Label-based Tree Representation

Reuven Cohen[1] Pierre Fraigniaud[2] David Ilcinkas[2]
Amos Korman[1] David Peleg[1]

[1]Dept. of Computer Science, Weizmann Institute, Israel

[2]CNRS, LRI, Université Paris-Sud, France

LOCALITY '05
September 26, 2005

Brief view of the problem

Many (distributed) algorithms on graphs use spanning trees.

- Broadcast
- Convergecast
- Graph exploration
- *etc.*

Need for locality

Local knowledge of the tree.

Our goal

Minimize the total memory used to store the tree.

Brief view of the problem

Many (distributed) algorithms on graphs use spanning trees.

- Broadcast
- Convergecast
- Graph exploration
- *etc.*

Need for locality

Local knowledge of the tree.

Our goal

Minimize the total memory used to store the tree.

Brief view of the problem

Many (distributed) algorithms on graphs use spanning trees.

- Broadcast
- Convergecast
- Graph exploration
- *etc.*

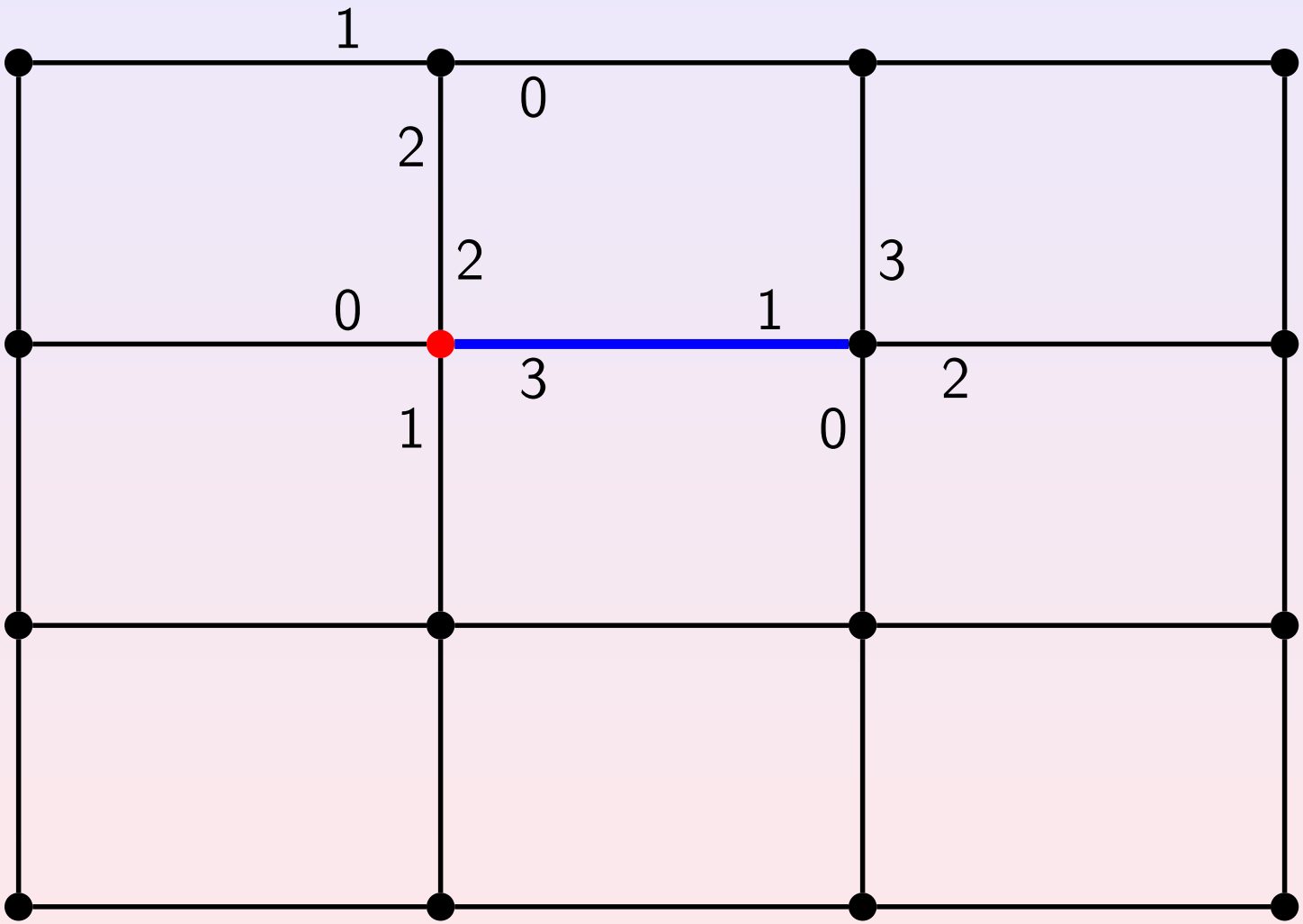
Need for locality

Local knowledge of the tree.

Our goal

Minimize the total memory used to store the tree.

Local labeling of the edges



Tree representations (1)

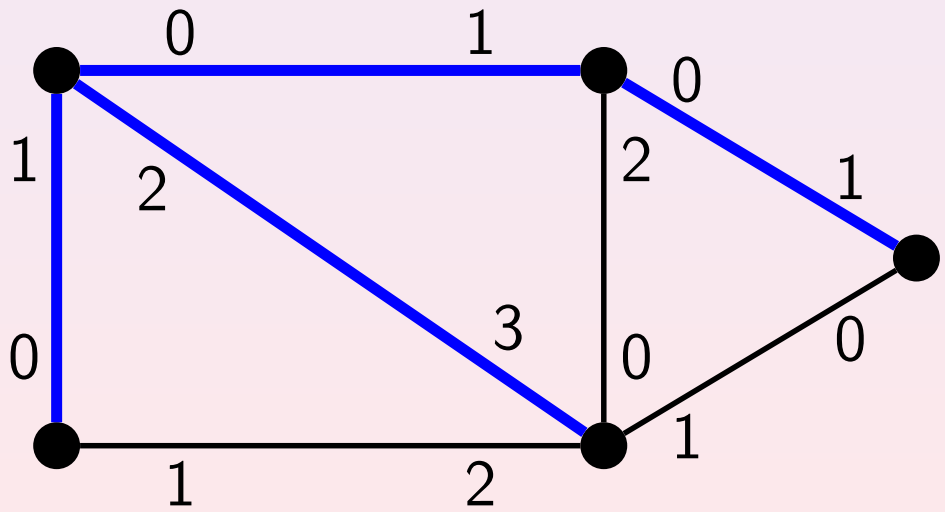
All-port tree representation

- Label: all port numbers of the incident tree edges
- Two measures for a spanning tree T
  - Maximum used space: $M_{all}(T)$
  - Total used space: $S_{all}(T)$

Tree representations (1)

All-port tree representation

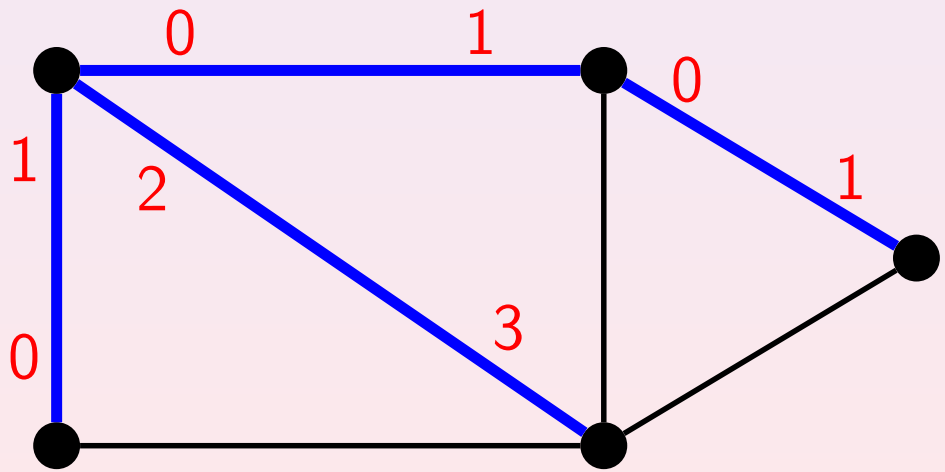
- Label: all port numbers of the incident tree edges
- Two measures for a spanning tree T
  - Maximum used space: $M_{all}(T)$
  - Total used space: $S_{all}(T)$



Tree representations (1)

All-port tree representation

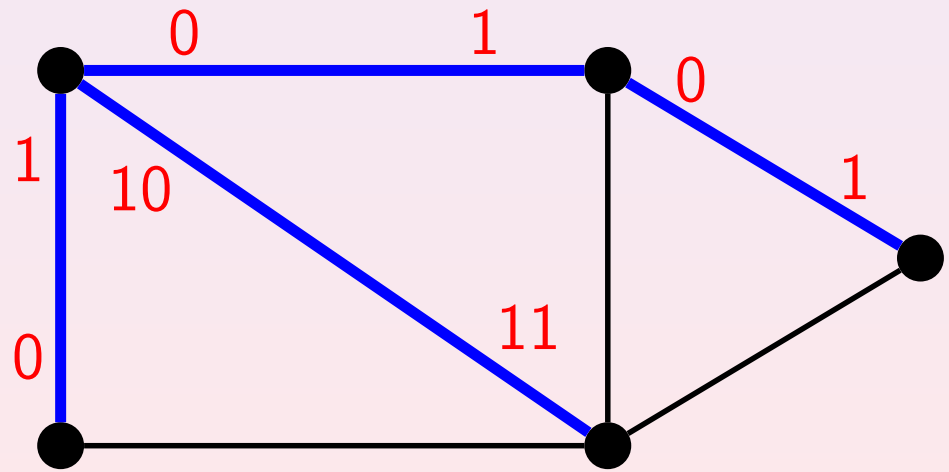
- Label: all port numbers of the incident tree edges
- Two measures for a spanning tree T
  - Maximum used space: $M_{all}(T)$
  - Total used space: $S_{all}(T)$



Tree representations (1)

All-port tree representation

- Label: all port numbers of the incident tree edges
- Two measures for a spanning tree T
  - Maximum used space: $M_{all}(T)$
  - Total used space: $S_{all}(T)$



$M_{all}(T) = 4$ bits

$S_{all}(T) = 10$ bits

Tree representations (2)

