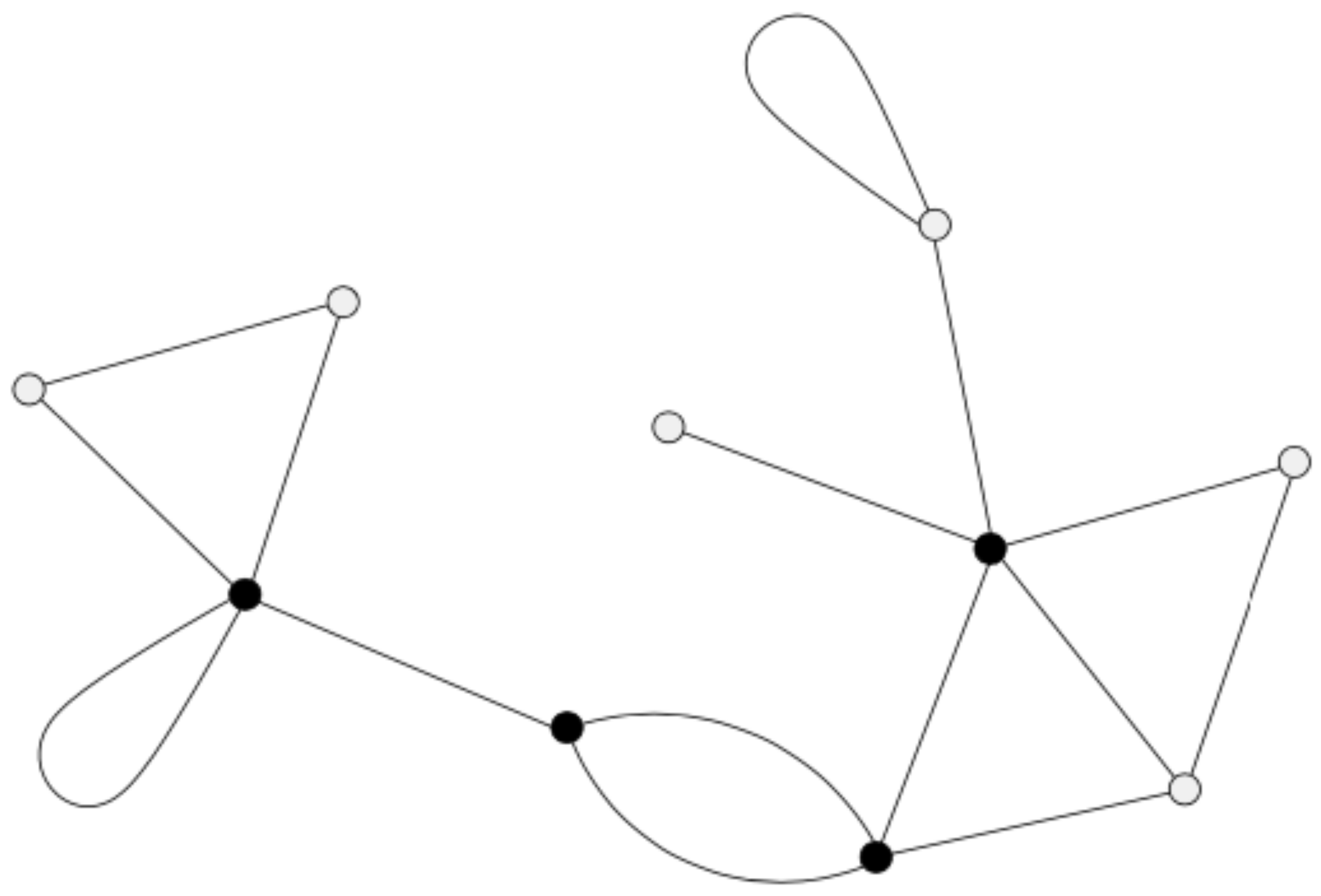
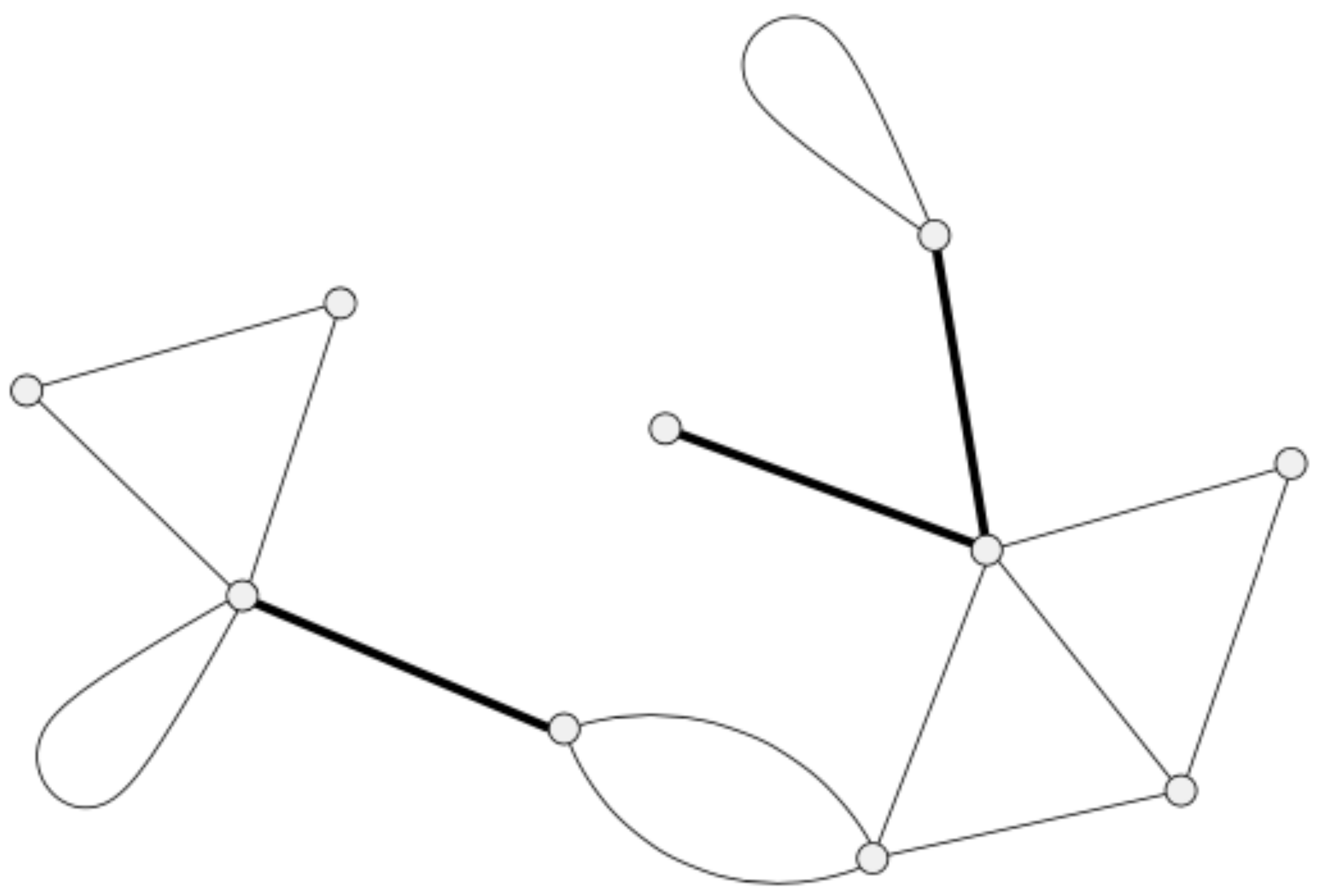
Let G be a connected graph.

If $v \in V(G)$ and $G \backslash v$ is disconnected, then v is called a *cut vertex*.

If $e \in E(G)$ and $G \backslash e$ is disconnected, then e is called a *cut edge*.



Cut vertices



Cut edges

Let G be a connected graph.

