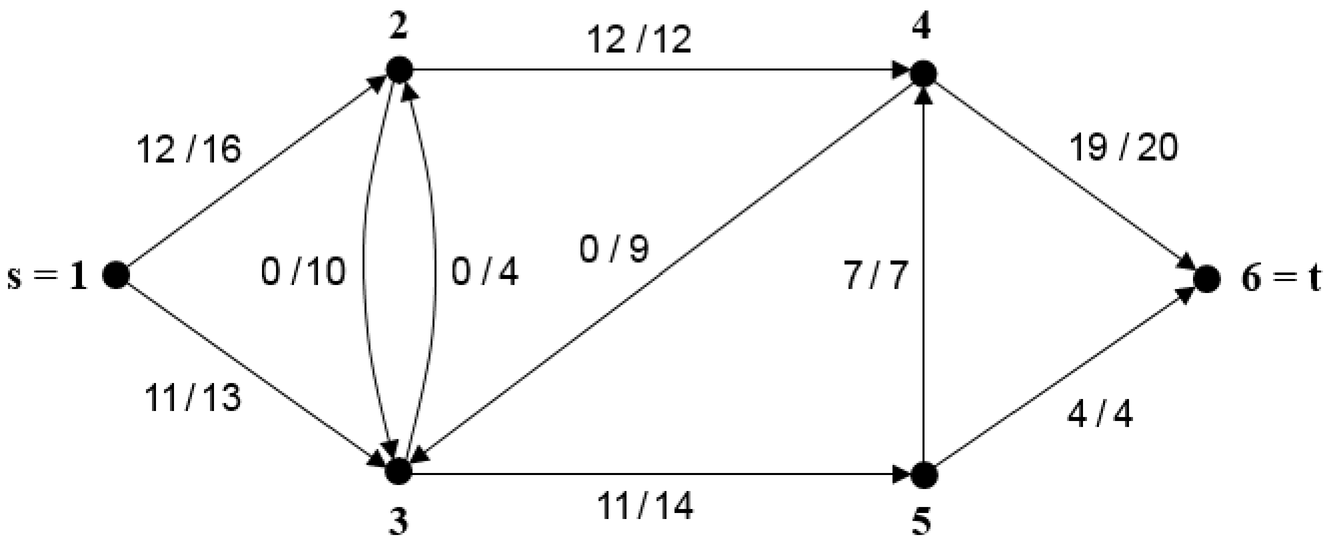


szélességi fa:

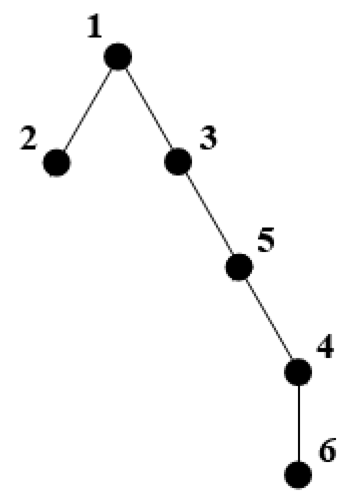


reziduális hálózat:

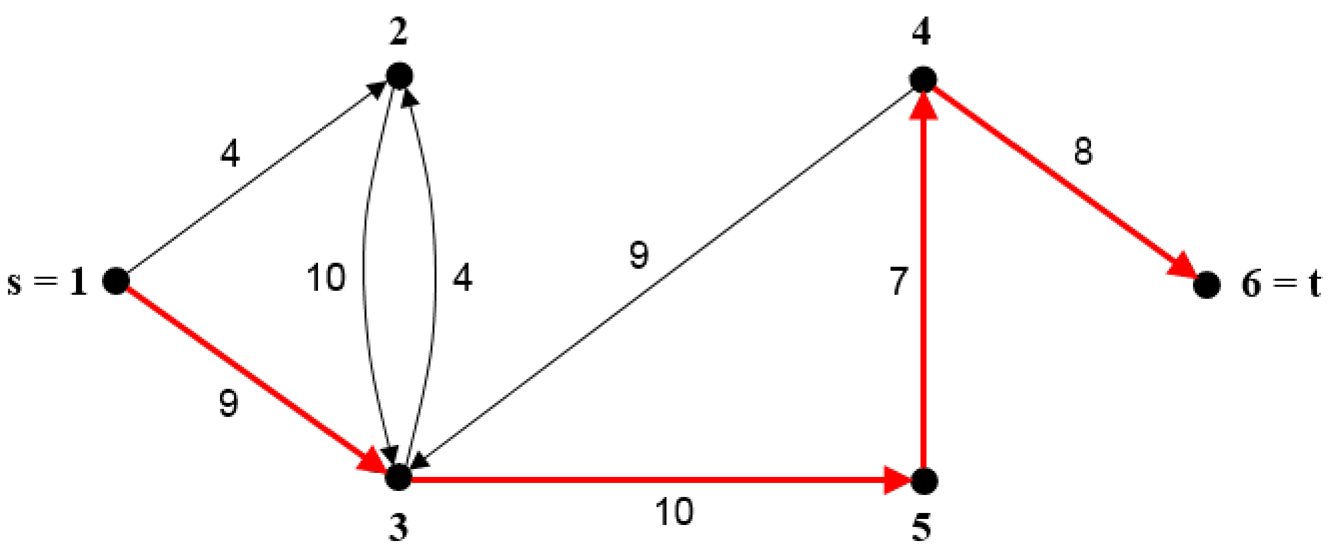


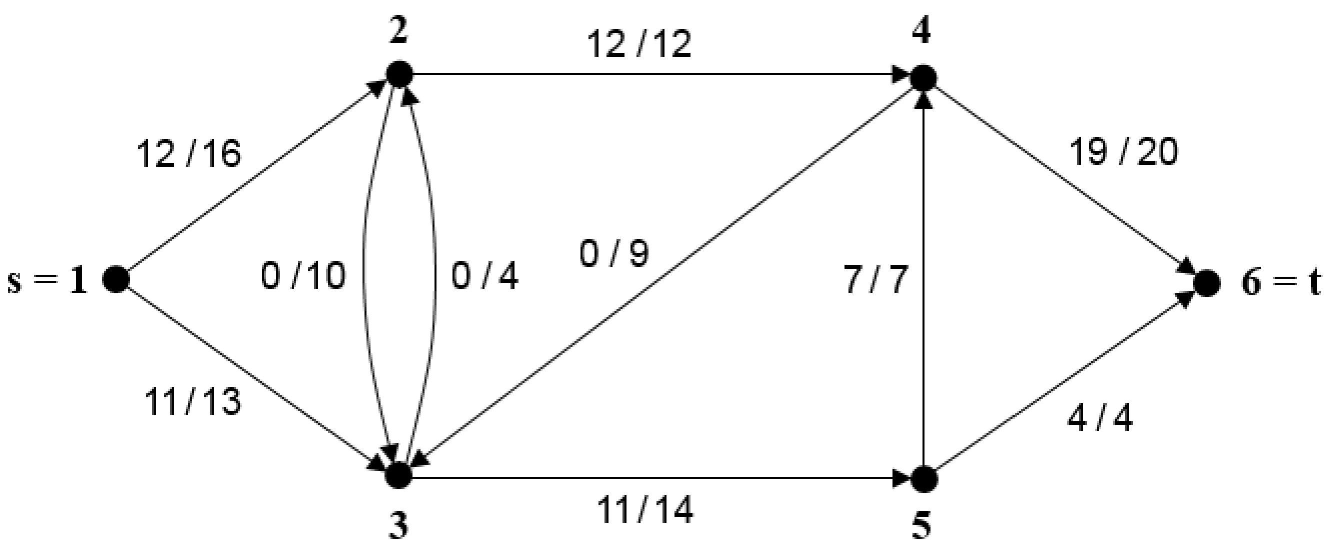