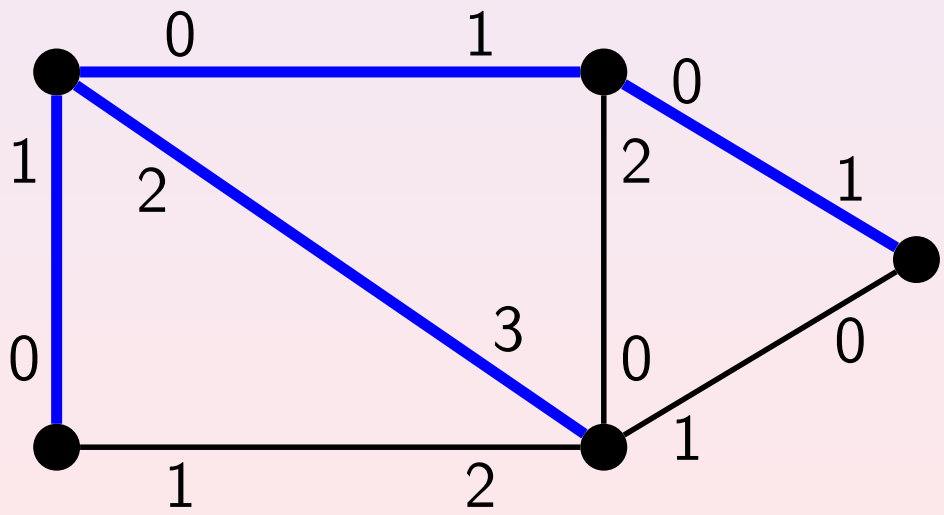
Upward tree representation

- Label: port number leading to the root
- Two measures for a spanning tree T
  - Maximum used space: $M_{up}(T)$
  - Total used space: $S_{up}(T)$

Tree representations (2)

Upward tree representation

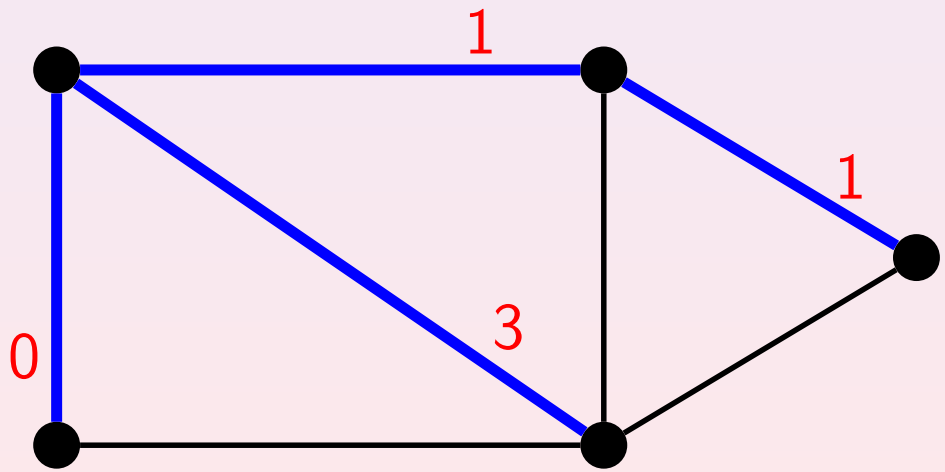
- Label: port number leading to the root
- Two measures for a spanning tree T
  - Maximum used space: $M_{up}(T)$
  - Total used space: $S_{up}(T)$



Tree representations (2)

Upward tree representation

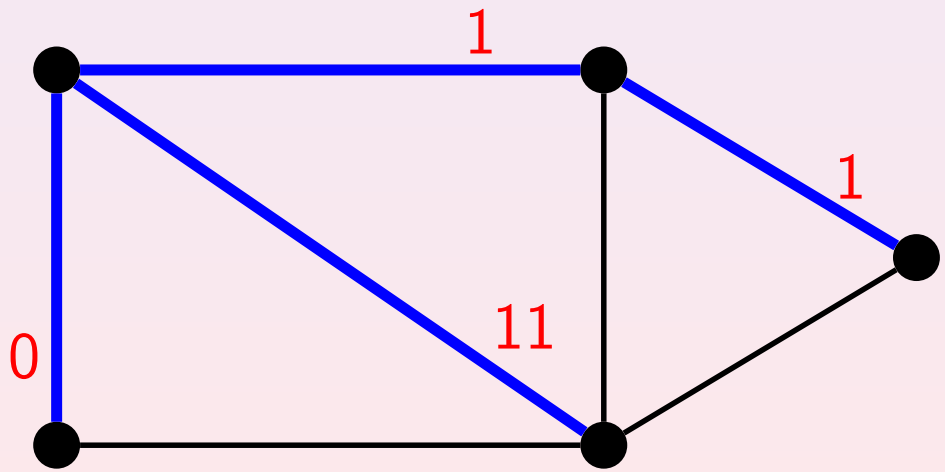
- Label: port number leading to the root
- Two measures for a spanning tree T
  - Maximum used space: $M_{up}(T)$
  - Total used space: $S_{up}(T)$



Tree representations (2)

Upward tree representation

- Label: port number leading to the root
- Two measures for a spanning tree T
  - Maximum used space: $M_{up}(T)$
  - Total used space: $S_{up}(T)$



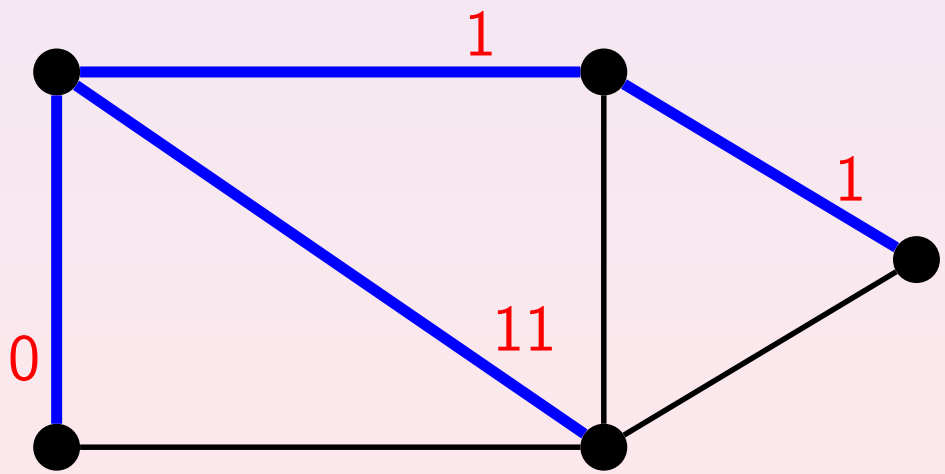
$M_{up}(T) = 2$ bits

$S_{up}(T) = 5$ bits

Tree representations (2)

Upward tree representation

- Label: port number leading to the root
- Two measures for a spanning tree T
  - Maximum used space: $M_{up}(T)$
  - Total used space: $S_{up}(T)$



$M_{up}(T) = 2$ bits

$S_{up}(T) = 5$ bits

Applications: upcast

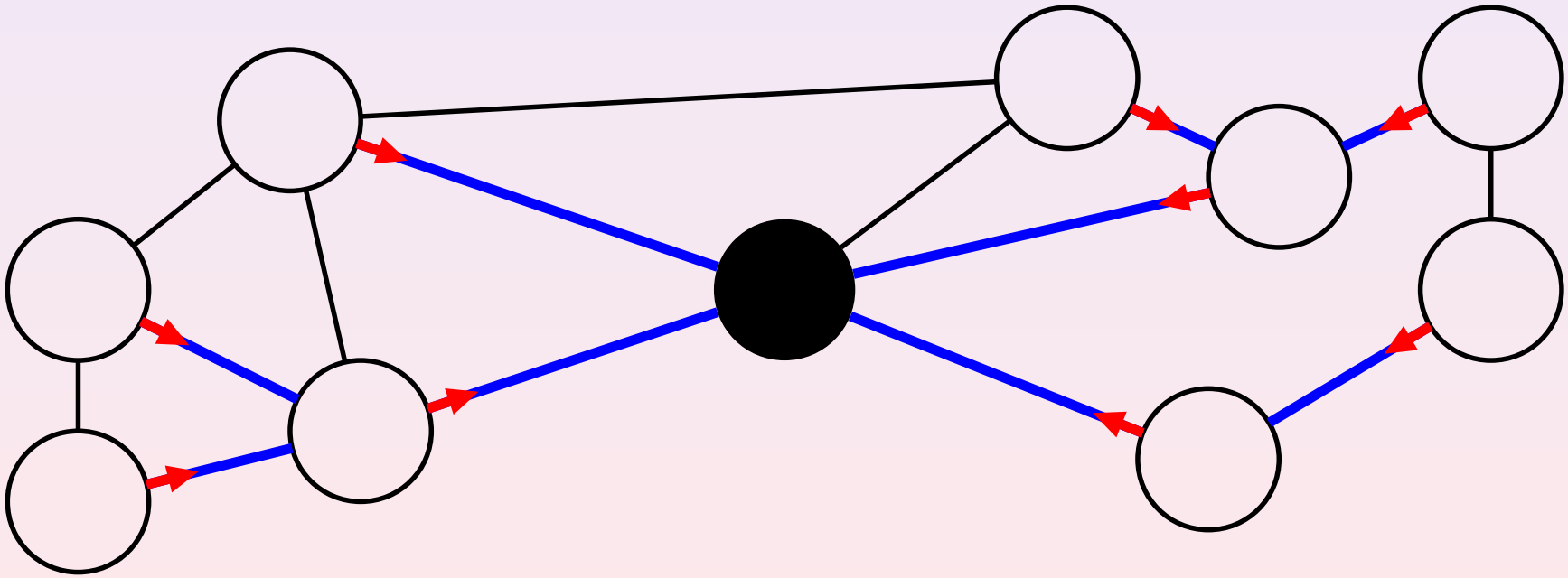
Goal

Send a message to the root.