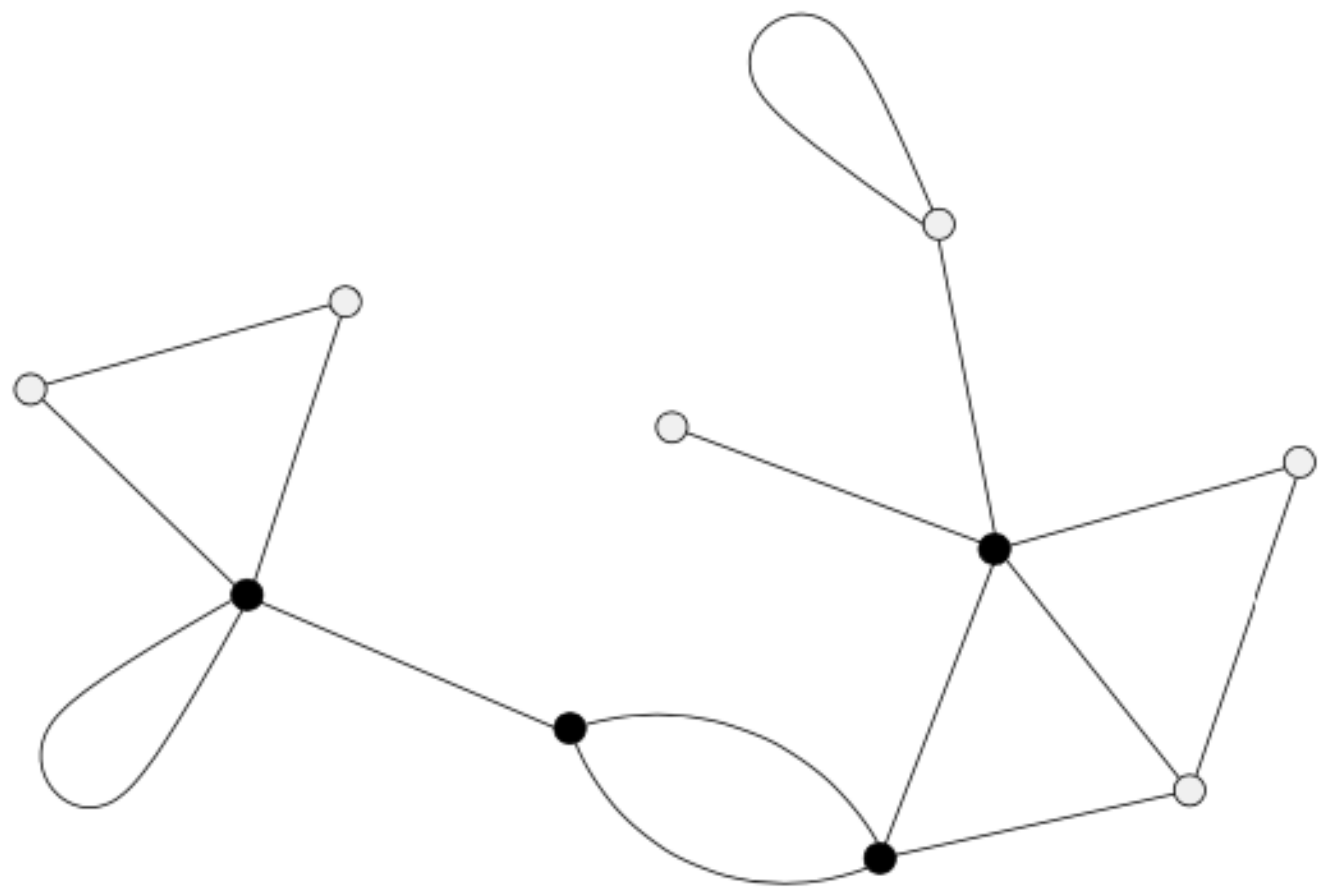
If $S \in V(G)$ and $G \backslash S$ is disconnected, then S is called a *cut set*.

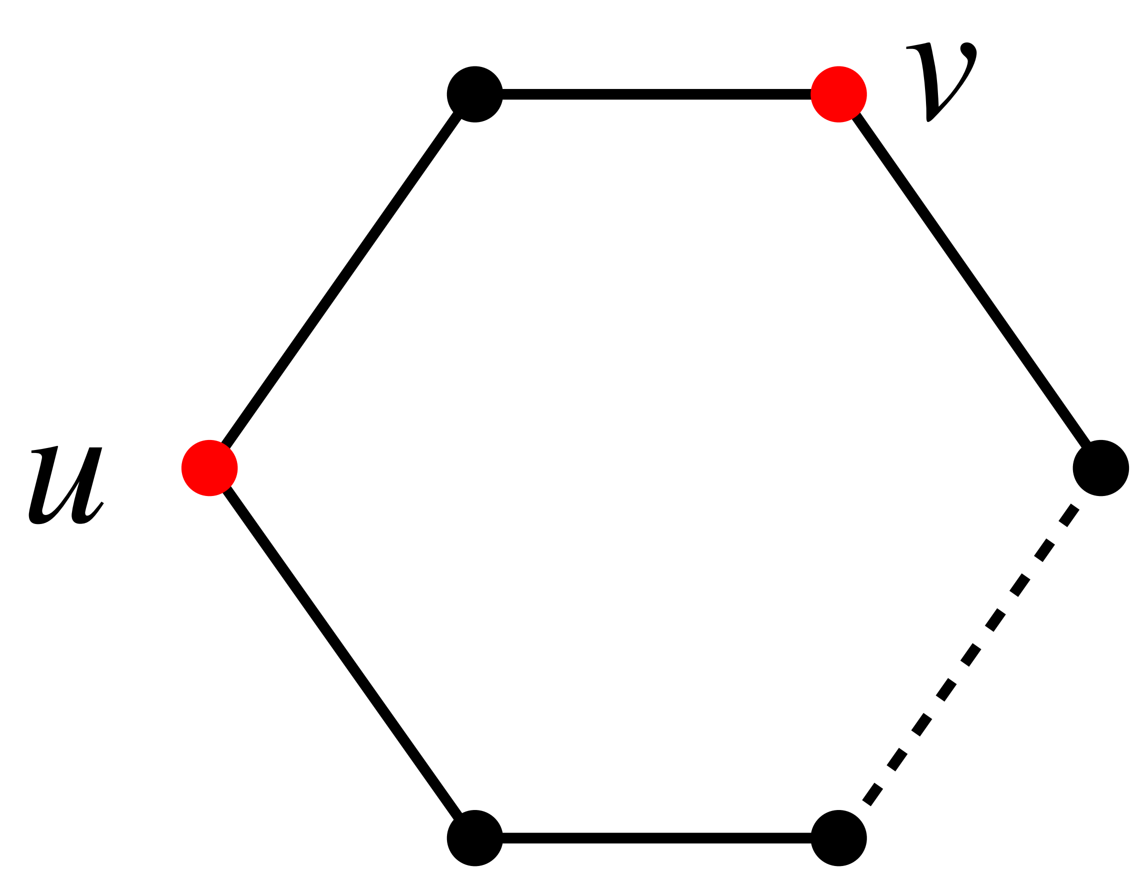
For a connected graph G with at least two nonadjacent vertices, define

$$\kappa(G) = \min\{\, |S| : S \text{ is a cut set of } G\},$$

called *connectivity* (连通度) of G.

A graph G is called *k-connected* if $\kappa(G) \geq k$.

