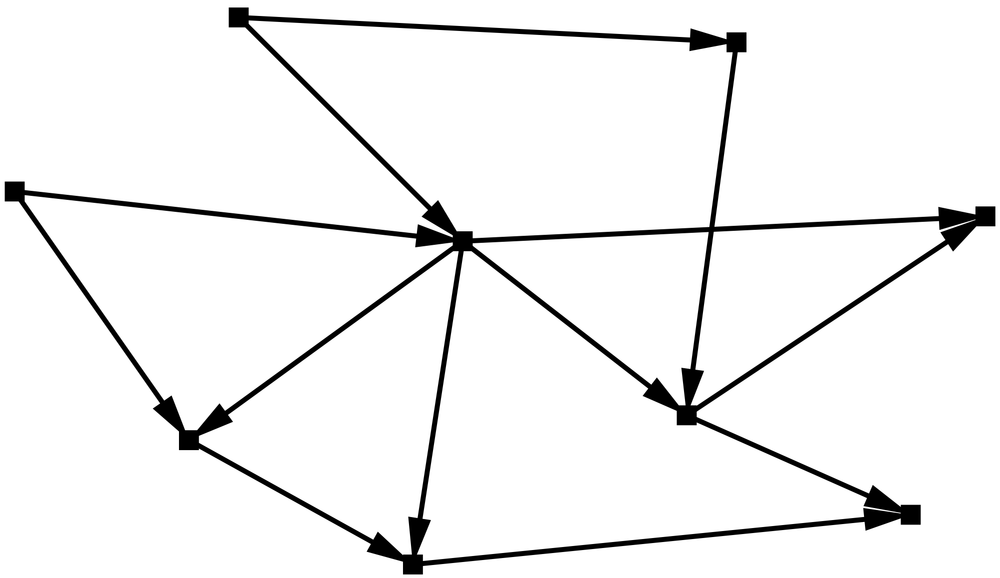
Suunatud graaf $G = (V, E)$ on *suunatud tsükliteta*, kui iga kahe tipu $u, v \in V$ jaoks ei leidu teed $u$-st $v$-sse või $v$-st $u$-sse.

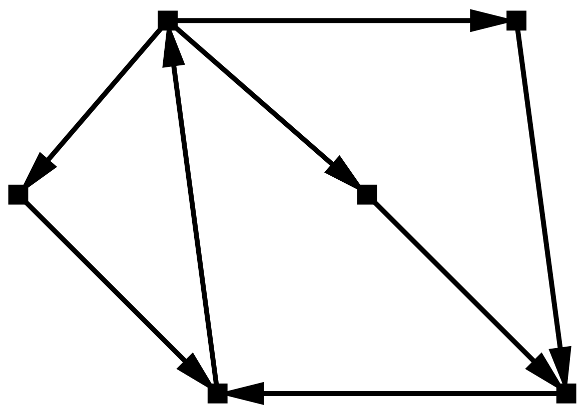
$G$ on *tugevalt sidus*, kui iga kahe tipu $u, v \in V$ jaoks leidub tee nii $u$-st $v$-sse kui ka $v$-st $u$-sse.

Suunatud graafi $G$ *tugevalt sidusad komponendid* on tema maksimaalsed alamgraafid, mis on tugevalt sidusad.

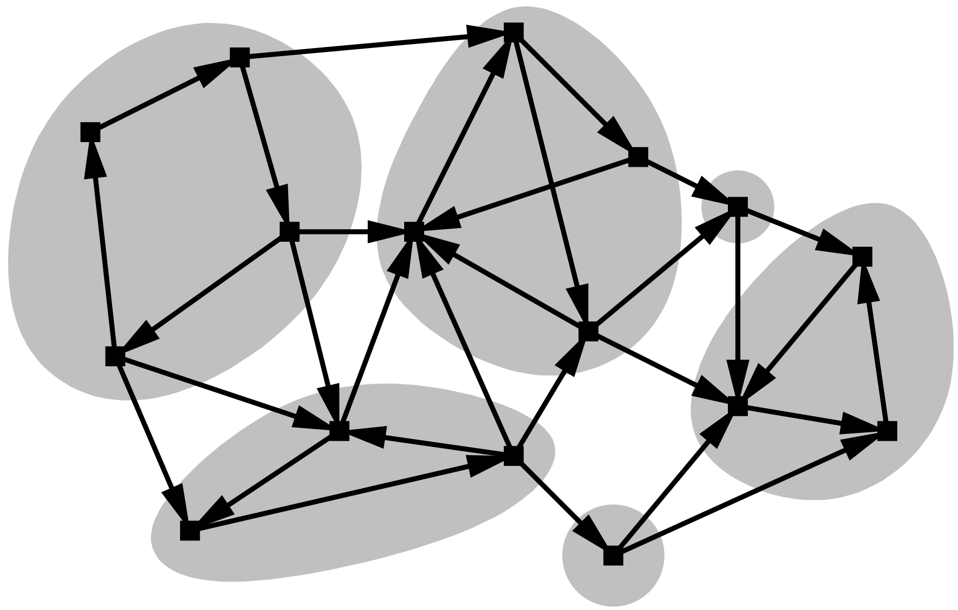
- Maksimaalne tipuhulkade sisalduvuse mõttes.



Suunatud tsükliteta graaf.



Tugevalt sidus graaf.

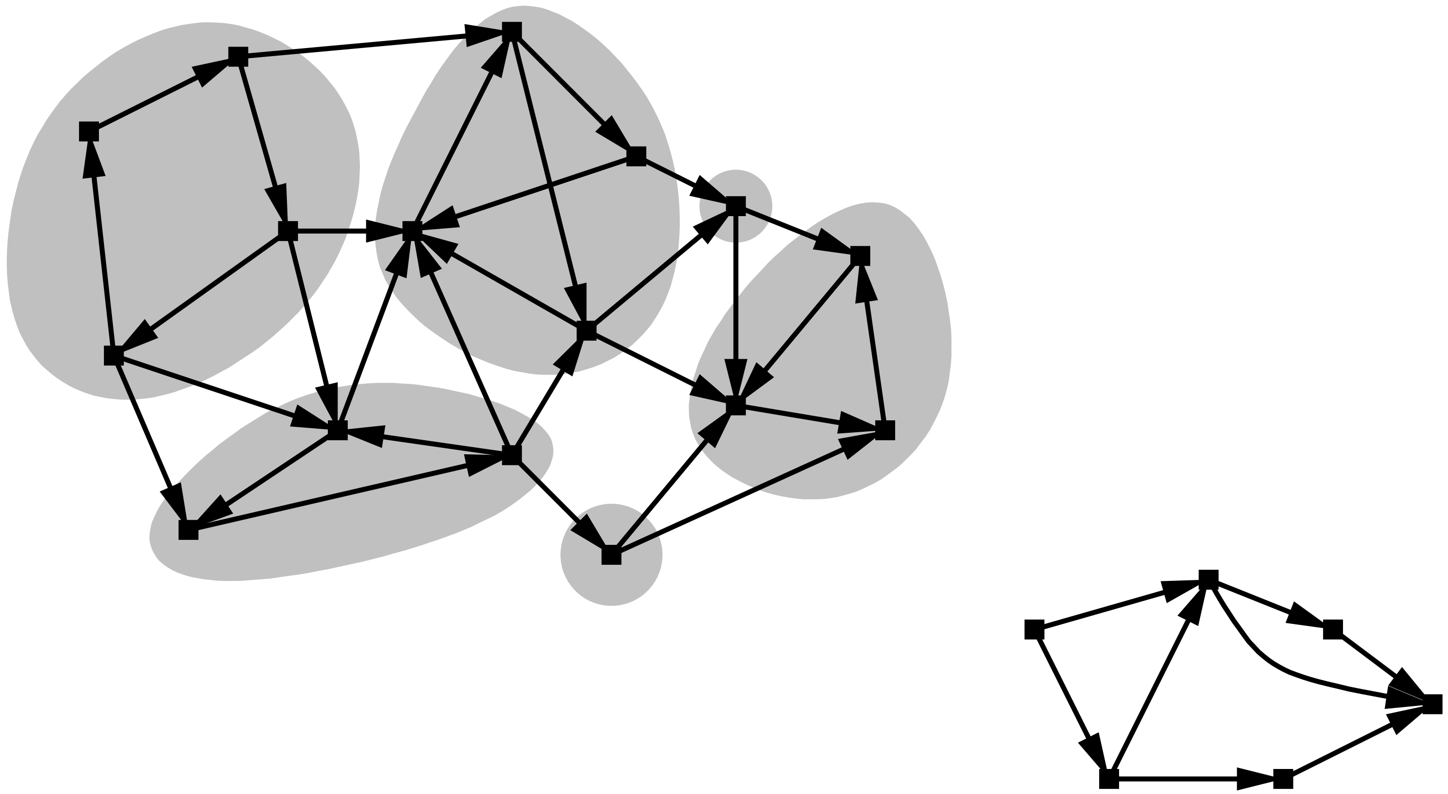


Graafi tugevalt sidusad komponendid.

Graafi $G = (V, E)$ *komponentgraafiks* nimetame graafi $G^{\mathrm{komp}}$, mille

- tippudeks on graafi $G$ tugevalt sidusad komponendid;
- kaar graafi $G^{\mathrm{komp}}$ tipust $V_i$ tippu $V_j$ (siin $V_i, V_j \subseteq V$) leidub parajasti siis, kui $G$-s leidub serv mõnest $V_i$-sse kuuluvast tipust mõnda $V_j$-i kuuluvasse tippu.

Graaf $G^{\mathrm{komp}}$ on suunatud tsükliteta.

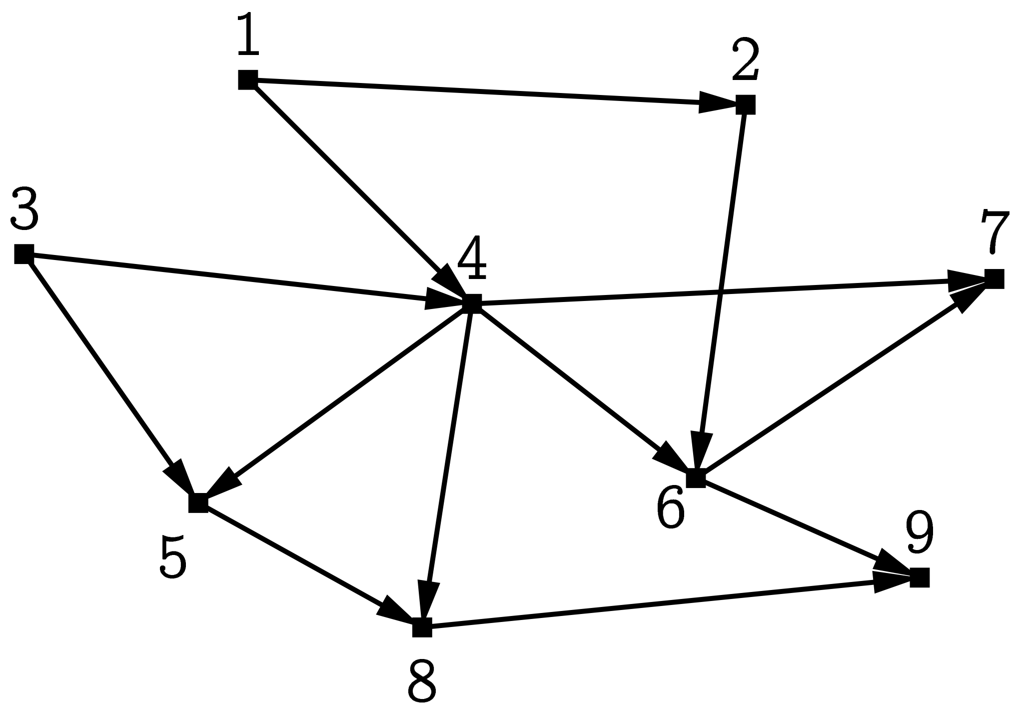


Näide komponentgraafist.

Kui graaf on suunatud tsükliteta, siis saab tema tippe selliselt järjestada, et igast tipust lähevad servad ainult selles järjestuses kaugemal olevatesse tippudesse.

Kui selline järjestus on antud, siis ütleme, et tipud on *topoloogiliselt sorteeritud*.

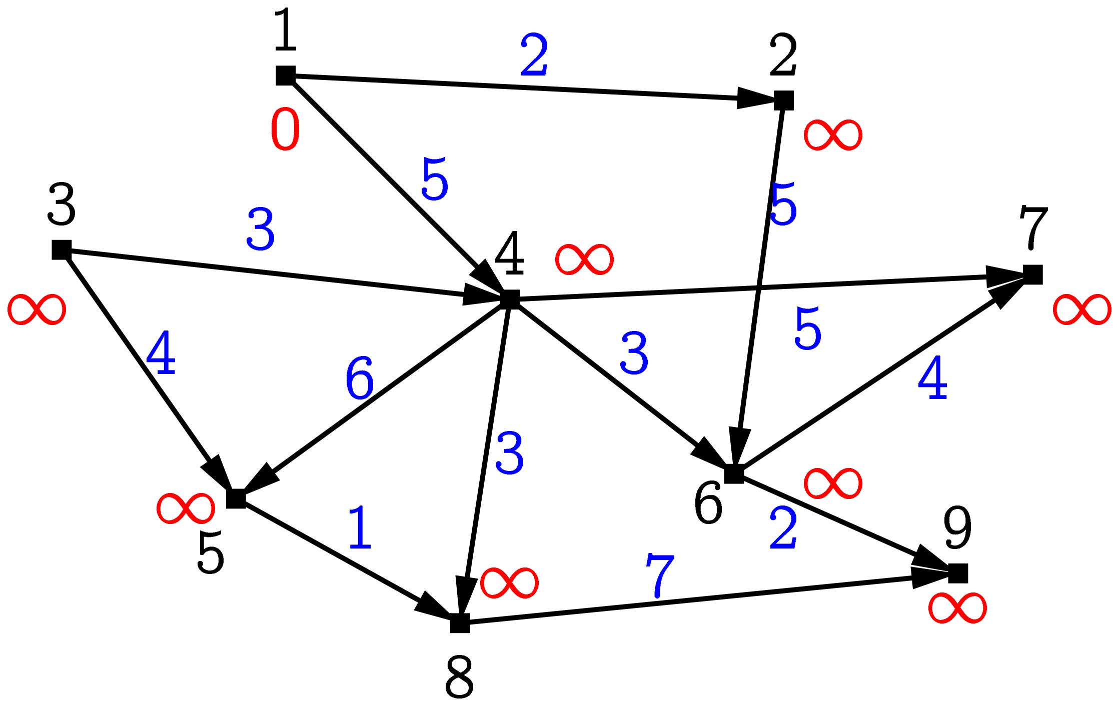
Algoritm tippude topoloogiliselt sorteerimiseks: Kiho konspekt, joonis 6.2. Tema keerukus on $O(|E|)$.



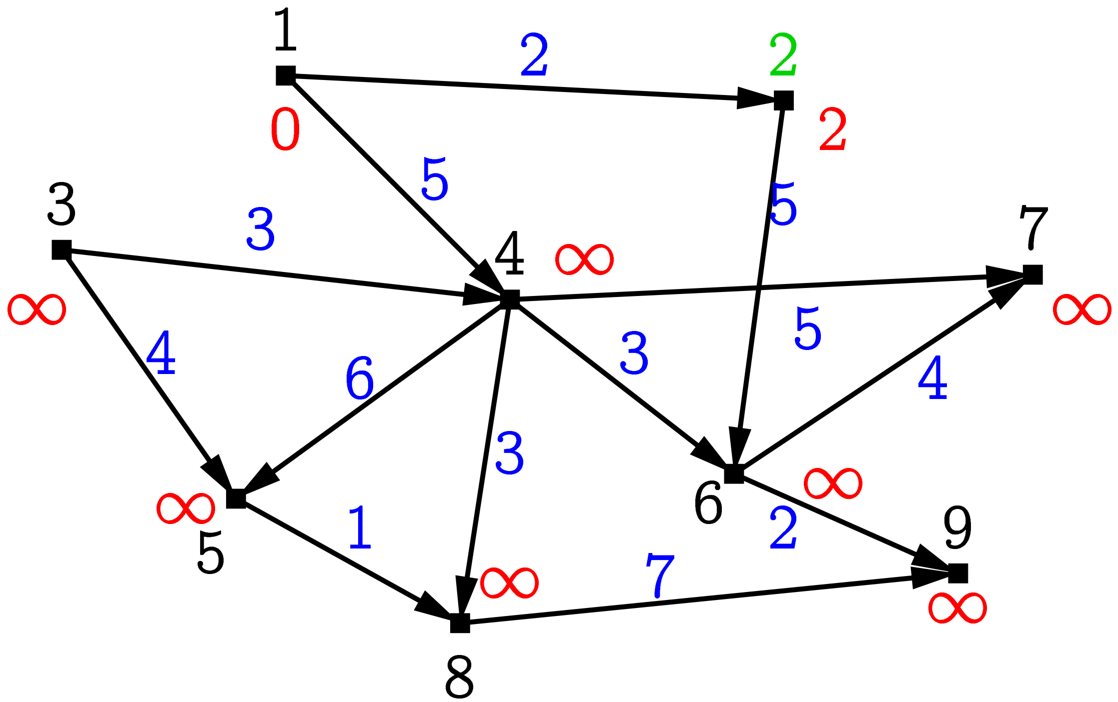

Näide suunatud tsükliteta graafi tippude topoloogilisest sorteerimisest.

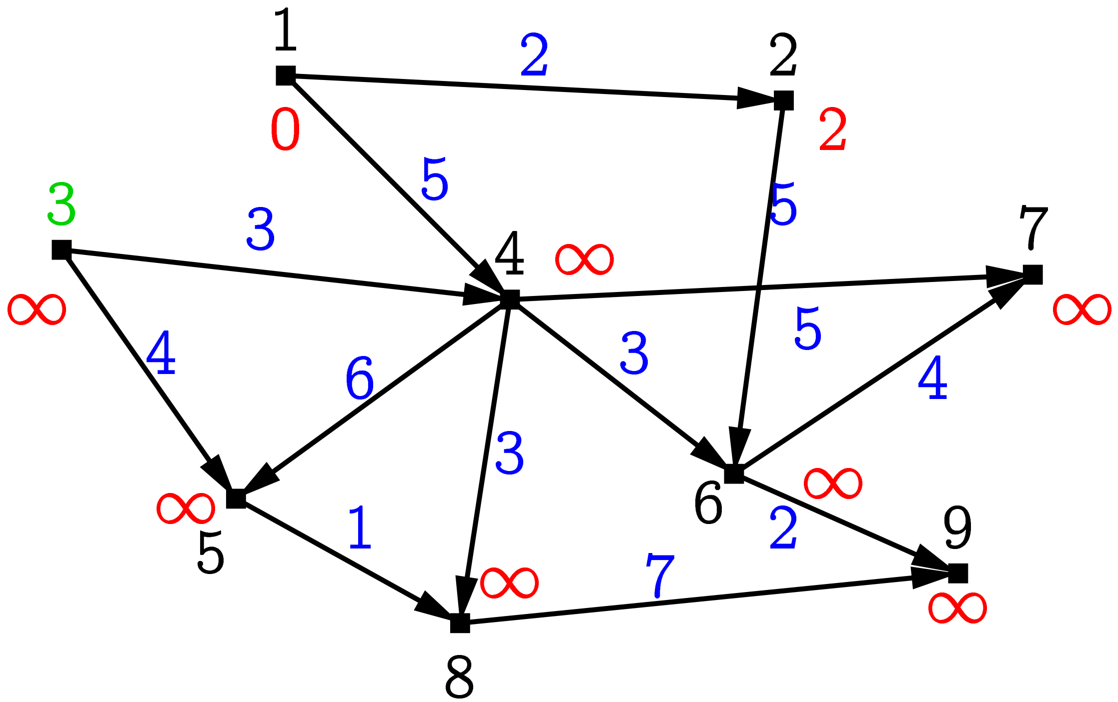
Suunatud tsükliteta graafis saab lühimaid teid mingist fikseeritud tipust $u$ kõigisse teistesse tippudesse leida ajaga $O(|E|)$.

$\{v_1, \ldots, v_n\} :=$ topol_sorteeri$(G)$
**for all** $v \in V$ **do** $D[v] := \infty$
$D[u] := 0$
**for** $i := 1$ **to** $n$ **do**
    **for all** $w \in G^{-1}v_i$ **do**
        $D[v_i] := \min(D[v_i], D[w] + \ell(w, v_i))$
**return** $D$

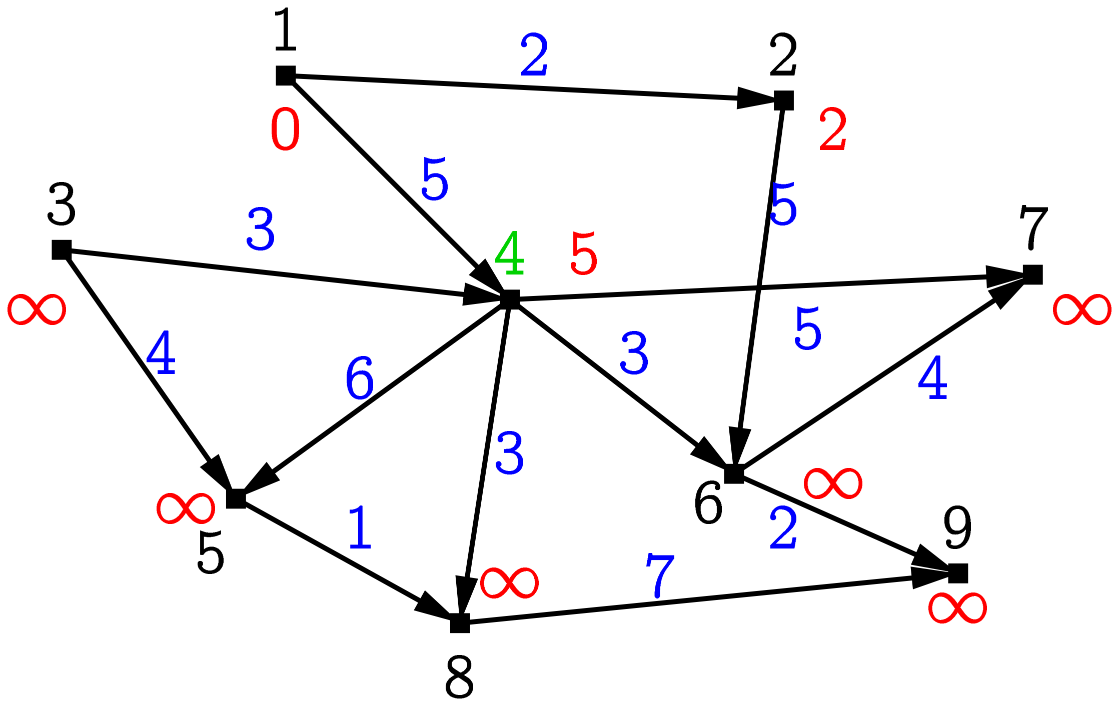


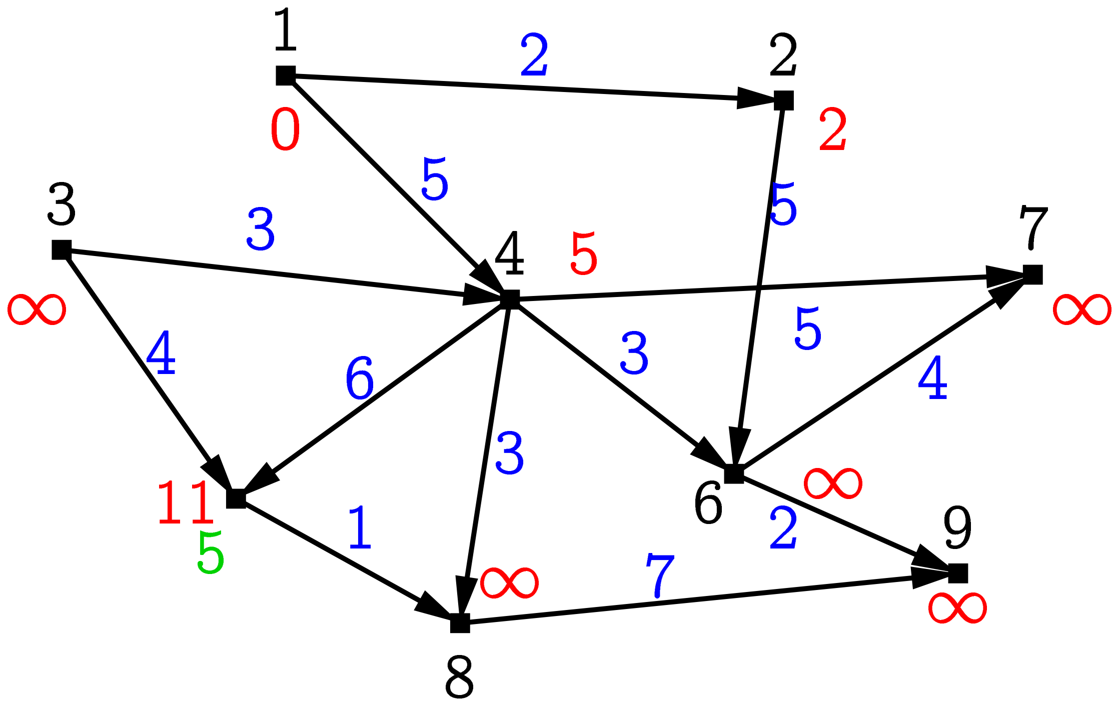
1
2
3
4
5
6
7
8
9
0
∞
2
5
3
5
4
6
3
3
5
4
1
2
7


1
0
2
2
2
3
∞
3
5
4
∞
5
7
∞
4
6
3
3
5
4
6
∞
∞
5
1
2
9
∞
7
8
∞

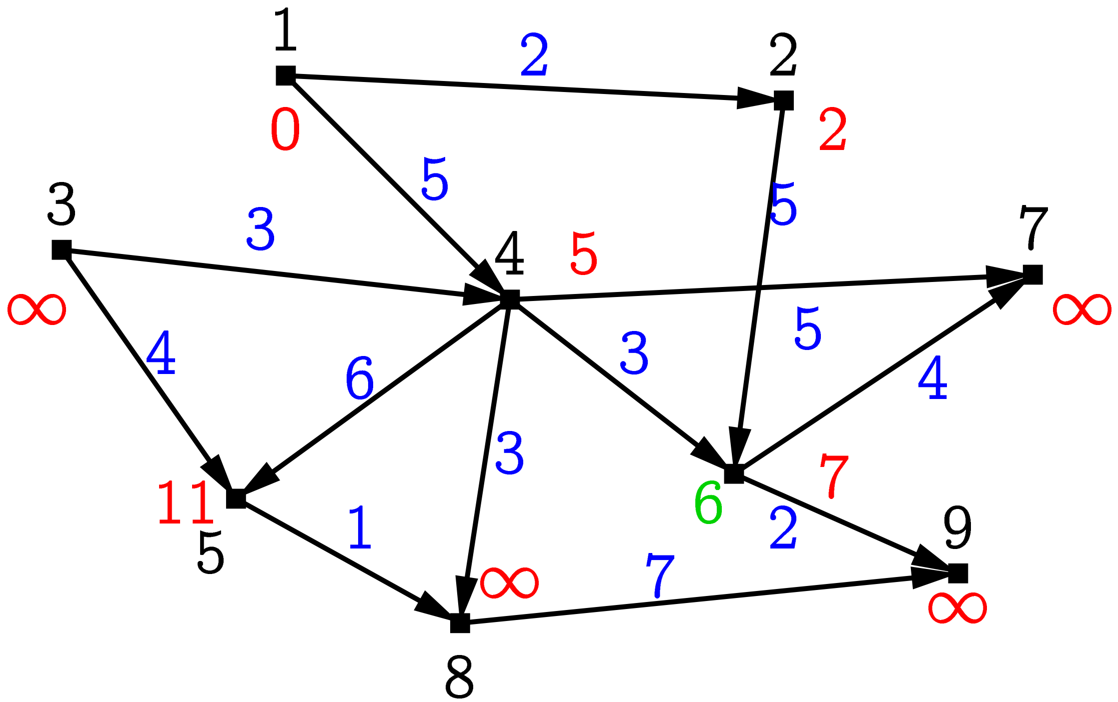


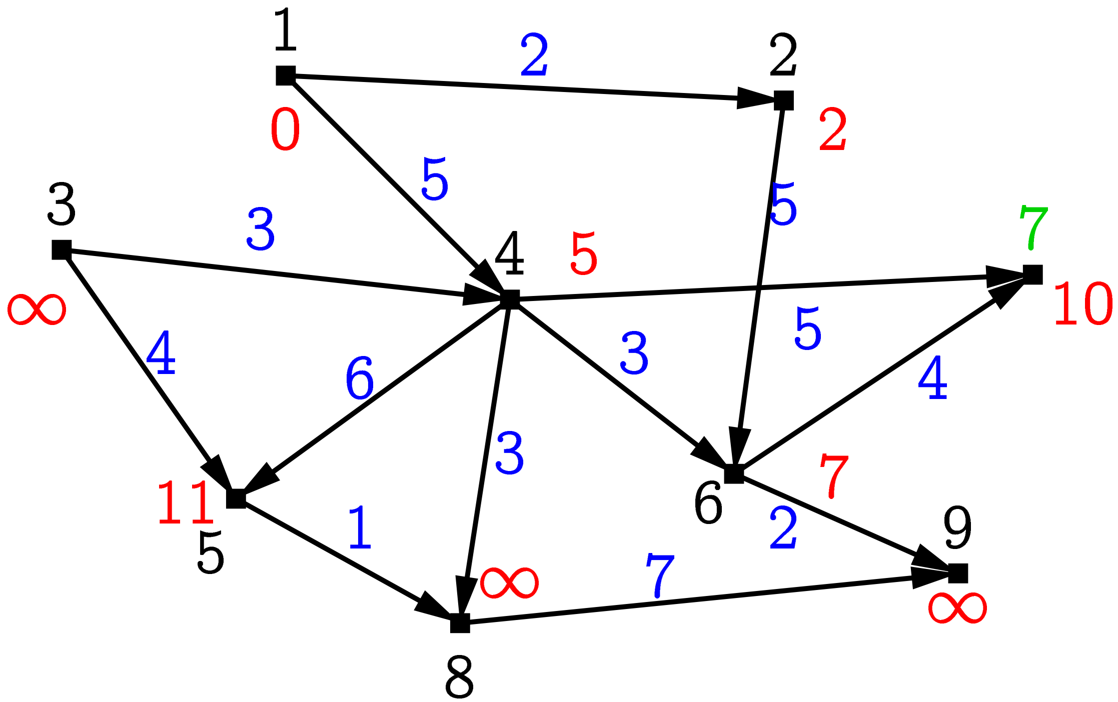
1
0
2
2
2
5
5
3
3
∞
4
∞
7
∞
5
4
6
3
3
4
∞
5
1
6
∞
2
9
7
∞
∞
8