Applications: upcast

Goal

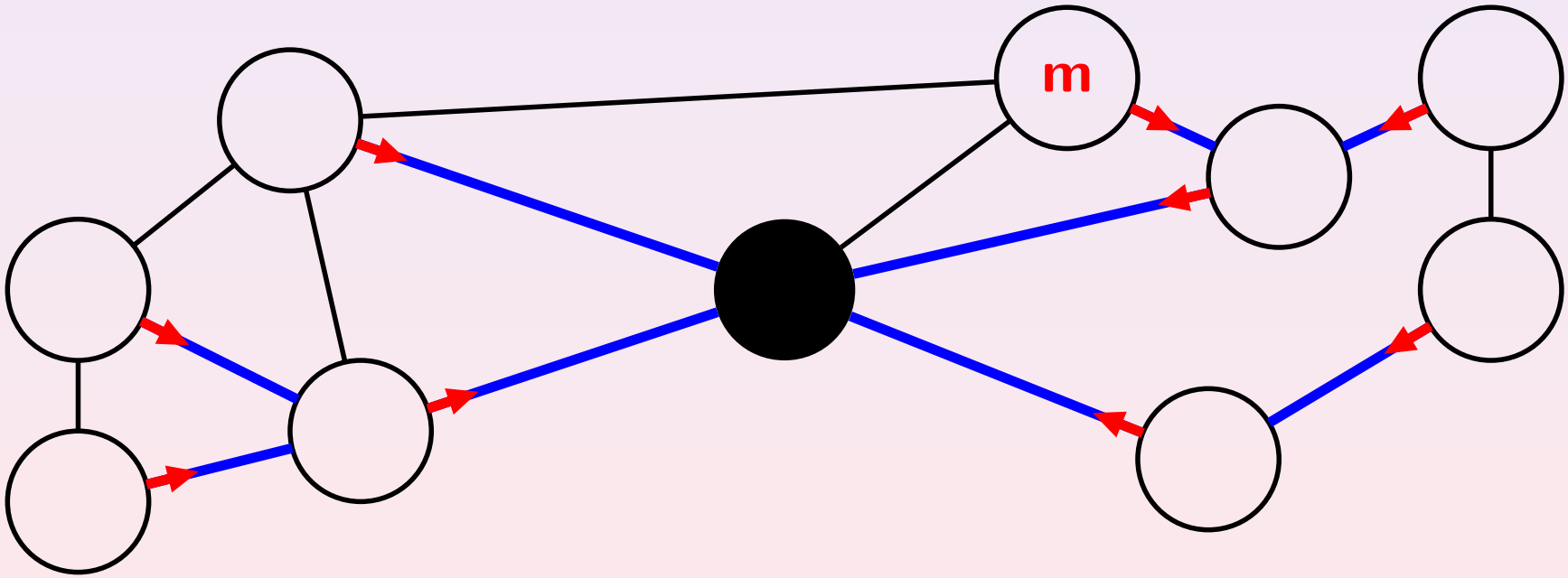
Send a message to the root.



Applications: upcast

Goal

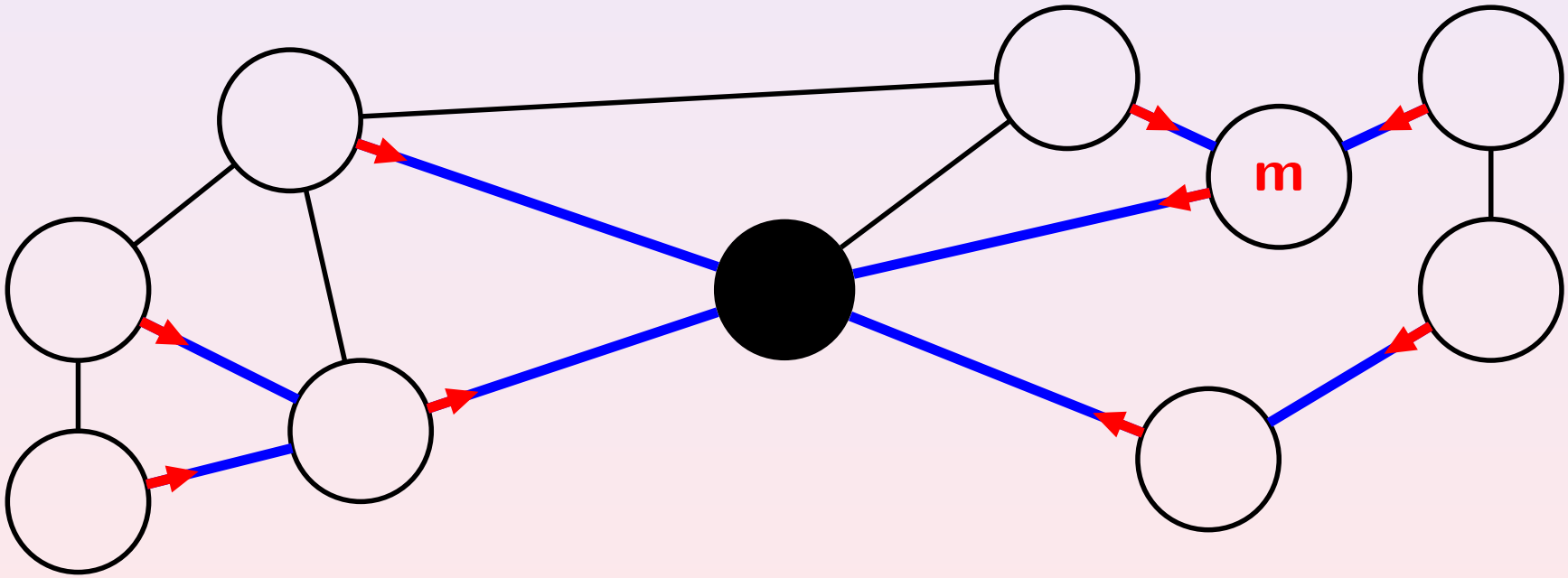
Send a message to the root.



Applications: upcast

Goal

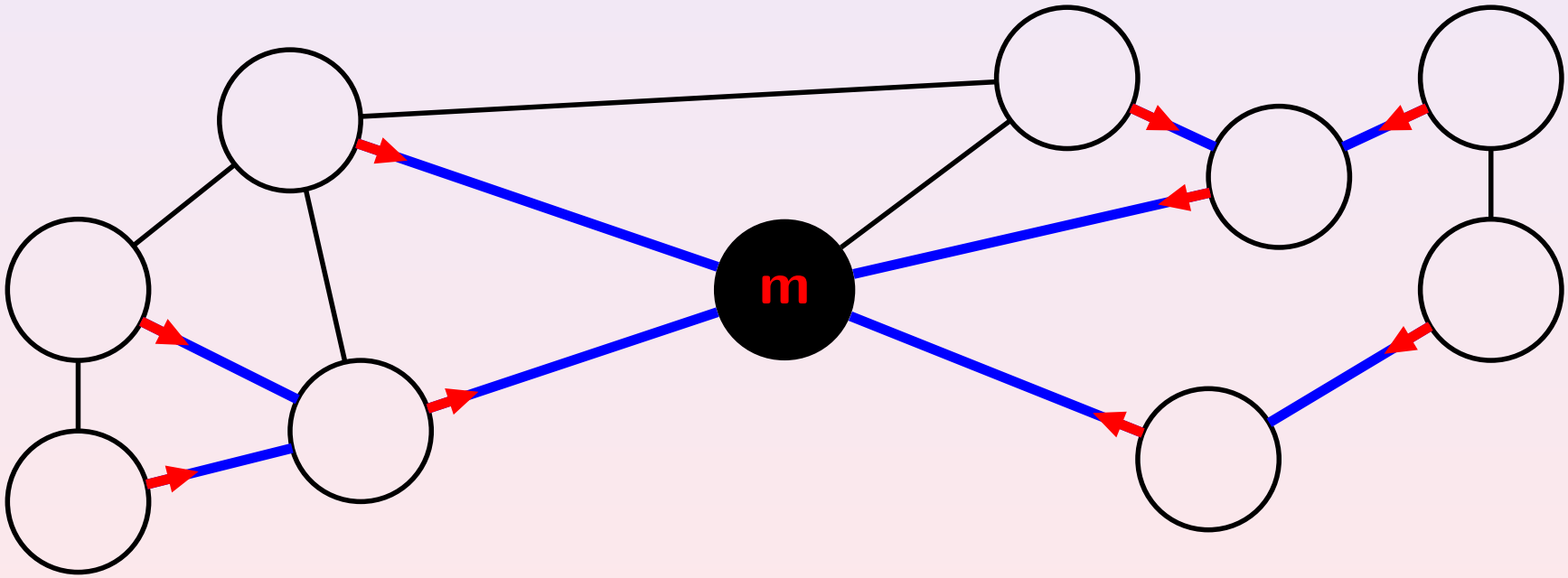
Send a message to the root.



Applications: upcast

Goal

Send a message to the root.