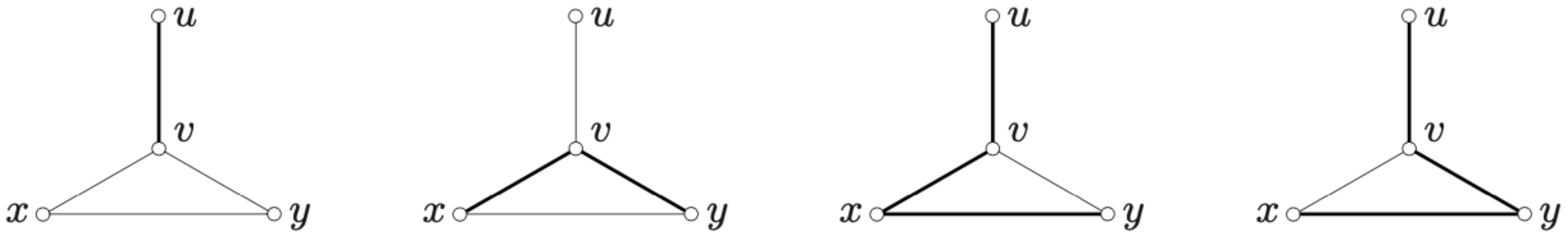




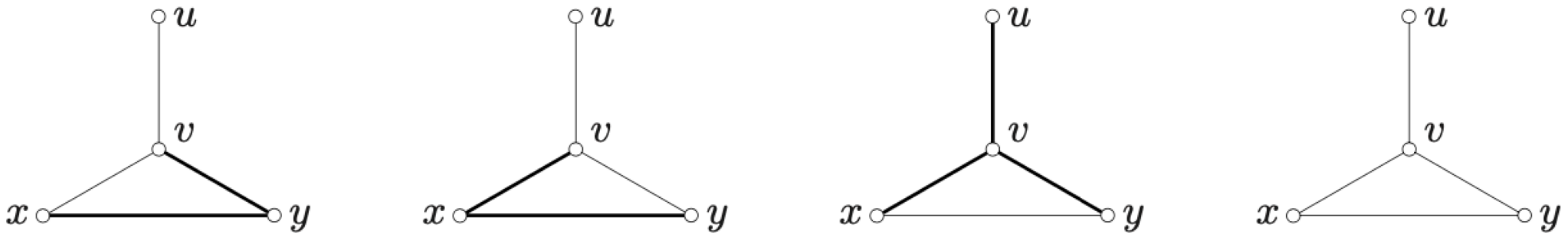
Let G be a graph. If $X \subset V(G)$ and $Y = V(G) \backslash X$, then the edge set

$$E(X, Y) = \{uv : uv \in E(G), u \in X \text{ and } v \in Y\}$$

is called the *edge cut* of G associated with X, write as $\partial(X)$.



$\partial(u)$ $\quad$ $\partial(u,v)$ $\quad$ $\partial(u,x)$ $\quad$ $\partial(u,y)$



$\partial(u,v,x)$ $\quad$ $\partial(u,v,y)$ $\quad$ $\partial(u,x,y)$ $\quad$ $\partial(u,v,x,y)$

Let T be a rooted tree with root r. For any $v \in V(T)$,

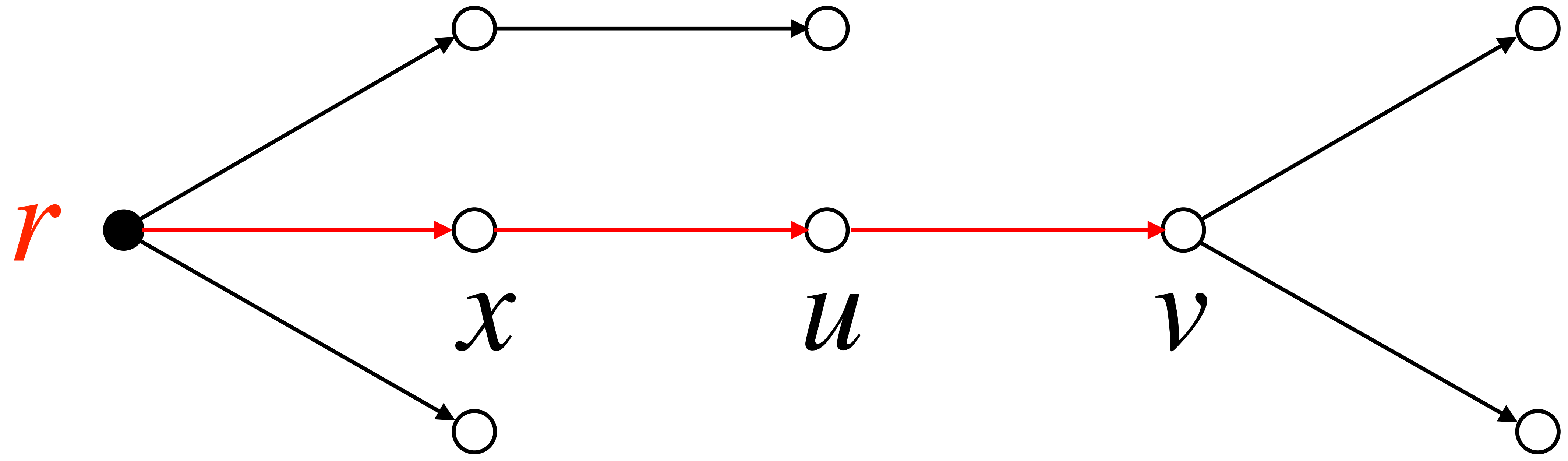
rTv denotes the unique path connecting r and v,

$\ell(v)$ is the length of rTv,

$p(v)$ is the predecessor of v in rTv (from the root to v) for any $V \in V(T) \backslash r$.

The (oriented) edge set of a rooted tree $T = (V(T), E(T))$ is determined by its predecessor function p:

$$E(T) = \left\{ p(v)v : v \in V(T) \backslash r \right\}.$$



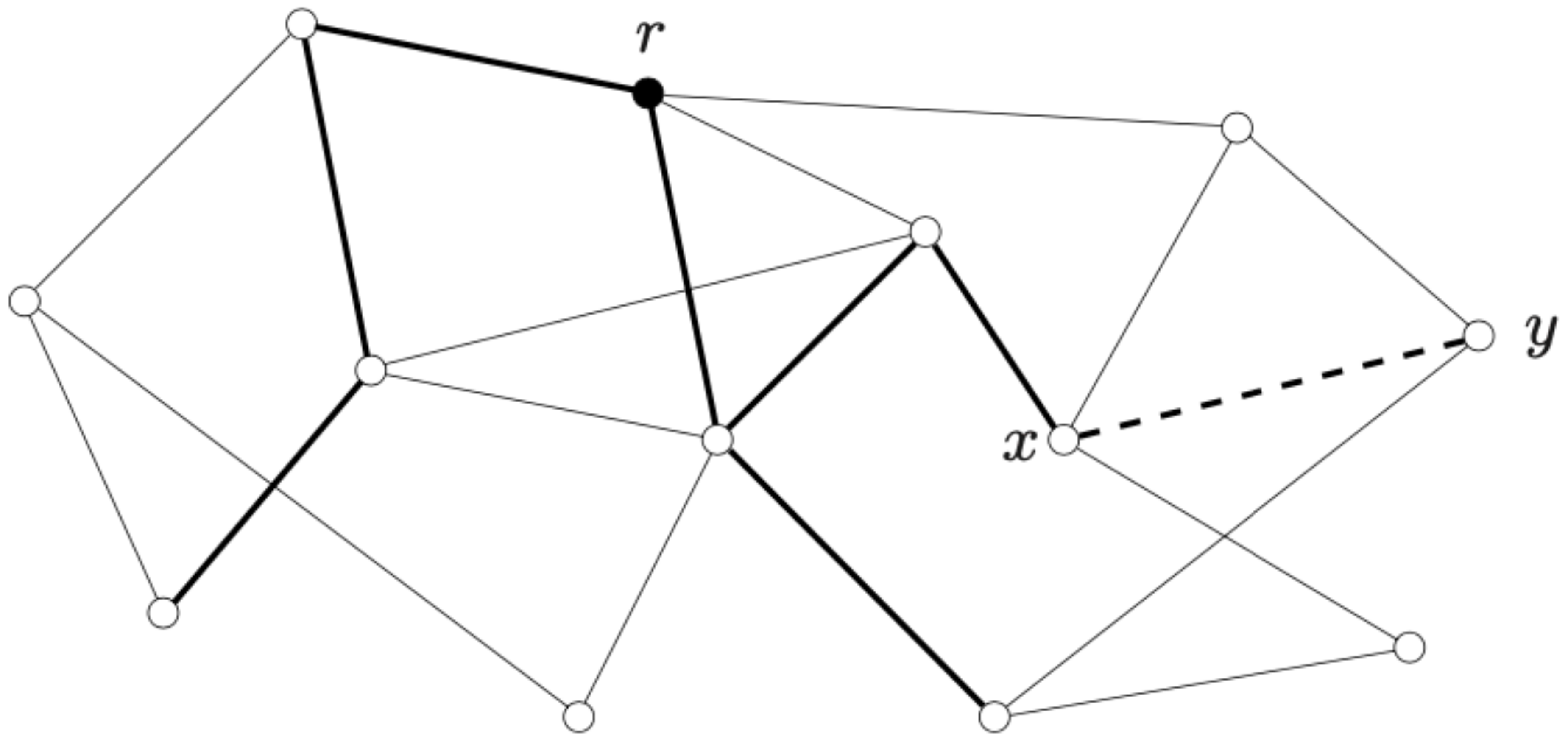
Trees in Graphs

How to determine whether a graph is connected?