1
2
3
4
5
6
7
8
9
0
2
5
∞
2
5
3
5
3
4
6
3
5
4
1
2
7
∞
∞
∞
∞
∞
∞

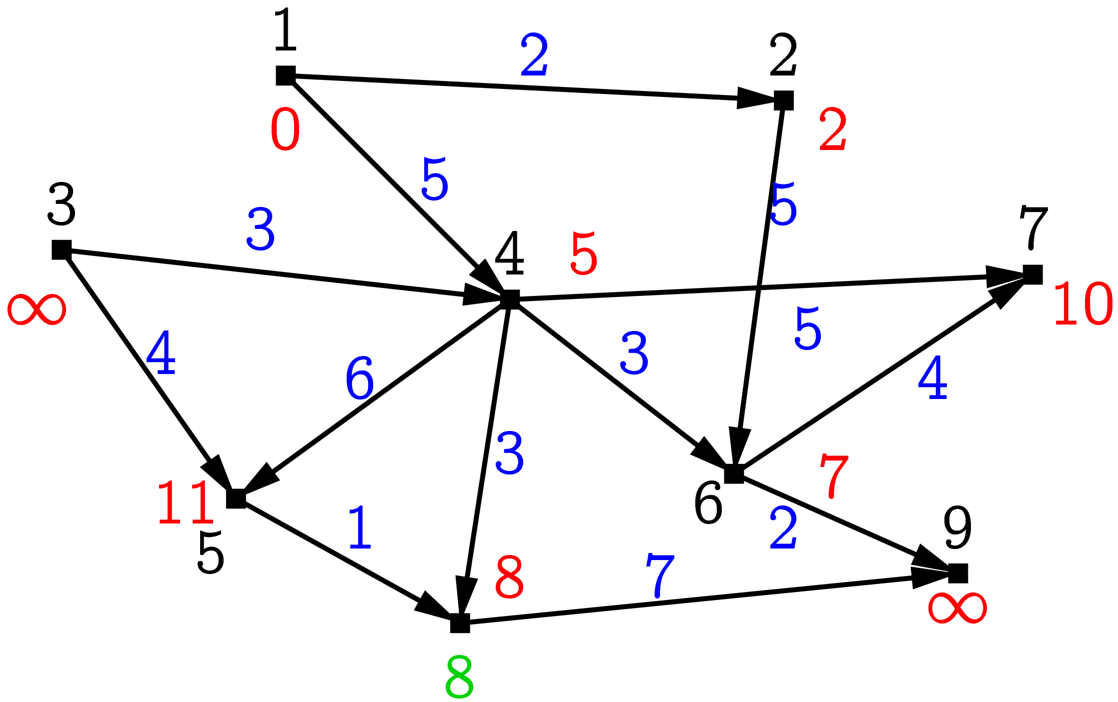


1
0
2
2
3
∞
4
5
7
∞
6
∞
11
5
8
∞
9
∞
2
5
3
5
5
4
3
6
3
4
1
7
2

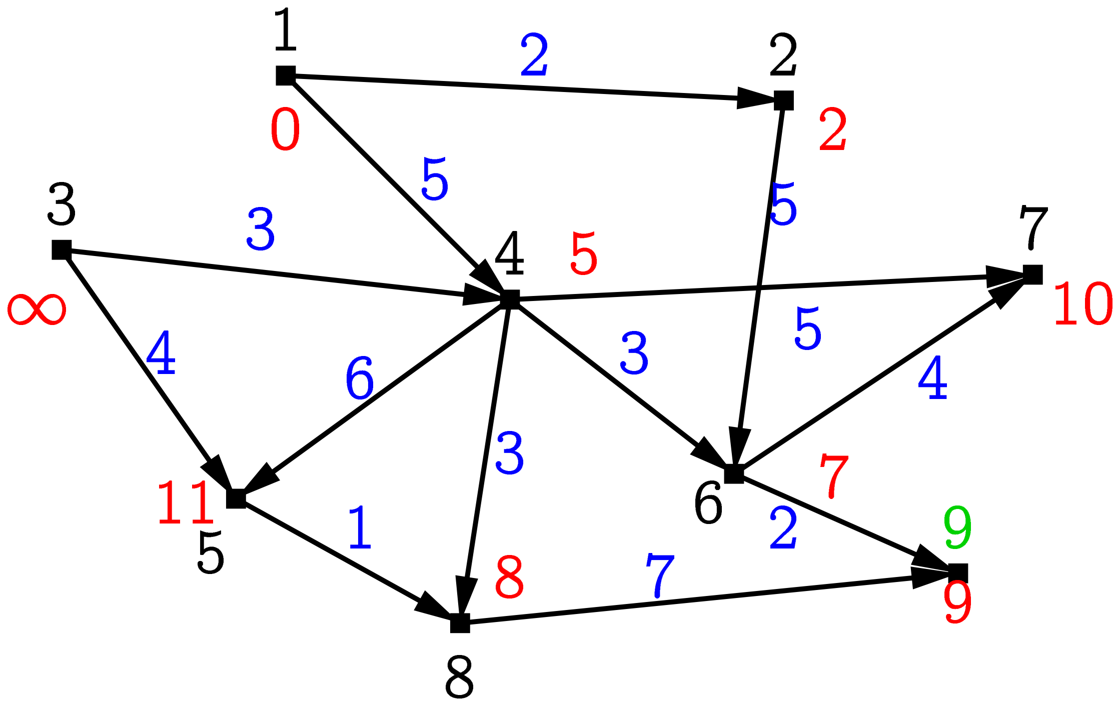

1
2
0
2
5
3
3
5
5
4
7
∞
5
5
∞
4
3
6
4
3
7
11
6
1
2
9
5
∞
7
∞
8



1
2
0
2
5
3
3
5
4
5
7
5
10
∞
4
6
3
3
5
4
7
11
1
6
2
9
5
∞
7
∞
8



1
2
0
2
5
3
5
3
4
5
7
∞
5
10
4
3
6
4
3
7
11
6
1
2
9
5
8
7
∞
8



1
2
0
2
3
3
5
5
4
5
7
∞
10
4
6
3
5
4
3
7
11
6
1
2
9
5
8
7
9
8

Tsükkel ridades 4–5 toimub üle kõigi servade. Sellest ka keerukus $O(|E|)$.

Seda tsüklit võib teha suvalises sellises järjekorras, et kui serv $(w_1, w_2)$ vaadatakse läbi enne serva $(w_3, w_4)$, siis $w_1 \leqslant w_4$.

Siis võib kindel olla, et $(w_3, w_4)$ läbivaatus ei mõjuta $w_1$ kaugust $u$-st.

Muuhulgas siis ka...

4 **for** $i := 1$ **to** $n$ **do**<br>
5     **for all** $w \in Gv_i$ **do**<br>
6         $D[w] := \min(D[w], D[v_i] + \ell(v_i, w))$

Kuidas leida graafi tugevalt sidusaid komponente?

Naiivne algoritm — leiame kõigi tippude vahelised kaugused, kontrollime kõigi paaride $u, v \in V$ jaoks, kas $d(u, v) < \infty$ ja $d(v, u) < \infty$.

Vaatame järgmist graafi sügavuti läbimise algoritmi, mis oma töö tulemusena järjestab tipud vastava tipu töötlemise *lõpetamise* järjekorras.

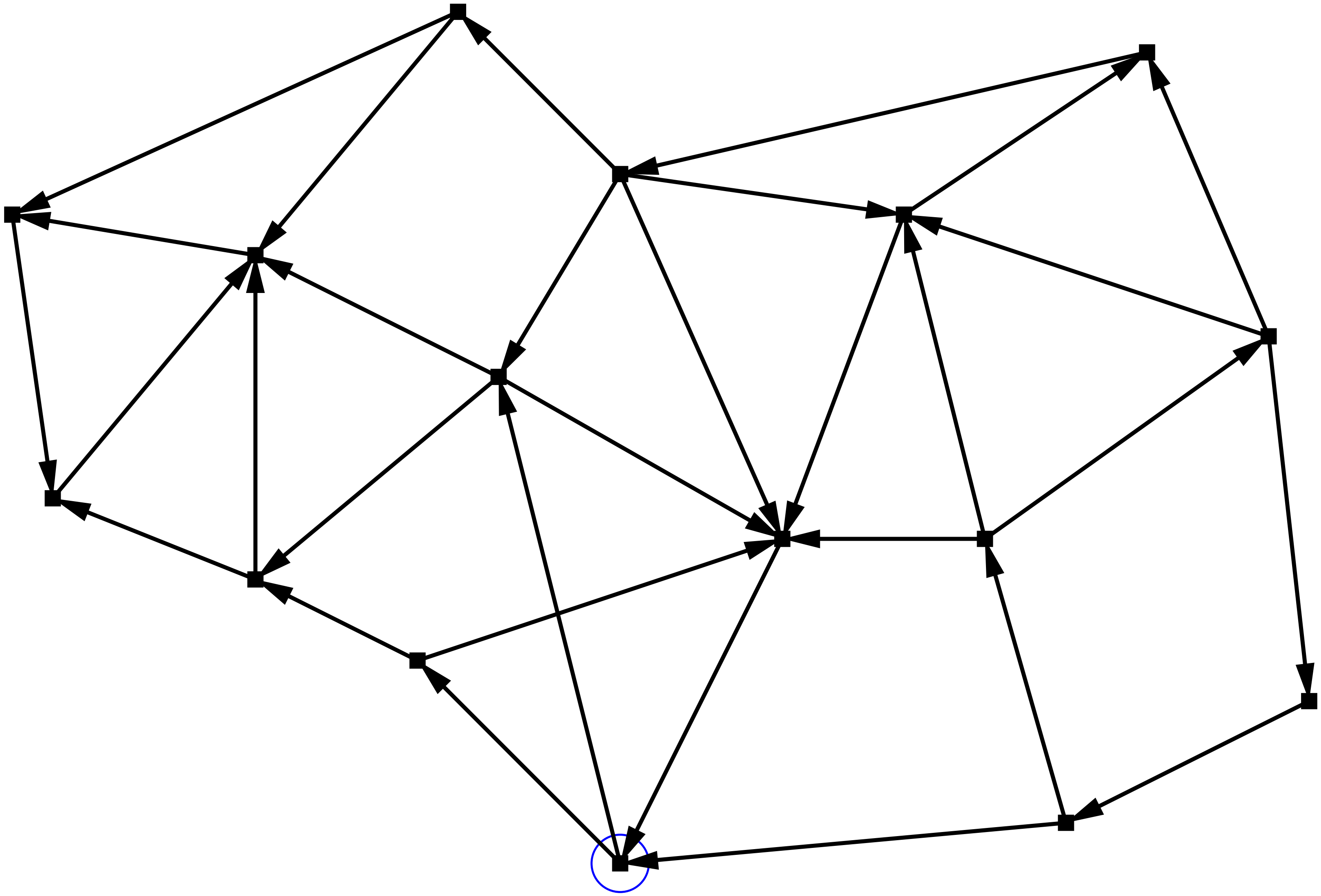
Igal tipul $v$ olgu kaks töövälja: $v.läbitud$ ja $v.jrknr$.

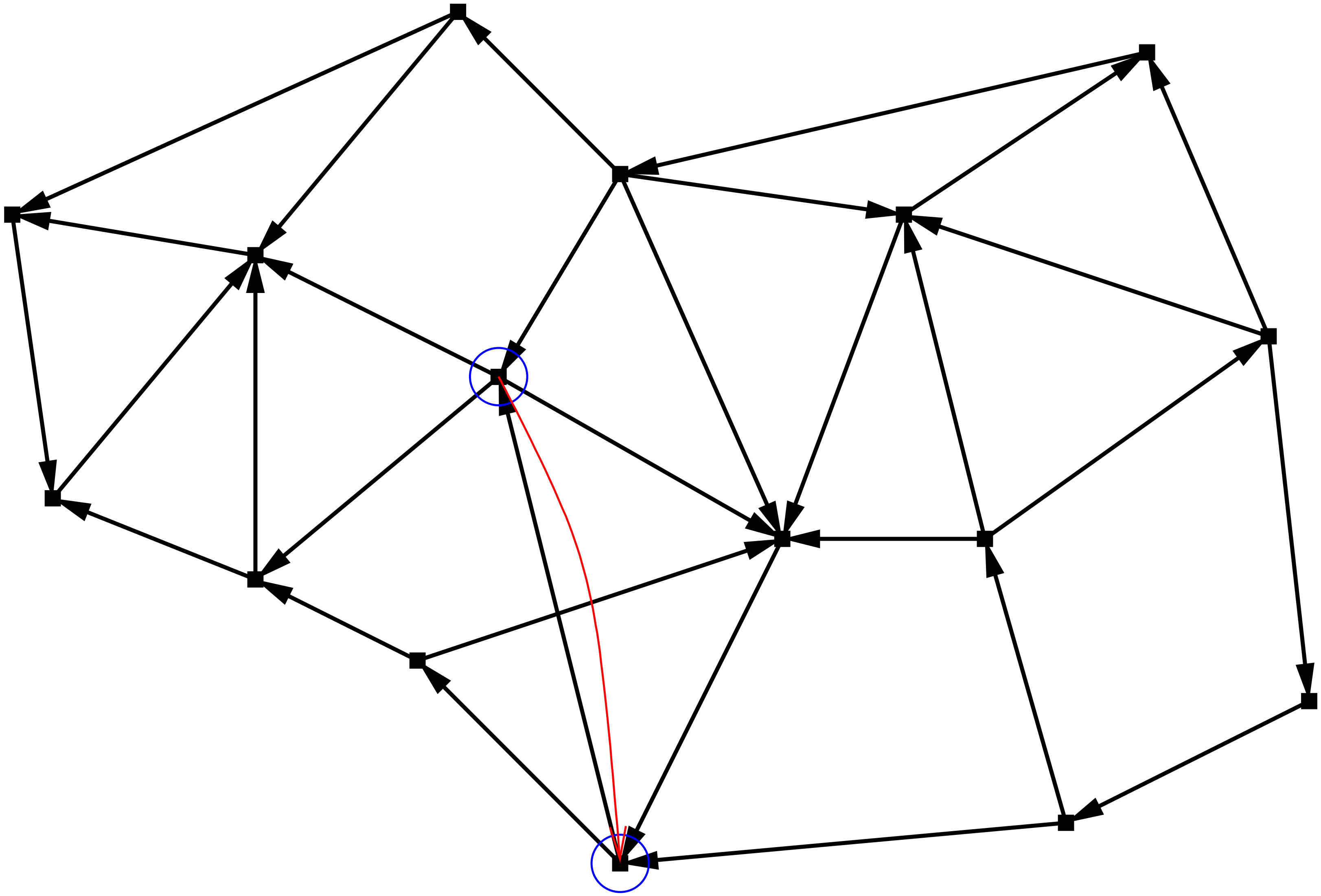
sügavuti($G$) on

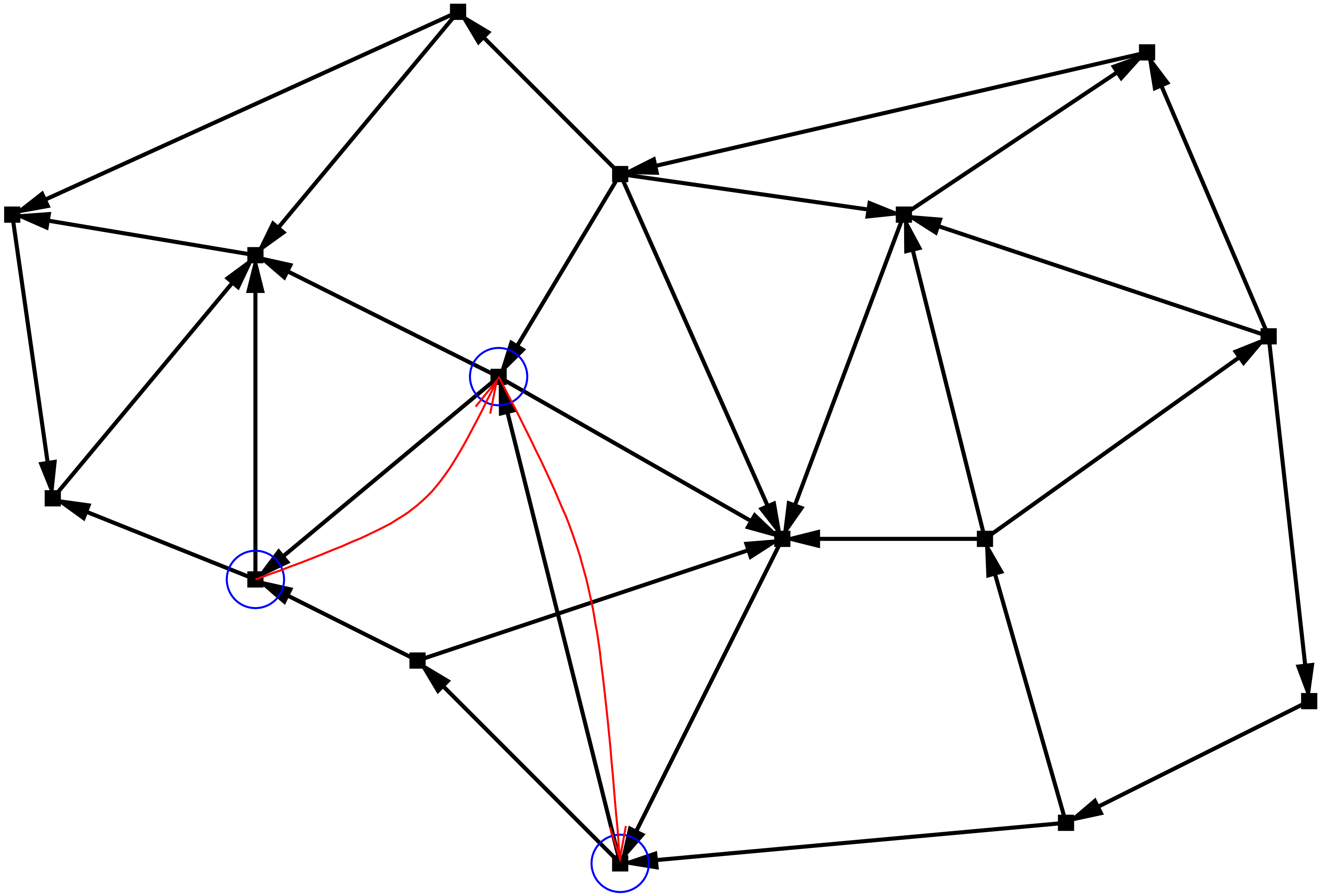
**for all** $v \in V$ **do** $v.\textit{läbitud} :=$ false
$aeg := 0$
**for all** $v \in V$ **do**
    **if** $\neg v.\textit{läbitud}$ **then** külasta($v$)

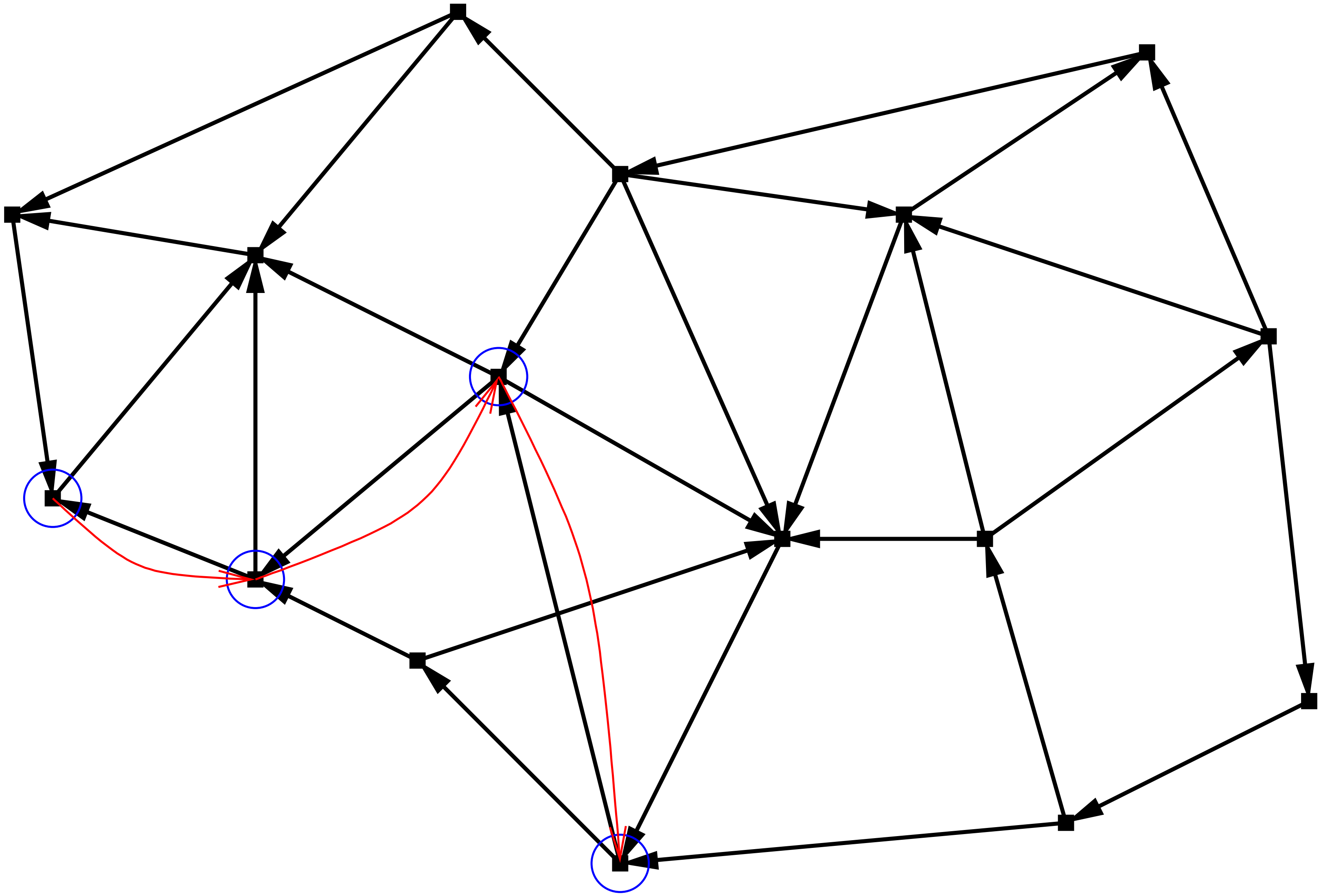
külasta($v$) on

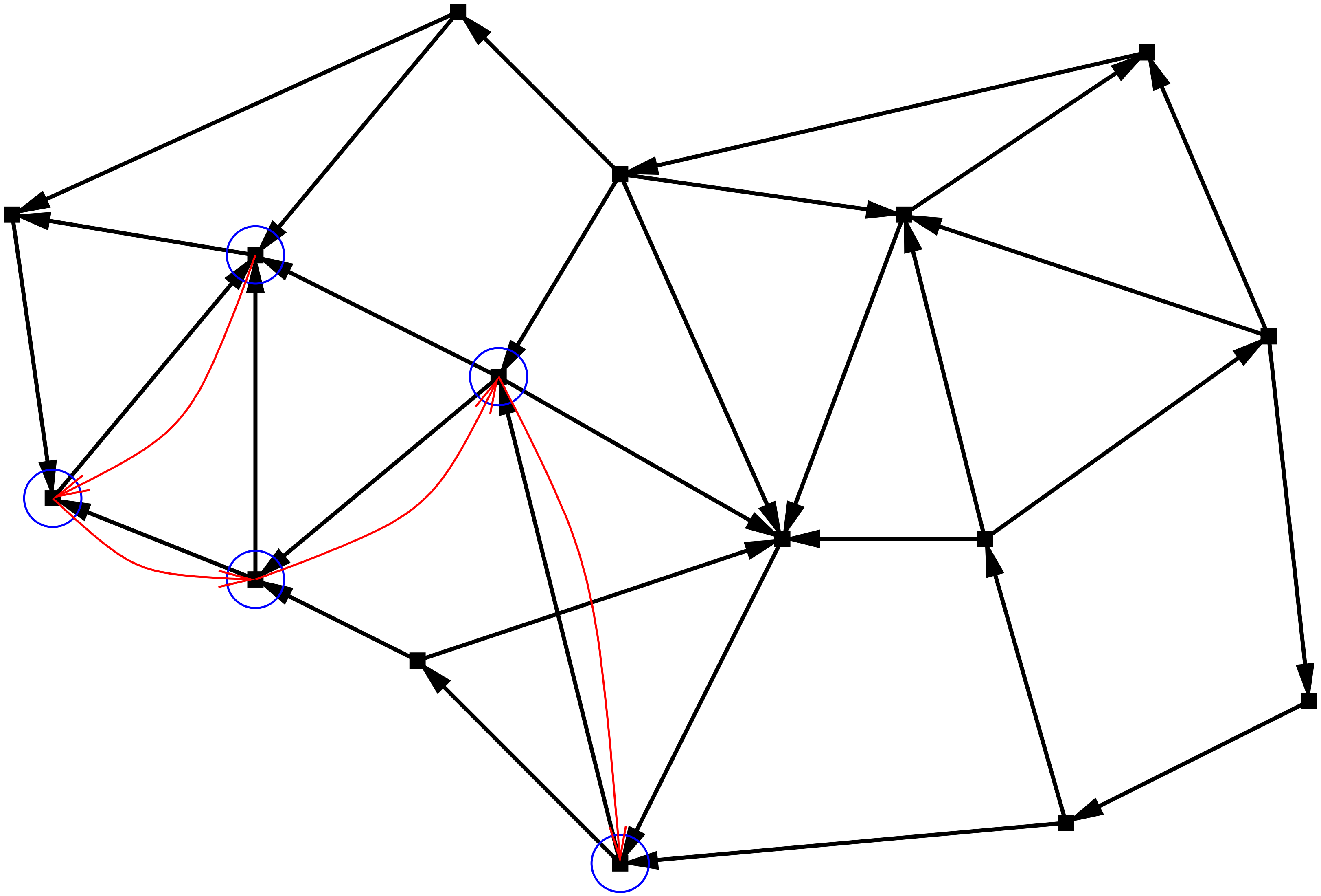
$v.\textit{läbitud} :=$ true
**for all** $w \in Gv$ **do**
    **if** $\neg w.\textit{läbitud}$ **then** külasta($w$)
$aeg := aeg + 1;\ v.jrknr := aeg$