Applications: convergecast

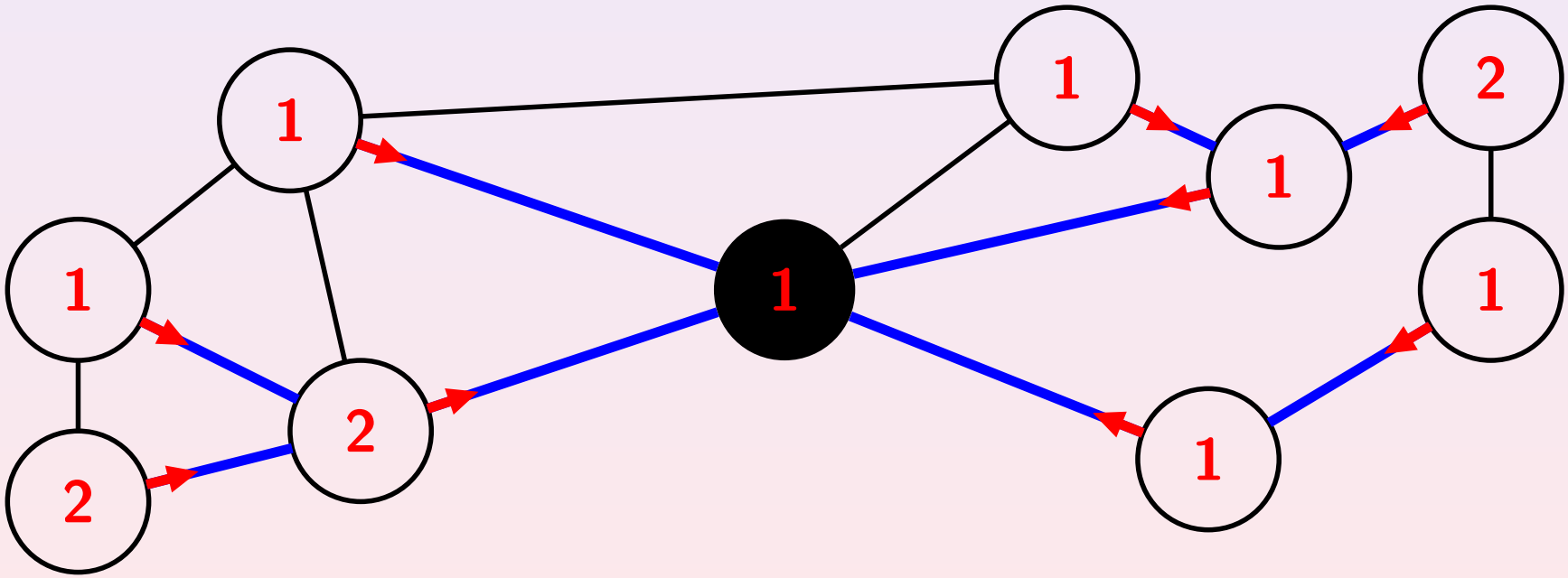
Goal

Collecting information to the root

Applications: convergecast

Goal

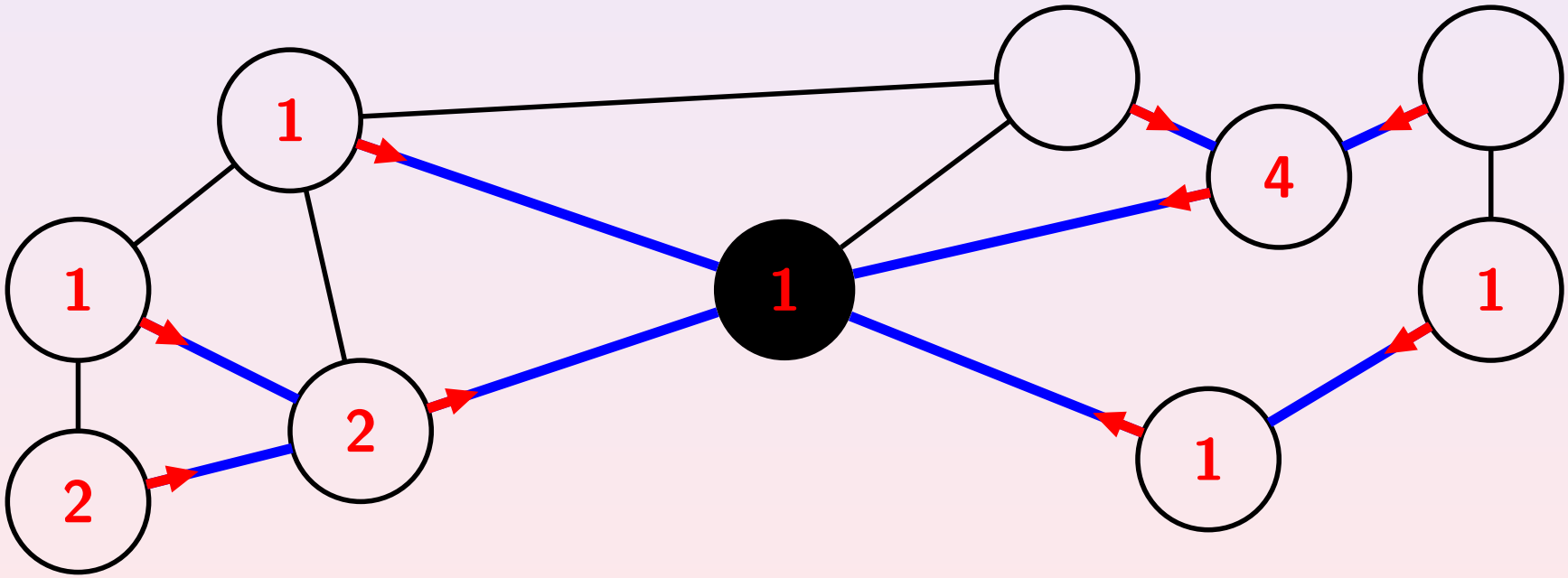
Collecting information to the root



Applications: convergecast

Goal

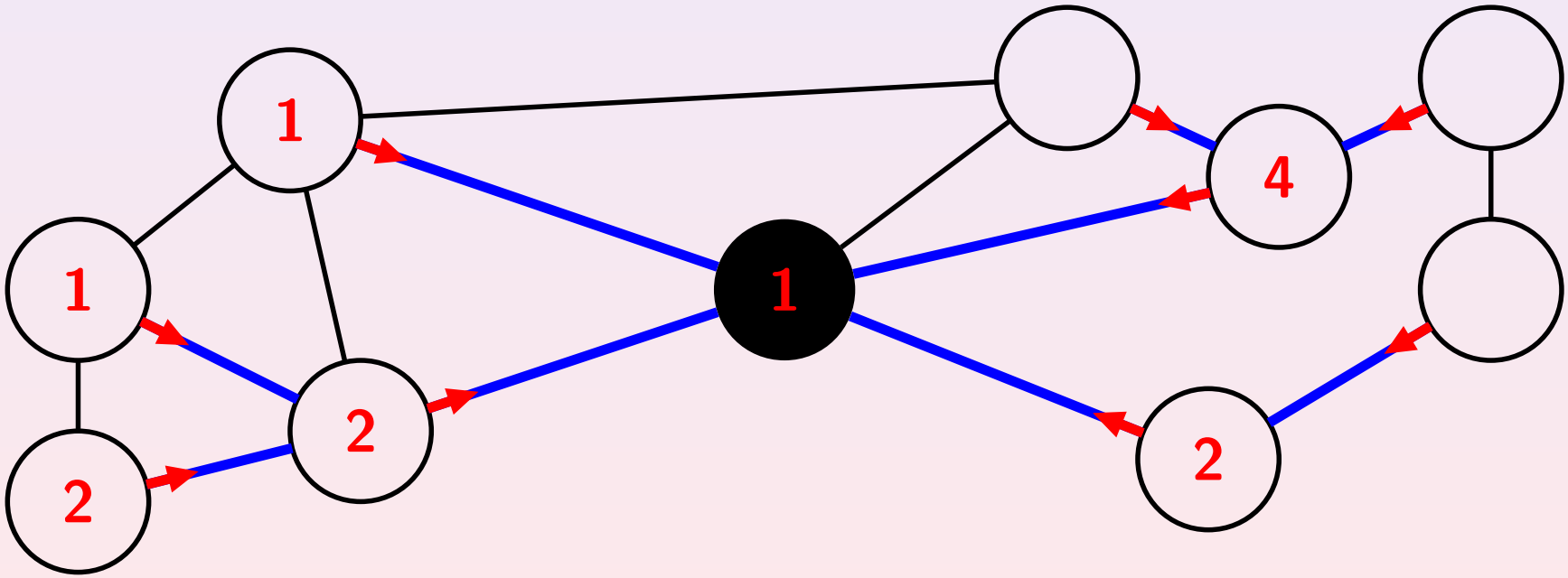
Collecting information to the root



Applications: convergecast

Goal

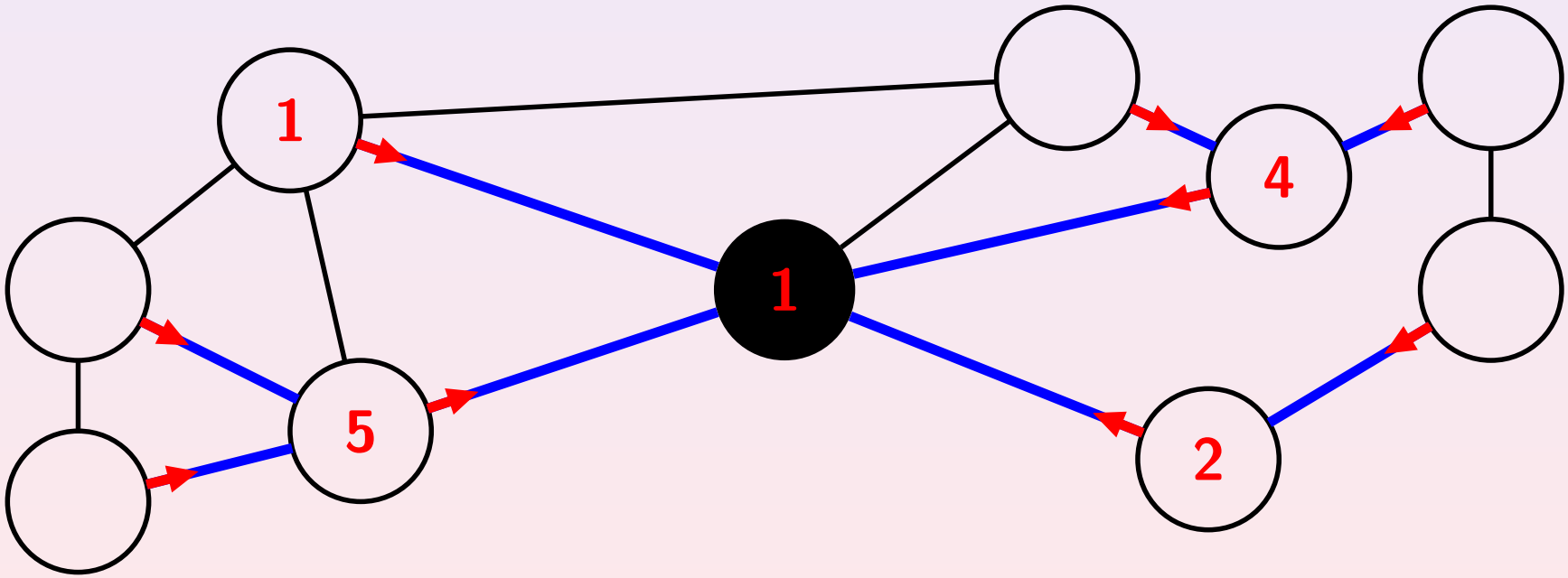