Let T be a tree in a graph G.

If $V(T) = V(G)$, then T is a spanning tree of G, and so G is connected.

If $V(T) \subset V(G)$, there are two possibilities: either $\partial(T) = \emptyset$, in which case G is disconnected, or $\partial(T) \neq \emptyset$. In the latter case, for any edge $xy \in \partial(T)$, where $x \in V(T)$ and $y \in V(G) \backslash V(T)$, $T + xy$ is again a tree in G.



Breadth-First Search Algorithm (BFS)

1. Starting with $i = 0,\ Q = \emptyset$;

2. Take $T = r$, color r black; Set $t(r) = i + 1, \ell(r) = 0, Q = r$;

3. If $Q \neq \emptyset$, check the head x of the queue Q:

(1) If x has an uncolored neighbor y, then color y black, set

$$t(y) = i + 1, \ell(y) = \ell(x) + 1,\ p(y) = x,$$

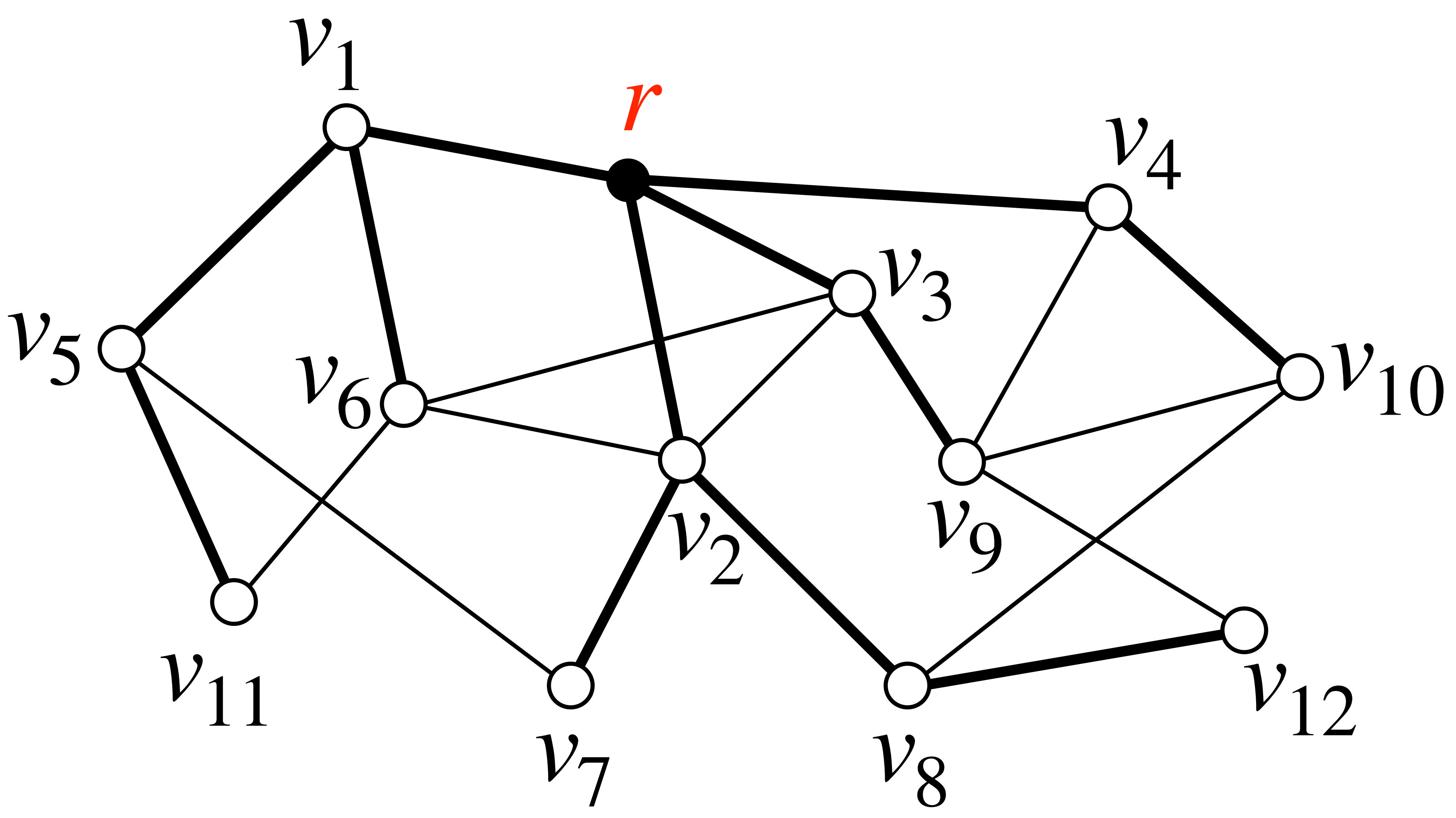
and append y to Q;

(2) If x has no uncolored neighbors, then remove x from Q.

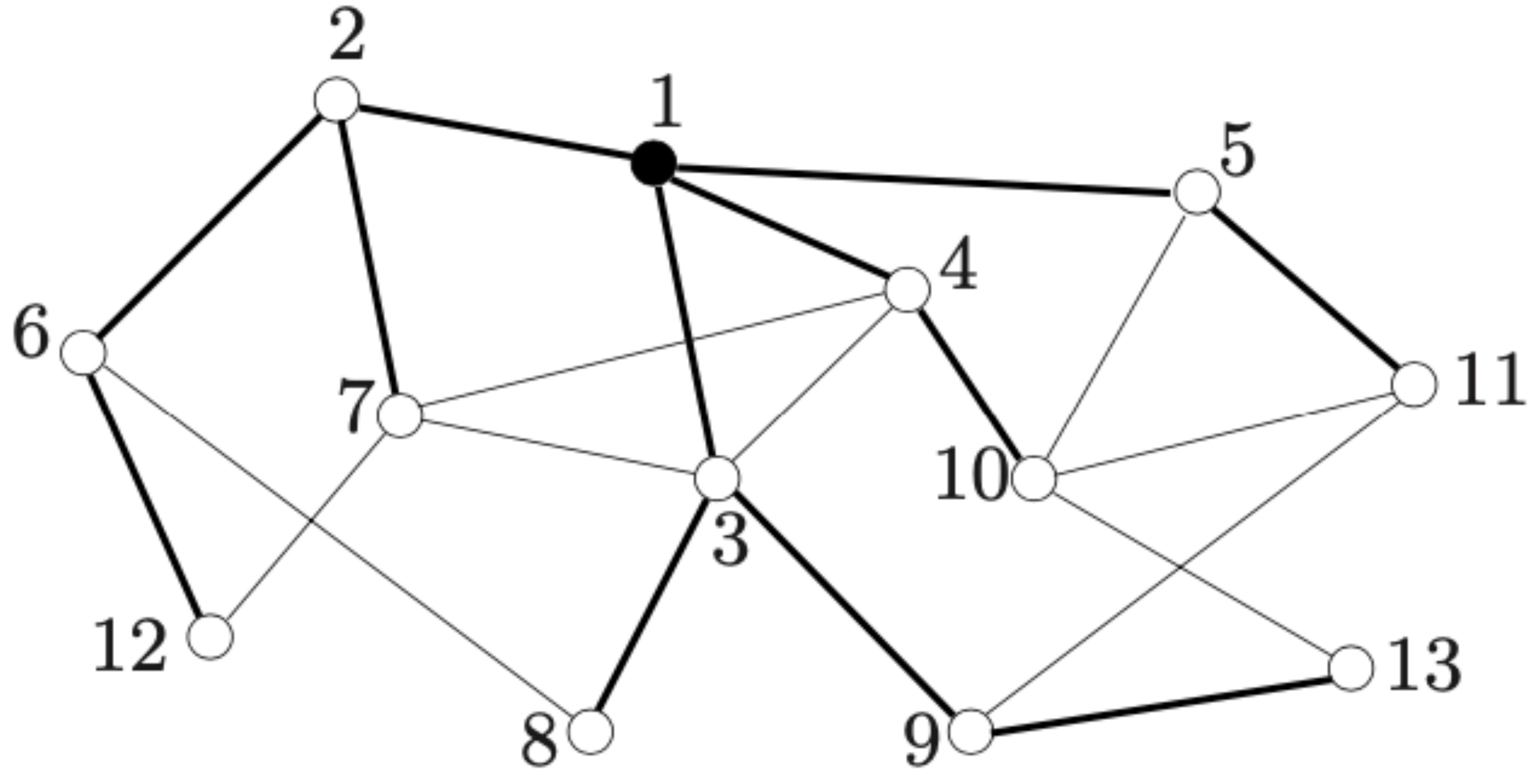
4. Repeat **3** until $Q = \emptyset$.