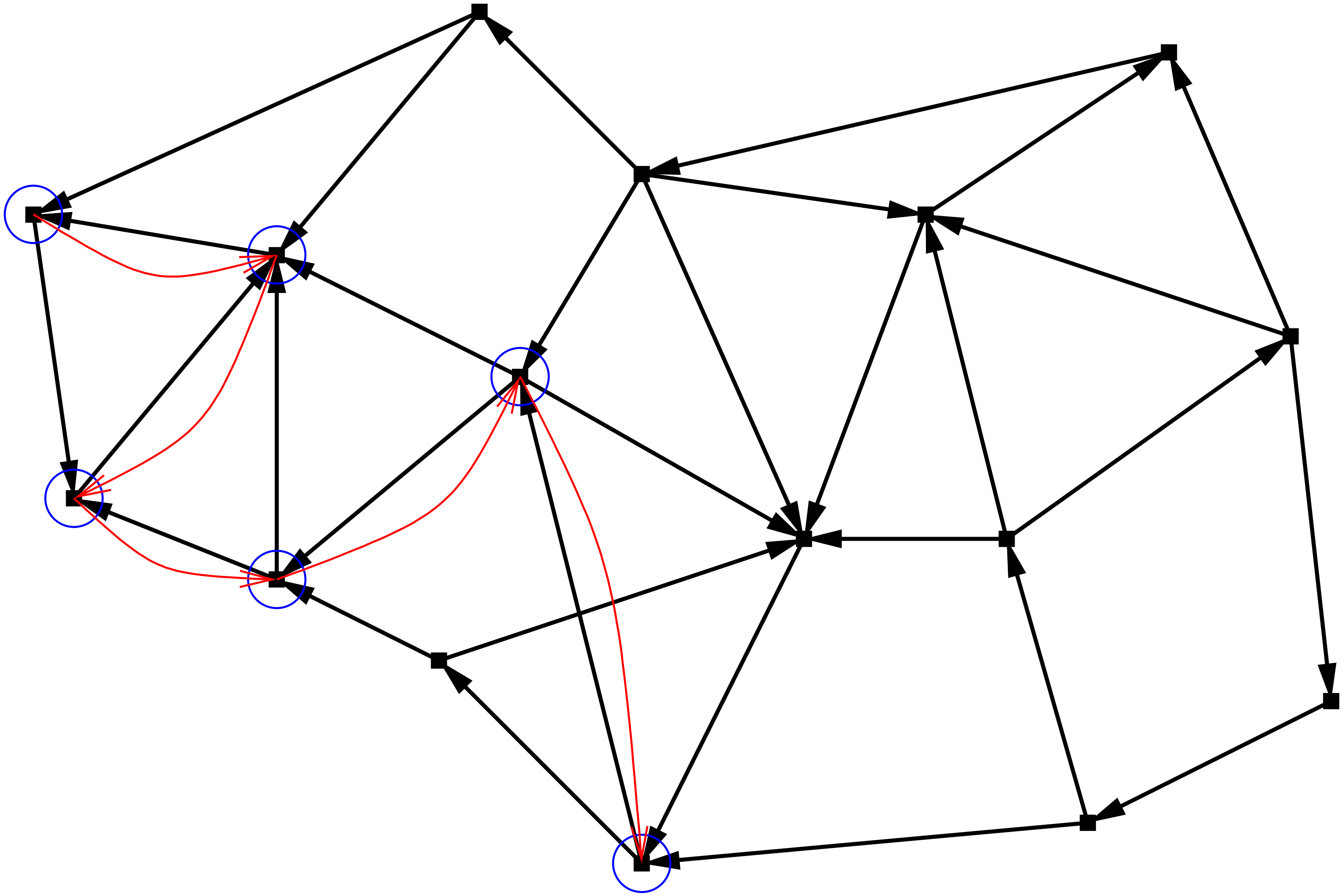


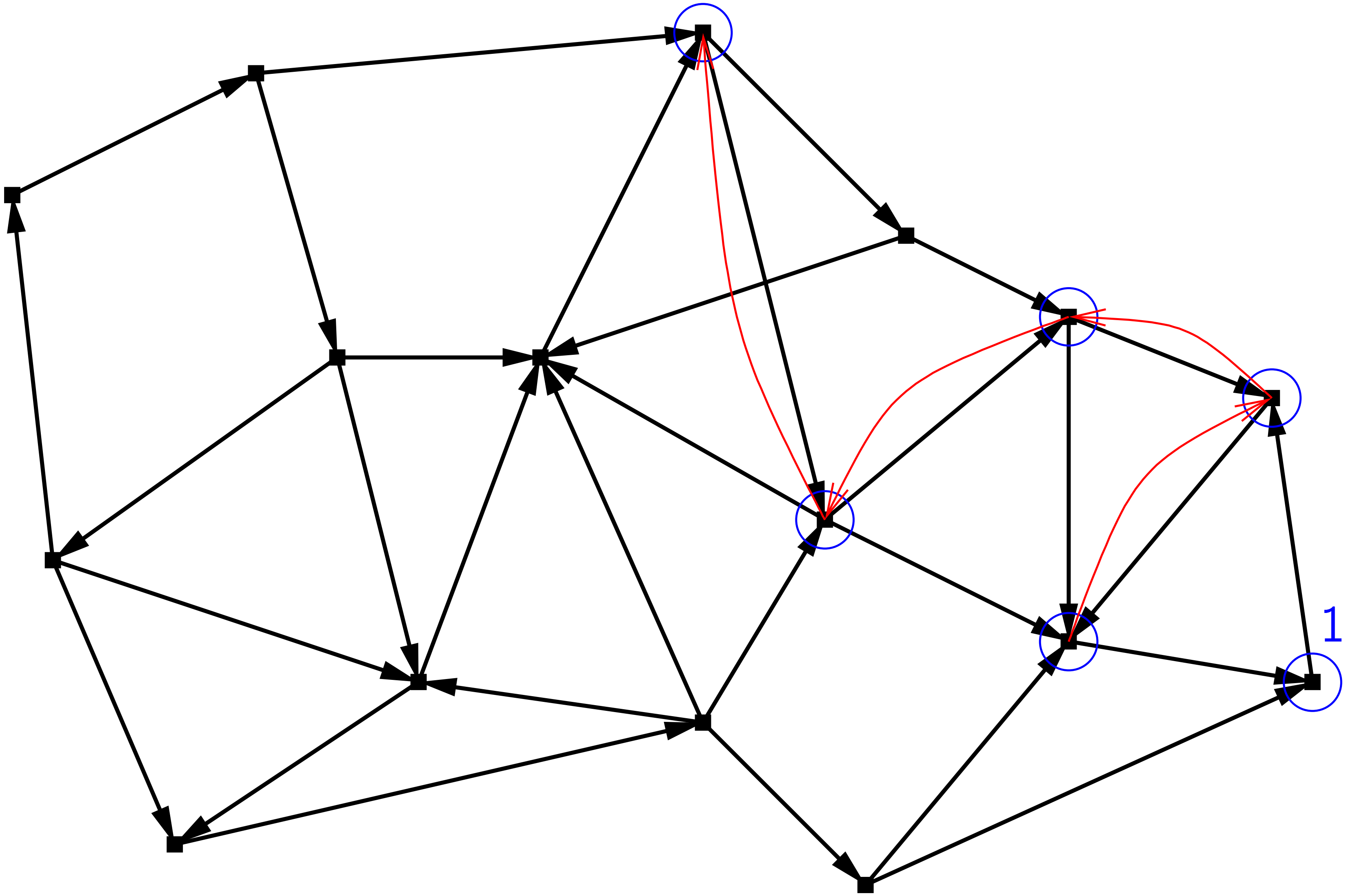


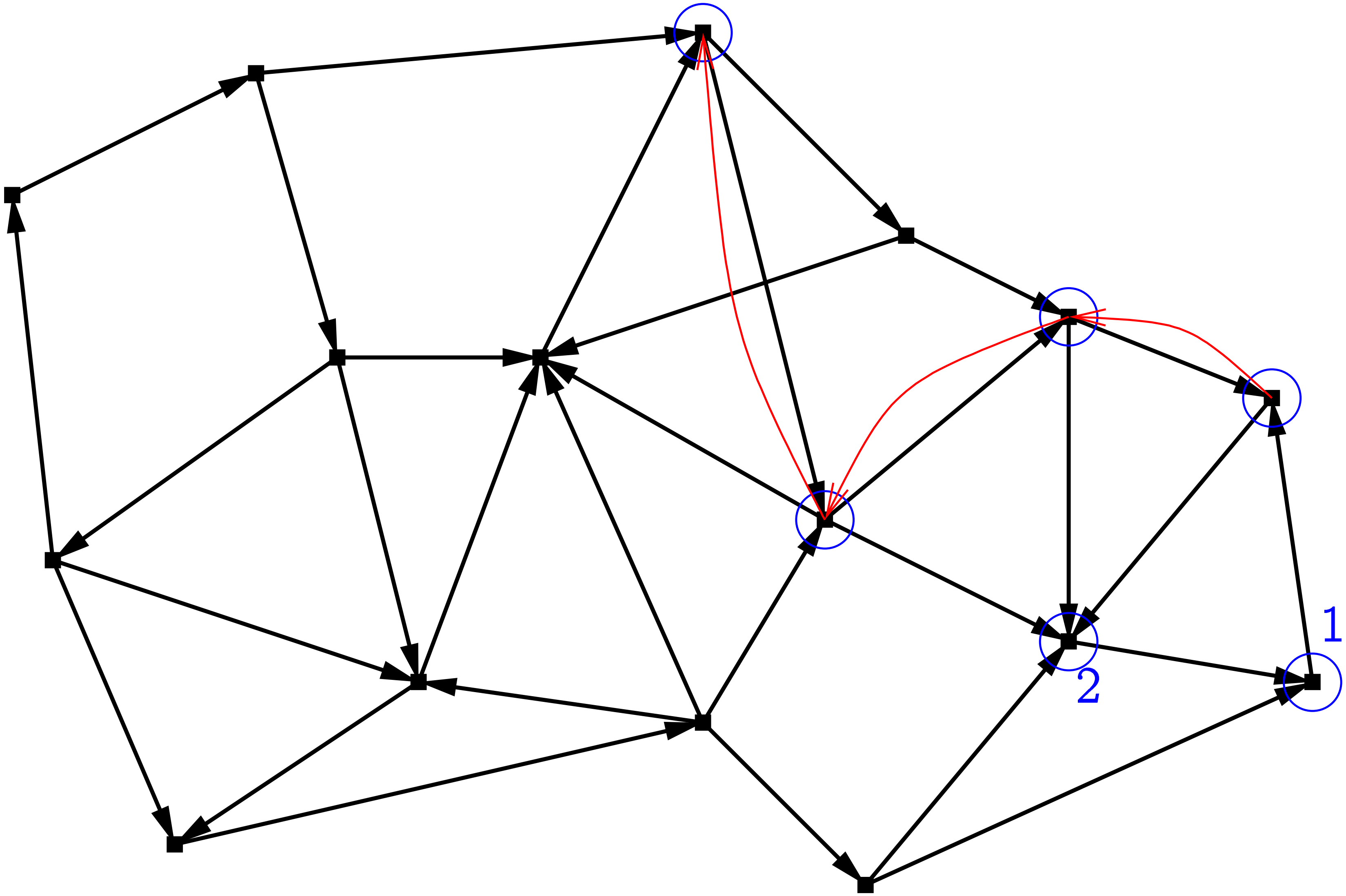


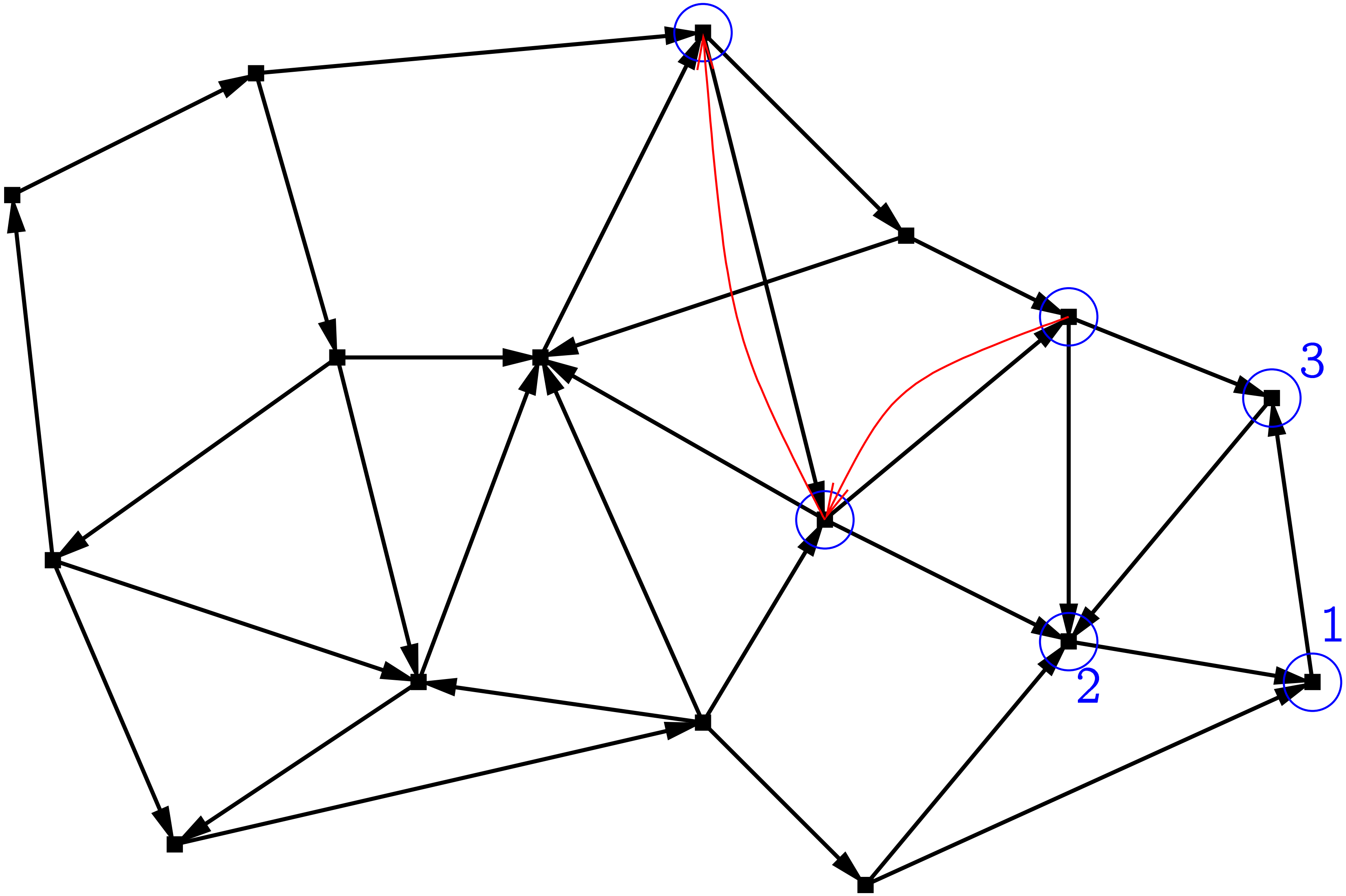


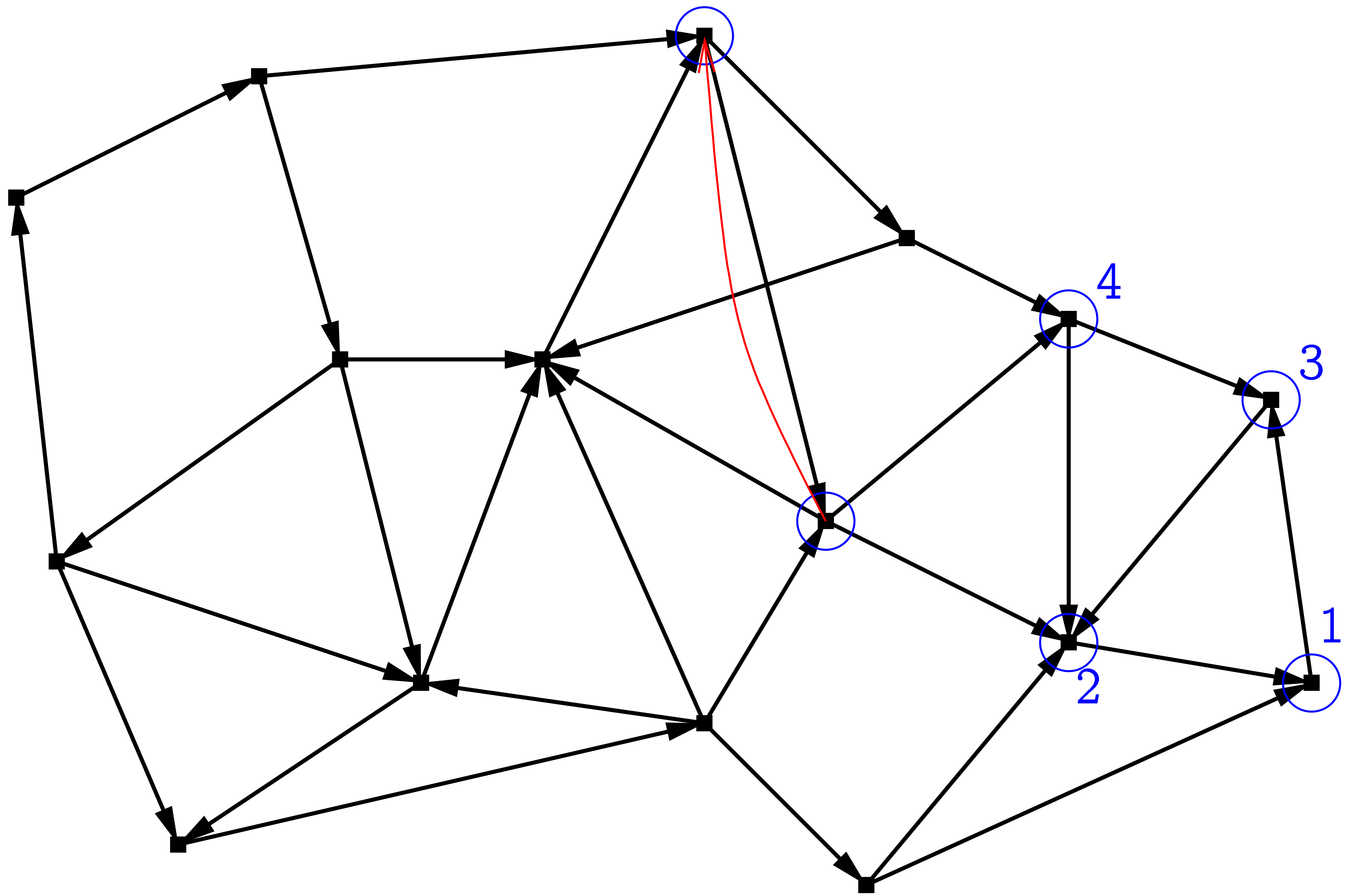


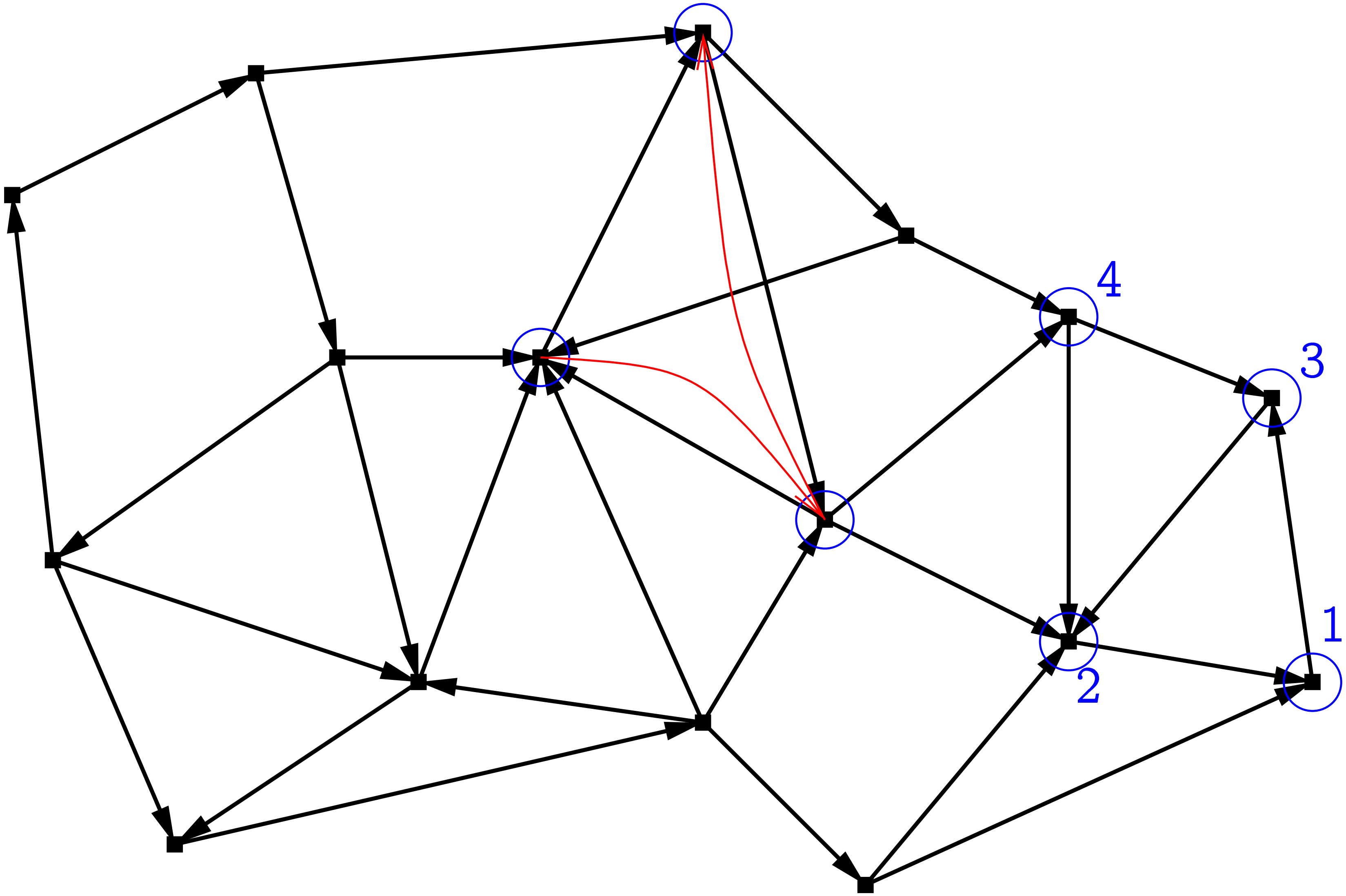


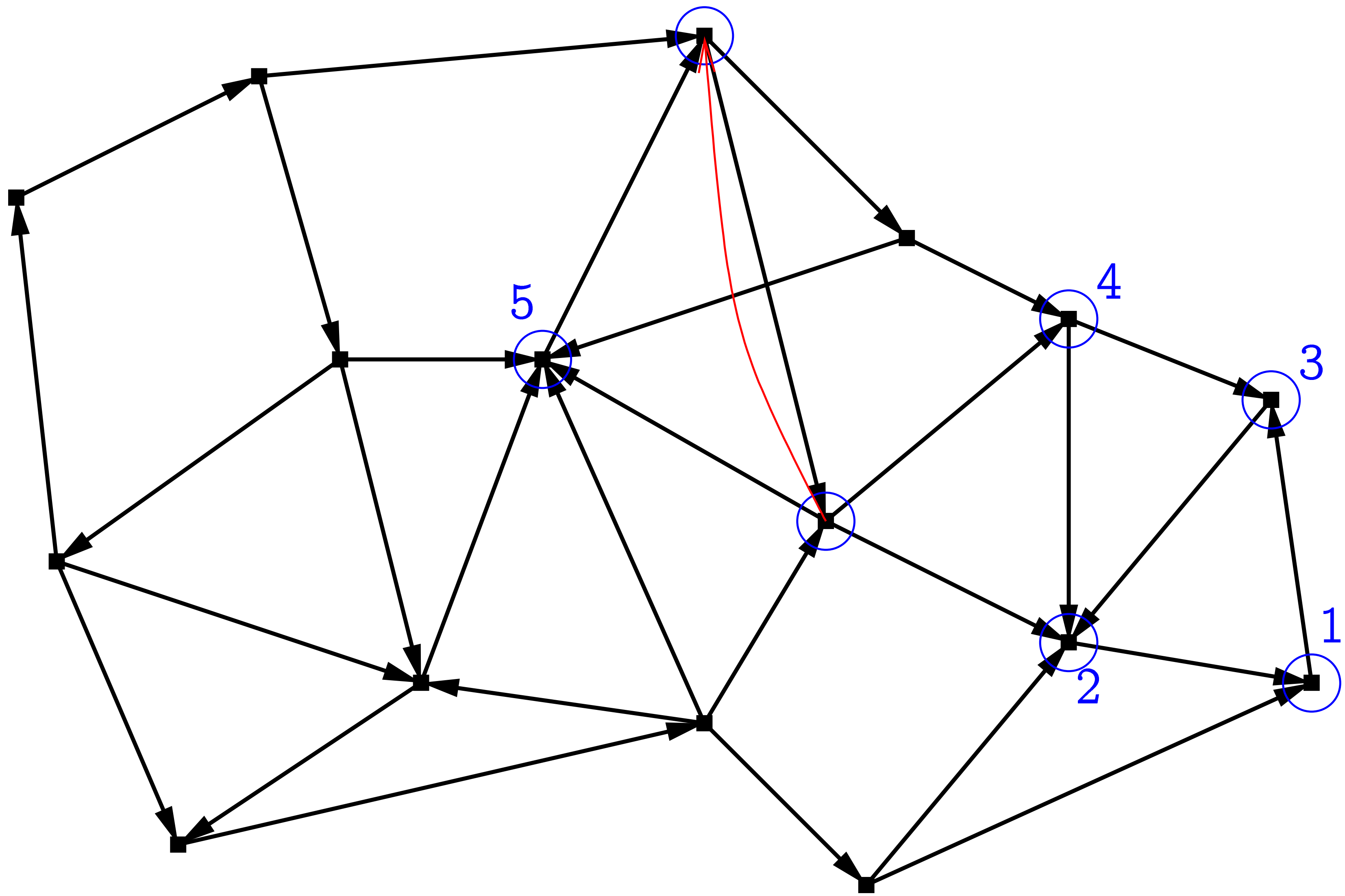

1

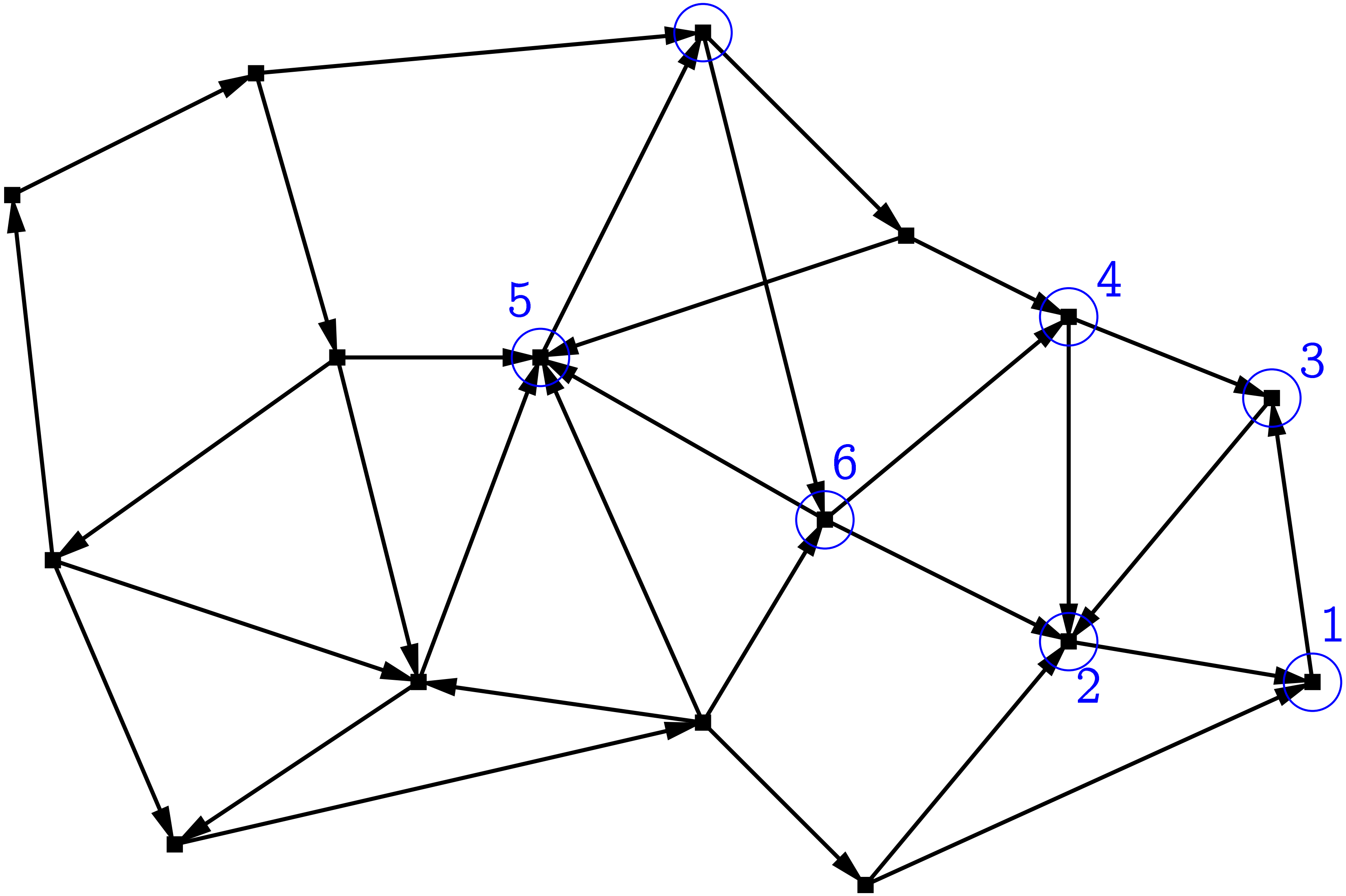

1
2


3
1
2

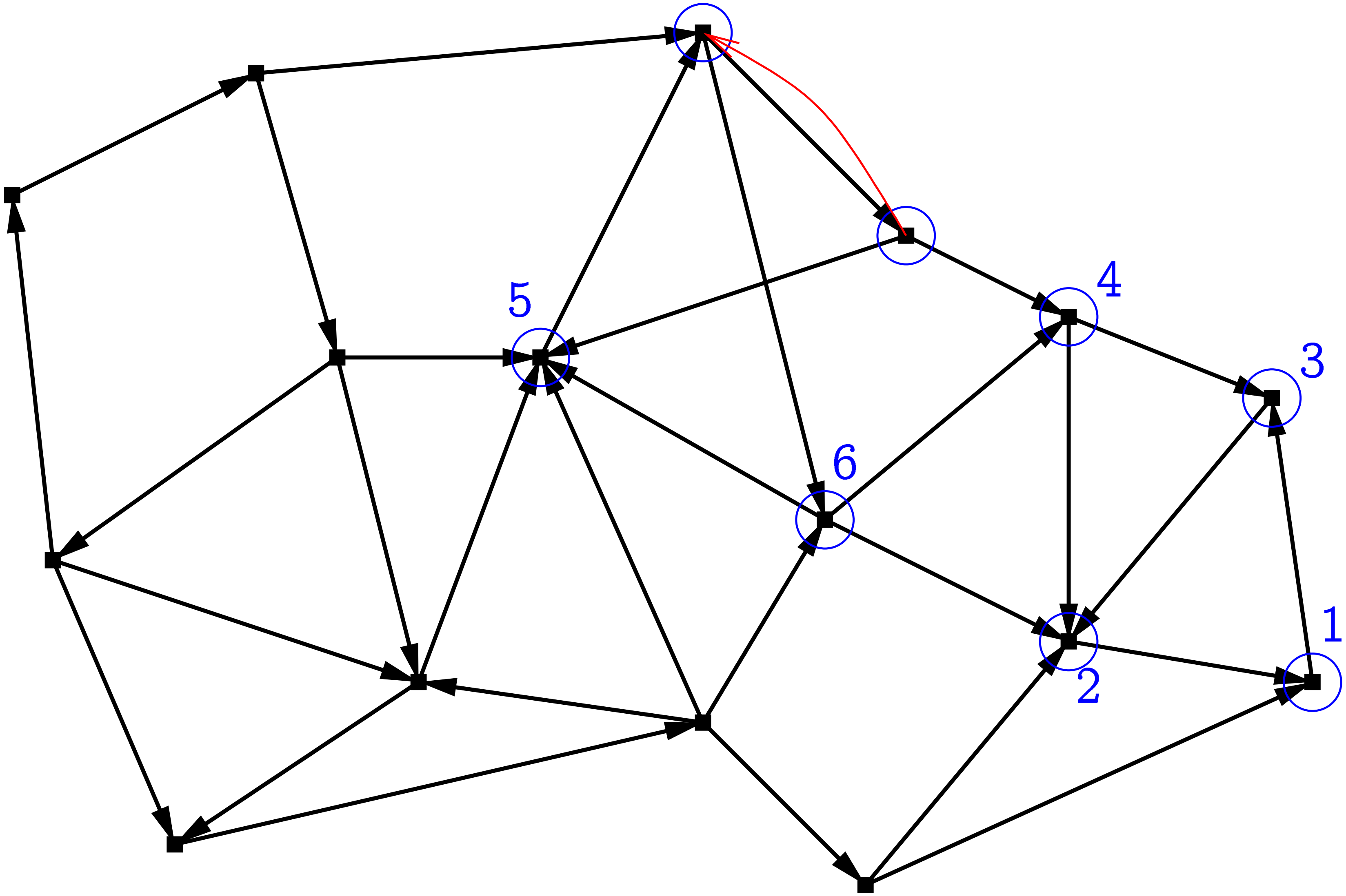

4
3
1
2

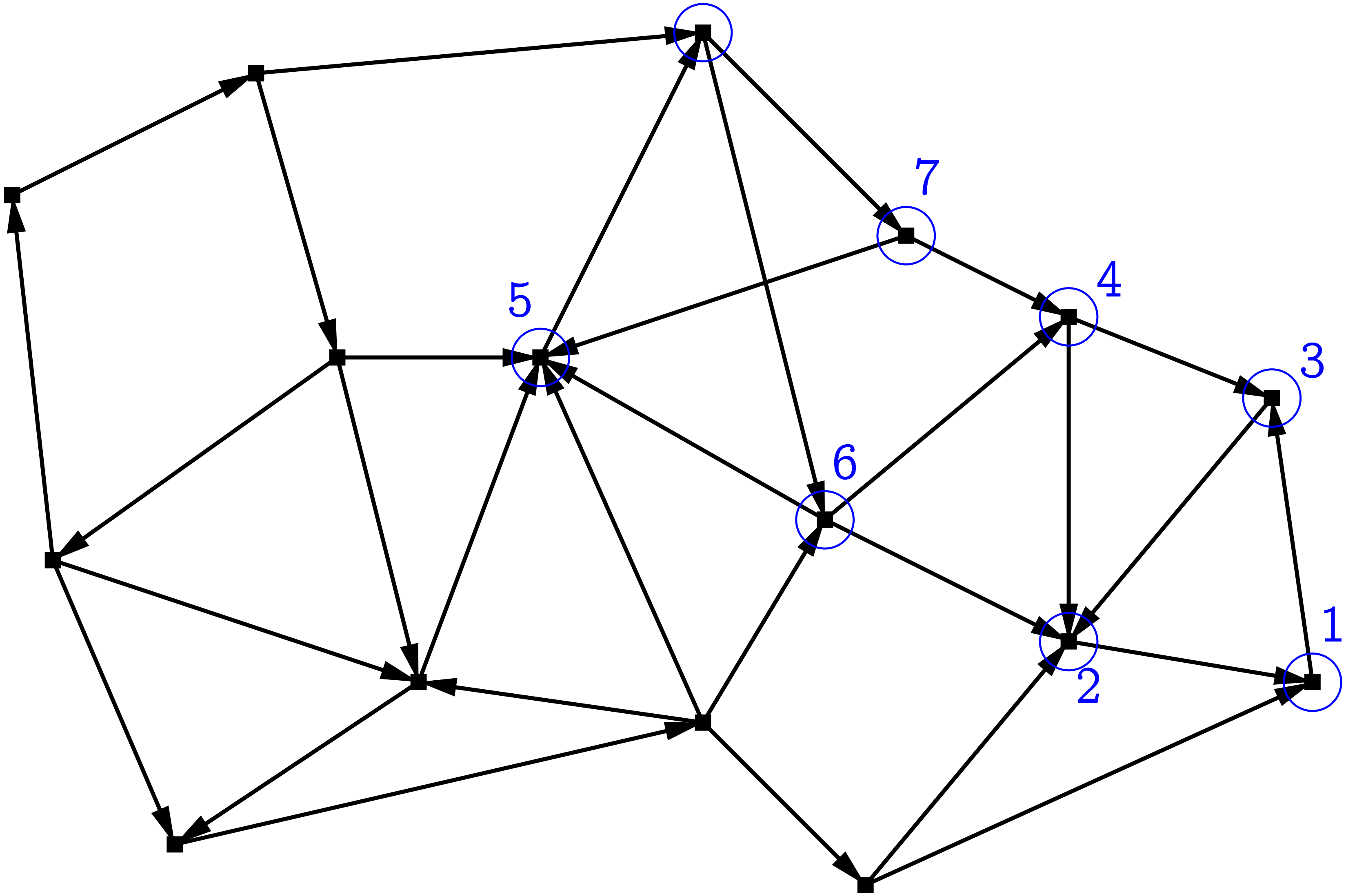

4
3
1
2

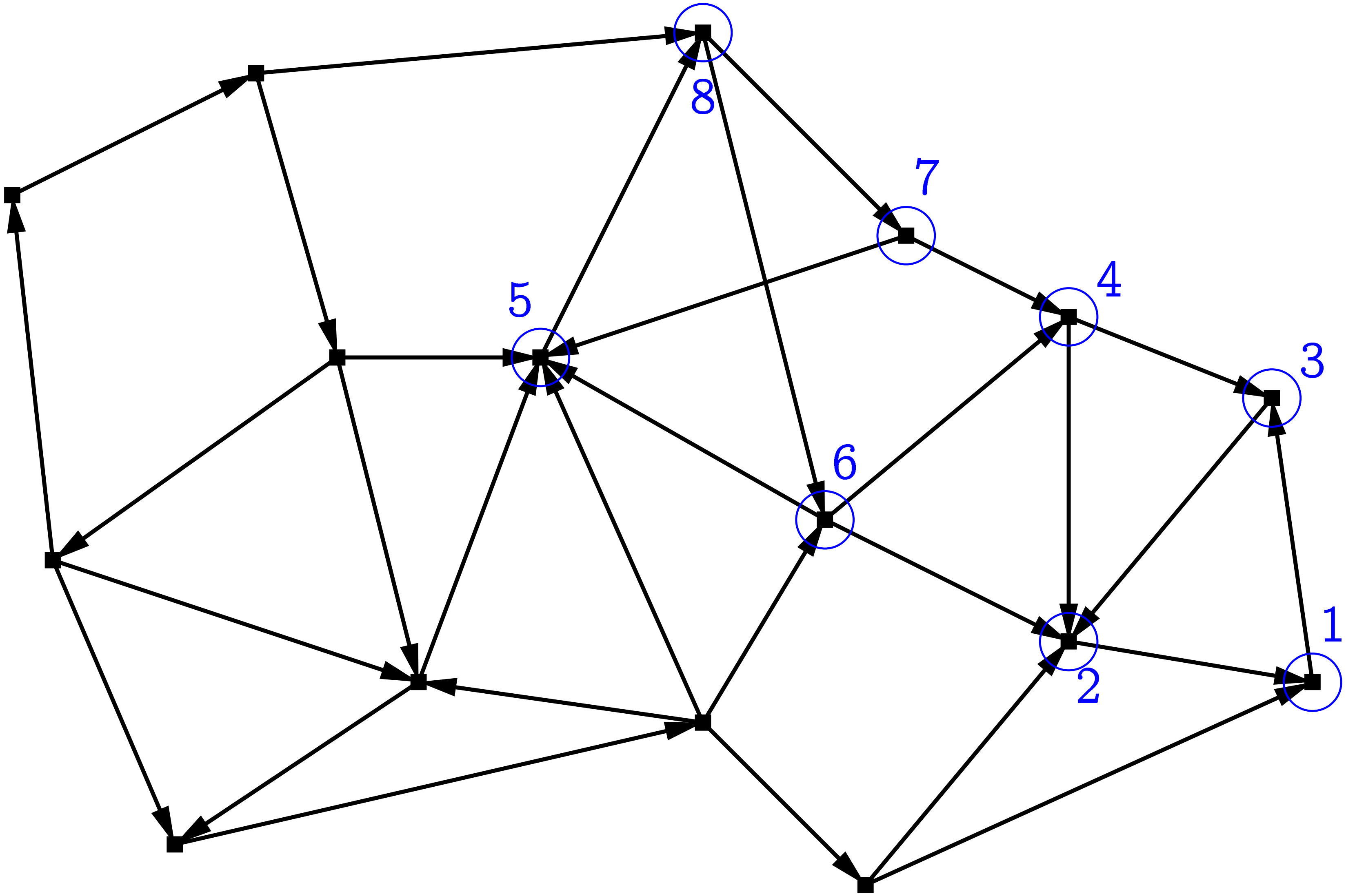

5
4
3
1
2

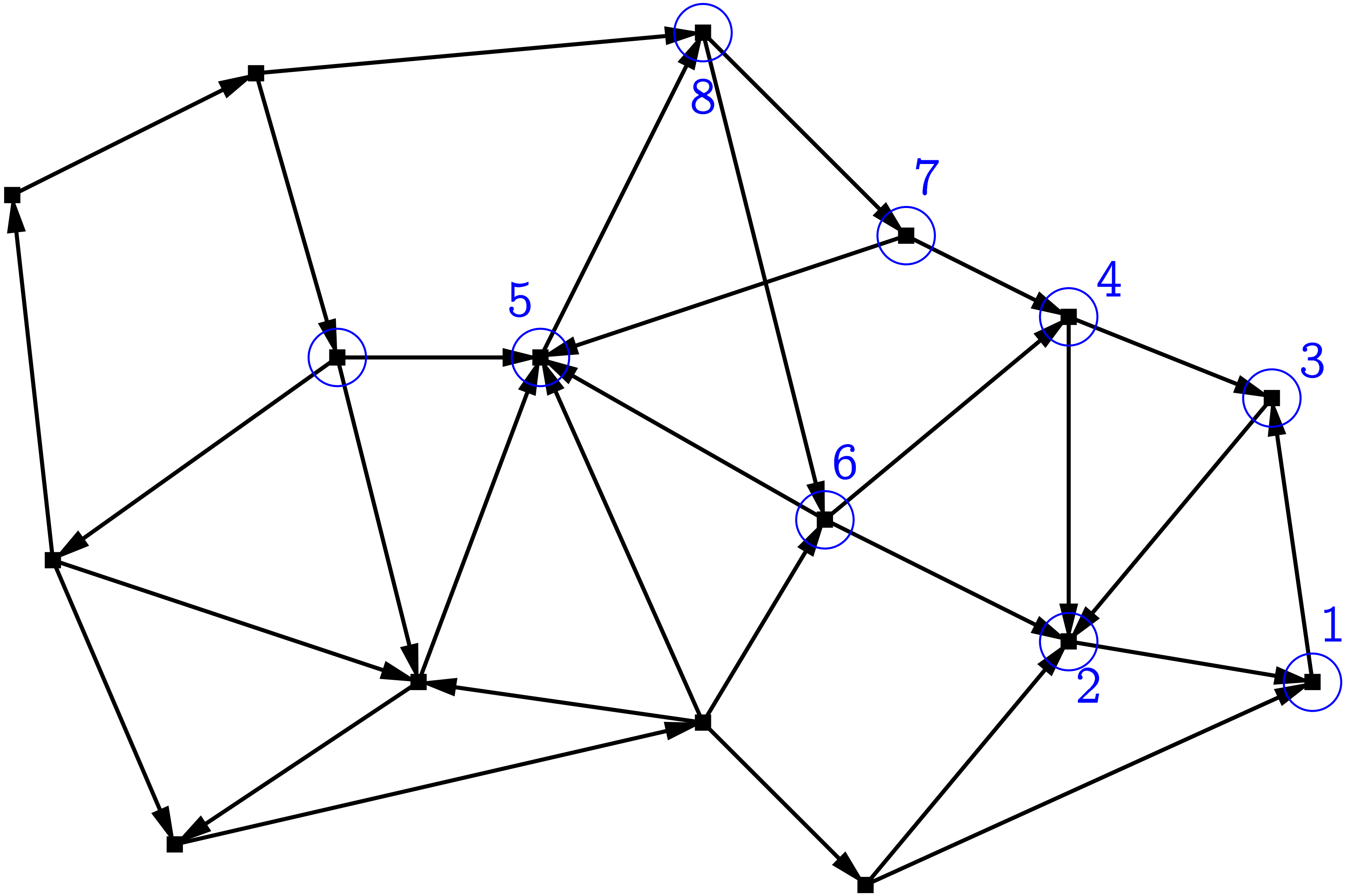


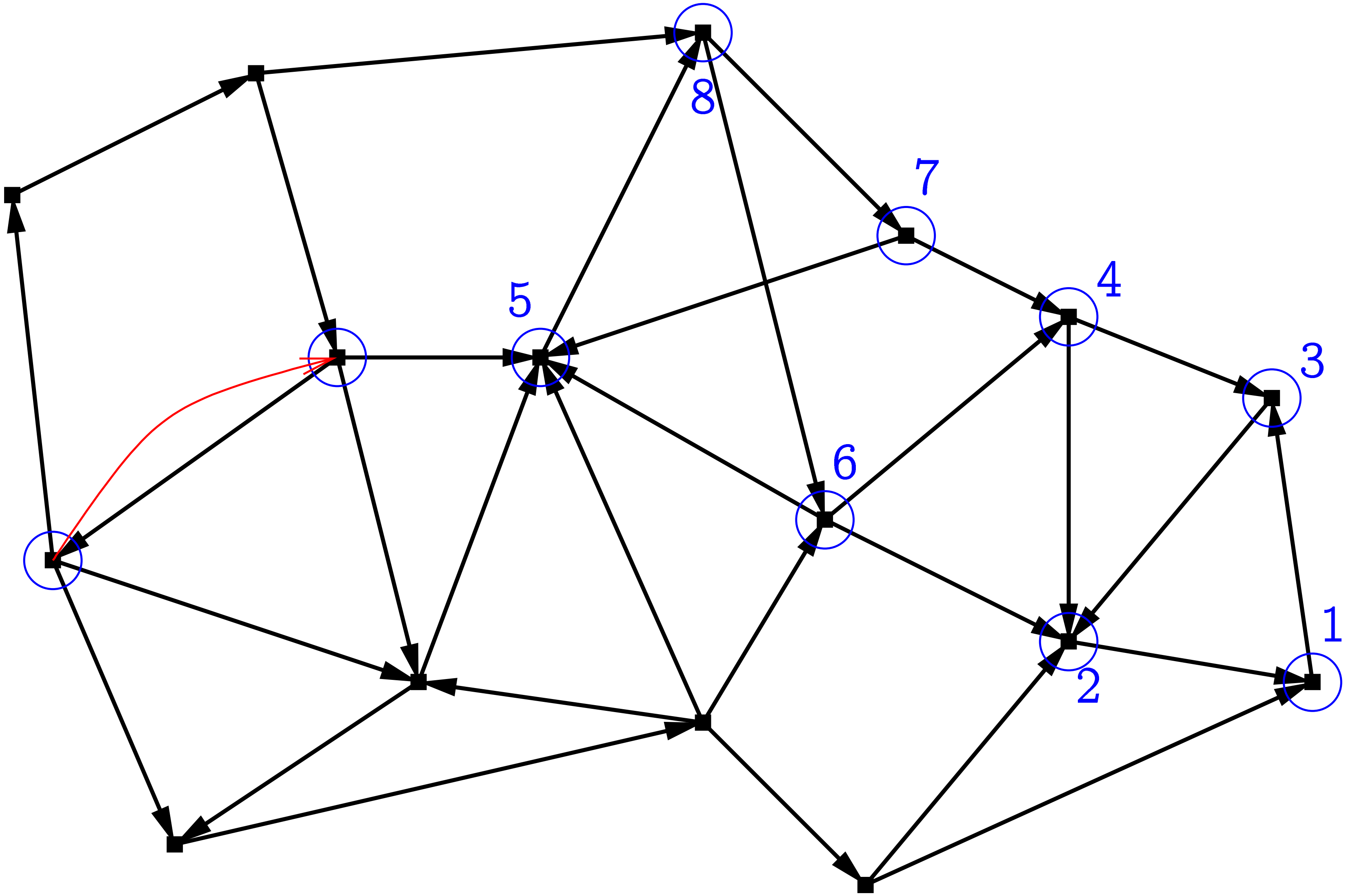
5
4
3
6
1
2


5
4
3
6
1
2

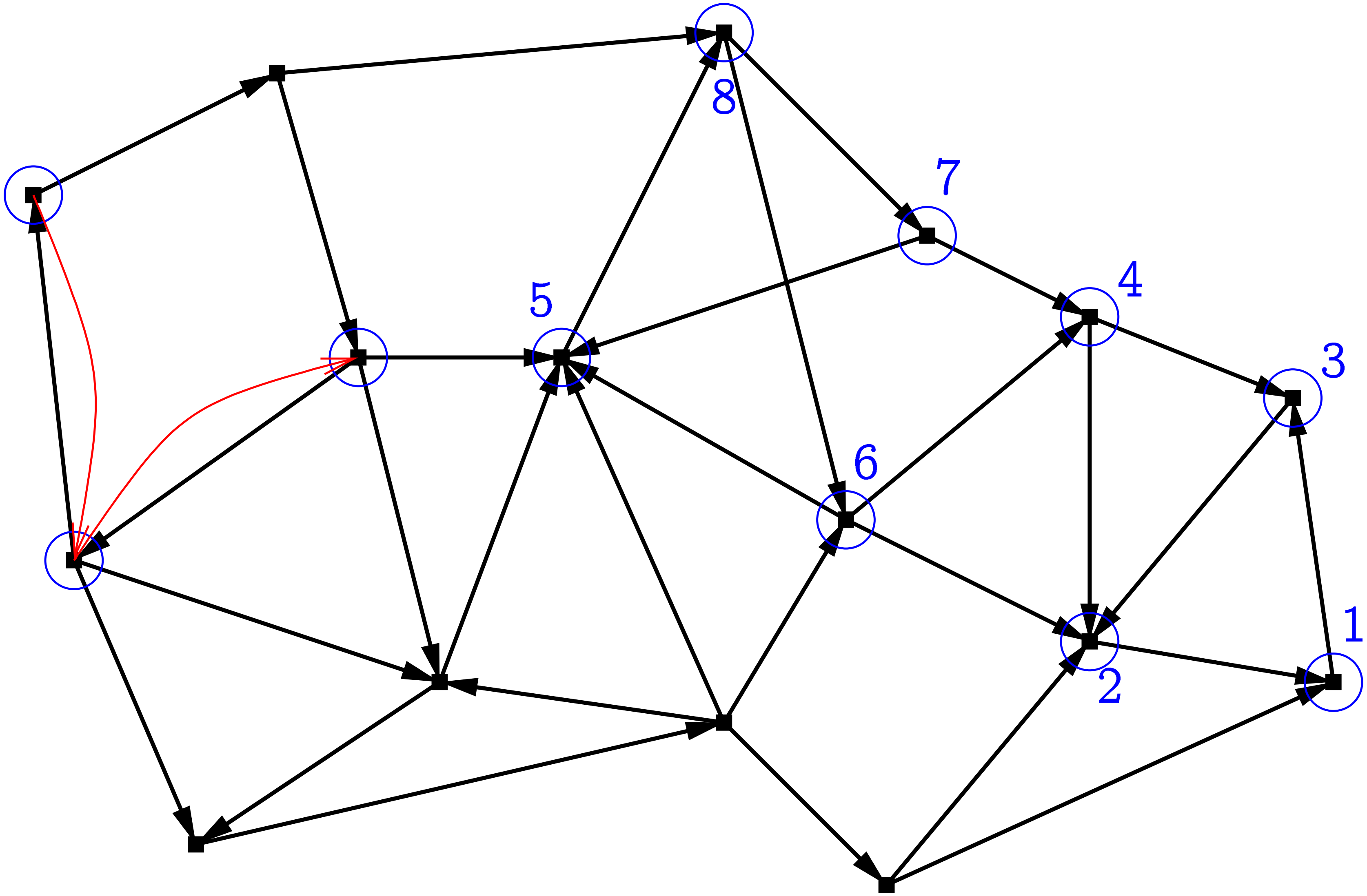

7
4
5
3
6
1
2

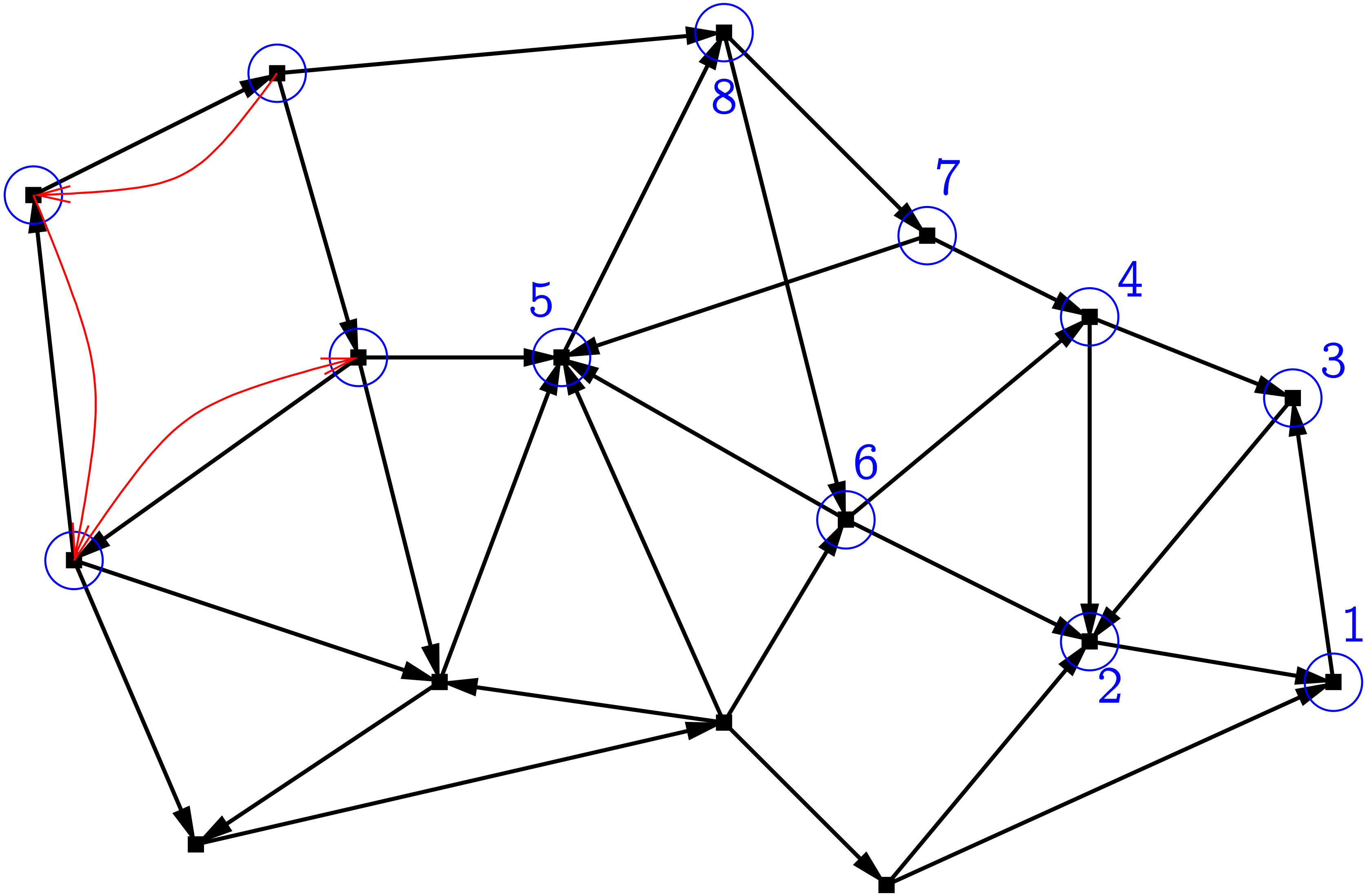

8
7
4
5
3
6
1
2

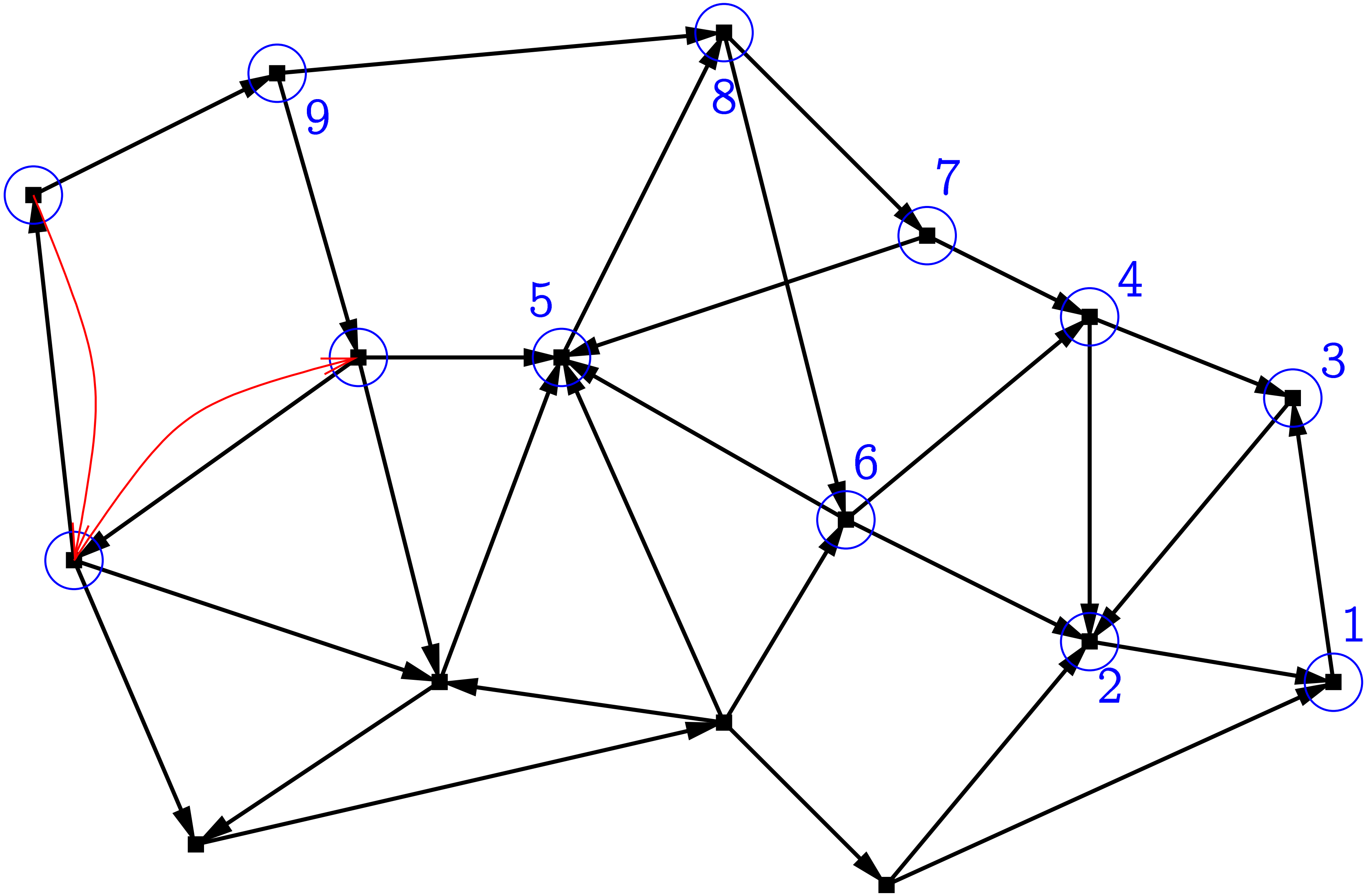

8
7
4
5
3
6
1
2

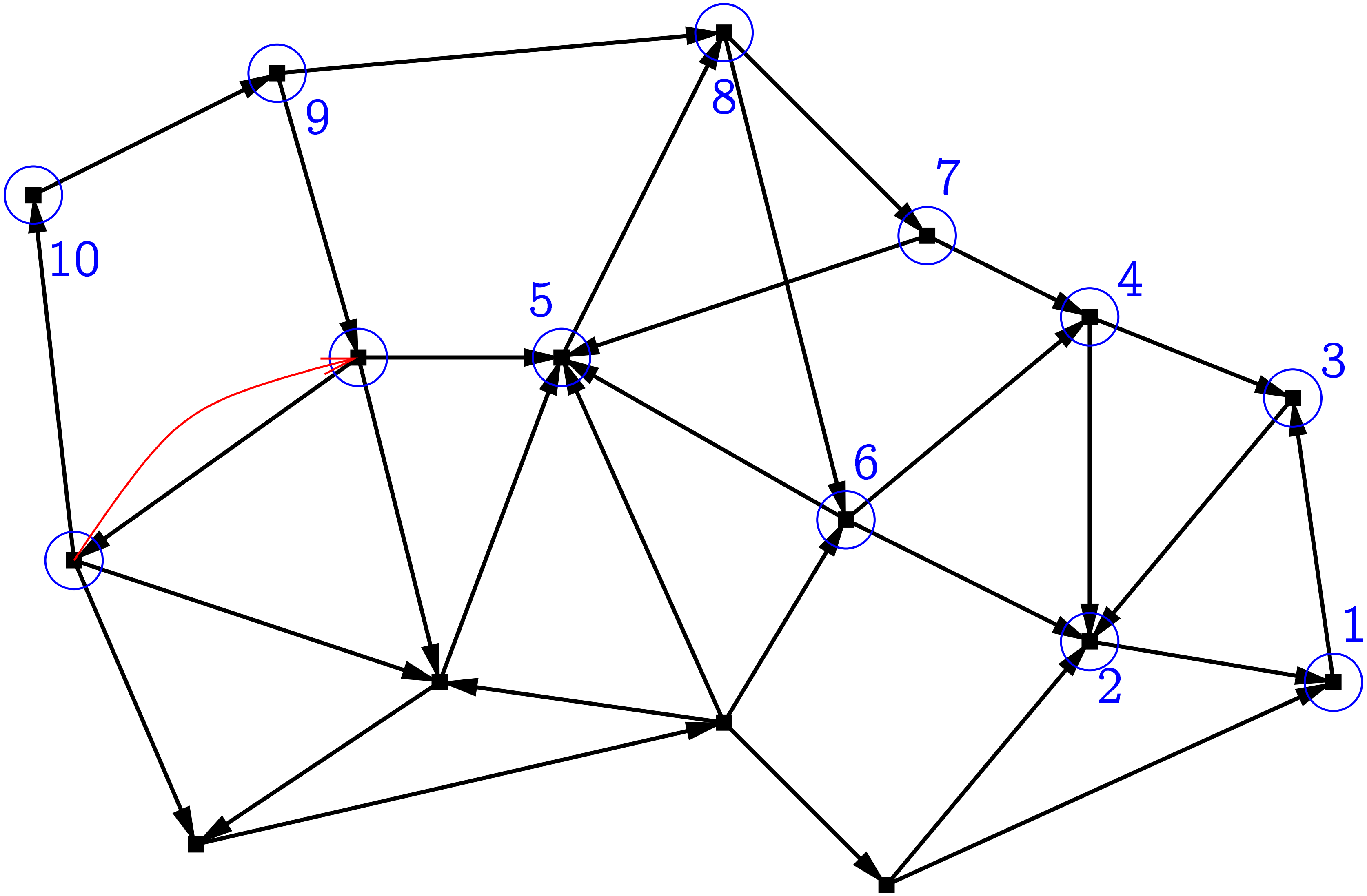


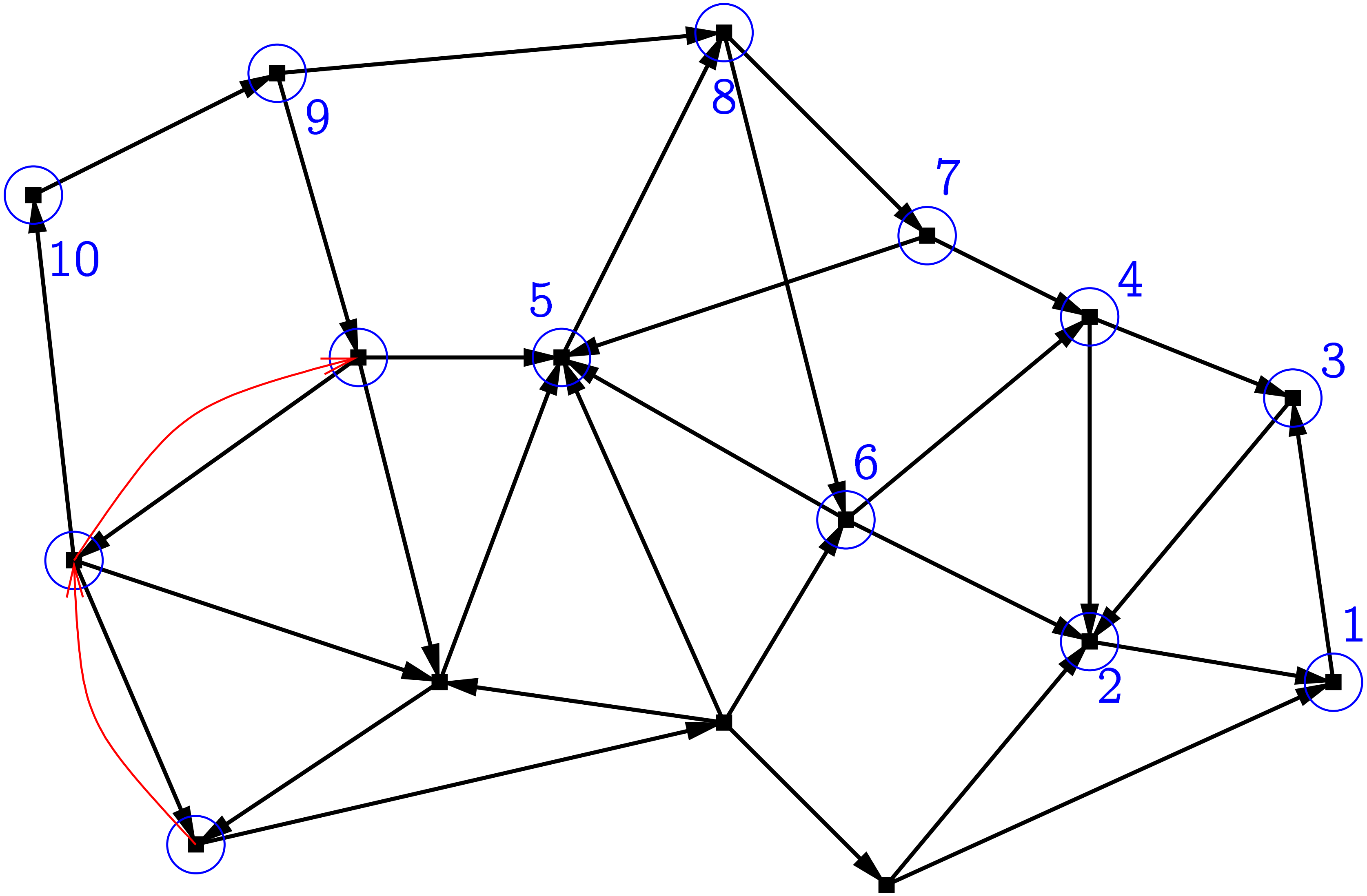
8
7
5
4
3
6