Collecting information to the root



Applications: convergecast

Goal

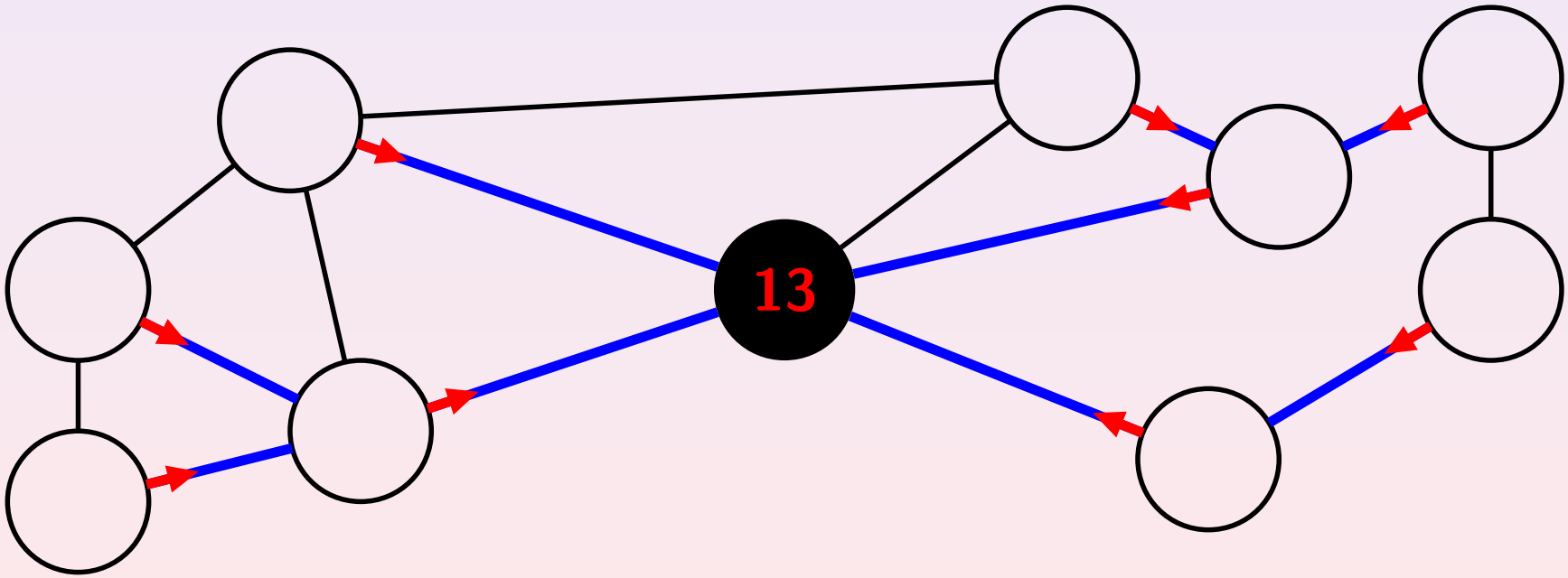
Collecting information to the root



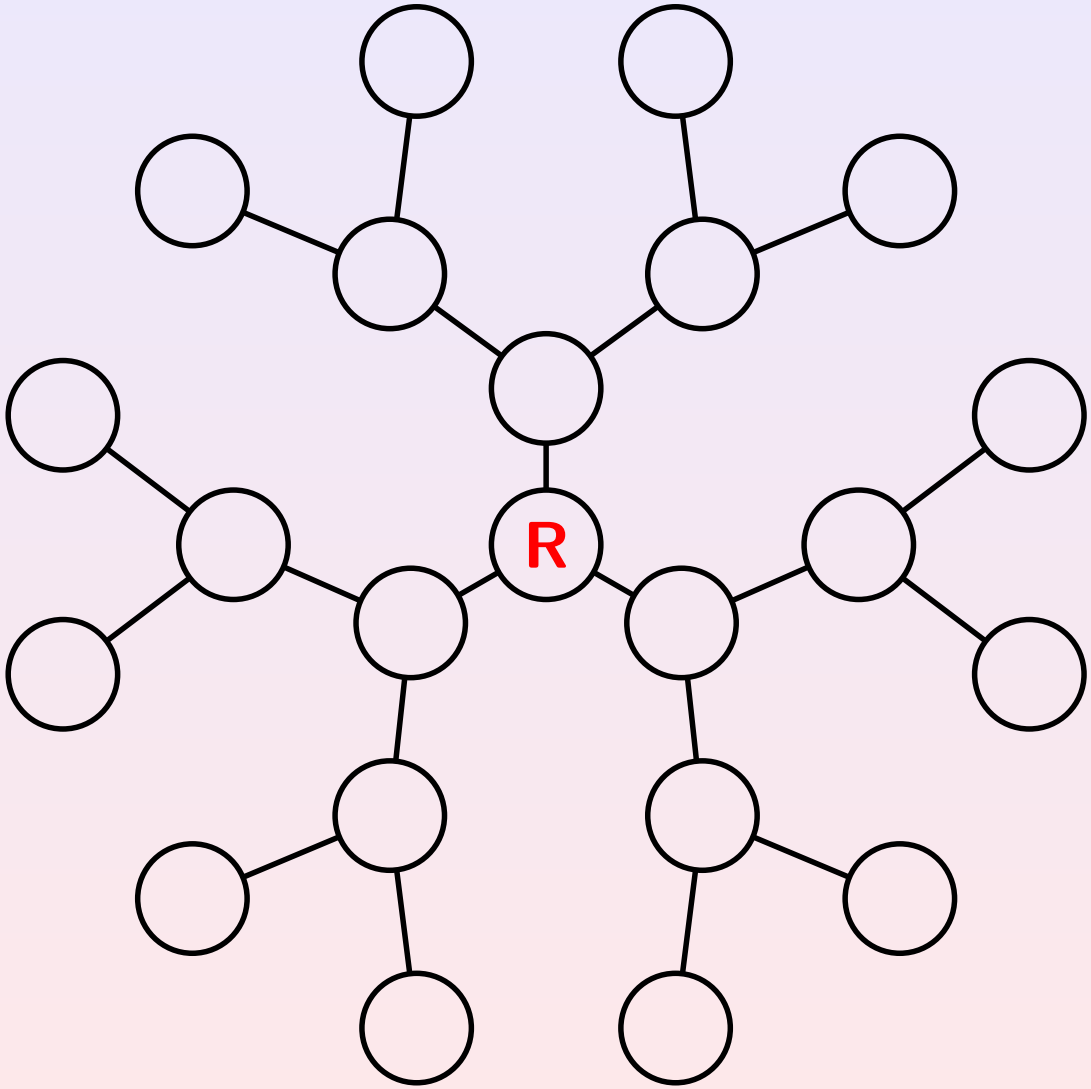
Applications: convergecast

Goal

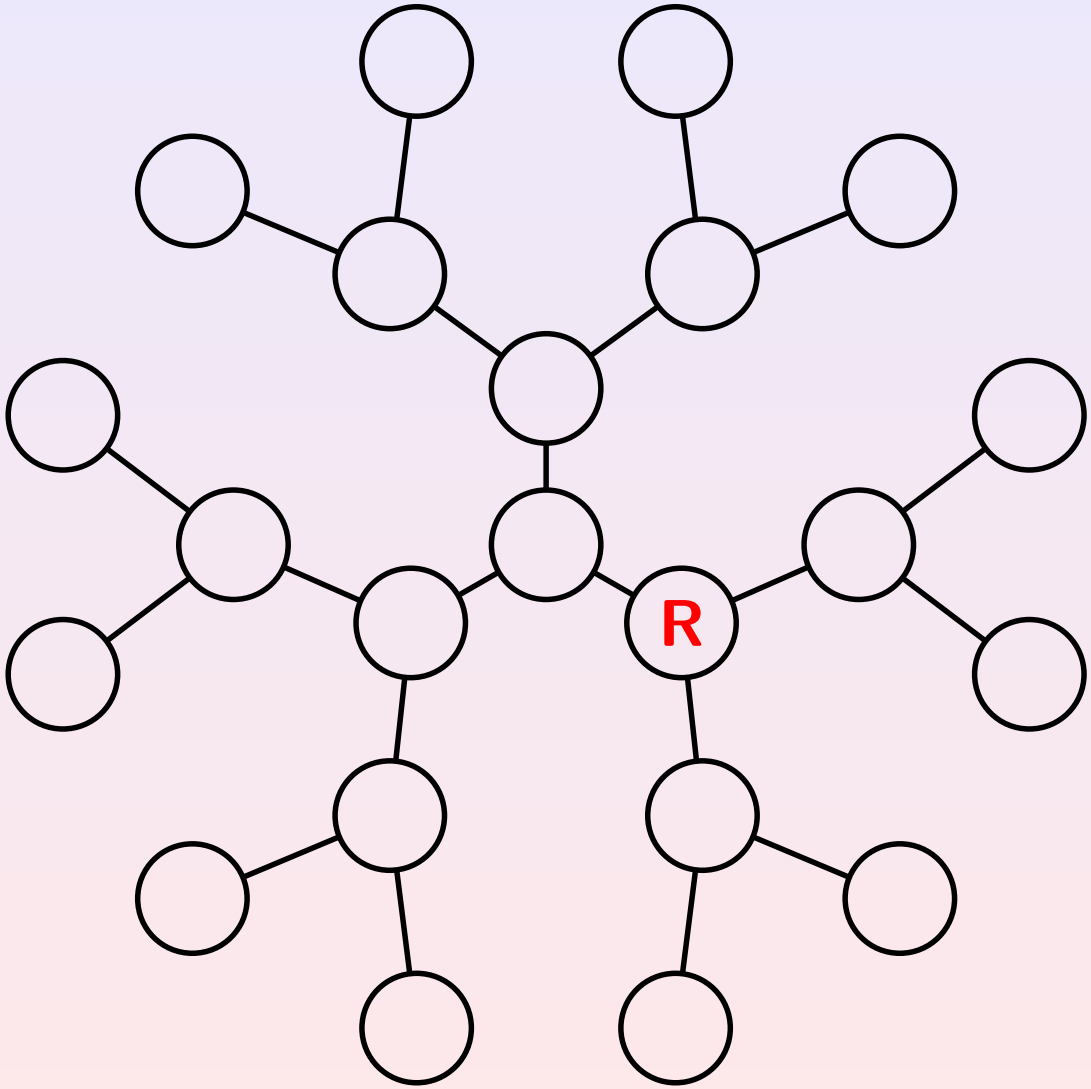
Collecting information to the root



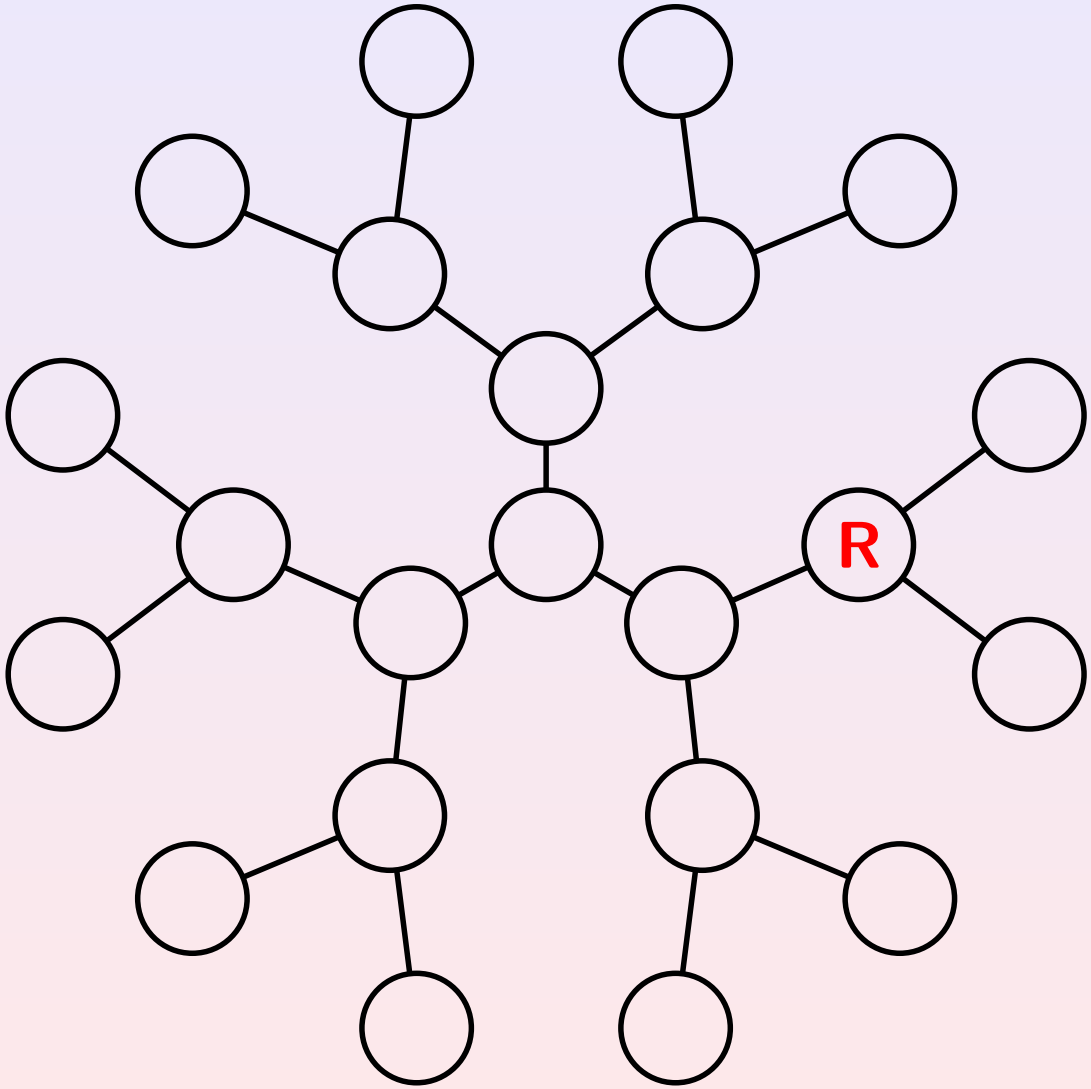
Applications: graph exploration (1)