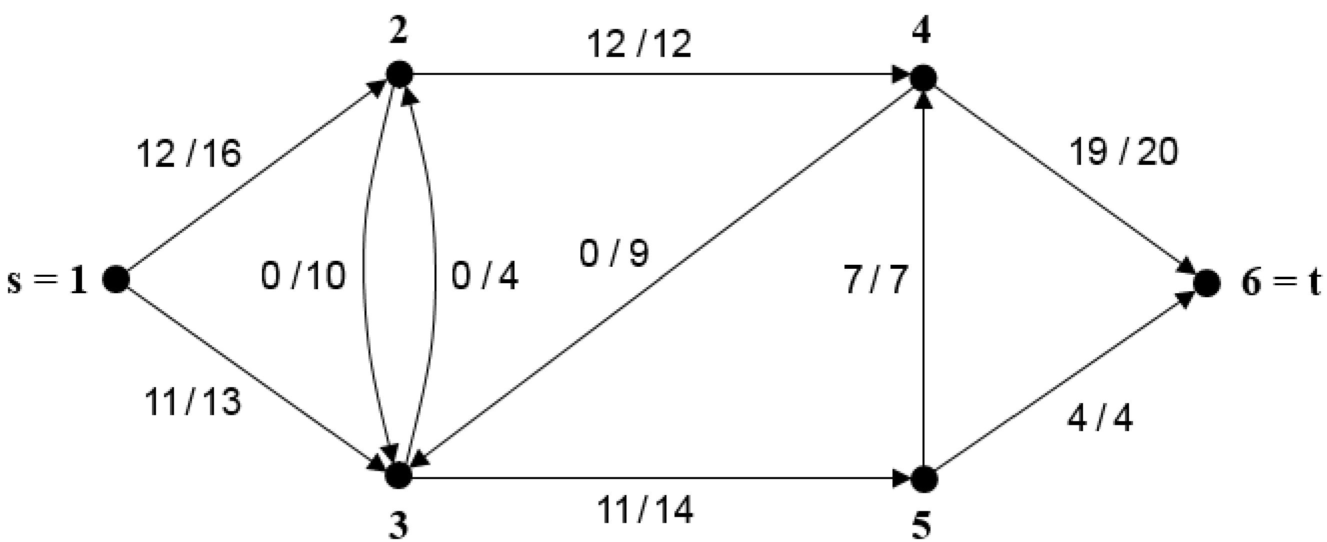
szélességi fa:



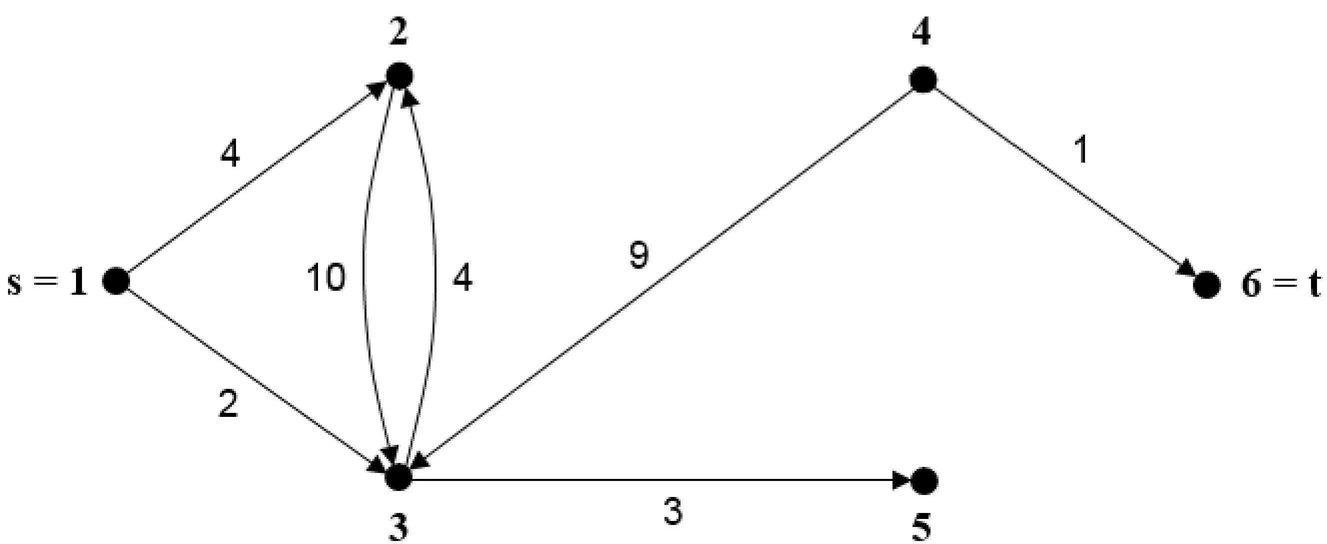
reziduális hálózat:

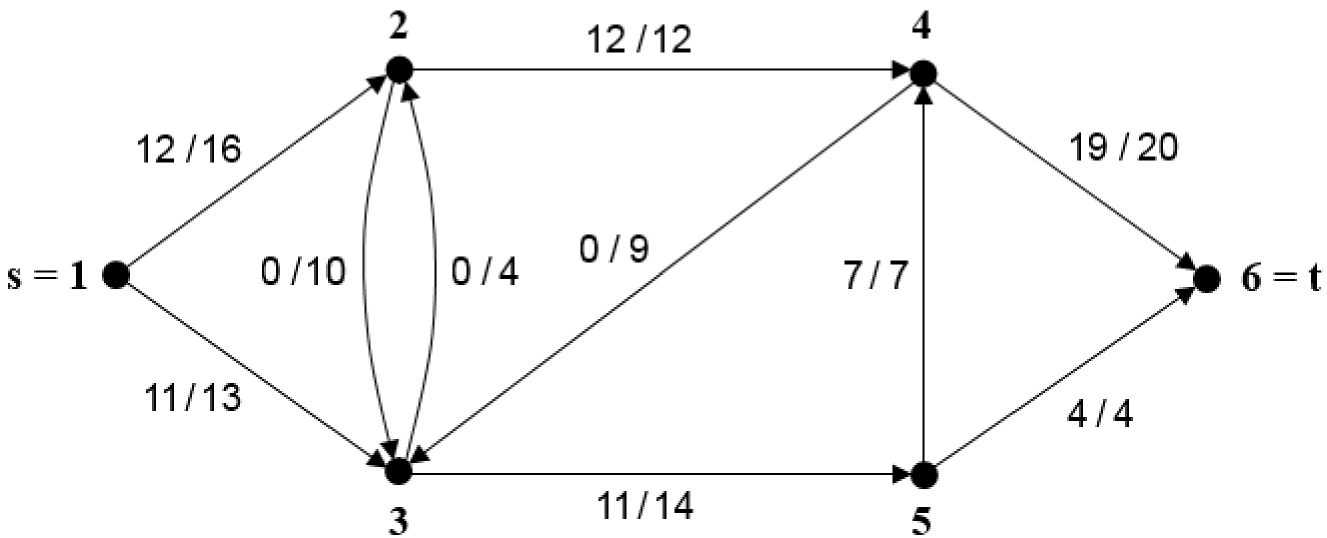



2
12 / 12
4
12 / 16
19 / 20
s = 1
0 / 10
0 / 4
0 / 9
7 / 7
6 = t
11 / 13
4 / 4
3
11 / 14
5



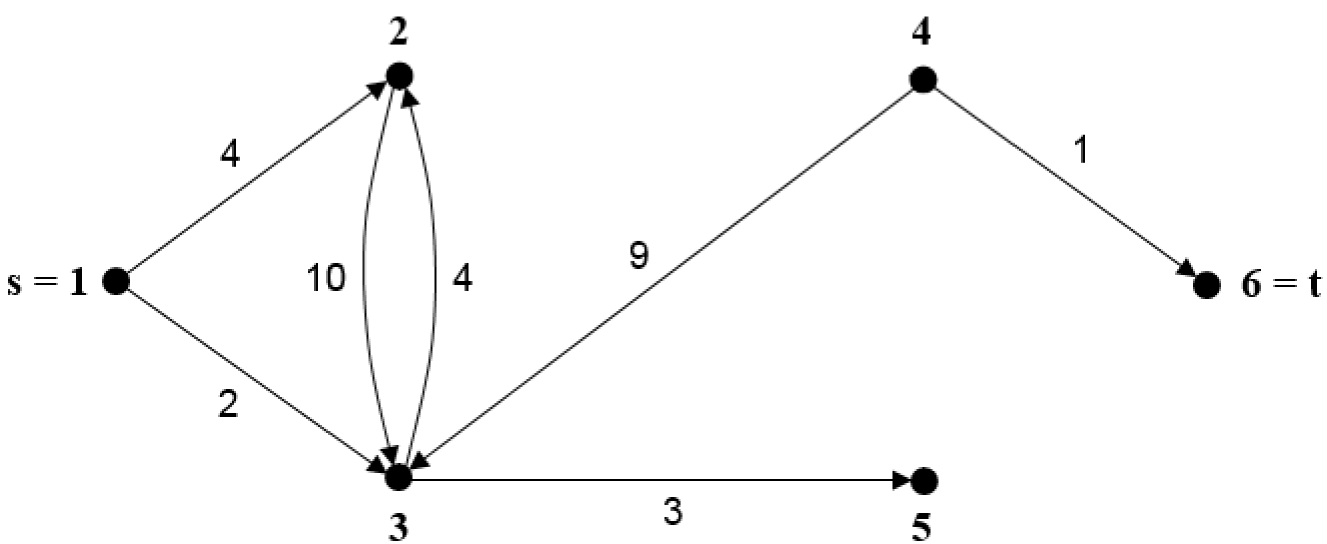
reziduális hálózat:





A maximális folyam értéke: **23**

reziduális hálózat:



A hálózati folyam probléma általánosításai

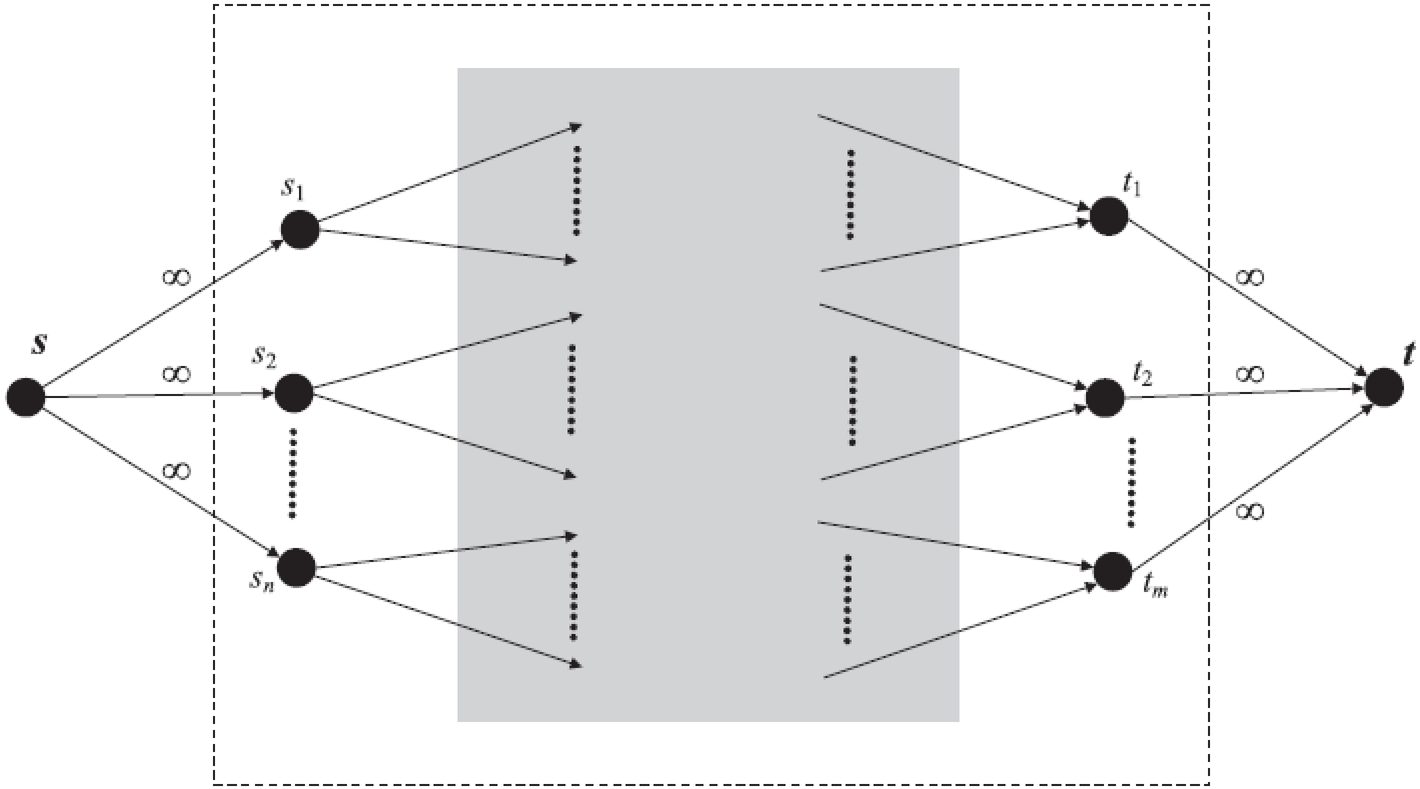
Három olyan helyzetet vizsgálunk meg, amikor a feladat könnyen visszavezethető a hagyományos folyam-problémára:

A hálózati folyam probléma általánosításai

Három olyan helyzetet vizsgálunk meg, amikor a feladat könnyen visszavezethető a hagyományos folyam-problémára:

1) Több forrás – több nyelő

Ha a hálózati gráf több forrást és nyelőt tartalmaz, akkor az ilyen feladat könnyen visszavezethető a klasszikus „egy forrás – egy nyelő“ problémára. A megoldás abban áll, hogy felveszünk egy virtuális *szuperforrás*t, amelyet végtelen kapacitású ki-élekkel hozzákötünk minden valódi forráshoz. Hasonlóképpen járunk el a nyelők esetében is, azaz minden valódi nyelőt összekötünk, végtelen kapacitású ki-éleken keresztül, egy virtuális *szupernyelő*vel. Az így nyert hálózati gráfban meghatározzuk a szuperforrásból a szupernyelőbe átfolyó maximális folyamot. A kapott eredmény a gráf valódi forrásoktól valódi nyelőkig levő szakaszán éppen az eredeti feladat megoldását jelenti.

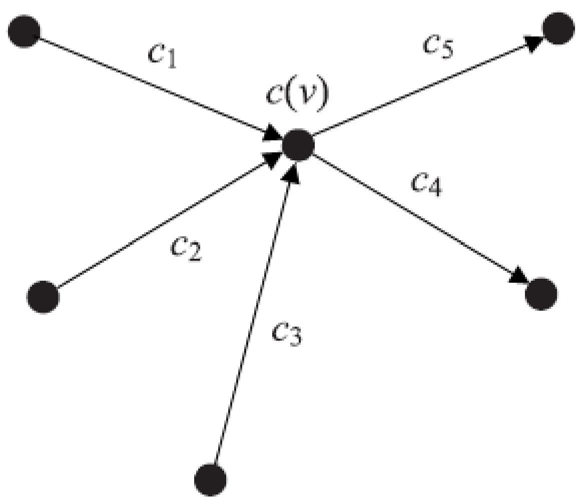

s
∞
∞
∞
s_1
s_2
s_n
t_1
t_2
t_m
∞
∞
∞
t

2) Amikor a csúcspontoknak is van kapacitása

Ha a csúcspontok is rendelkeznek kapacitásértékekkel, akkor megtehetjük, hogy minden csúcspontot helyettesítünk a pont kapacitásával azonos kapacitású éllel. Ezzel a feladatot visszavezetjük a klasszikus feladatra, ahol a csúcspontoknak nincs kapacitásértékük. A helyettesített csúcspont be-élei az új él kezdőpontjához, a ki-élei pedig az új él végpontjához fognak illeszkedni.