1
2


8
7
5
4
3
6
1
2

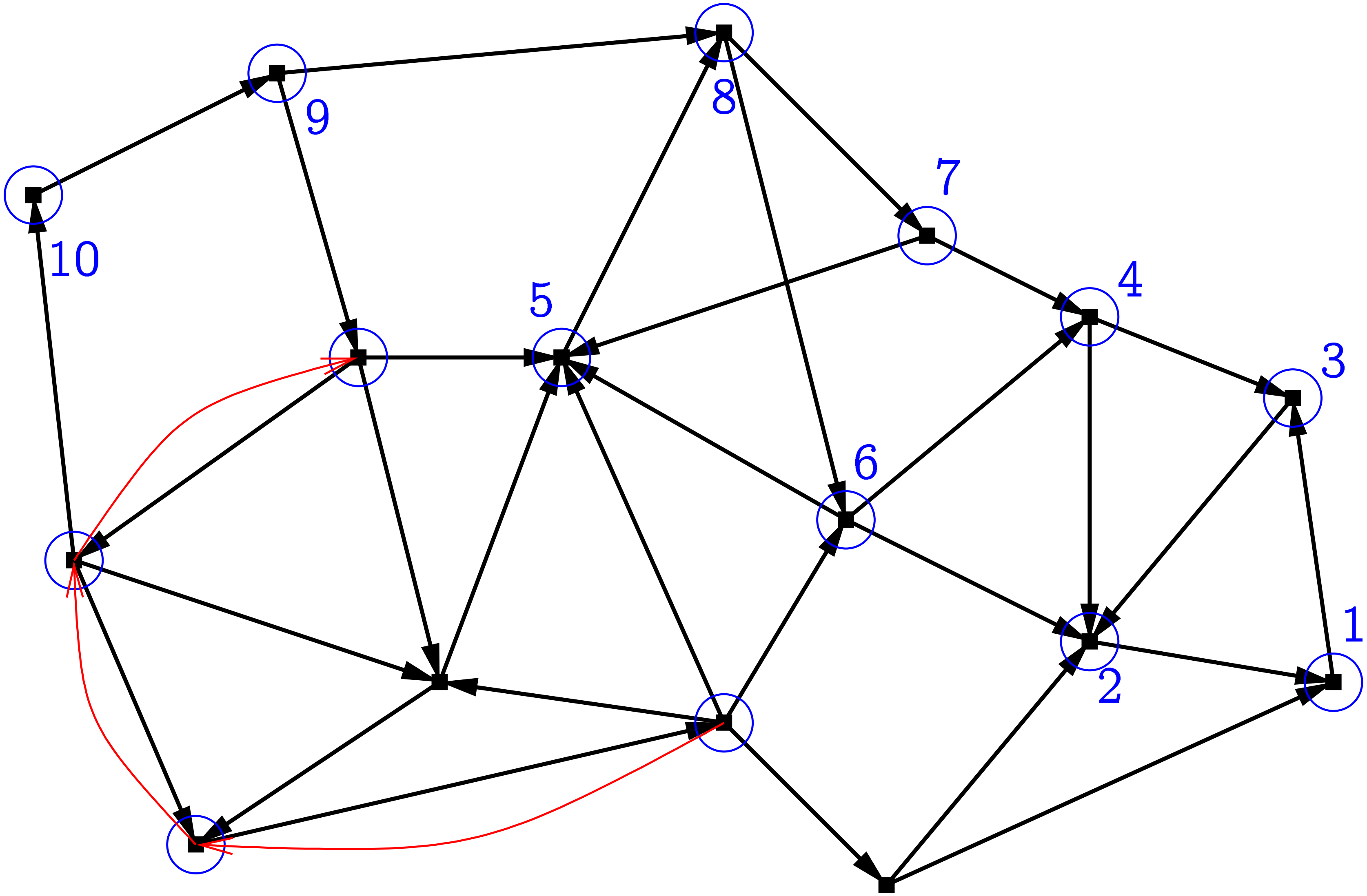

8
7
4
5
3
6
1
2

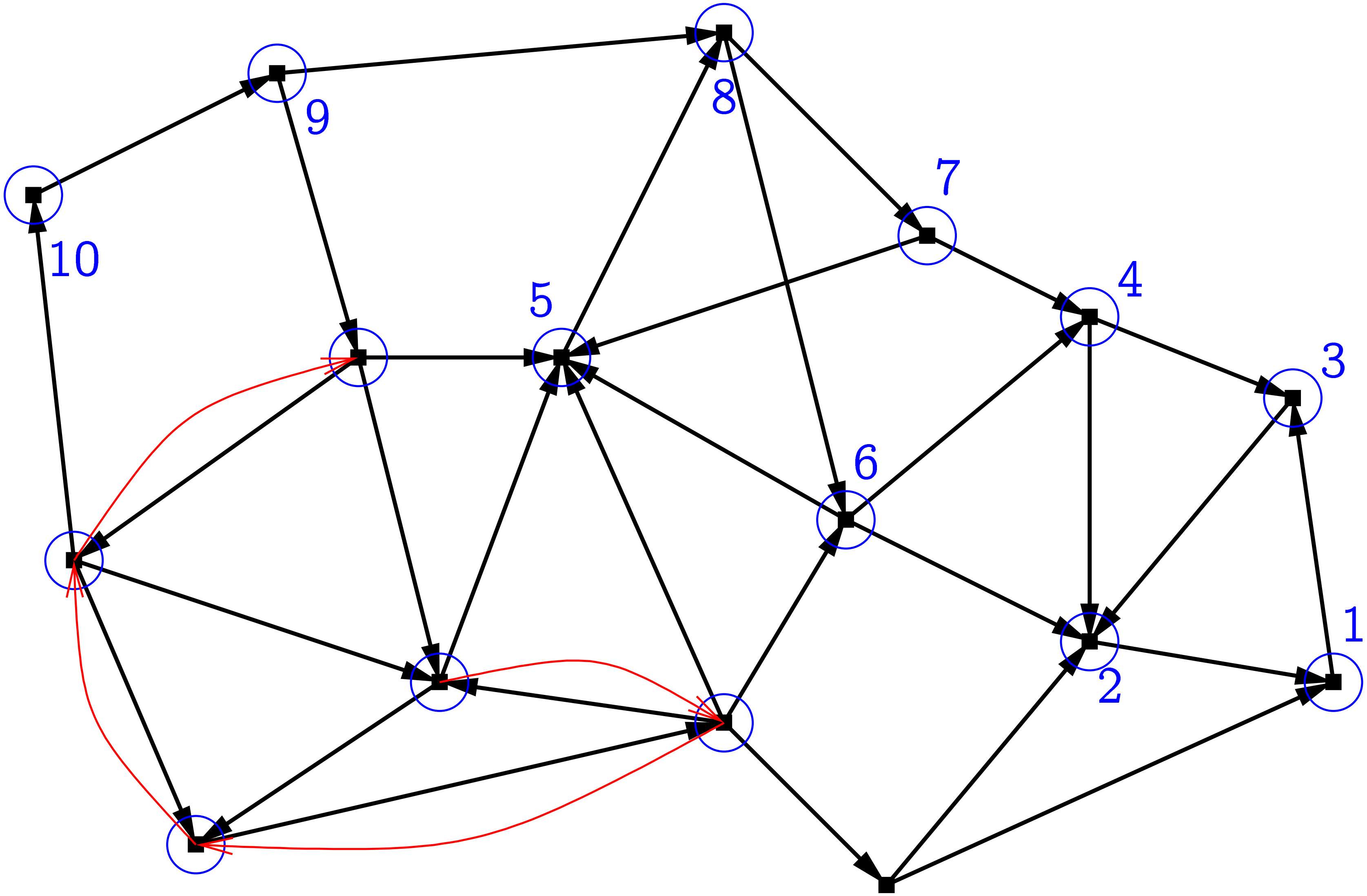

9
8
7
5
4
3
6
1
2


9
8
7
10
5
4
3
6
1
2

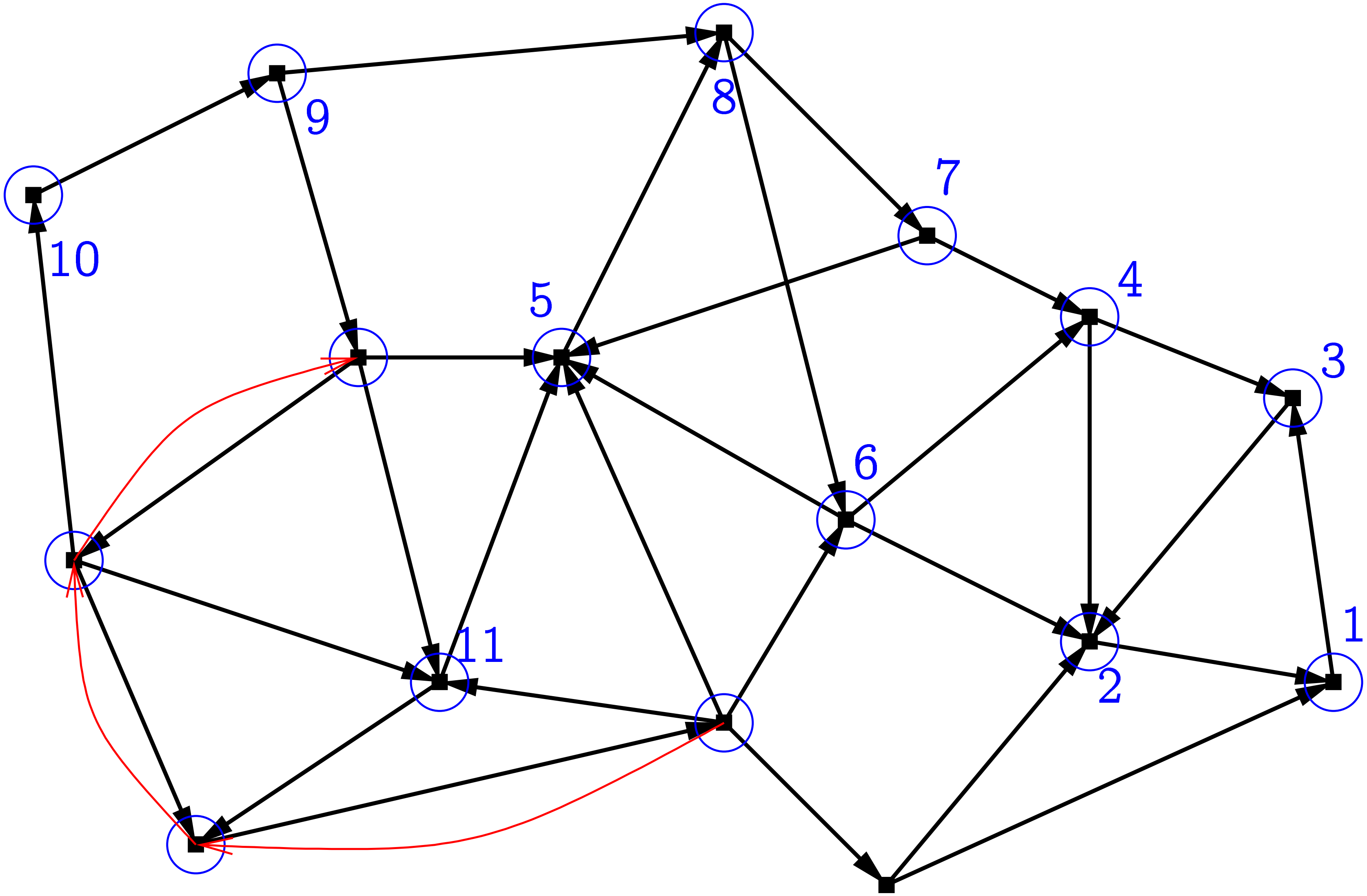


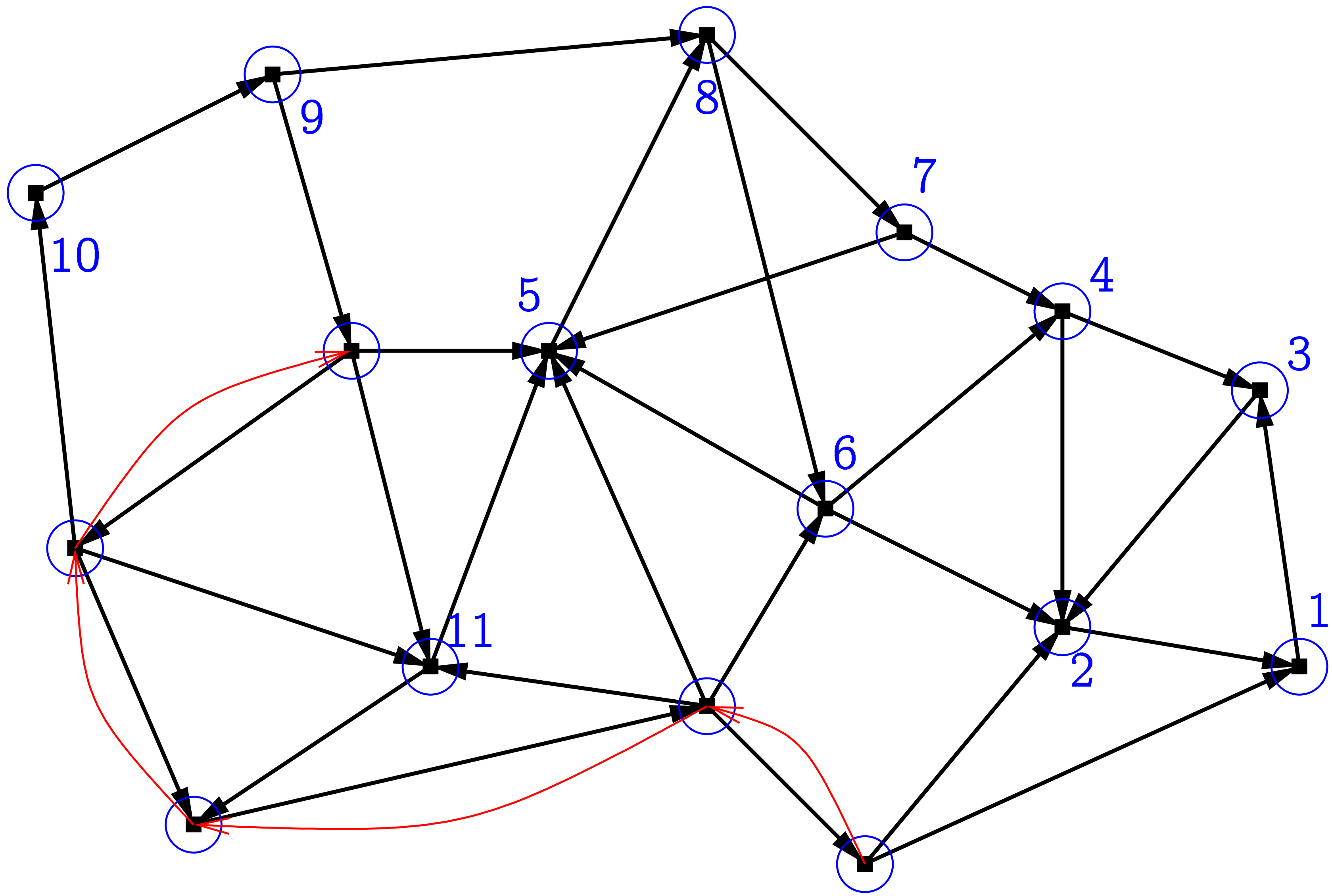
8
9
7
10
5
4
3
6
1
2


9
8
7
10
5
4
3
6
1
2

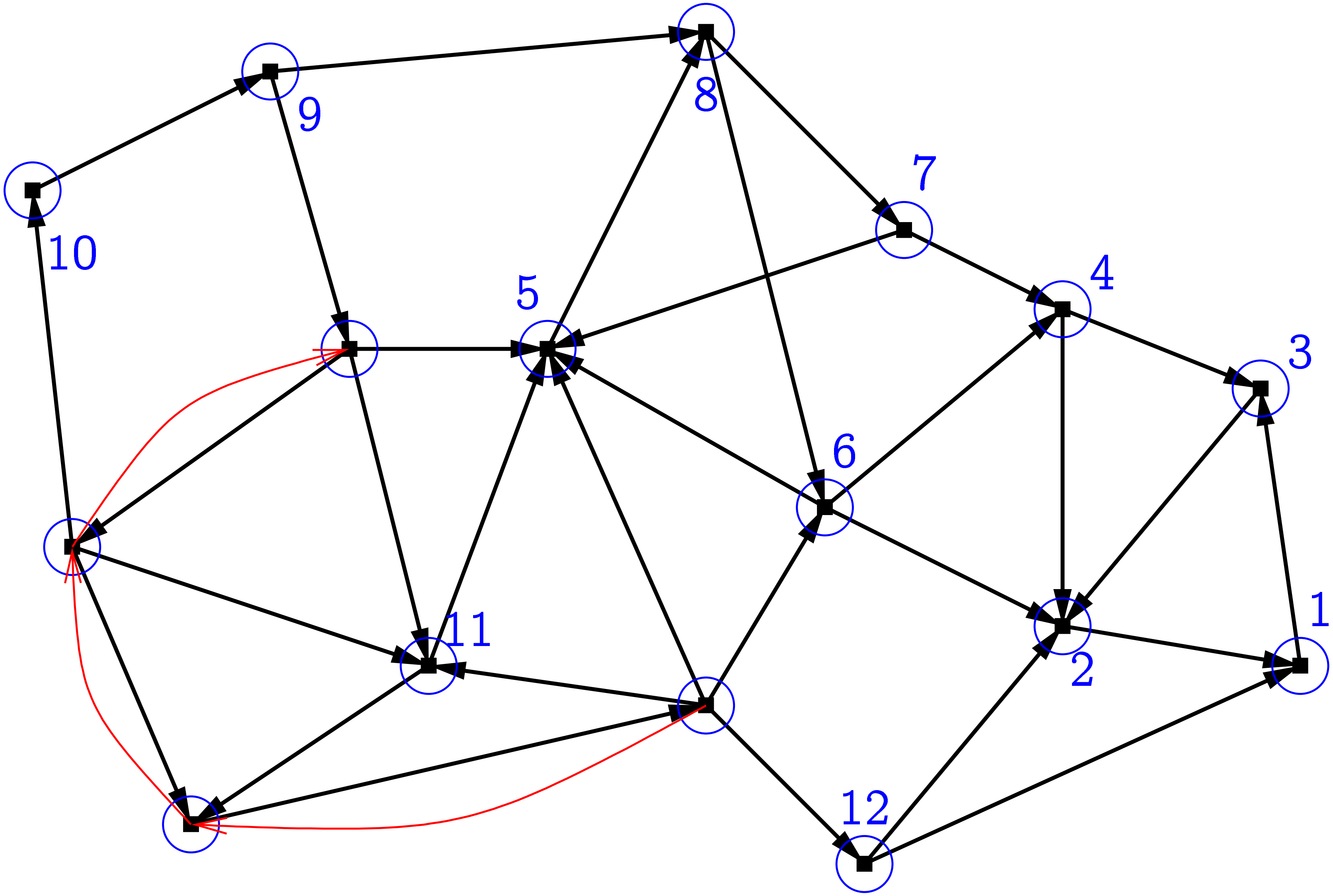


8
9
7
10
4
5
3
6
1
2

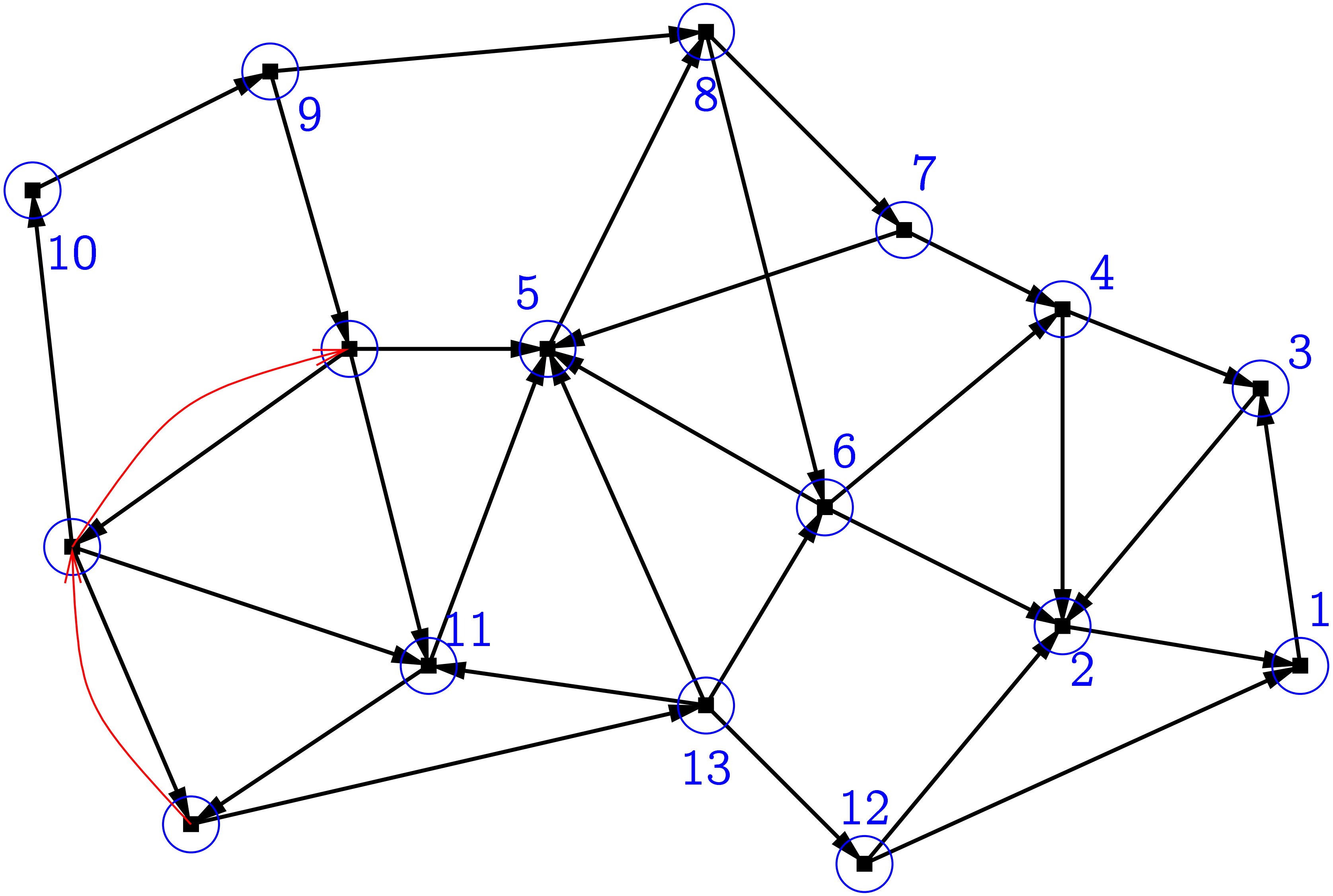

1
2
3
4
5
6
7
8
9
10
11

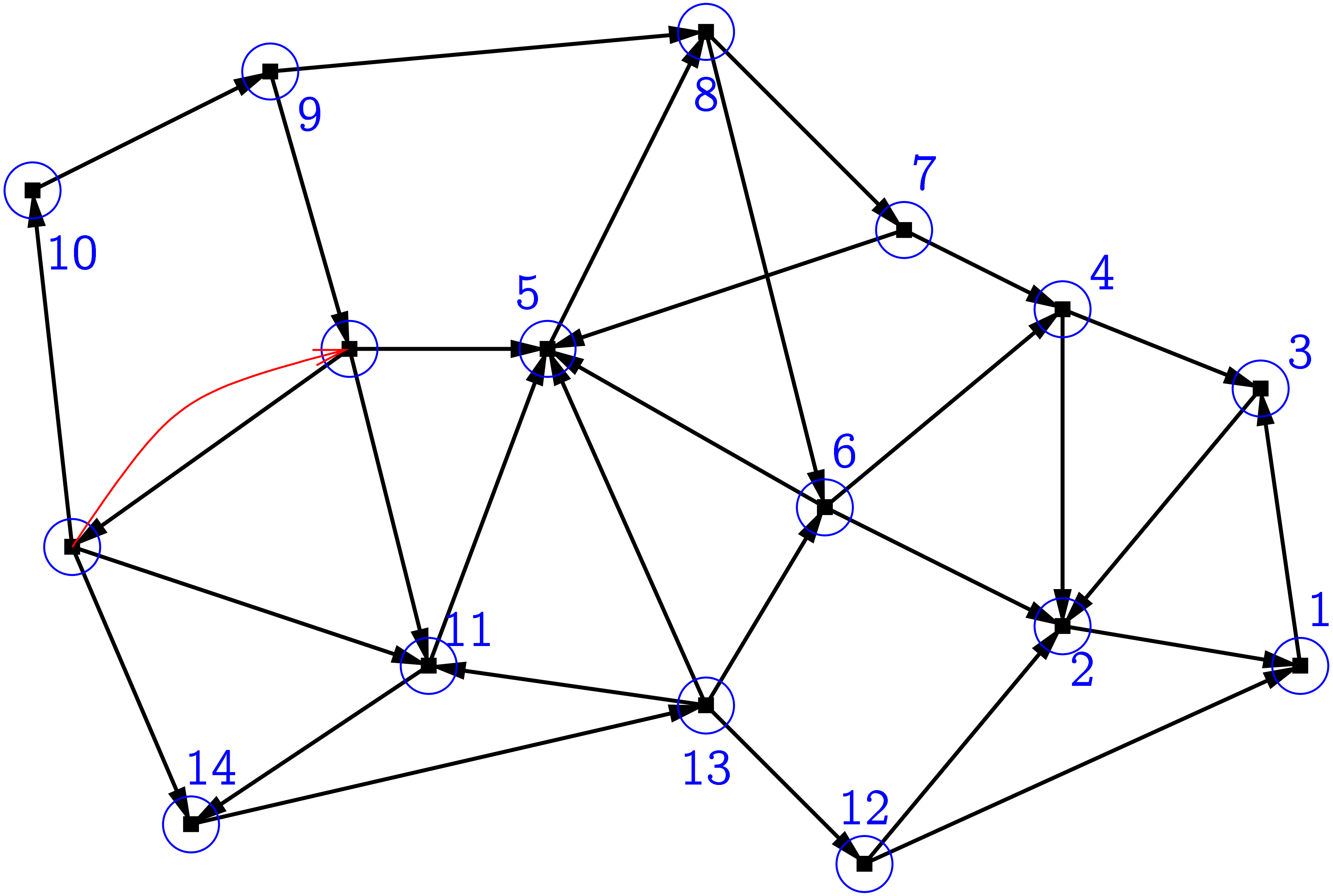


1
2
3
4
5
6
7
8
9
10
11

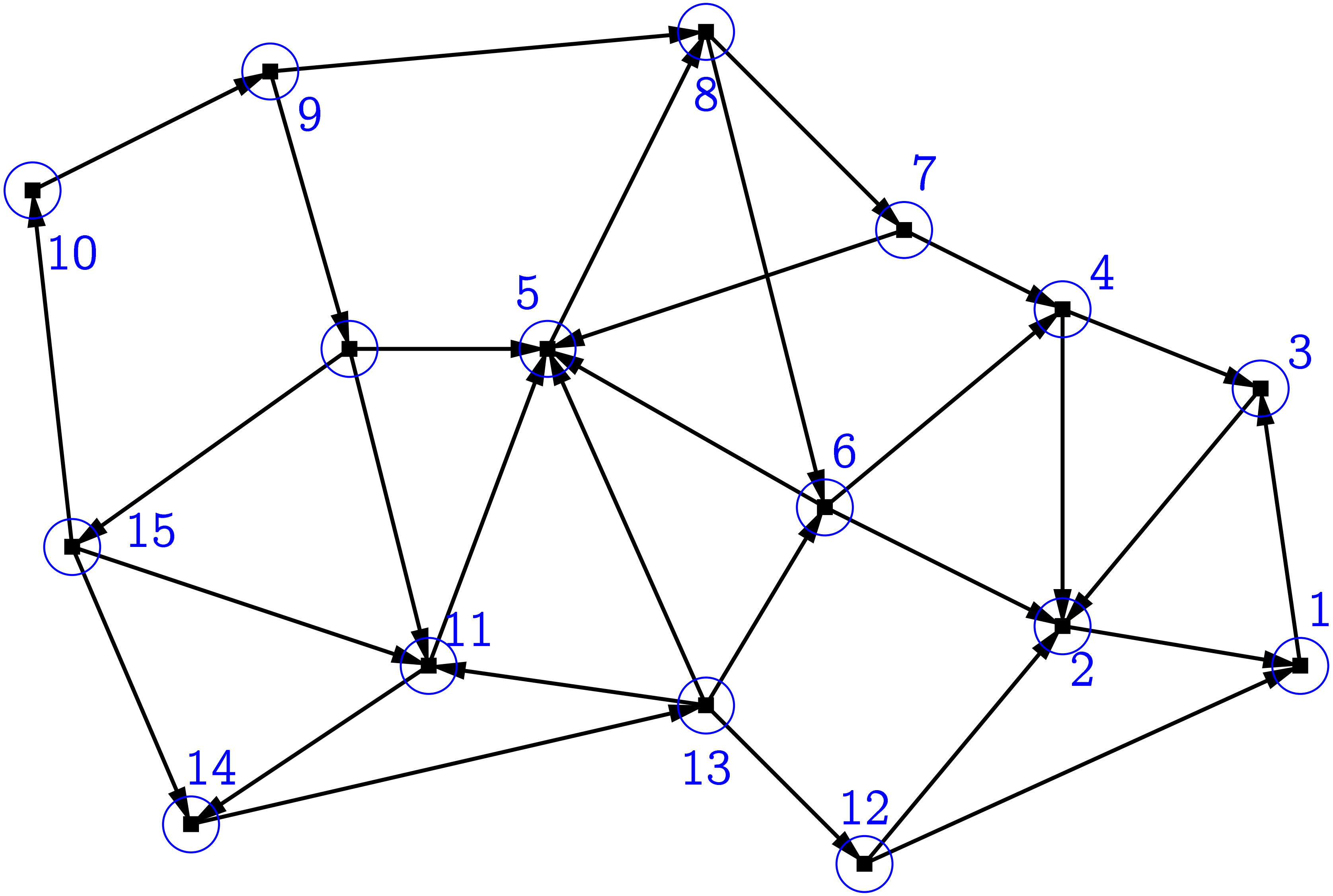


1
2
3
4
5
6
7
8
9
10
11
12

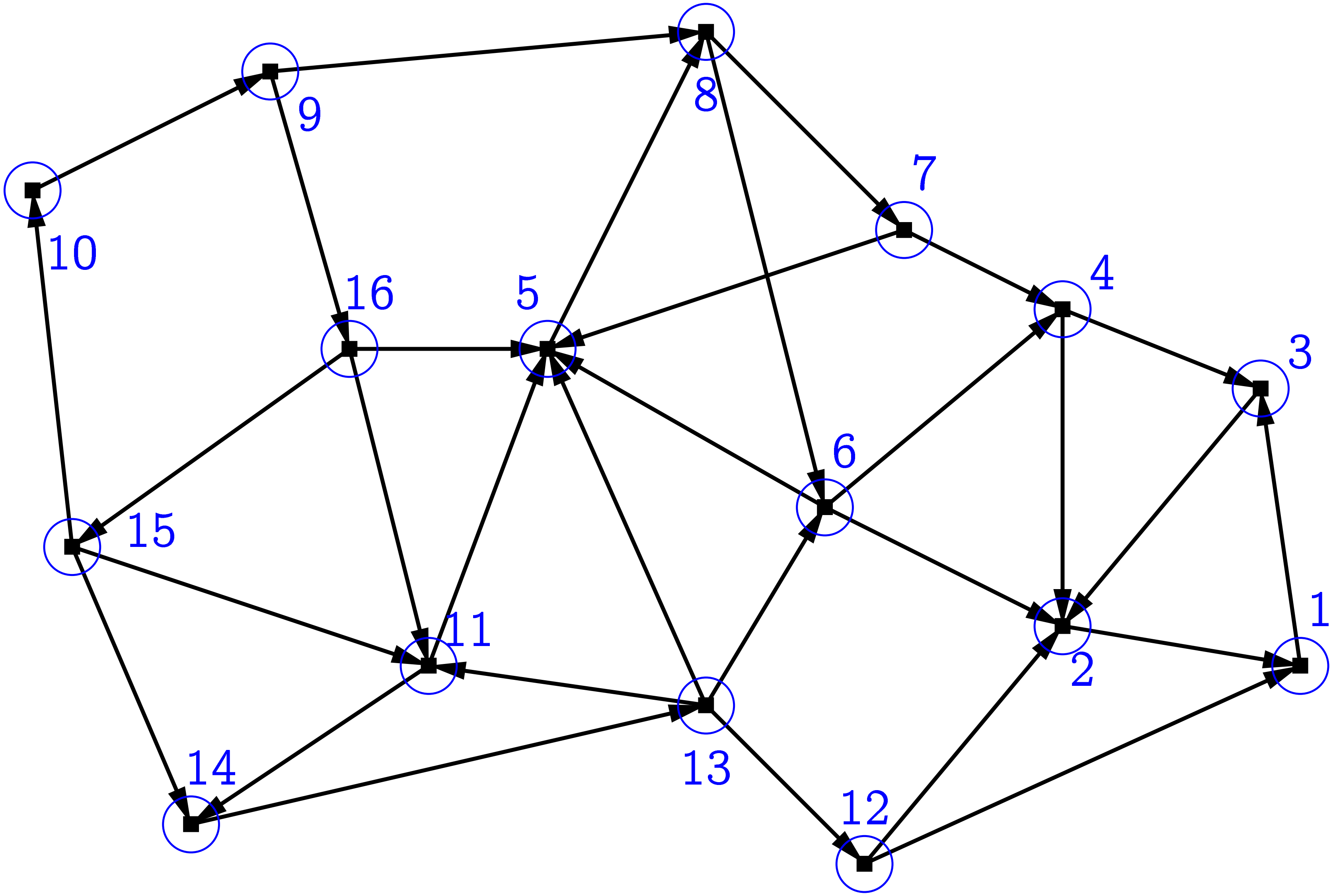

9
8
7
10
5
4
3
6
1
11
2
13
12



9
8
7
10
4
5
3
6
1
11
2
14
13
12



1
2
3
4
5
6
7
8
9
10
11
12
13
14
15



1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

Tähelepanekuid algoritmi kohta:

Iga tipu $v$ jaoks, mille jaoks „sügavuti“ 4. real „külasta“ välja kutsutakse, käiakse enne tagasipöördumist läbi kõik tipud, kuhu $v$-st jõuda võib ja mis ei ole veel läbi käidud.

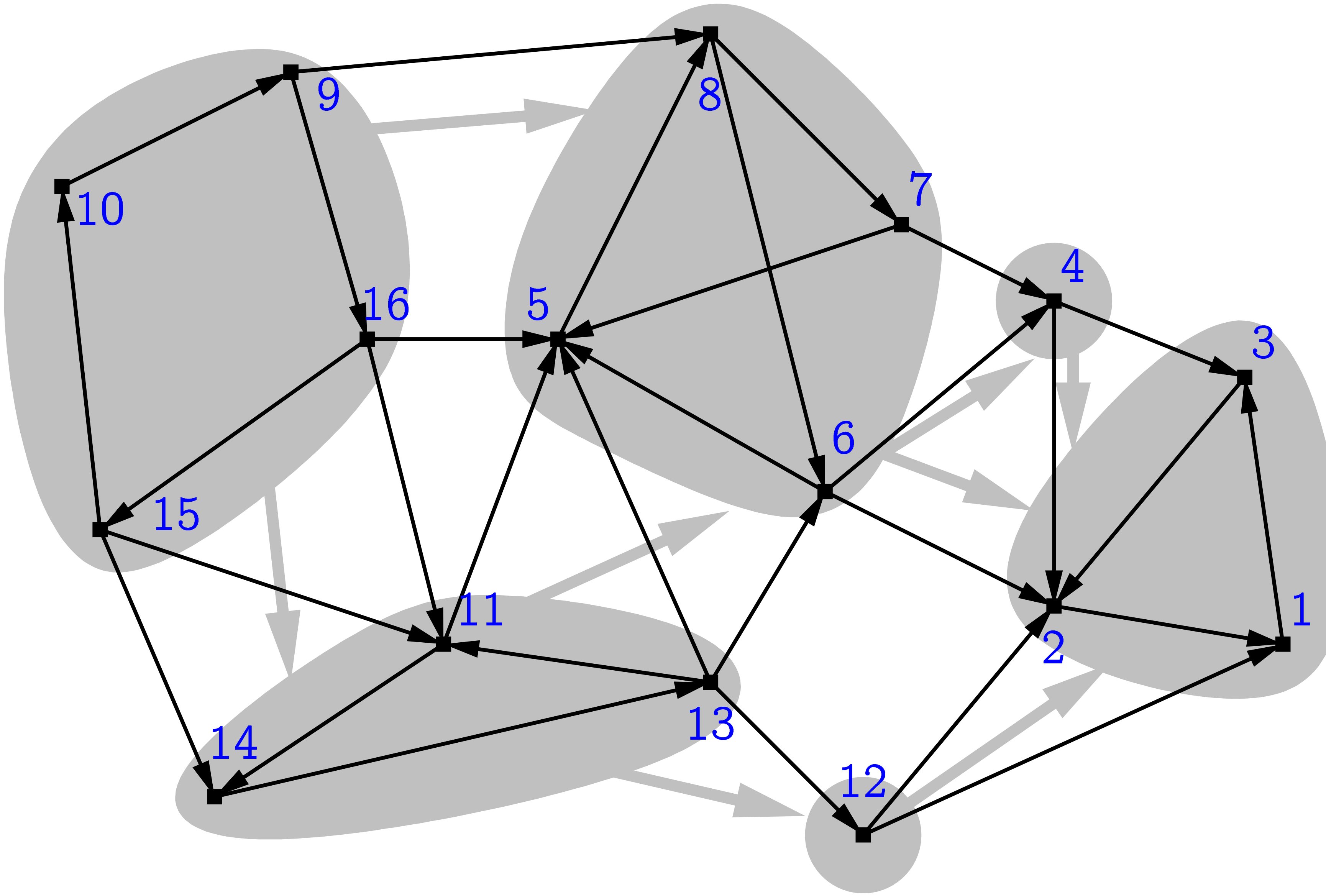
S.t. iga $v$ jaoks, mille jaoks „külasta“ kutsuti välja „sügavuti“-st, ja iga $w$ jaoks, kuhu on $v$-st võimalik minna, kehtib $v.jrknr > w.jrknr$.

Iga tugevalt sidus komponent käiakse läbi sama „külasta“ „sügavuti“-st väljakutse ajal.

Kui $v$ on esimene tipp mingist tugevalt sidusast komponendist, mille jaoks „külasta“ välja kutsutakse, siis enne tagasipöördumist käiakse läbi see tugevalt sidus komponent ning samuti kõik tugevalt sidusad komponendid, kuhu sealt saab ning mis ei ole veel läbi käidud.

Olgu $V_1, V_2$ kaks tugevalt sidusat komponenti, nii et $V_1$-st saab $V_2$-e. Olgu $v_1 \in V_1$ ja $v_2 \in V_2$ max. $.jrknr$-ga tipud neist komponentides. Siis $v_1.jrknr > v_2.jrknr$.