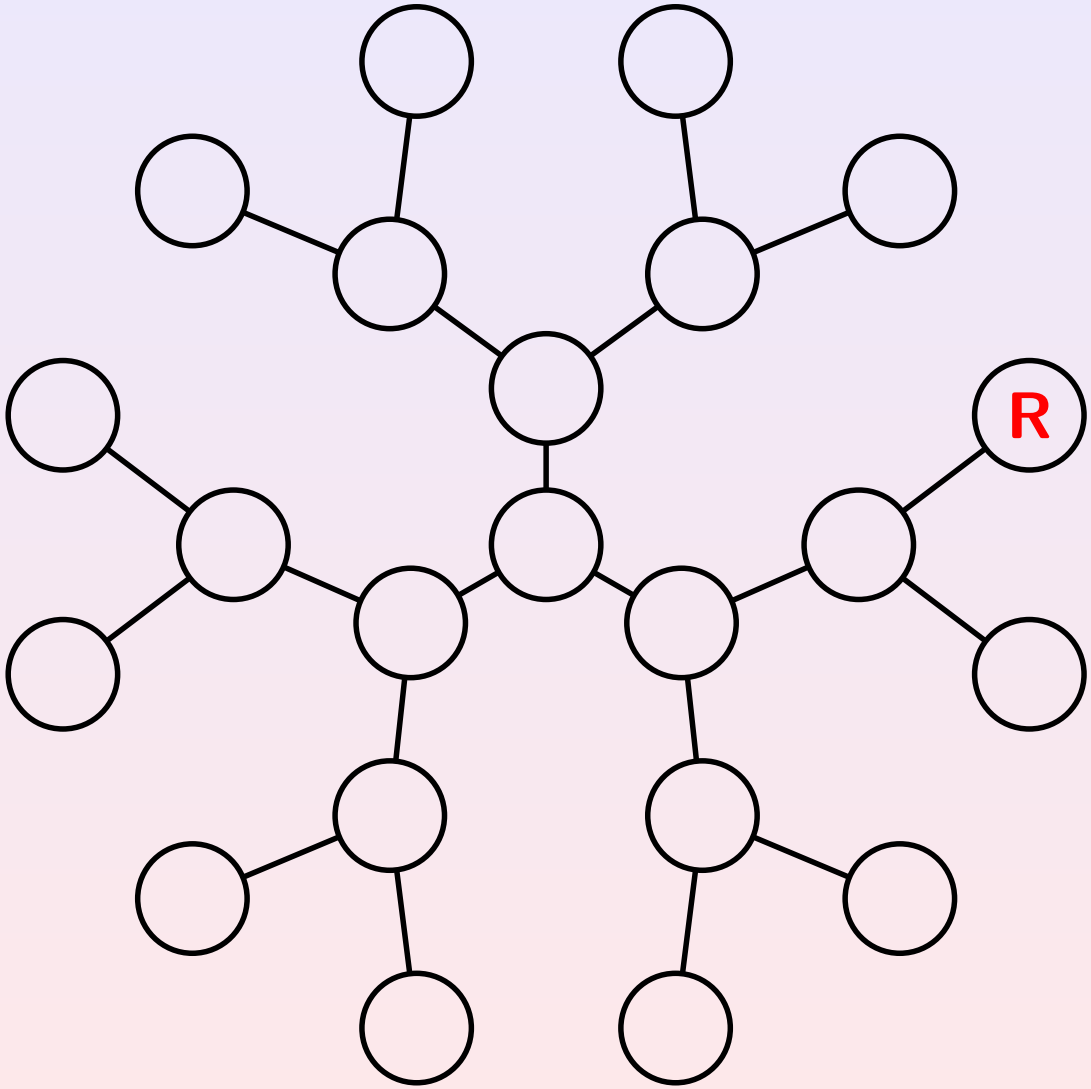
Applications: graph exploration (1)



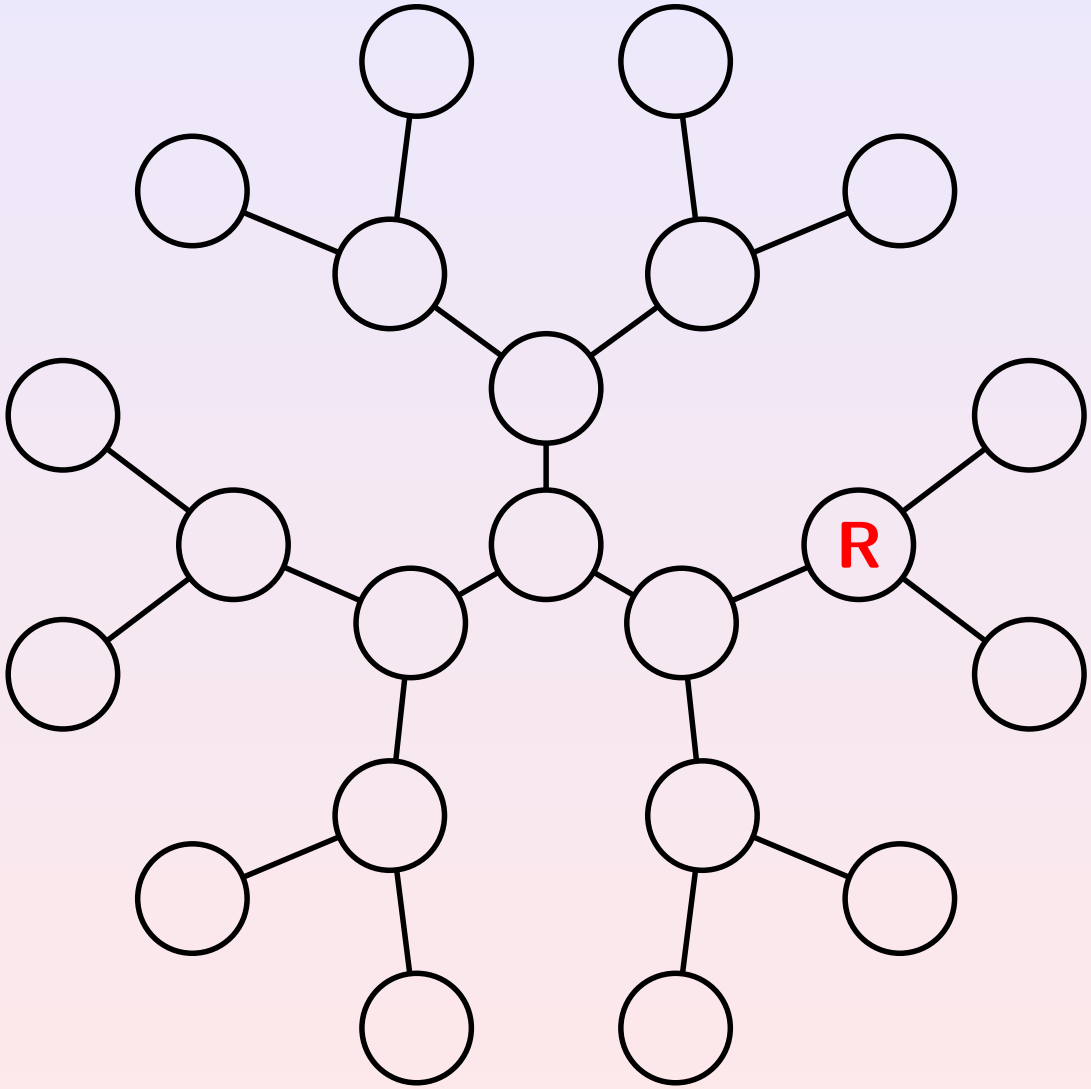
Applications: graph exploration (1)