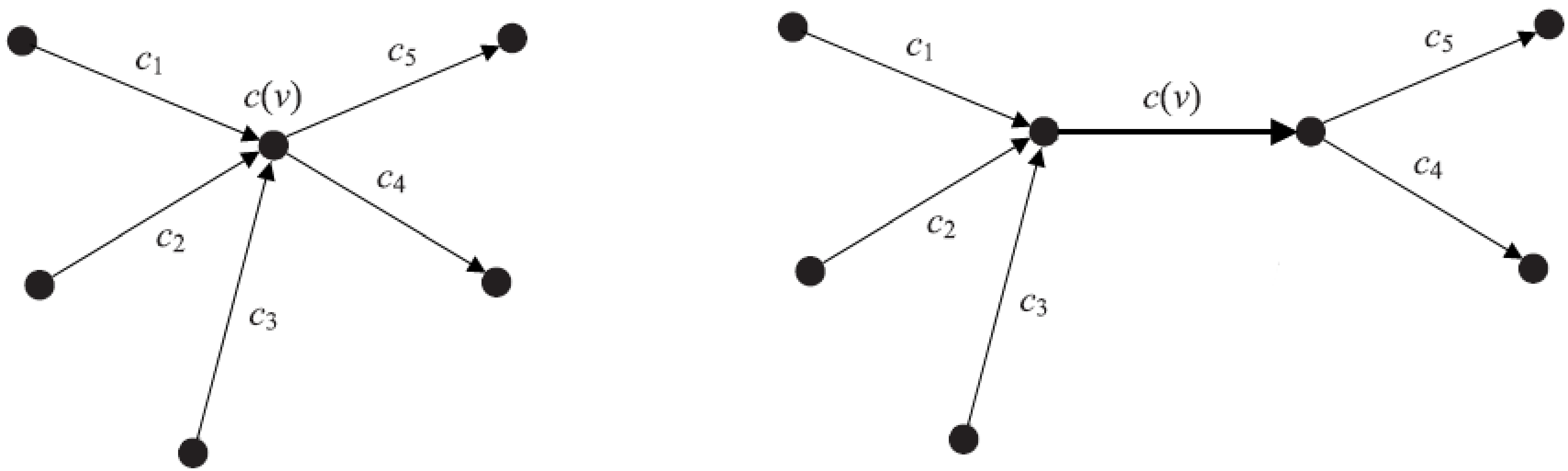
2) Amikor a csúcspontoknak is van kapacitása

Ha a csúcspontok is rendelkeznek kapacitásértékekkel, akkor megtehetjük, hogy minden csúcspontot helyettesítünk a pont kapacitásával azonos kapacitású éllel. Ezzel a feladatot visszavezetjük a klasszikus feladatra, ahol a csúcspontoknak nincs kapacitásértékük. A helyettesített csúcspont be-élei az új él kezdőpontjához, a ki-élei pedig az új él végpontjához fognak illeszkedni.



2) Amikor a csúcspontoknak is van kapacitása

Ha a csúcspontok is rendelkeznek kapacitásértékekkel, akkor megtehetjük, hogy minden csúcspontot helyettesítünk a pont kapacitásával azonos kapacitású éllel. Ezzel a feladatot visszavezetjük a klasszikus feladatra, ahol a csúcspontoknak nincs kapacitásértékük. A helyettesített csúcspont be-élei az új él kezdőpontjához, a ki-élei pedig az új él végpontjához fognak illeszkedni.

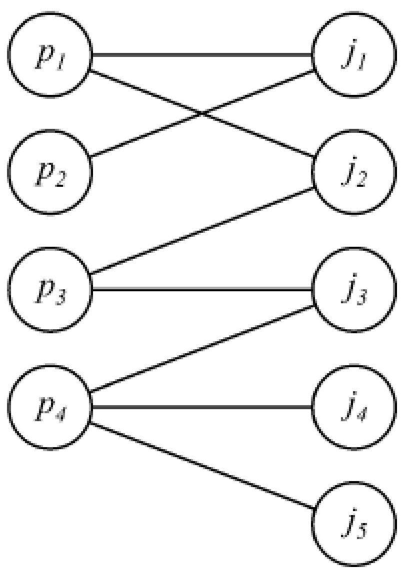


3) Maximális párosítás keresése páros gráfban

Adjunk irányítást a $G = (A, B)$ páros gráfnak. Legyenek az élek kezdőpontjai az A halmazbeli csúcspontok, a végpontokat pedig vegyük a B halmazból. Vegyünk fel egy virtuális szuperforrást, amelyből induljon irányított él minden A halmazbeli csúcsponthoz. Hasonlóképpen, legyen egy virtuális szupernyelő is, amelyhez minden B halmazbeli csúcsponttól érkezik egy-egy irányított él. Továbbá tekintsük a gráf minden élén a kapacitás értékét 1-nek.