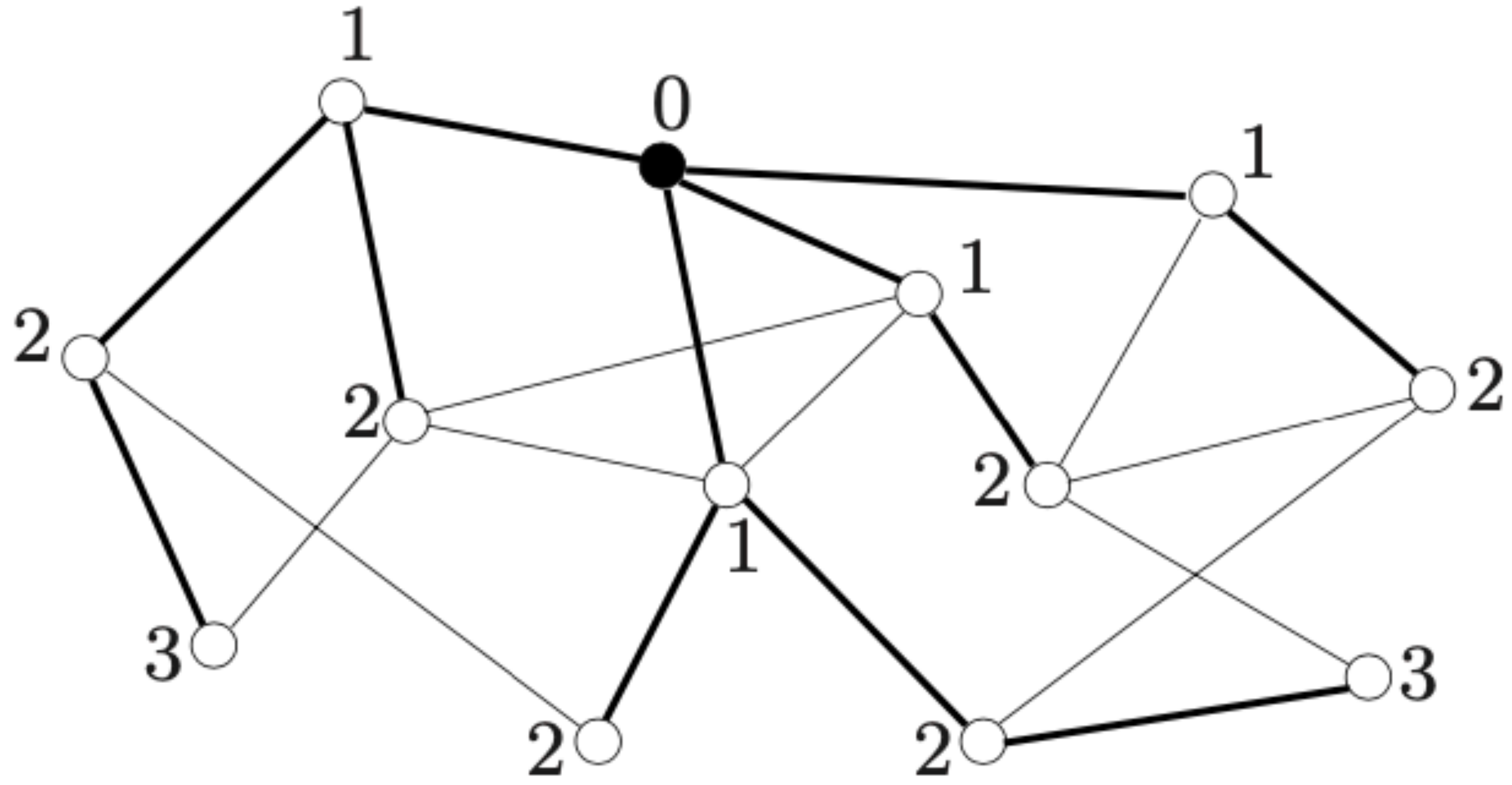
This algorithm (BFS) outputs a spanning tree.

The tree obtained by BFS algorithm is called a BFS-tree.



$$\varnothing \to r \to rv_1 \to rv_1v_2 \to rv_1v_2v_3 \to rv_1v_2v_3v_4$$
$$\to v_1v_2v_3v_4 \to v_1v_2v_3v_4v_5 \to v_1v_2v_3v_4v_5v_6$$
$$\to v_2v_3v_4v_5v_6 \to v_2v_3v_4v_5v_6v_7 \to v_2v_3v_4v_5v_6v_7v_8$$
$$\to v_3v_4v_5v_6v_7v_8 \to v_3v_4v_5v_6v_7v_8v_9 \to v_4v_5v_6v_7v_8v_9$$
$$\to v_4v_5v_6v_7v_8v_9v_{10} \to v_5v_6v_7v_8v_9v_{10}$$
$$\to v_5v_6v_7v_8v_9v_{10}v_{11} \to v_6v_7v_8v_9v_{10}v_{11}$$
$$\to v_7v_8v_9v_{10}v_{11} \to v_8v_9v_{10}v_{11} \to v_8v_9v_{10}v_{11}v_{12}$$
$$\to v_9v_{10}v_{11}v_{12} \to v_{10}v_{11}v_{12} \to v_{11}v_{12} \to v_{12} \to \varnothing$$





Theorem 7. Let T be a BFS-tree of a connected graph G, with root r. Then

(1) For any $v \in V(G)$, $\ell(v) = d_T(v, r)$.

(2) For any $uv \in E(G)$, $|\ell(u) - \ell(v)| \leq 1$.

Proof. (1) By the definition of $\ell(v)$, it is the length of the unique path rTv in T connecting r and v, and so the result follows.

(2) If u and v are any two vertices such that $\ell(u) < \ell(v)$, then u joined Q before v.

Assume that $uv \in E(G)$ and $\ell(u) < \ell(v)$. If $u = p(v)$, then $\ell(u) = \ell(v) - 1$. If not, set $y = p(v)$. Because v was added to T by the edge yv, and not by the edge uv, the vertex y joined Q before u, hence $\ell(y) \leq \ell(u)$ by the argument above. Therefore, $\ell(v) - 1 = \ell(y) \leq \ell(u) \leq \ell(v) - 1$, which implies that $\ell(u) = \ell(v) - 1$.

Theorem 8. Let T be a BFS-tree of a connected graph G, with root r, and $\ell(v)$ be the level function by BFS algorithm. Then

$$\ell(v) = d_G(v, r) \text{ for all } v \in V(G).$$

Proof. Obviously, $\ell(v) = d_T(r, v) \geq d_G(r, v)$.

We show $\ell(v) \leq d_G(r, v)$ by induction on the length of a shortest (r, v)-path.