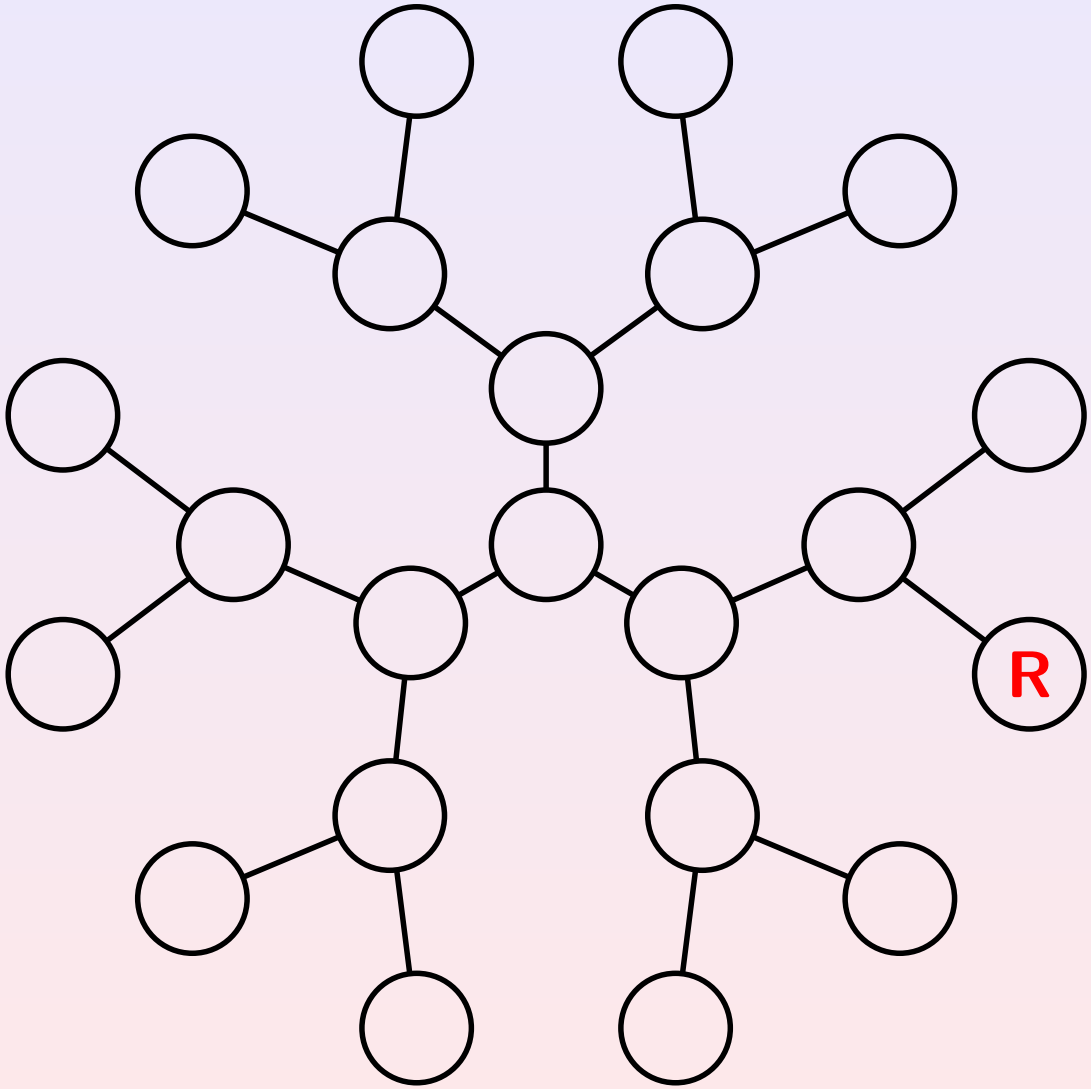
Applications: graph exploration (1)



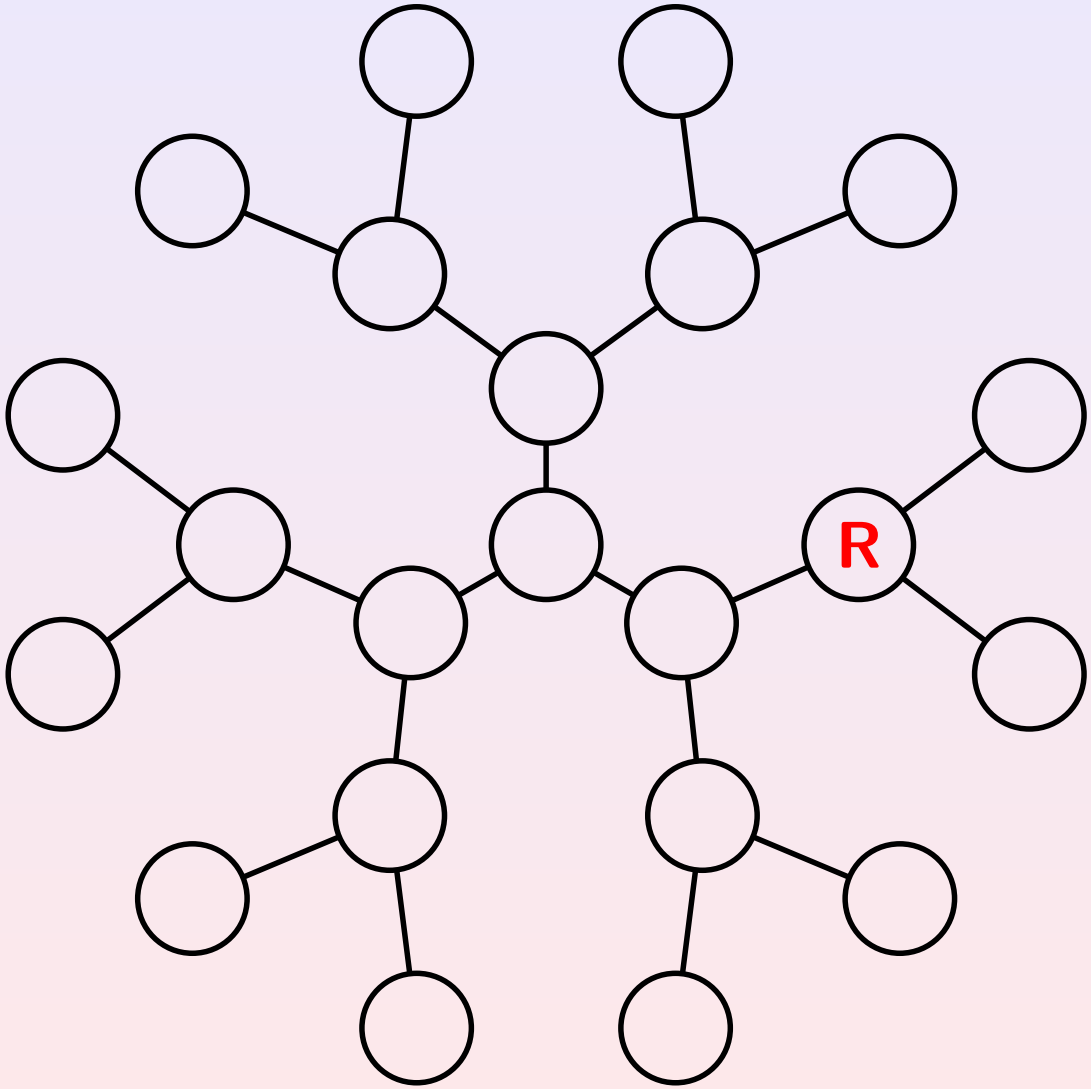
Applications: graph exploration (1)



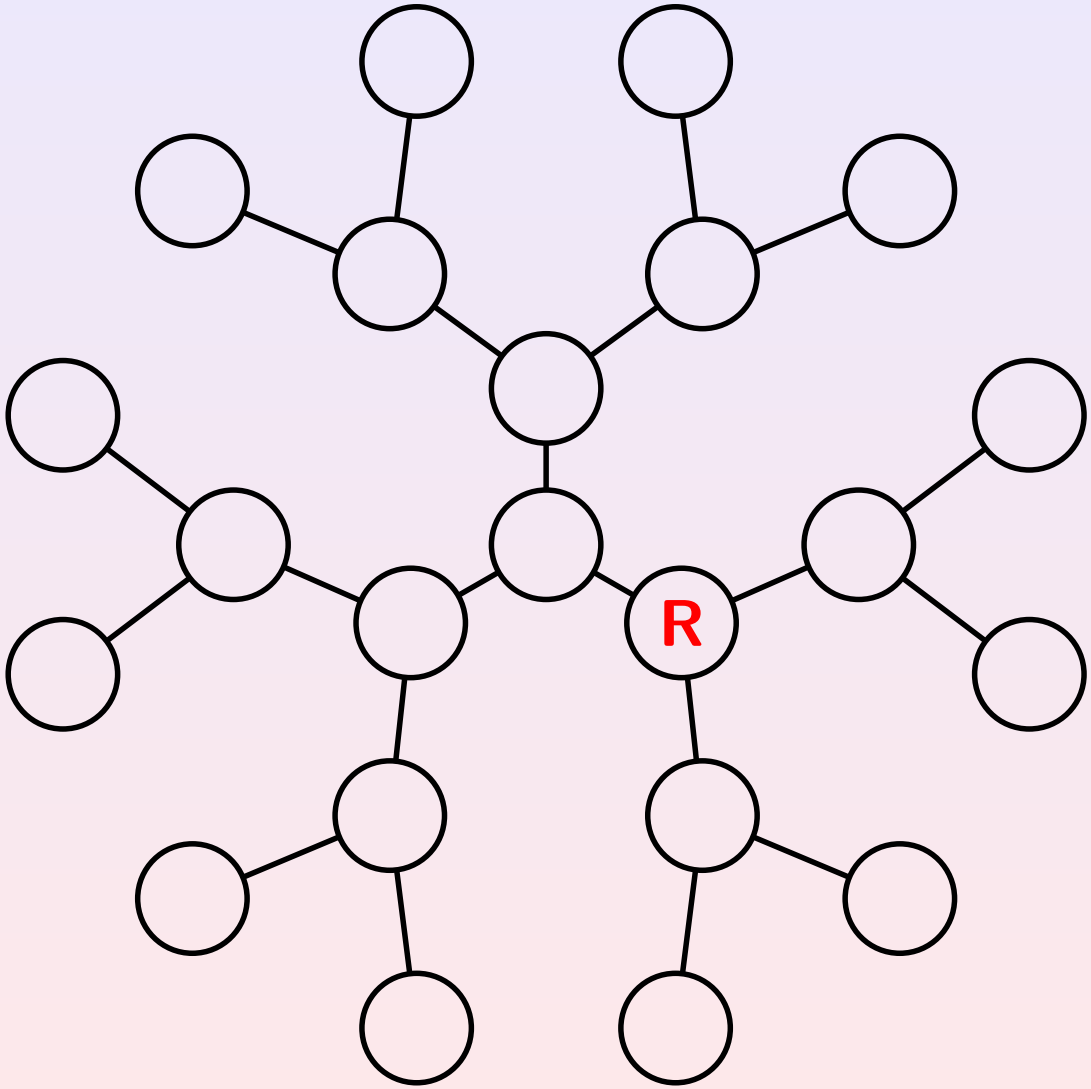
Applications: graph exploration (1)