Kui seame $G^{\mathrm{komp}}$ igale tipule (s.t. esialgse graafi tugevalt sidusale komponendile) $U \subseteq V$ vastavusse arvu $\max\limits_{v \in U} v.jrknr$, siis saame $G^{\mathrm{komp}}$ tipud topoloogiliselt sorteeritud (kahanevas järjekorras).



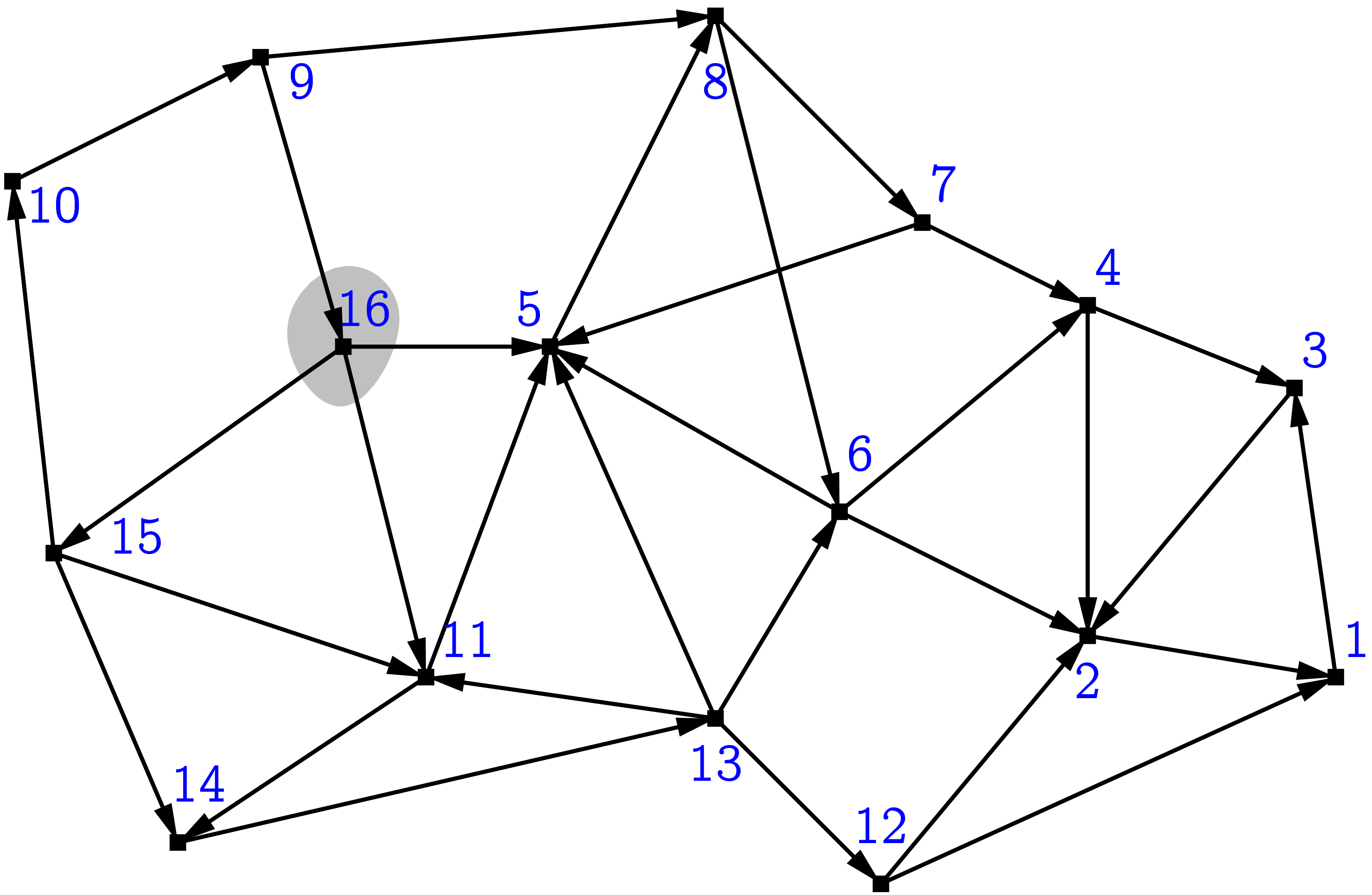
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

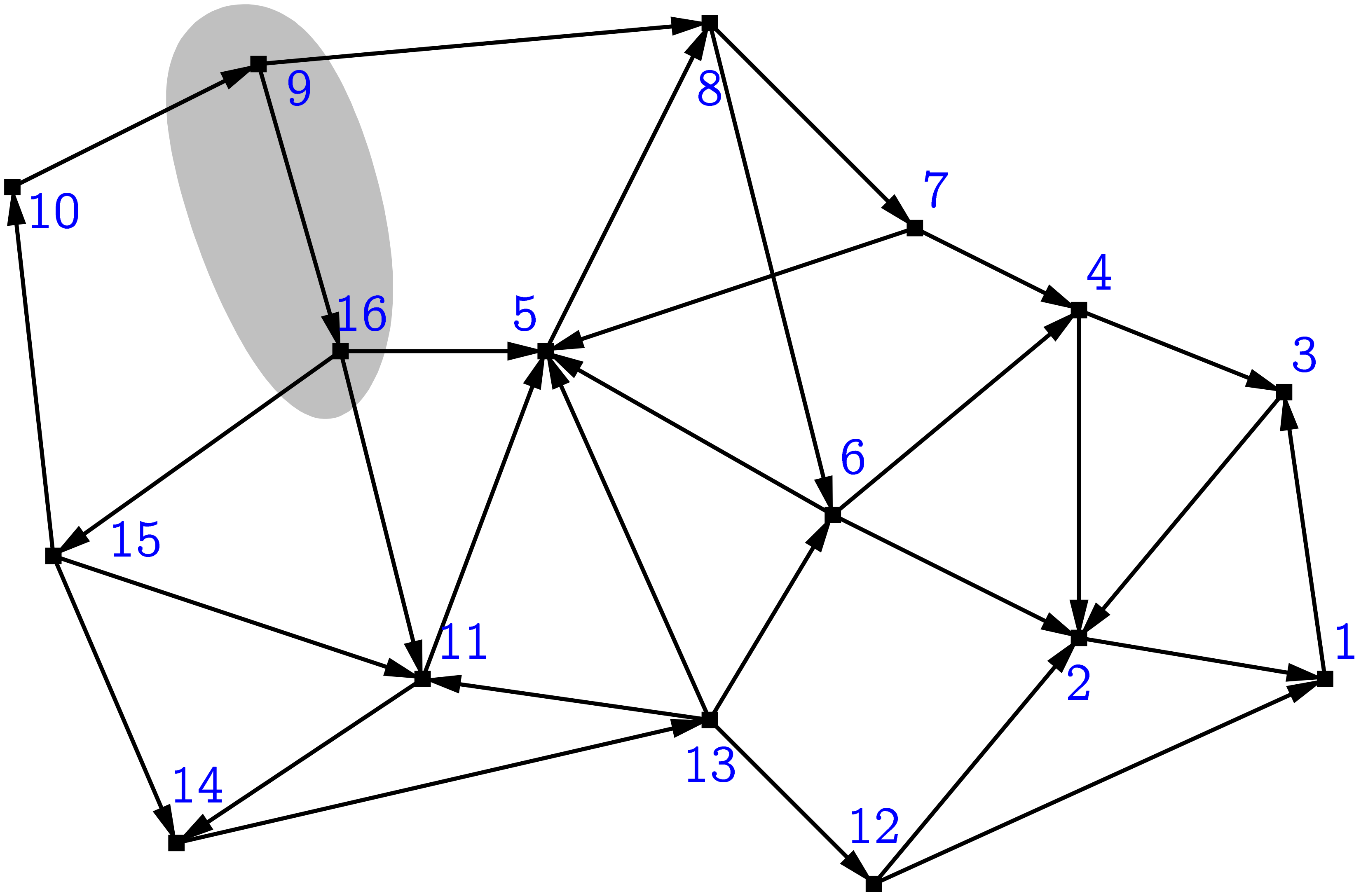
Algoritm tugevalt sidusate komponentide leidmiseks:

sügavuti($G$)
**for all** $v \in V$ **do** $v.\textit{läbitud} :=$ false; $s[v.jrknr] := v$
$K := \emptyset$
**for** $i := |V|$ **downto** 1 **do**
    **if** $\neg s[i].\textit{läbitud}$ **then** $K := K \cup \{\text{komponent}(s[i])\}$
**return** $K$

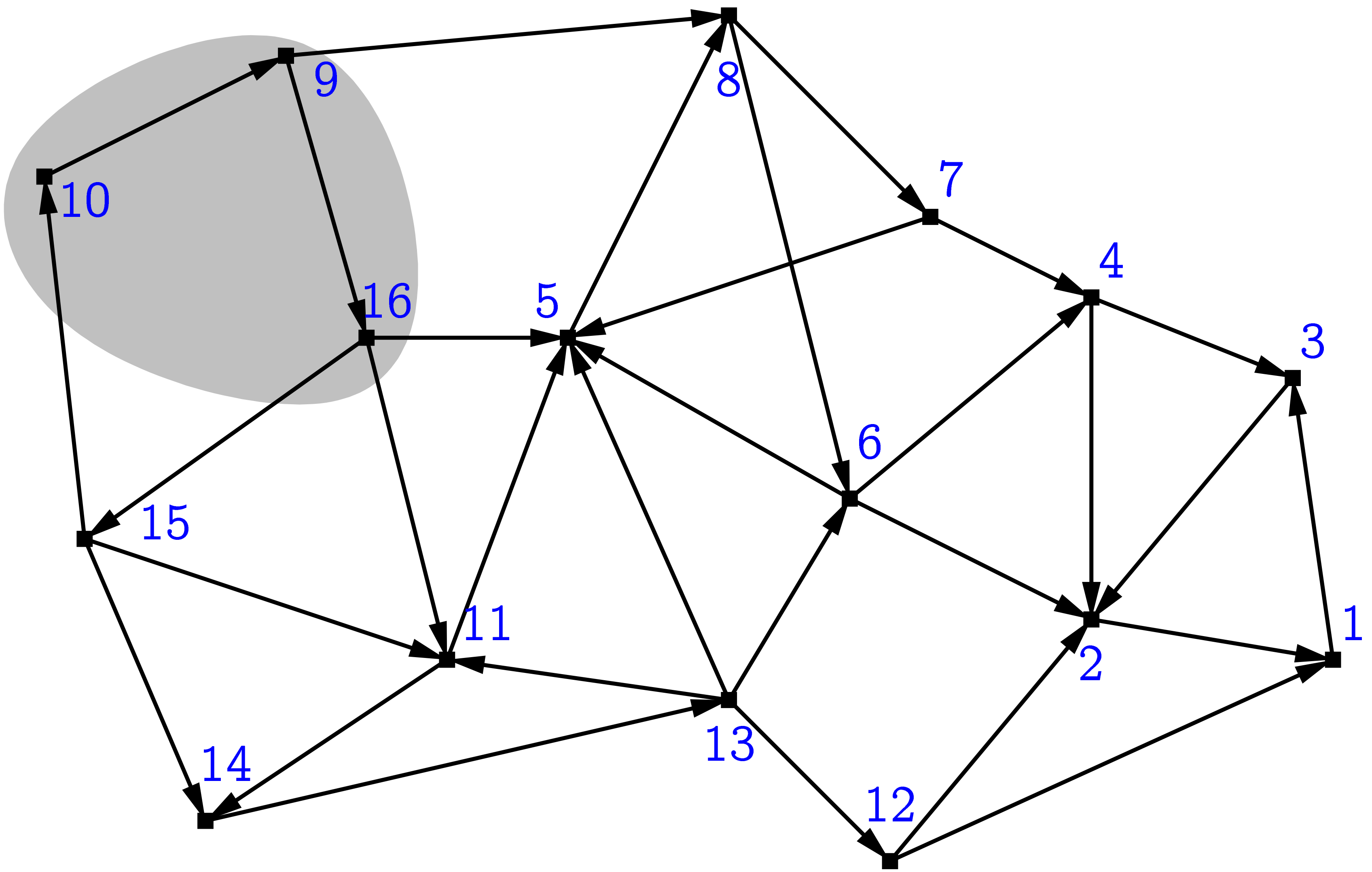
komponent($v$) on

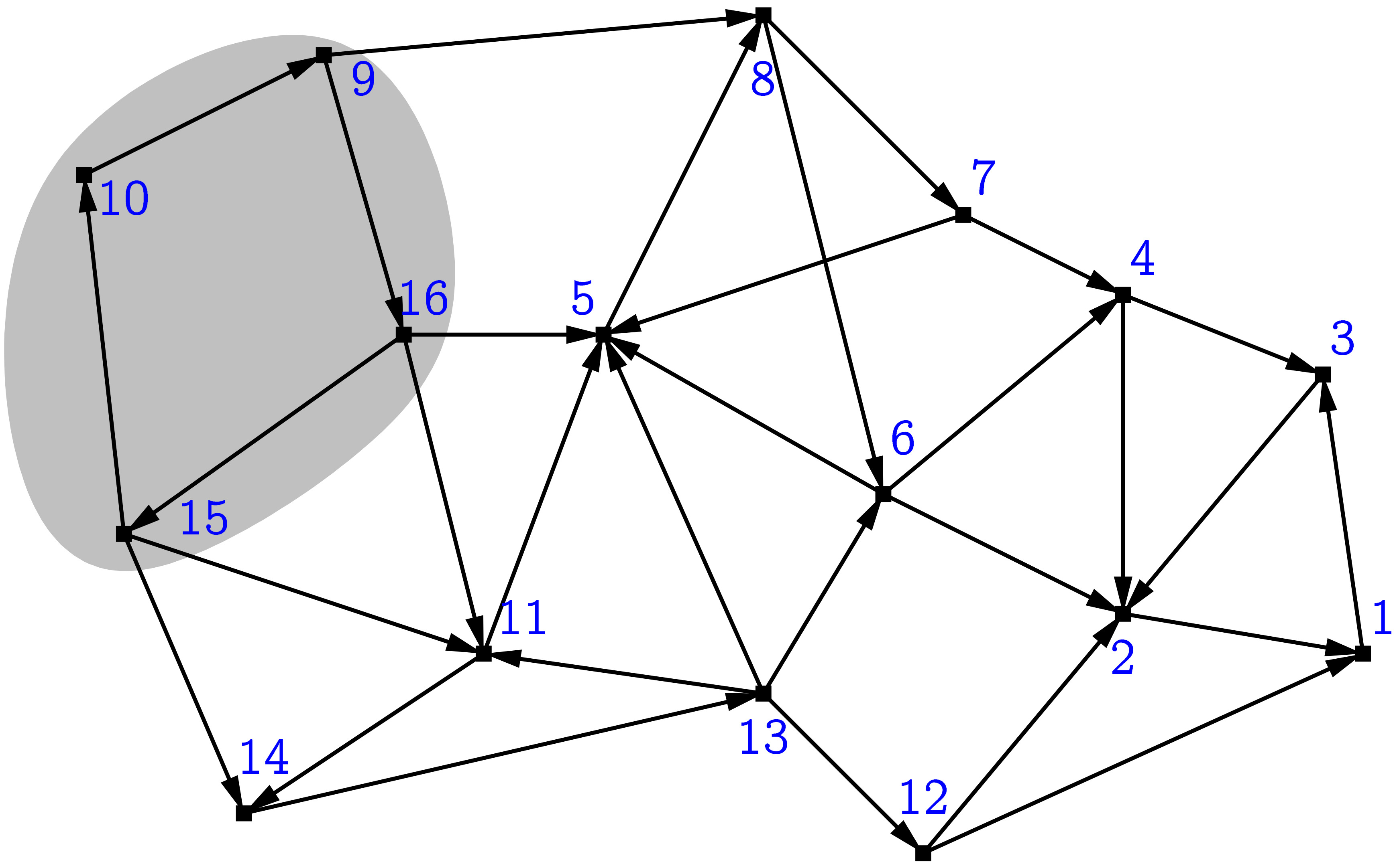
$v.\textit{läbitud} :=$ true
$k := \{v\}$
**for all** $w \in G^{-1}v$ **do**
    **if** $\neg w.\textit{läbitud}$ **then** $k := k \cup \text{komponent}(w)$