Let P be a shortest (r, v)-path in G, where $v \neq r$, and let u be the predecessor of v on P. Then rPu is a shortest (r, u)-path, and $d_G(r, u) = d_G(r, v) - 1$. By induction, $\ell(u) \leq d_G(r, u)$, and by Theorem 7, $\ell(v) - \ell(u) \leq 1$. Therefore,

$$\ell(v) \leq \ell(u) + 1 = d_G(r, u) + 1 = d_G(r, v).$$

Depth-First Search Algorithm (DFS)

1. Starting with $i = 0,\ S = \varnothing$;

2. Take $T = r$, color r black; Set $f(r) = i + 1$, $S = r$;

3. If $S \neq \varnothing$, check the tail x of the queue S:

(1) If x has an uncolored neighbor y, then color y black, set

$$f(y) = i + 1,\ p(y) = x,$$

and append y to S;

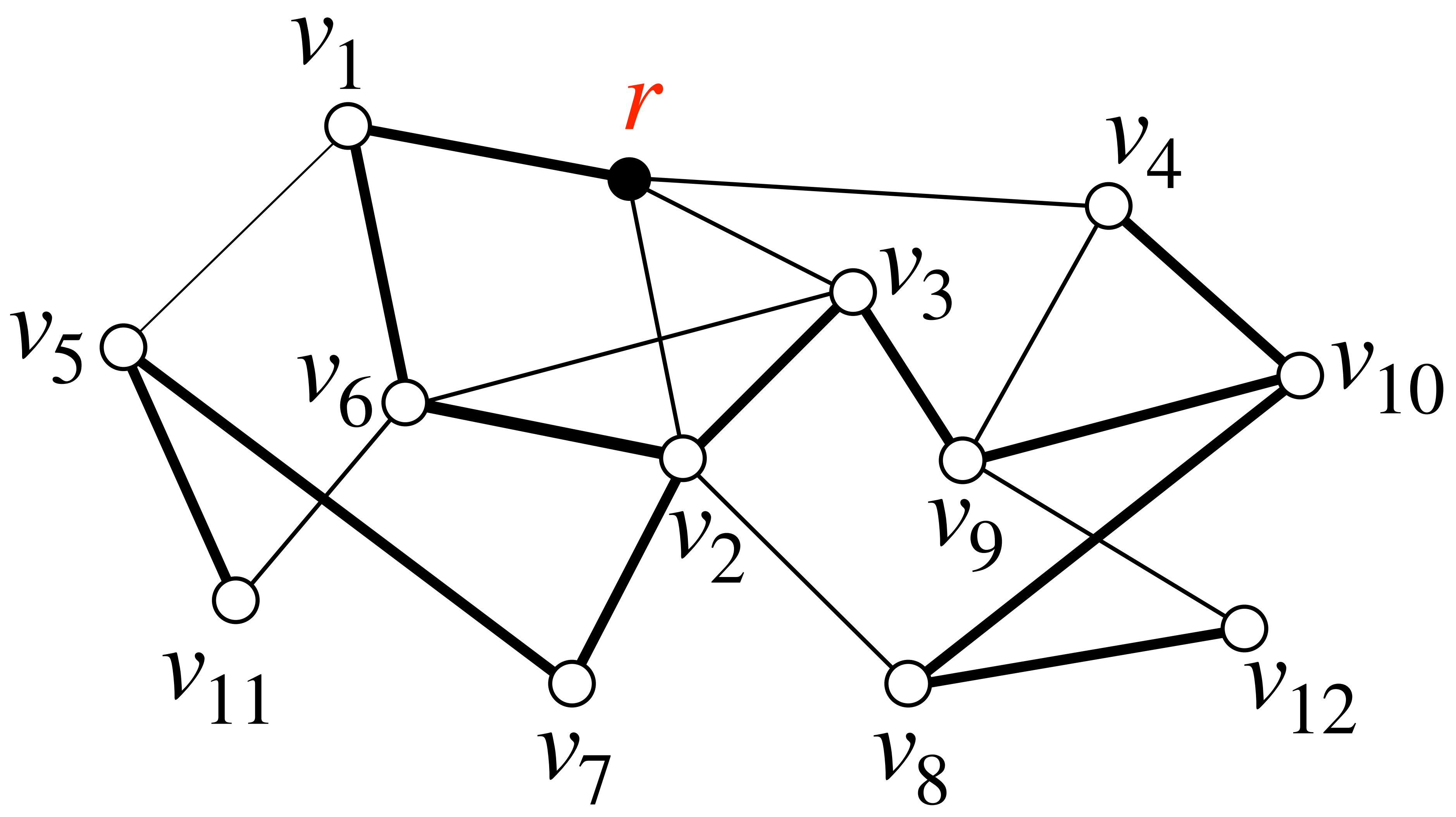
(2) If x has no uncolored neighbors, then remove x from S and set

$$\ell(x) = i + 1.$$

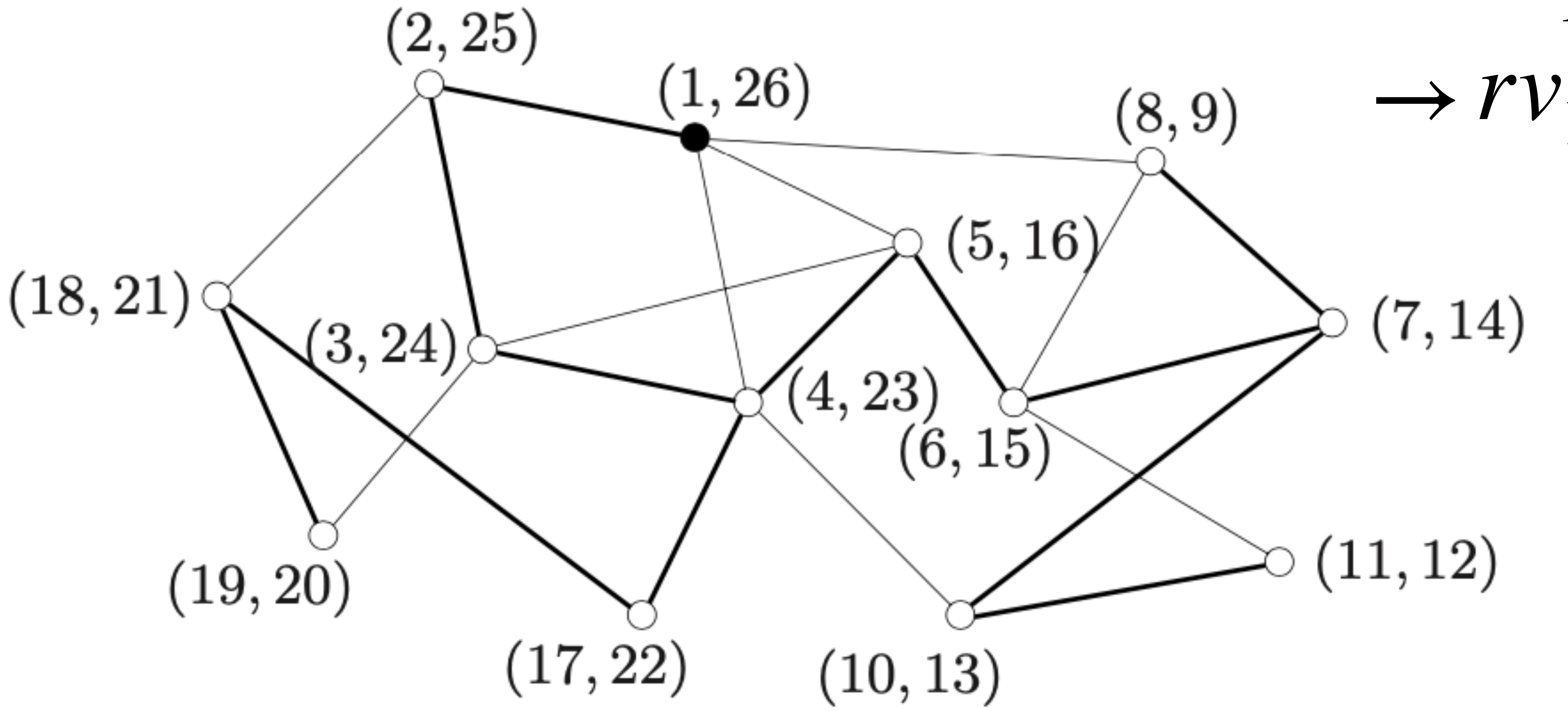
4. Repeat **3** until $S = \varnothing$.