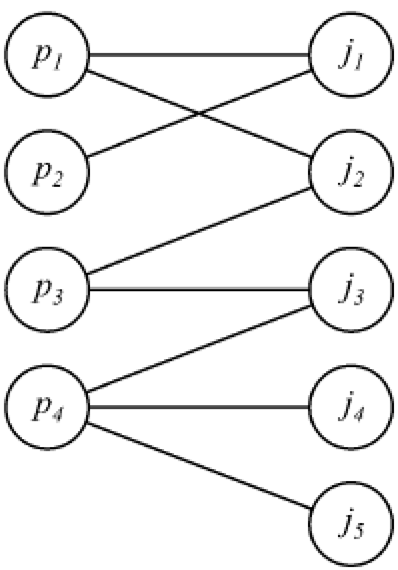
3) Maximális párosítás keresése páros gráfban

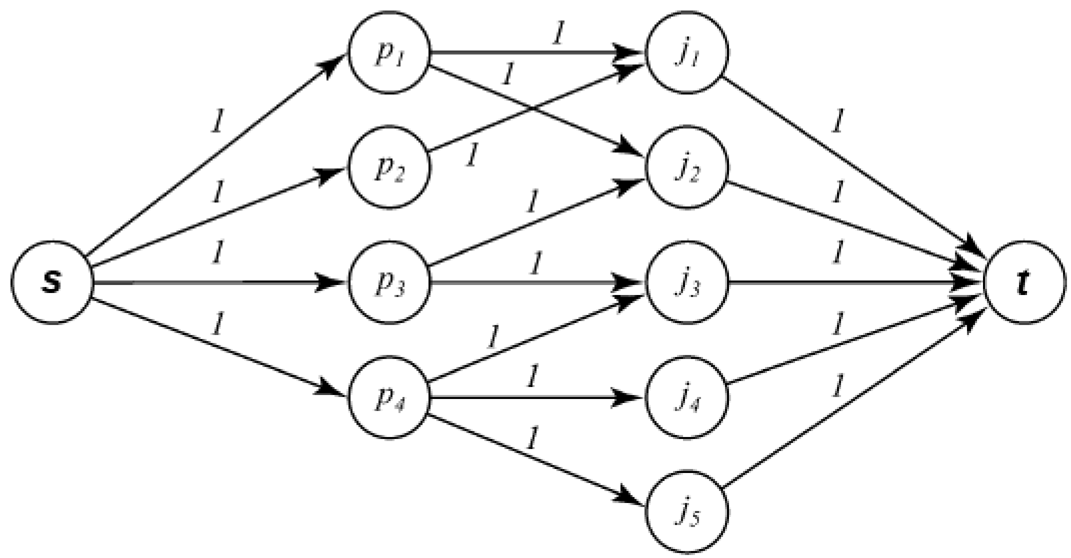
Adjunk irányítást a $G = (A, B)$ páros gráfnak. Legyenek az élek kezdőpontjai az A halmazbeli csúcspontok, a végpontokat pedig vegyük a B halmazból. Vegyünk fel egy virtuális szuperforrást, amelyből induljon irányított él minden A halmazbeli csúcsponthoz. Hasonlóképpen, legyen egy virtuális szupernyelő is, amelyhez minden B halmazbeli csúcsponttól érkezik egy-egy irányított él. Továbbá tekintsük a gráf minden élén a kapacitás értékét 1-nek.



3) Maximális párosítás keresése páros gráfban

Adjunk irányítást a $G = (A, B)$ páros gráfnak. Legyenek az élek kezdőpontjai az A halmazbeli csúcspontok, a végpontokat pedig vegyük a B halmazból. Vegyünk fel egy virtuális szuperforrást, amelyből induljon irányított él minden A halmazbeli csúcsponthoz. Hasonlóképpen, legyen egy virtuális szupernyelő is, amelyhez minden B halmazbeli csúcsponttól érkezik egy-egy irányított él. Továbbá tekintsük a gráf minden élén a kapacitás értékét 1-nek.





Ha meghatározzuk ebben a hálózatban a maximális folyam értékét, akkor ez egyenlő lesz a G gráf maximális párosításának élszámával. Másfelől a maximális folyamhoz tartozó él-idegen utak A és B halmazok közötti élei éppen egy maximális párosítást adnak meg.

4) Irányítatlan hálózati gráf

Irányítatlan gráfok esetében egyszerűen helyettesítünk minden irányítatlan élt irányított oda-vissza-élekkel. Bár ez azt fogja eredményezni, hogy a forrásnak is lesznek be-élei és a nyelőnek is ki-élei, mindez nem fogja megzavarni az algoritmus működését. Ezek az élek csak visszamutató élként kerülhetnének bármely javítóútra. Ez viszont nem fog soha bekövetkezni, hiszen visszamutató élként alapállásból telítettek (a kezdeti folyamértékük nulla).