9
8
10
7
4
16
5
3
6
15
11
1
2
14
13
12

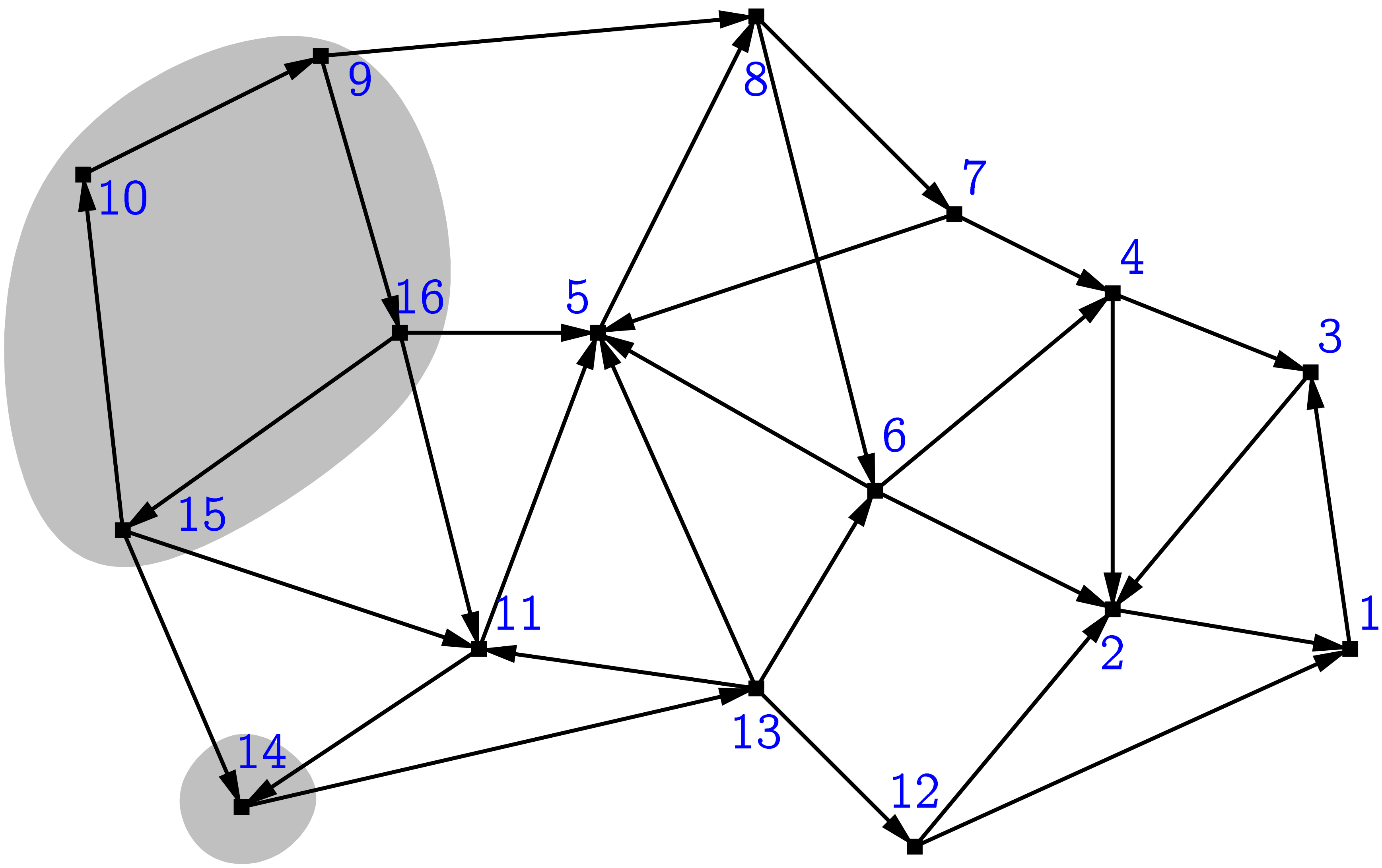


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

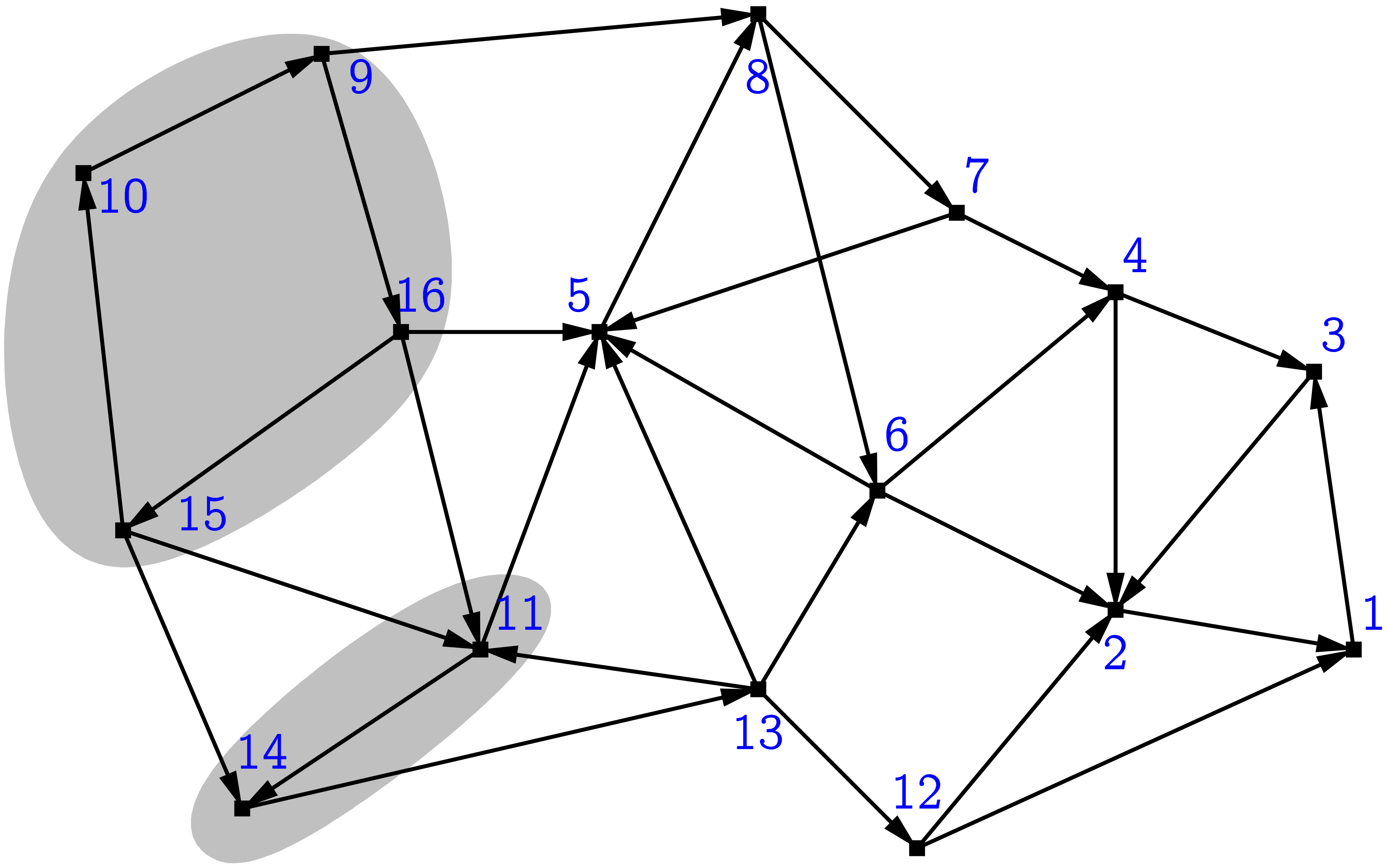

9
8
10
7
4
16
5
3
6
15
11
1
2
13
14
12

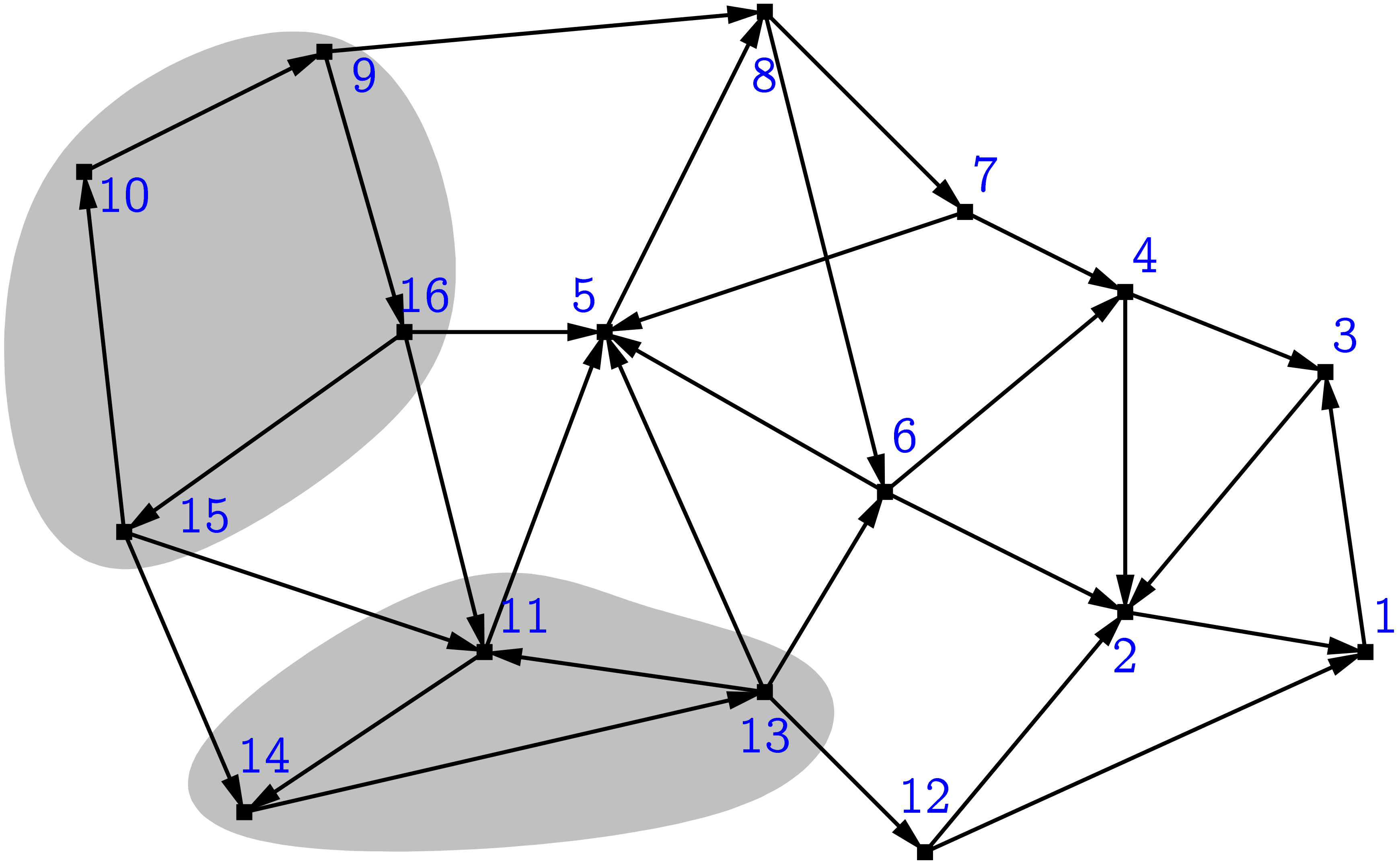


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

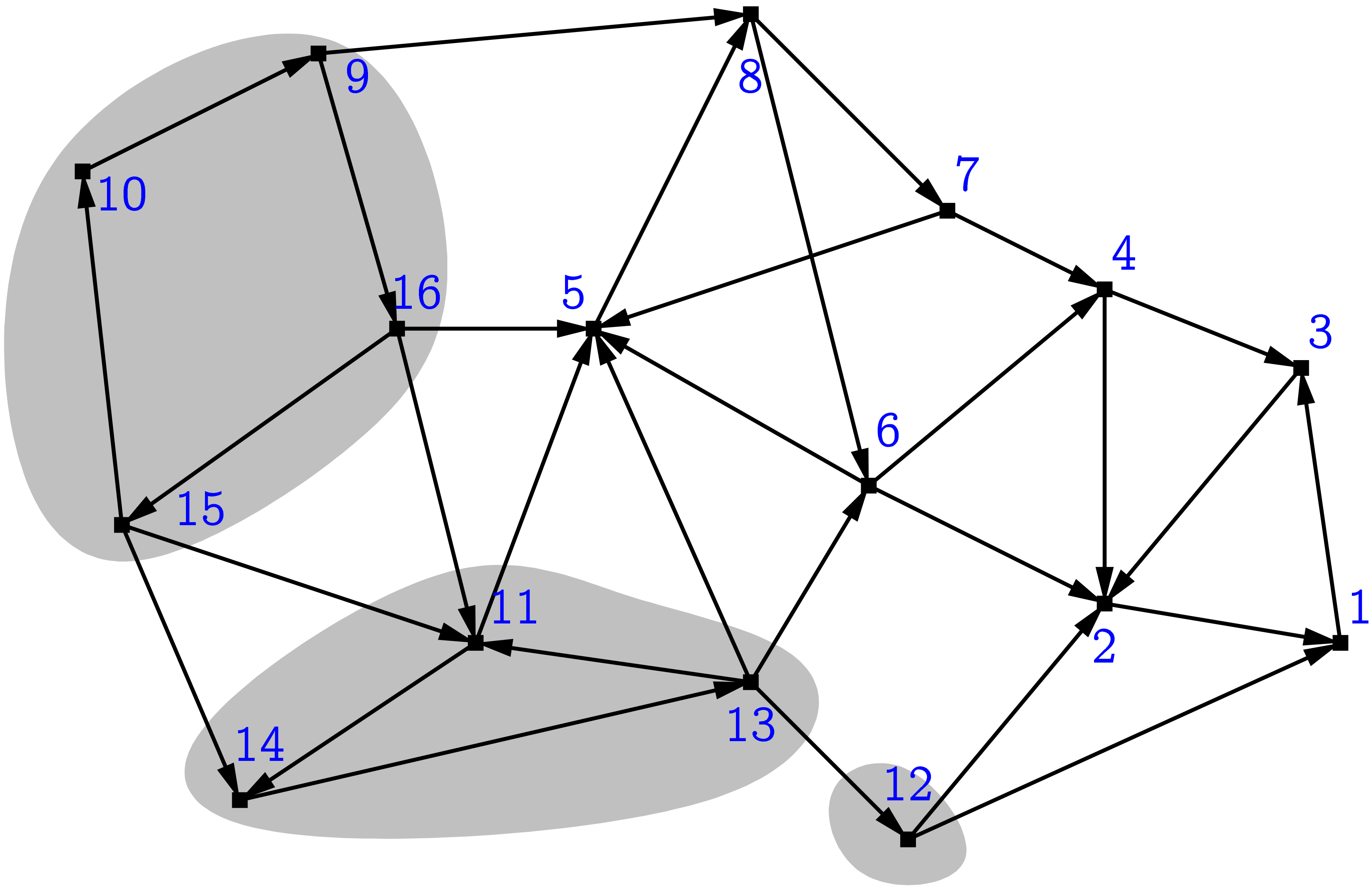


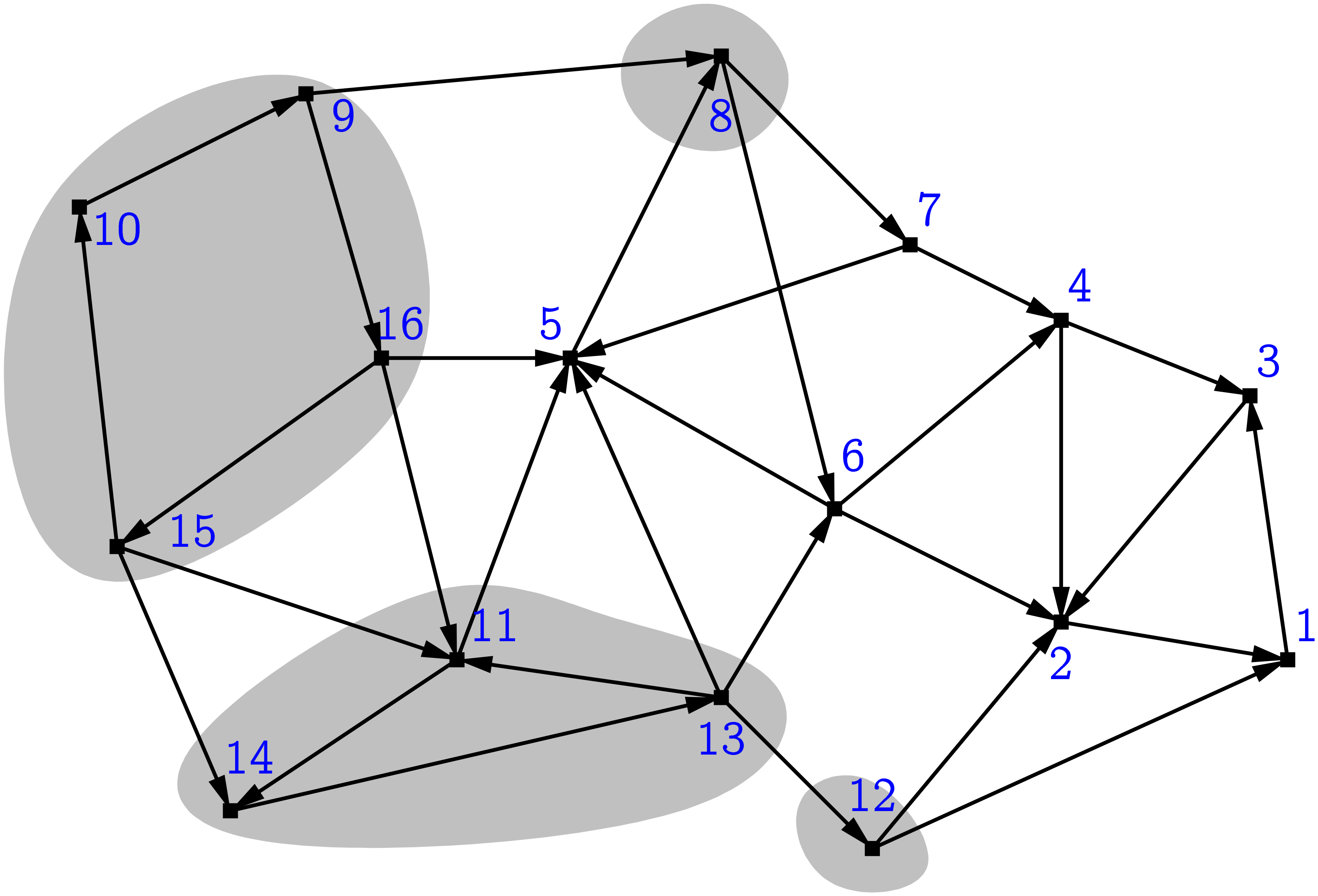
9
10
16
5
8
7
4
3
6
15
11
1
2
13
14
12


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

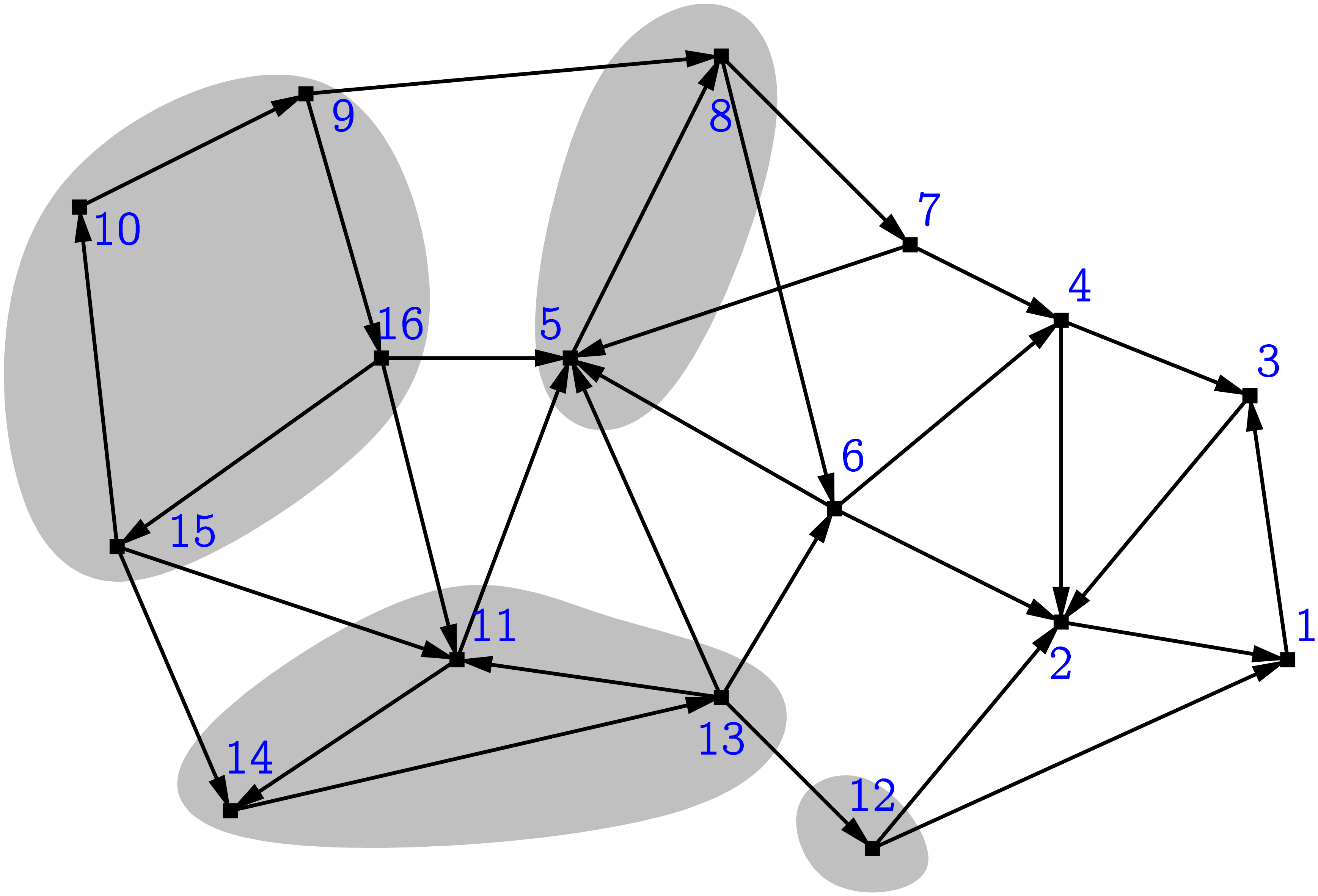


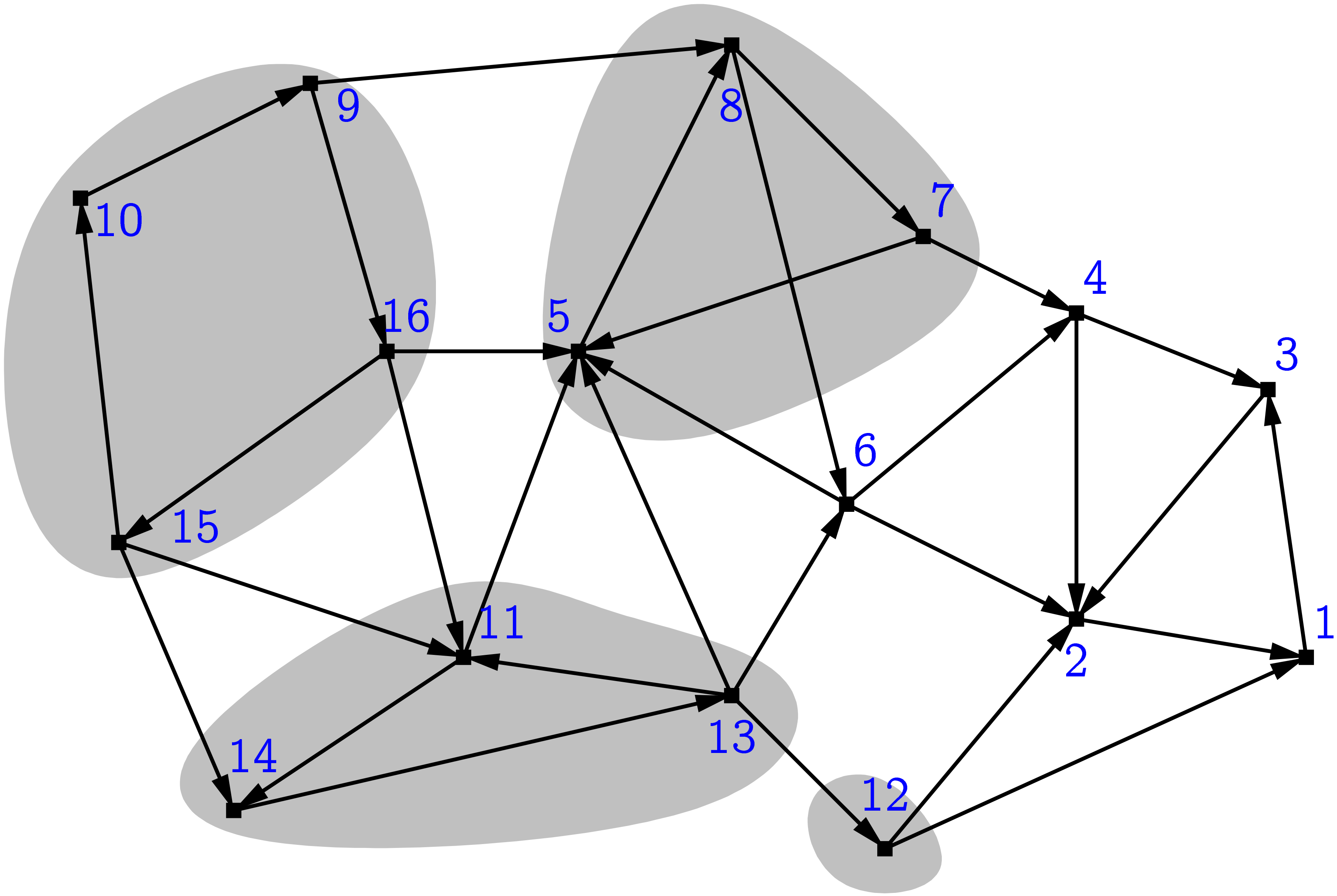
9
8
10
7
4
16
5
3
6
15
11
1
2
14
13
12


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

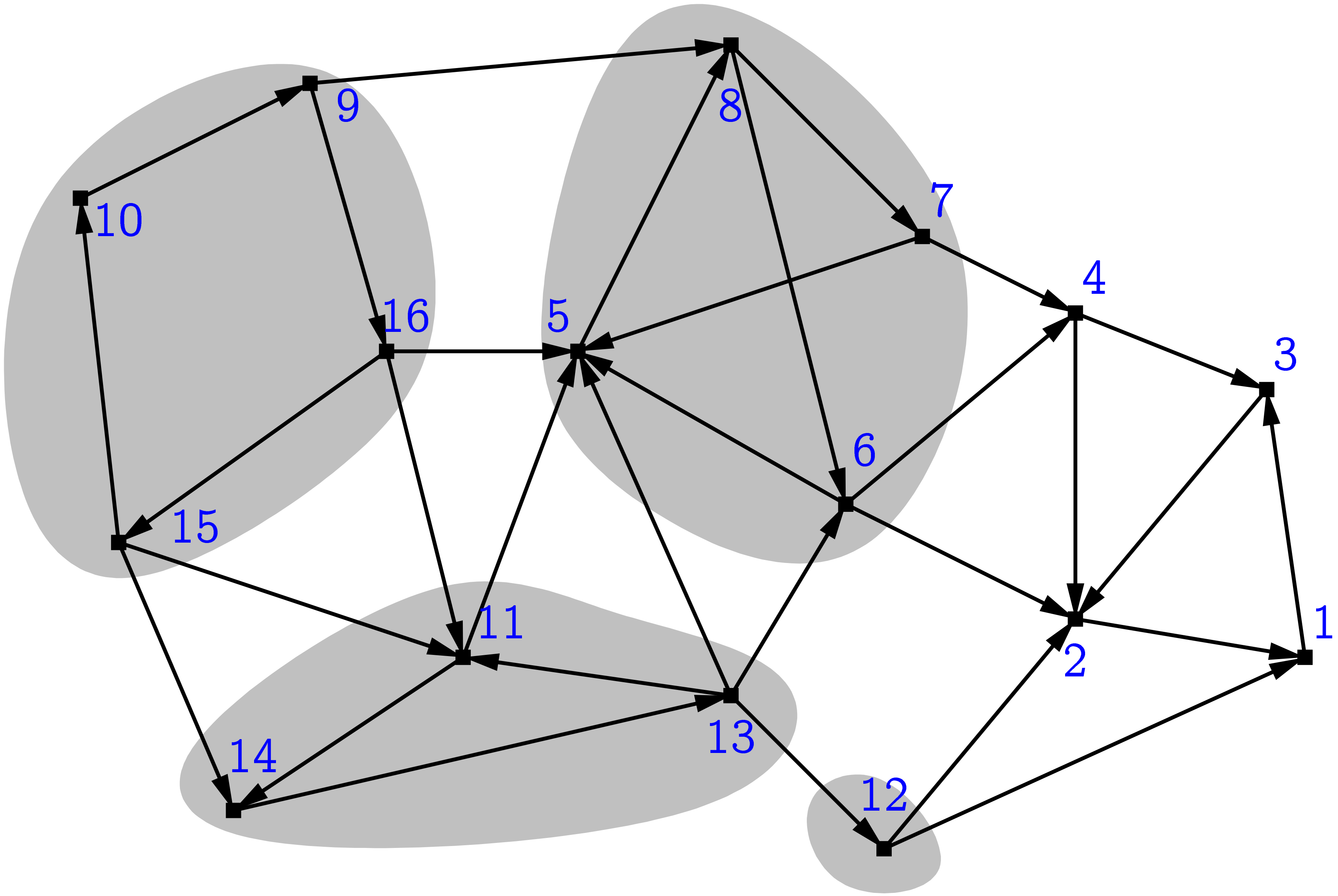


8
9
10
7
4
16
5
3
6
15
11
1
2
13
14
12

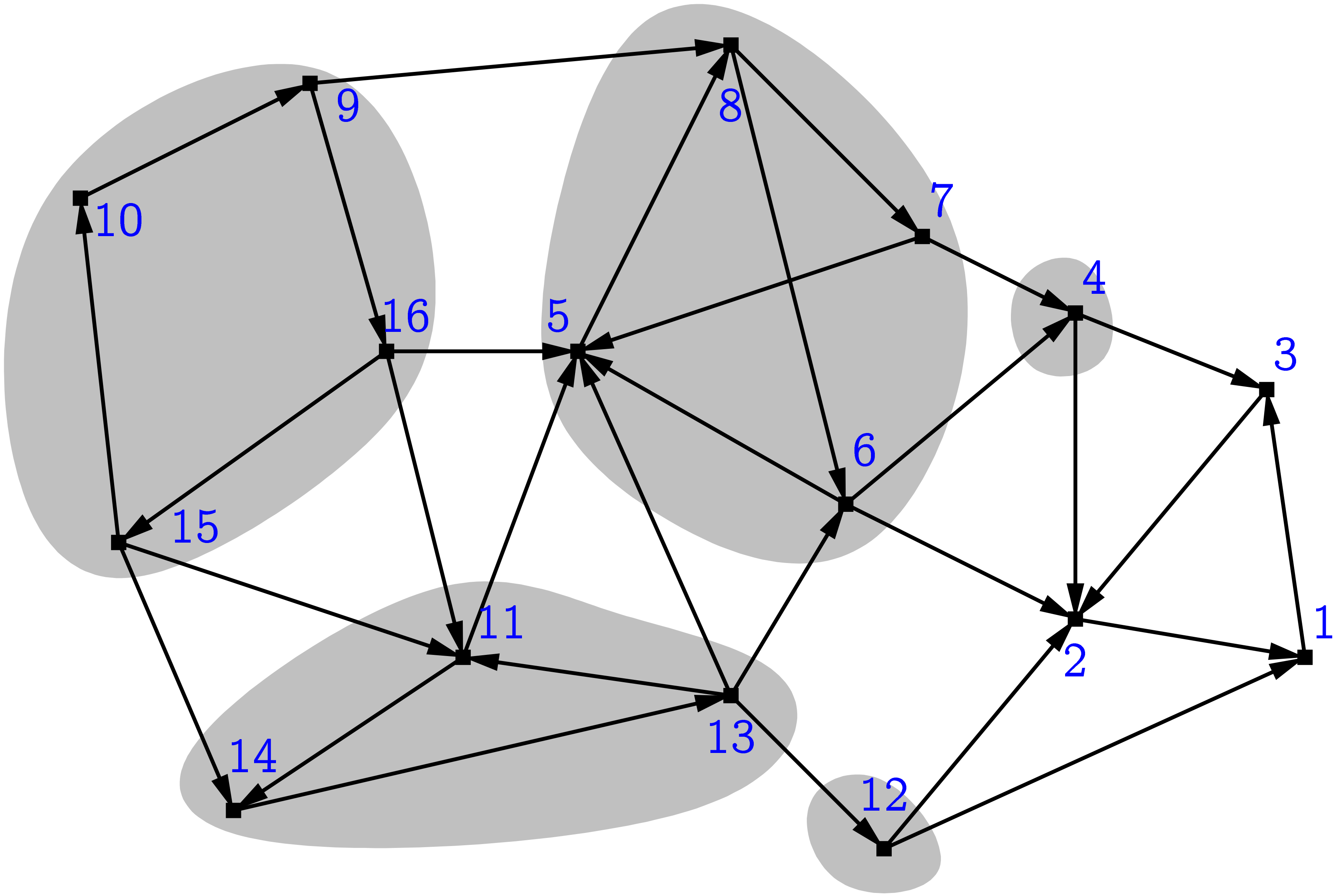

9
8
10
7
16
5
4
3
6
15
11
1
2
13
14
12



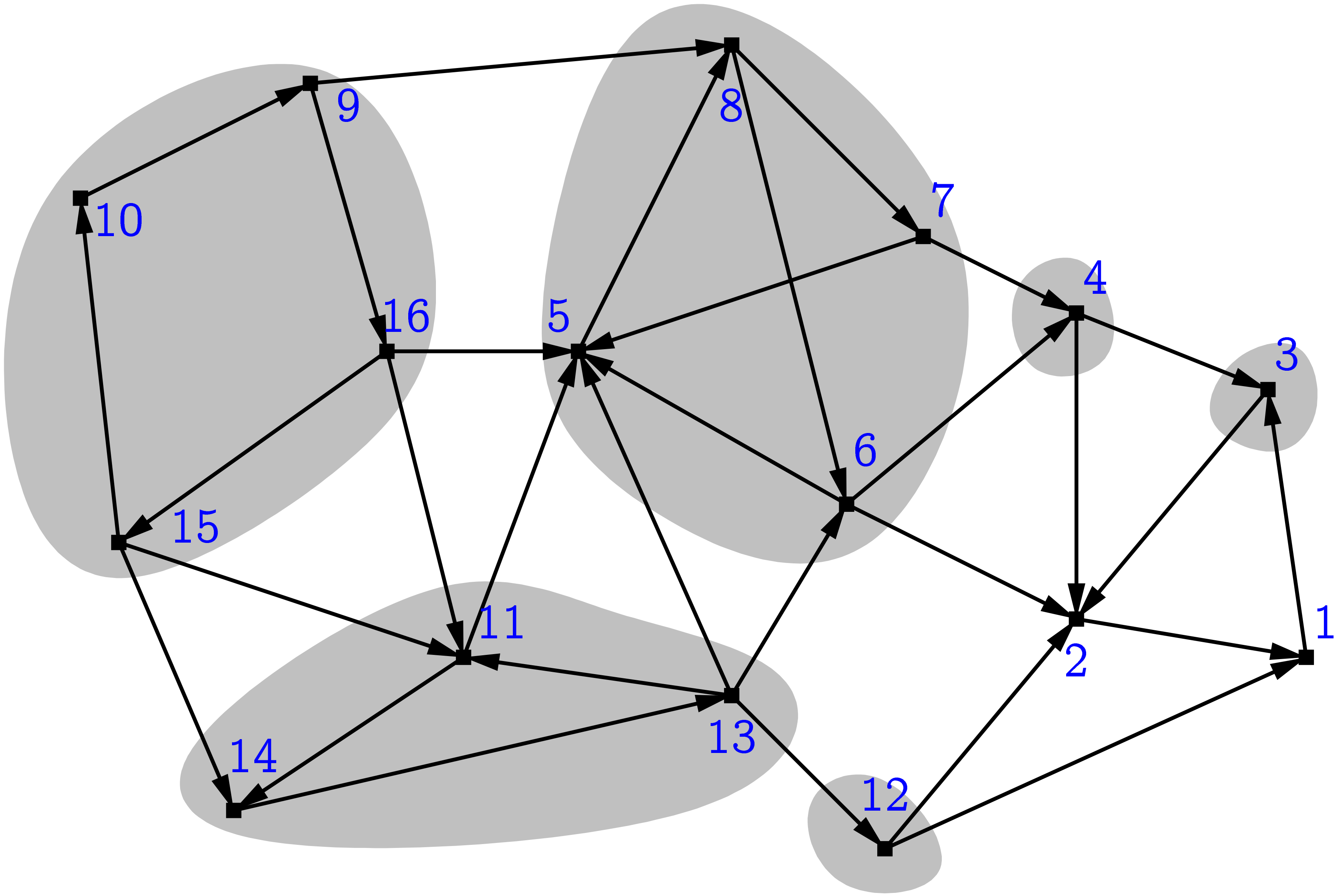
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16