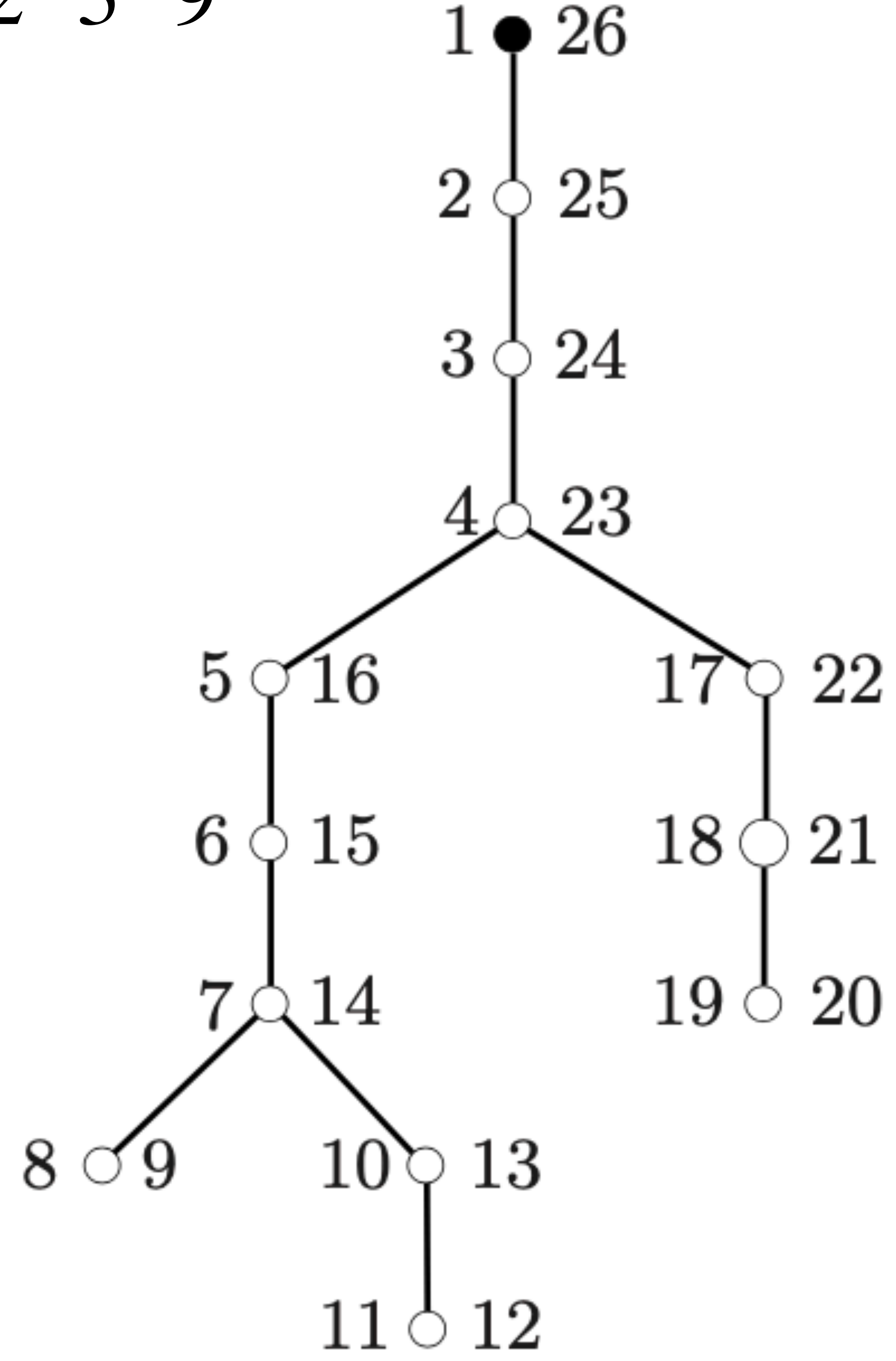
This algorithm (DFS) outputs a spanning tree.

The tree obtained by DFS algorithm is called a DFS-tree.



$$\varnothing \to r \to rv_1 \to rv_1v_6 \to rv_1v_6v_2 \to rv_1v_6v_2v_3$$
$$\to rv_1v_6v_2v_3v_9 \to rv_1v_6v_2v_3v_9v_{10} \to rv_1v_6v_2v_3v_9v_{10}v_4$$
$$\to rv_1v_6v_2v_3v_9v_{10} \to rv_1v_6v_2v_3v_9v_{10}v_8$$
$$\to rv_1v_6v_2v_3v_9v_{10}v_8v_{12} \to rv_1v_6v_2v_3v_9v_{10}v_8$$
$$\to rv_1v_6v_2v_3v_9v_{10} \to rv_1v_6v_2v_3v_9$$
$$\to rv_1v_6v_2v_3 \to rv_1v_6v_2$$
$$\to rv_1v_6v_2v_7 \to rv_1v_6v_2v_7v_5$$
$$\to rv_1v_6v_2v_7v_5v_{11}$$
$$\to rv_1v_6v_2v_7v_5 \to rv_1v_6v_2v_7$$
$$\to rv_1v_6v_2 \to rv_1v_6$$
$$\to rv_1 \to r \to \varnothing$$





The following proposition provides a link between the input G, its DFS-tree T, and the two time functions $f(v)$ and $\ell(v)$ returned by DFS algorithm.

Proposition 3. Let u and v be two vertices of G, with $f(u) < f(v)$.

(1) If $uv \in E(G)$, then $\ell(v) < \ell(u)$.

(2) u is an ancestor of v in T if and only if $\ell(v) < \ell(u)$.

Theorem 9. Let T be a DFS-tree of a connected graph G.

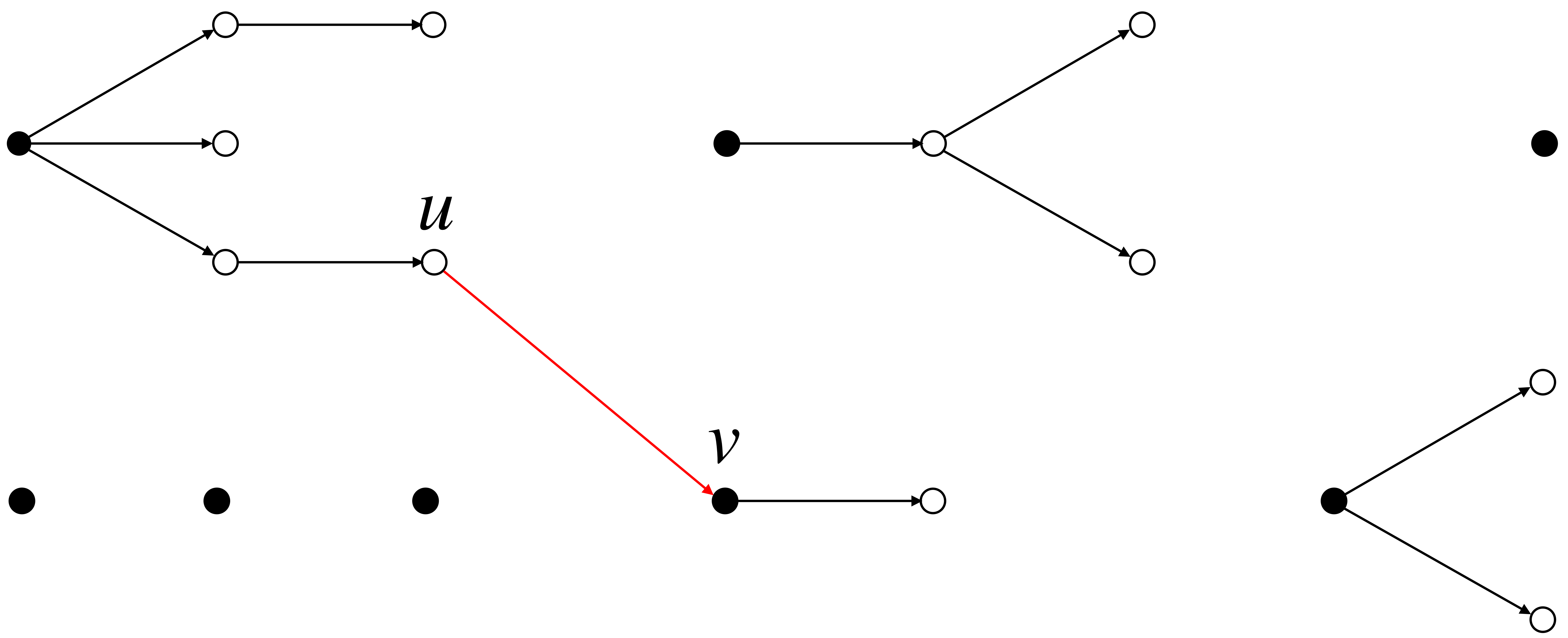
The root r of T is a cut vertex of G if and only if it has at least two children.

Any other vertex $v \neq r$ of T is a cut vertex of G if and only if it has a child no descendant of which is adjacent to a proper ancestor of the vertex v.

Theorem 10(Cayley's Formula). The number of labelled trees on n vertices is n^{n-2}.

Proof. We first show the number of labelled branchings on n vertices is n^{n-1}. To see this, consider the number of ways in which a labelled branching can be built up: Starting with an empty graph on n vertices, add one edge at a time. In order to end up with a branching, the subgraph constructed at each stage must be a branching forest. Initially, this branching forest has n components, each is an isolated vertex. At each stage, the number of components decreases by one.

If there are k components, the number of choices for the new edge (u, v) is $n(k-1)$: any one of the n vertices may be taken as u, whereas v must be the root of one of the $k-1$ components which do not contain u.



The total number of ways of constructing a branching on n vertices in this way is

$$\prod_{i=1}^{n-1} n(n-i) = n^{n-1}(n-1)!$$

Notice that any individual branching on n vertices is constructed exactly $(n-1)!$ times by this procedure, once for each of the orders in which its $n-1$ edges are selected, the number of labelled branchings on n vertices is

$$n^{n-1}.$$

Moreover, since each labelled spanning tree on n vertices corresponding to exactly n labelled branchings on n vertices, the result follows.

For a general graph, we have the following simple recursive formula to count the number of spanning trees of a graph G.

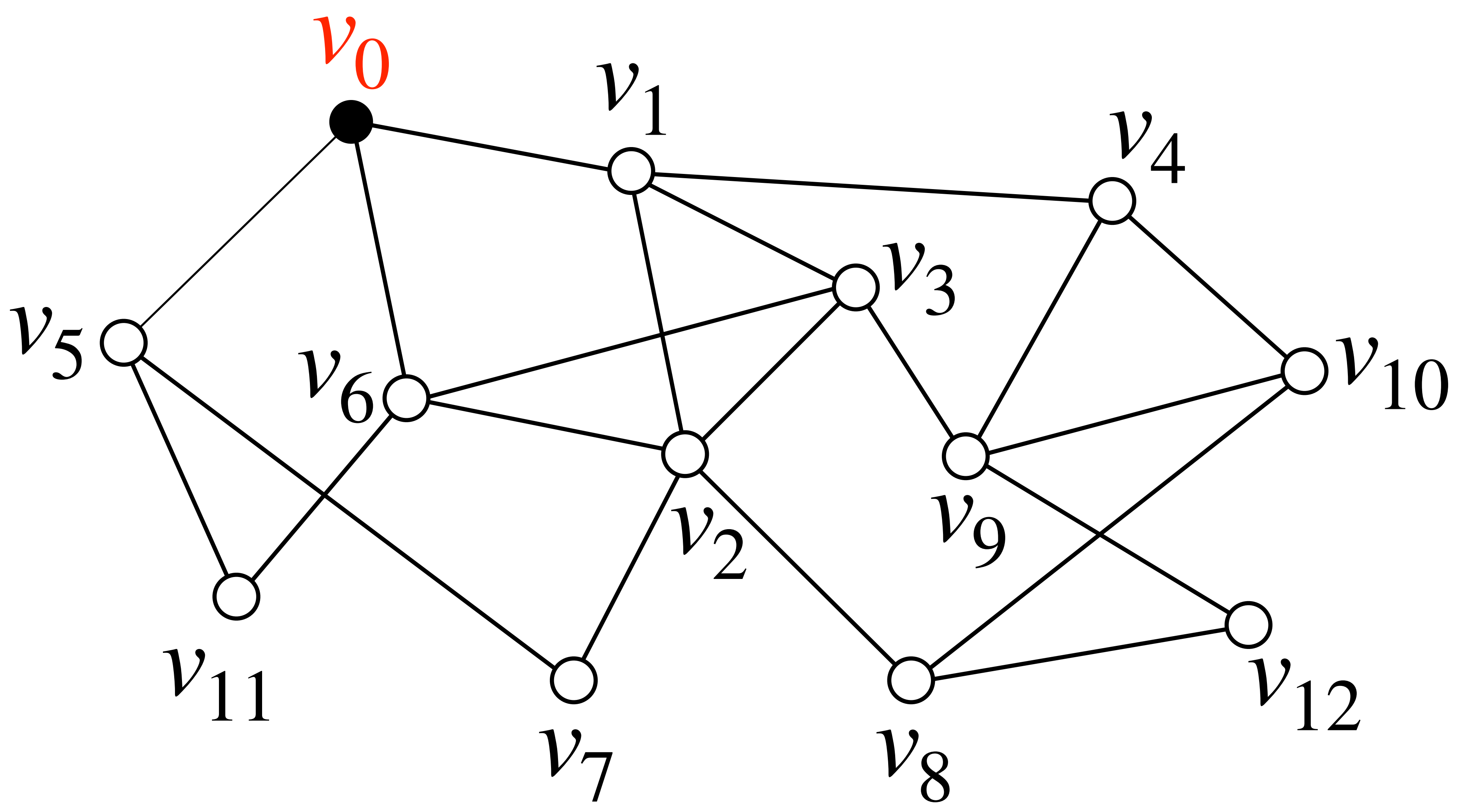
Proposition 4. Let G be a graph and e an edge of G. Then

$$t(G) = t(G\backslash e) + t(G/e).$$

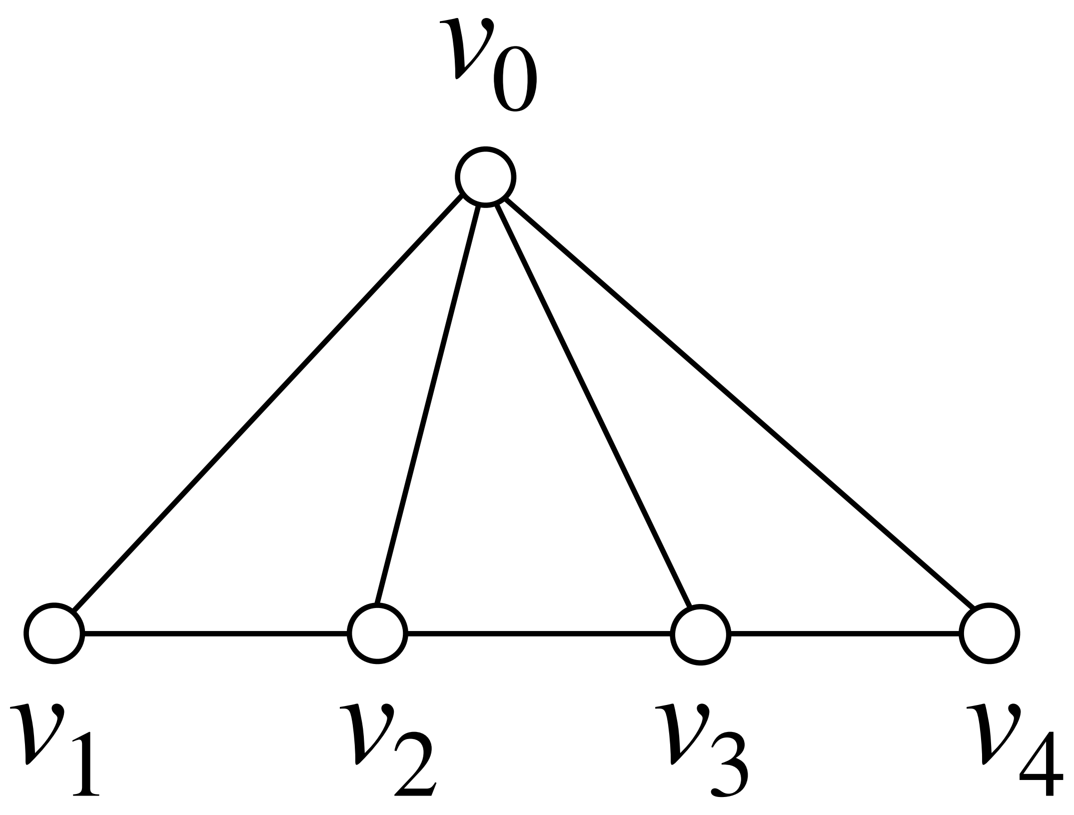
Exercise 2.

1. Find a BFS-tree and a DFS-tree in the graph G, take v_0 as root.

2. Count the number of spanning trees in the graph F.



G



F