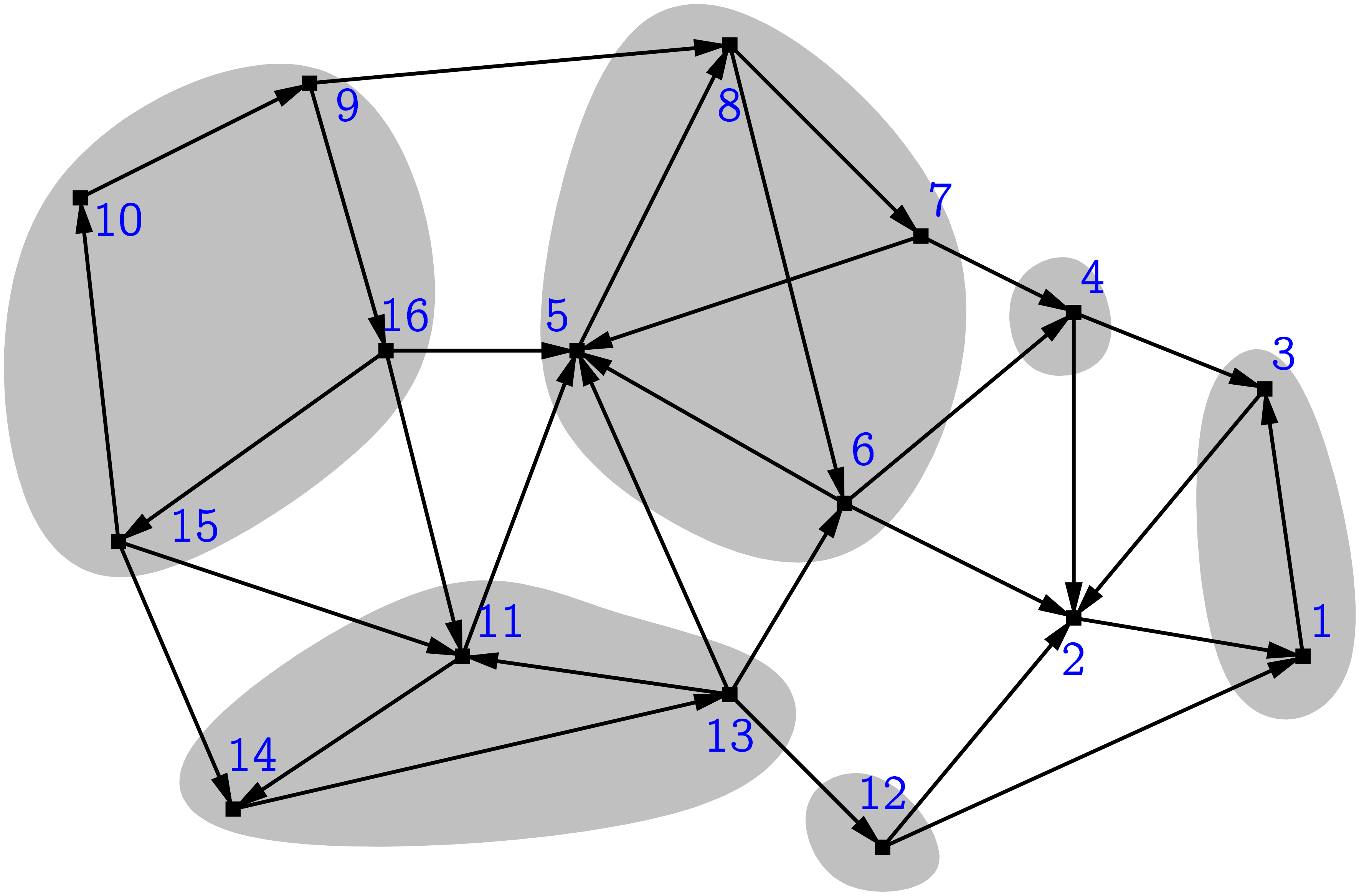
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16



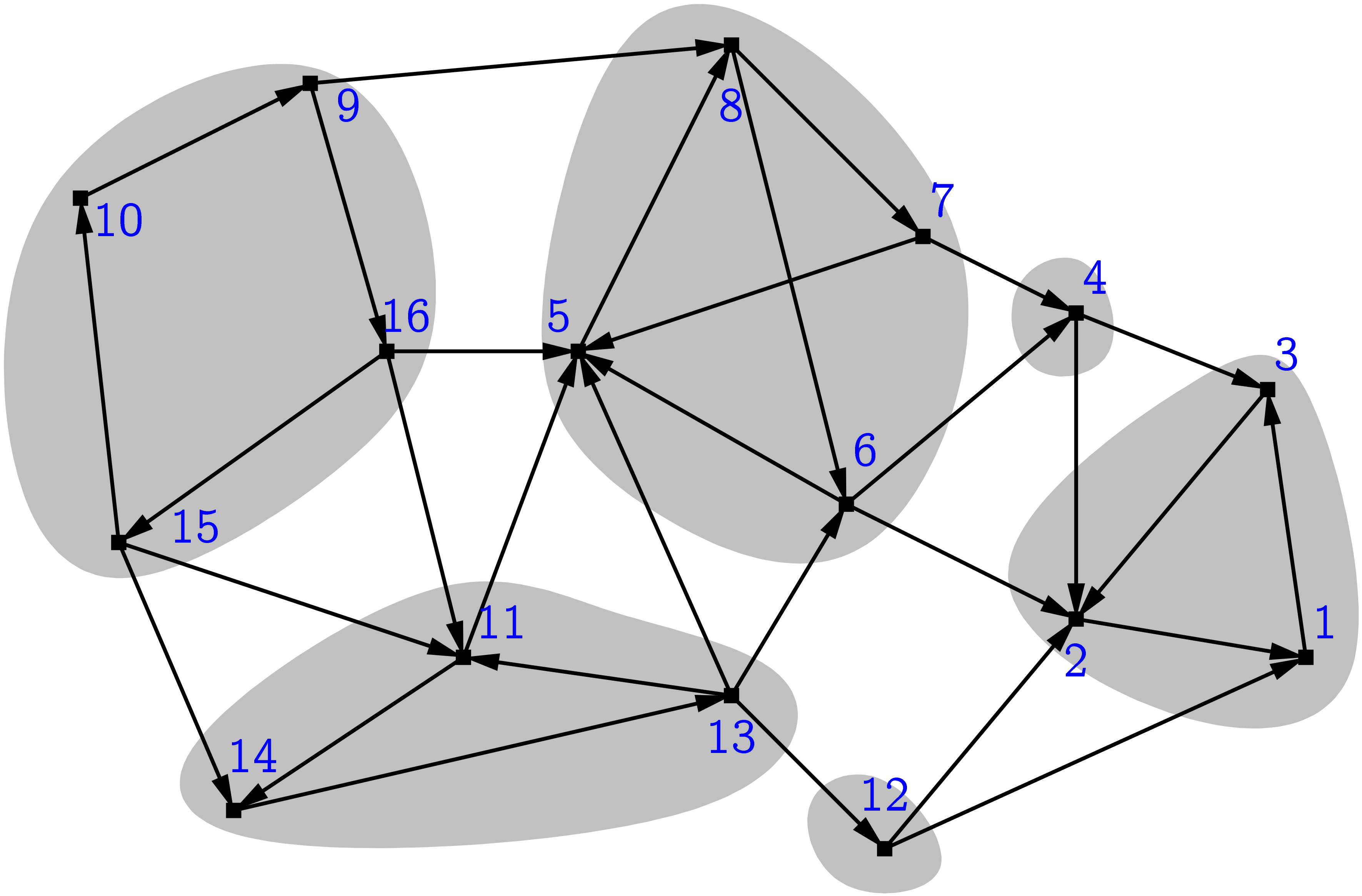
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16



1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16



1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16


9
10
8
7
4
16
5
3
6
15
11
1
2
13
14
12

Olgu meil antud objektid $x_1, \ldots, x_n$. Me soovime pidada andmestruktuuri, mis vastaks hulkade komplektile $(S_1, \ldots, S_k)$, nii et iga element $x_i$ kuuluks täpselt ühte hulkadest $S_j$.

Me soovime järgmiseid operatsioone:

- tee_klass($x$) lisab juba vaadeldavatele objektidele uue objekti $x$, samuti lisab ta uue hulka $S$ ja võtab $x$-i selle ainsaks elemendiks.
- ühenda($x, y$) ühendab hulgad, millesse kuuluvad $x$ ja $y$, üheks hulgaks.
- leia_esindaja($x$) tagastab selle hulga $S$, kuhu $x$ kuulub, mingi elemendi. Seejuures tagastab ta hulga $S$ iga elemendi jaoks sama elemendi.

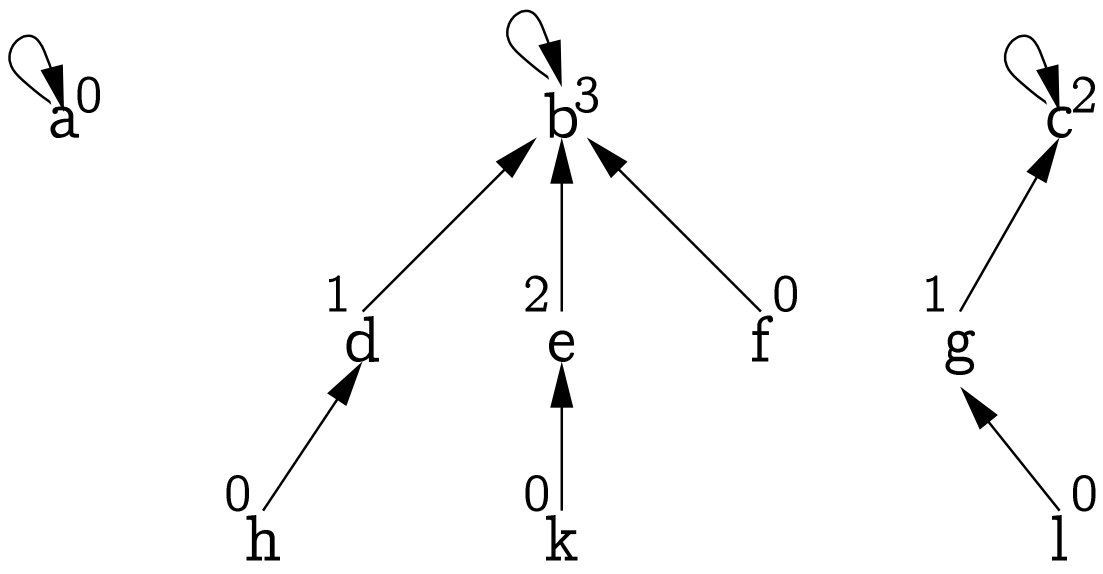
*Galler-Fisher*i meetodil klasside kujutamisel on igal objektil kaks abivälja — $.viit$ ja $.h$.

Klassi kujutatakse puuna tema elementidest, $.viit$ on viit ülemusele. Objekti väli $.h$ on mingi naturaalarv, mis on vähemalt sama suur kui alampuu, mille juureks see objekt on, kõrgus.

Kui mingi objekt on klassi kujutava puu juureks, siis tema $.viit$ viitab talle enesele.

leia_esindaja($x$) tagastab $x$-i sisaldava puu juure.

Näiteks võib ({a}, {b,d,e,f,h,k}, {c,g,l}) kujutatud olla järgmisel viisil:



tee_klass($x$) on

$x.viit := x;\ x.h := 0$

ühenda($x, y$) on

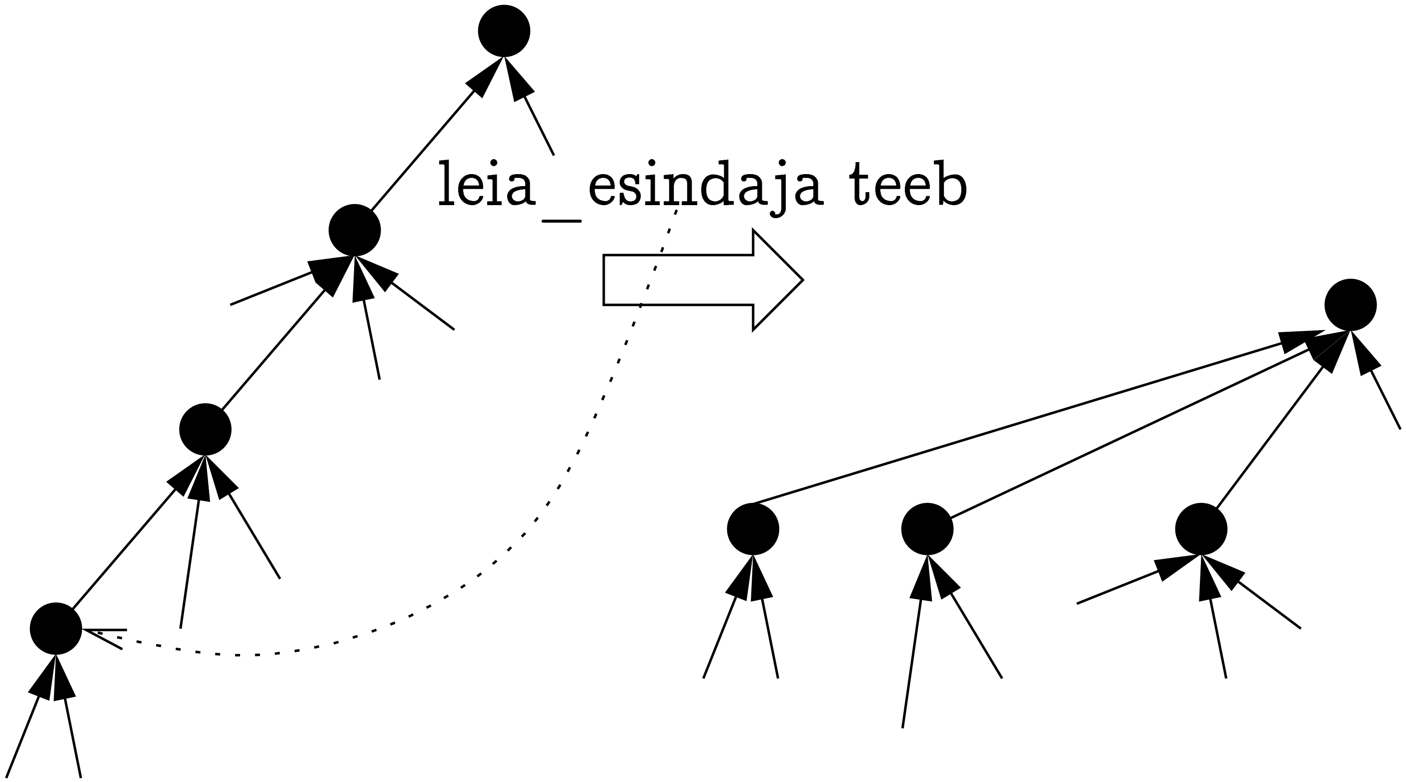
lingi(leia_esindaja($x$), leia_esindaja($y$))

lingi($x, y$) on

**if** $x.h > y.h$ **then** $y.viit := x$ **else** $x.viit := y$
**if** $x.h = y.h$ **then** $y.h := y.h + 1$

leia_esindaja($x$) on

```
if x ≠ x.viit then
    x.viit := leia_esindaja(x.viit)
return x.viit
```



Keerukus: kui meil on $n$ objekti ja $m$ operatsiooni tee_klass, ühenda ja leia_esindaja, siis ajakulu on $O(m \log^* n)$.

$\log^* n$ on selline $i$, et $0 < \underbrace{\log \cdots \log}_{i} n \leqslant 1$.

| $n$-i vahemik | $\log^* n$ |
|---|---|
| $n = 1$ | 0 |
| $n = 2$ | 1 |
| $3 \leqslant n \leqslant 4$ | 2 |
| $5 \leqslant n \leqslant 16$ | 3 |
| $17 \leqslant n \leqslant 65536$ | 4 |
| $65537 \leqslant n \leqslant 2^{65536}$ | 5 |

Me näitame, et $n$ objekti korral on $m$ operatsiooni tee_klass, lingi ja leia_esindaja ajakulu on $O(m \log^* n)$.

Siis on ka $m$ operatsiooni tee_klass, ühenda ja leia_esindaja ajakulu $O(m \log^* n)$, sest nad sisaldavad ülimalt $3m$ operatsiooni tee_klass, lingi ja leia_esindaja.

Puude struktuuri ja selle muutumise kohta töö jooksul võib öelda järgmist. Olgu $x$ mingi objekt.

- Alguses on $x$ mingi puu juur. Kui ta töö käigus mingil hetkel mittejuureks muutub, siis ei saa ta enam kunagi juureks.
- Kirjutusviis $x.viit = x$ tähendab „$x$ on mingi puu juur“.

Mõned tulemused väljade $.h$ väärtuste kohta ja nende muutumise kohta töö jooksul. Vaatame mingit objekti $x$.

- Kui $x.viit \neq x$, siis $x.h < x.viit.h$.
- Alguses on $x.h = 0$. Töö jooksul ta ei vähene. Alates hetkest, kus $x.viit \neq x$, $x.h$ enam ei muutu.
- $x.viit.h$ töö jooksul ei vähene.
- Olgu $P(x)$ tippude arv alampuus, mille juureks on $x$. Siis $P(x) \geqslant 2^{x.h}$.

Tõestused on induktsiooniga üle tehtud operatsioonide arvu.

Kehtigu eelmisel slaidil toodud väited peale mingit arvu operatsioone. Teeme järjekordse operatsiooni.

Kui see oli tee_klass($x$), siis saab $x.h$ väärtuseks 0. $x$ on mingi puu juur. $P(x) = 1 = 2^{x.h}$.

Kui see oli lingi$(x, y)$, siis muutused toimuvad ainult $x$-i ja $y$-i juures.

Kui enne operatsiooni $x.h \neq y.h$ (üldisust kitsendamata loeme, et $x.h < y.h$), siis pärast operatsiooni on juureks mittevõetud tipu $x$ väli $.h$ väiksem kui juureks võetud tipu $y$ väli $.h$.

Samuti on pärast operatsiooni $P(y) \geqslant 2^{y.h} + 2^{x.h} \geqslant 2^{y.h}$.

Kui enne operatsiooni $x.h = y.h$, siis pärast $y.h = x.h + 1 > x.h$. Samuti $P(y) \geqslant 2^{y.h-1} + 2^{x.h} = 2 \cdot 2^{y.h-1} = 2^{y.h}$.

lingi on ainus operatsioon, mis muudab välja $.h$ väärtusi. Muudetakse tipul, mis on puu juur.

Kui järjekordne operatsioon oli leia_esindaja($x$), siis võis $x.viit$ hakata viitama kõrgemale kui varem.

Kuna ennem oli $x.h < x.viit.h$ iga tipu $x$ jaoks, siis puus ülespoole liikudes välja $.h$ väärtused suurenevad.

Välja $x.viit$ muutes siis $x.viit.h$ suurenes.

**Järeldus.** Tippe, mille välja $.h$ väärtus on vähemalt $r$, on ülimalt $n/2^r$.

Tõestus. Kui mingi tipu $x$ jaoks omistatakse $x.h := r$, siis peab puus, mille juureks on $x$, olema vähemalt $2^r$ tippu. Nende tippude (v.a. $x$ ise) aste on väiksem kui $r$ ja enam ei muutu.

Kui mõne teise tipu $y$ jaoks omistatakse $y.h := r$, siis peab puus, mille juures on $y$, olema vähemalt $2^r$ tippu. Need tipud peavad olema erinevad puu, mille juureks on $x$, tippudest, sest $x$ ei kuulu puusse, mille juureks on $y$.

Seega saab ülimalt iga $2^r$-s tipp saada $.h$ väärtuseks $r$ või enam. Muuhulgas on siis iga tipu välja $.h$ väärtus $\leqslant \log n$.

Üks operatsioon tee_klass või lingi vajab konstantset aega.

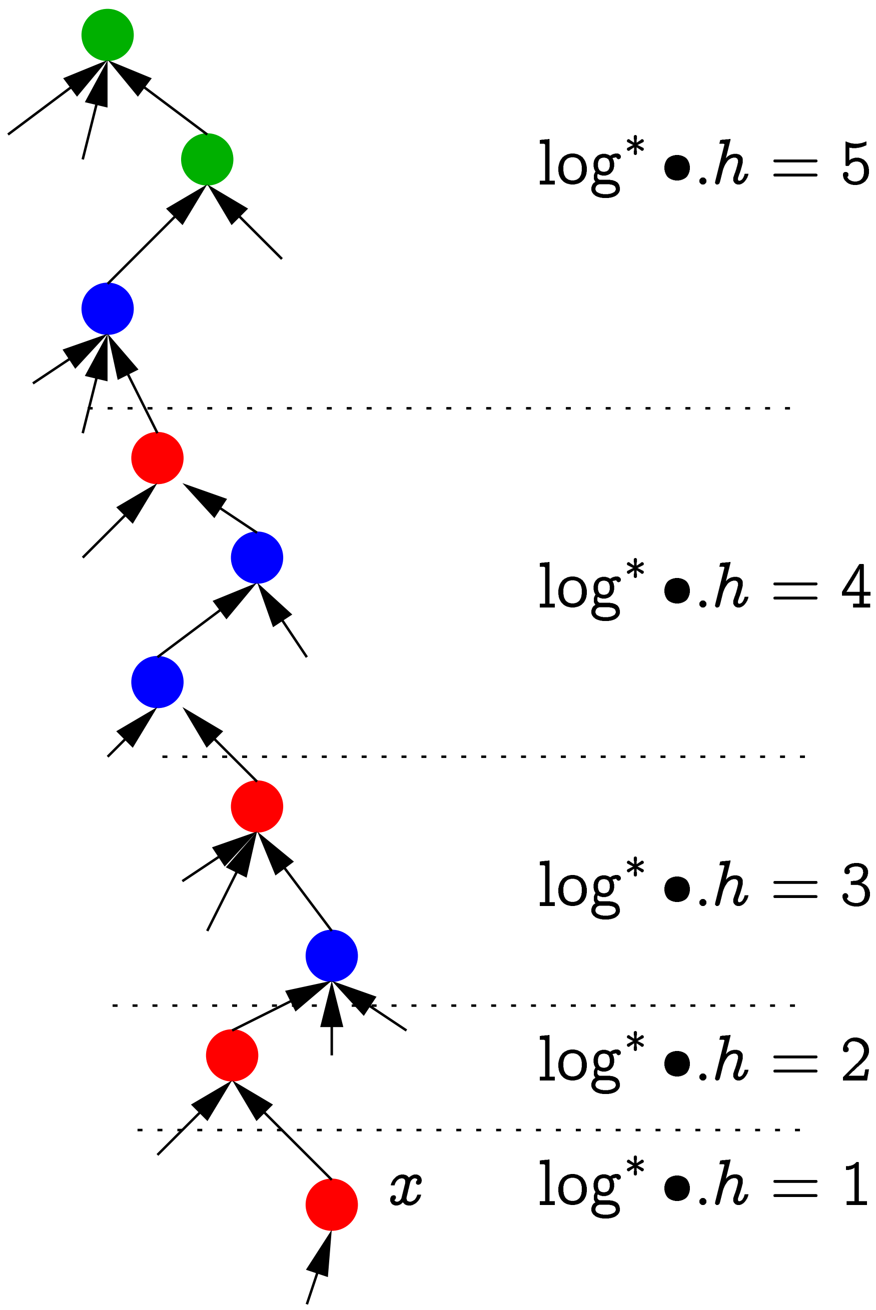
Ühe operatsiooni leia_esindaja($x$) tööaeg on proportsionaalne objekti $x$ kaugusega teda sisaldava puu juurest selle operatsiooni tegemise hetkel.

Me näitame, et peale $m$-i operatsiooni on nende kauguste summa $O(m \log^* n)$.

Vaatame mingit operatsiooni leia_esindaja($x$).

Vaatame ahelat tipust $x$ teda sisaldava puu juureni. Värvime sellel ahelal tipud järgmiselt:

- Juure ja tema vahetu järglase värvime roheliseks.
- Mõne teise tipu $v$ sellelt ahelalt värvime punaseks parajasti siis, kui $\log^* y.h \neq \log^* y.viit.h$.
  - Defineerime $\log^* 0 := -1$.
- Ülejäänud tipud värvime siniseks.
  - Neil tippudel $\log^* y.h = \log^* y.viit.h$.



$\log^* \bullet.h = 5$
$\log^* \bullet.h = 4$
$\log^* \bullet.h = 3$
$\log^* \bullet.h = 2$
$x$
$\log^* \bullet.h = 1$

Igal operatsioonil leia_esindaja on ülimalt kaks rohelist ja ülimalt $\log^* \log n = \log^* n - 1$ punast tippu.

Kui mingi tipp $x$ on kunagi olnud punane, siis ei saa ta enam kunagi edaspidi olla sinine, sest

- $x.h$ enam ei muutu, seega ei muutu ka $\log^* x.h$;
- $\log^* x.h < \log^* x.viit.h$;
- $x.viit.h$ saab üksnes suureneda, seega ka $\log^* x.viit.h$ saab üksnes suureneda.

Uurime, mitu korda saab üks tipp olla sinine.

Kui tipp $x$ on mingis operatsioonis leia_esindaja sinine, siis muutub tema otsene ülemus. S.t. $x.viit.h$ suureneb selle operatsiooni käigus.

Ta saab suureneda ainult senikaua kuni $\log^* x.viit.h = \log^* x.h$, muidu muutub tipp punaseks.

Olgu $S(x)$ kordade arv, mil tipp $x$ võib olla sinine. Siis on $m$ operatsiooni tehes operatsioonidel leia_esindaja vaadeldavate ahelate pikkuste summa ülimalt

$$\color{green}{2m} + \color{red}{m(\log^* n - 1)} + \color{blue}{\sum_{i=1}^{n} S(x_i)} \quad .$$

Tähistame: $\mathbf{2}^{*-2} := -1$, $\mathbf{2}^{*-1} := 0$, $\mathbf{2}^{*0} := 1$, $\mathbf{2}^{*i} := 2^{\mathbf{2}^{*i-1}}$.

S.t. $\mathbf{2}^{*i} = \left. 2^{2^{\cdot^{\cdot^{\cdot^{2}}}}} \right\} i$

Siis $\log^* n = i$ parasjagu siis, kui $\mathbf{2}^{*i-1} + 1 \leqslant n \leqslant \mathbf{2}^{*i}$.

Olgu $j = \log^* x.h$. Siis $x.viit.h$ võib suureneda ülimalt $\mathbf{2}^{*j} - \mathbf{2}^{*j-1} - 1$ korral, nii et $\log^* x.viit.h$ ei saa suuremaks kui $\log^* x.h$.

S.t. $S(x) \leqslant \mathbf{2}^{*\log^* x.h} - \mathbf{2}^{*(\log^* x.h)-1} - 1$.

Olgu $N(j) = |\{x : \log^* x.h = j\}|$. Siis on $m$ operatsiooni tehes operatsioonidel leia_esindaja vaadeldavate ahelate pikkuste summa ülimalt

$$2m + m(\log^* n - 1) + \sum_{j=-1}^{\log^* n - 1} N(j)(\mathbf{2}^{*j} - \mathbf{2}^{*j-1} - 1) \ .$$

Peale selle,

$$N(j) \leqslant \sum_{r=\mathbf{2}^{*j-1}+1}^{\mathbf{2}^{*j}} \frac{n}{2^r},$$

sest tippe, mille välja $.h$ väärtus on vähemalt $r$, on ülimalt $n/2^r$.

$$
\begin{aligned}
N(j) &\leqslant \sum_{r=\mathbf{2}^{*j-1}+1}^{\mathbf{2}^{*j}} \frac{n}{2^r} = \frac{n}{2^{\mathbf{2}^{*j-1}+1}} \sum_{r=0}^{\mathbf{2}^{*j}-\mathbf{2}^{*j-1}-1} \frac{1}{2^r} \\
&\leqslant \frac{n}{2^{\mathbf{2}^{*j-1}+1}} \sum_{r=0}^{\infty} \frac{1}{2^r} = \frac{n \cdot 2}{2^{\mathbf{2}^{*j-1}+1}} = \frac{n}{2^{\mathbf{2}^{*j-1}}} \\
&= \begin{cases} 2n, & \text{kui } j = -1 \\ \dfrac{n}{\mathbf{2}^{*j}}, & \text{kui } j \geqslant 0 \end{cases}
\end{aligned}
$$

$$2m + m(\log^* n - 1) + \sum_{j=-1}^{\log^* n - 1} N(j)(\mathbf{2}^{*j} - \mathbf{2}^{*j-1} - 1) \leqslant$$

$$m(\log^* n + 1) + \sum_{j=-1}^{\log^* n - 1} N(j)\mathbf{2}^{*j} \leqslant$$

$$m(\log^* n + 1) + 2n\mathbf{2}^{*-1} + \sum_{j=0}^{\log^* n - 1} \frac{n}{\mathbf{2}^{*j}}\mathbf{2}^{*j} =$$

$$m(\log^* n + 1) + \sum_{j=0}^{\log^* n - 1} n =$$

$$m(\log^* n + 1) + n \log^* n = O(m \log^* n)$$