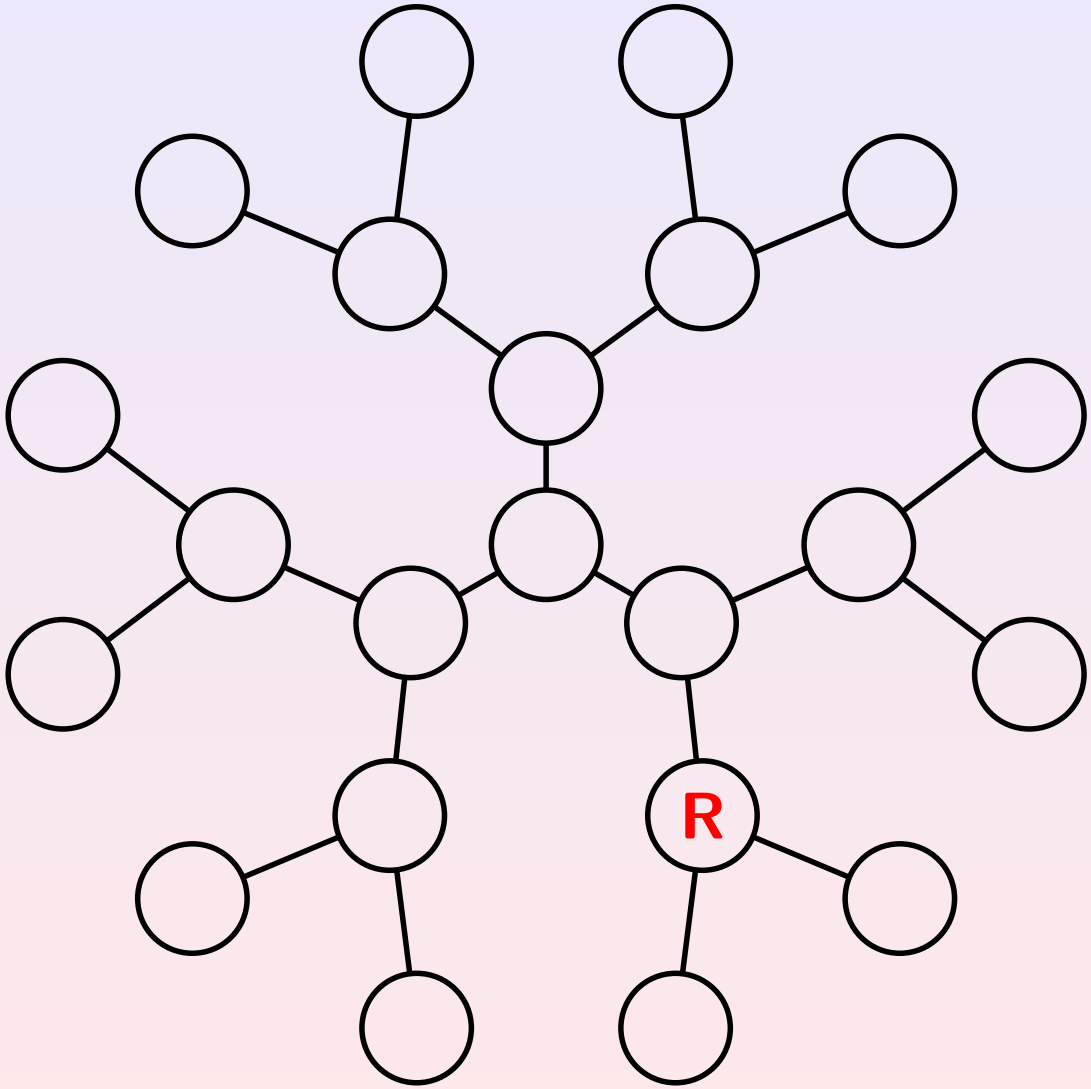
Applications: graph exploration (1)



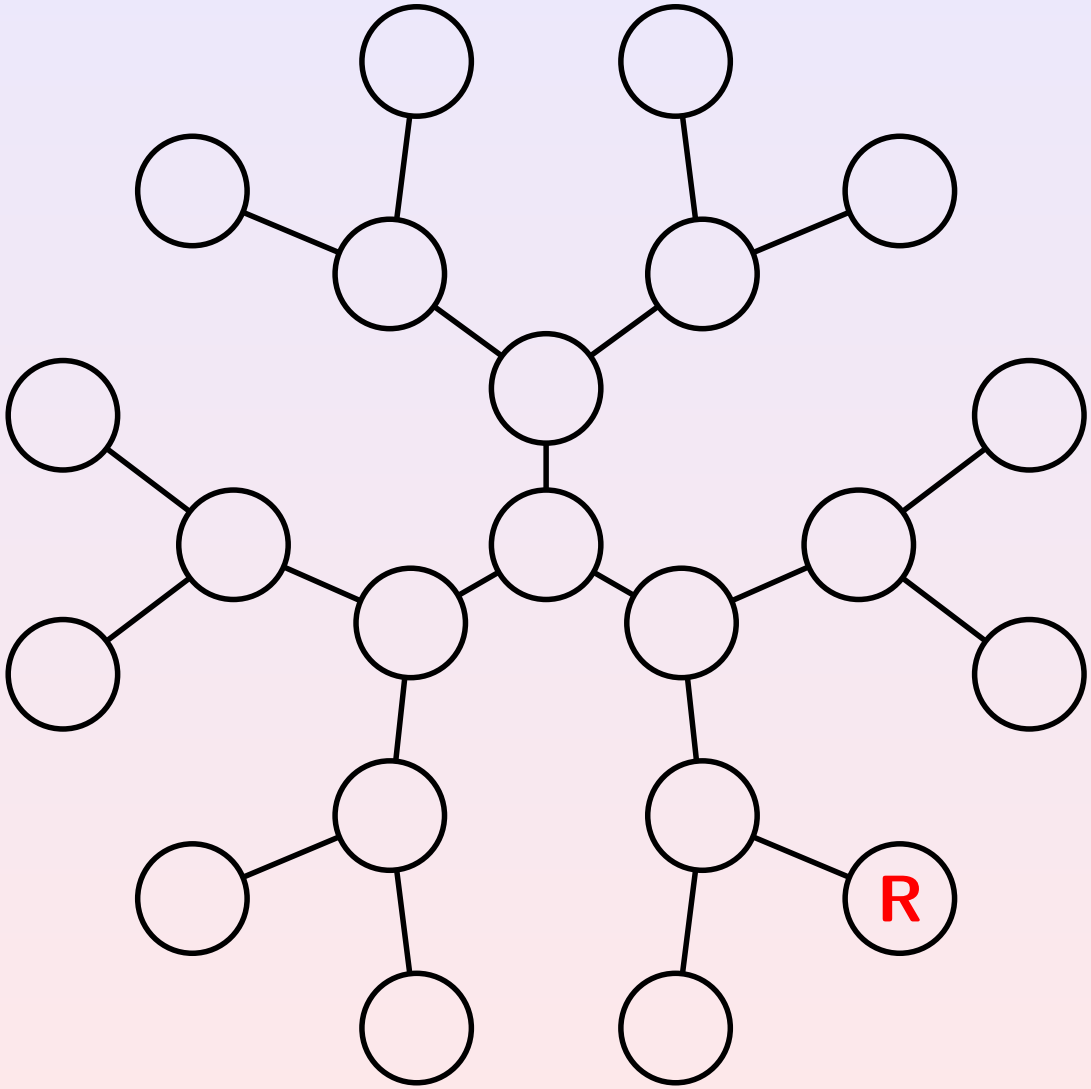
Applications: graph exploration (1)



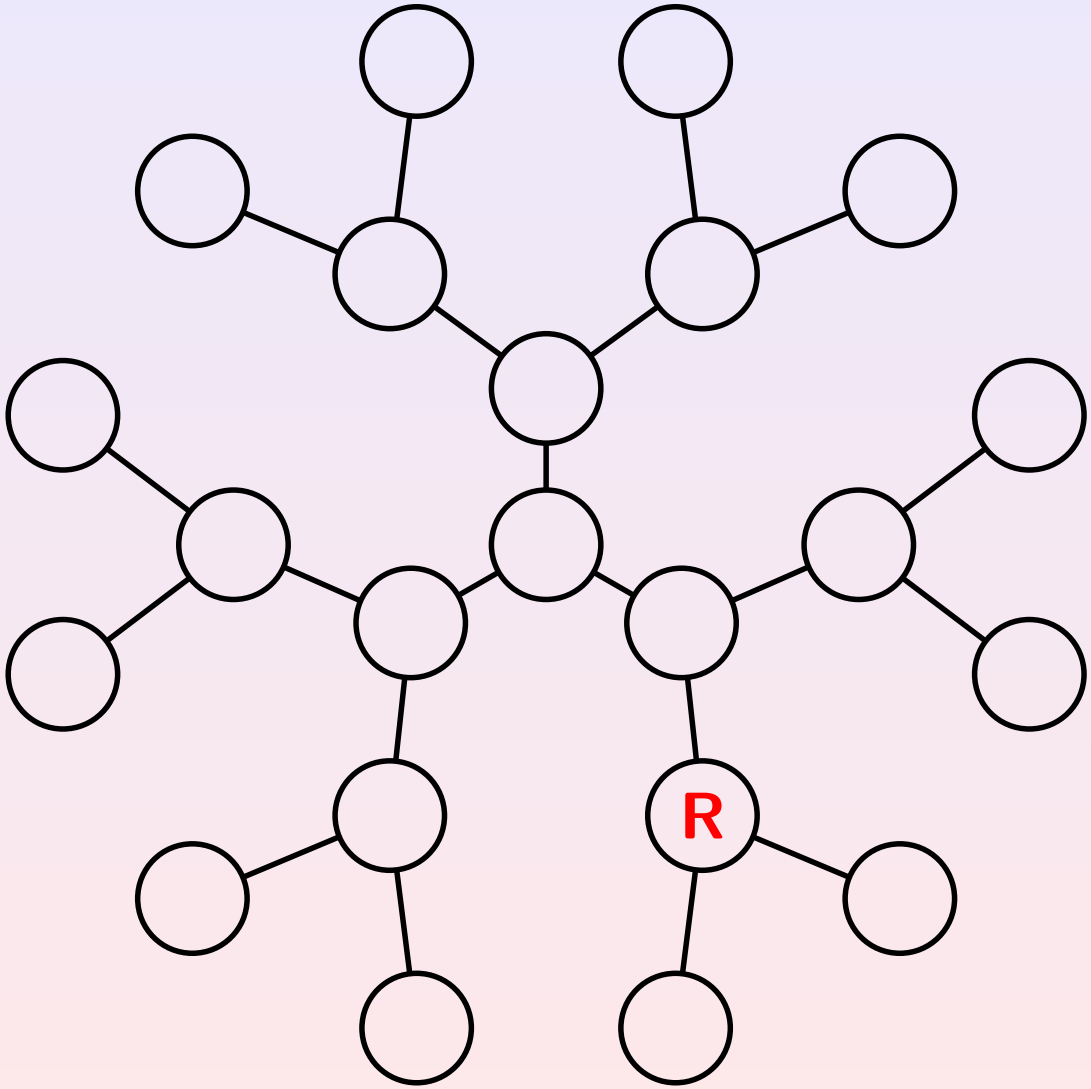
Applications: graph exploration (1)



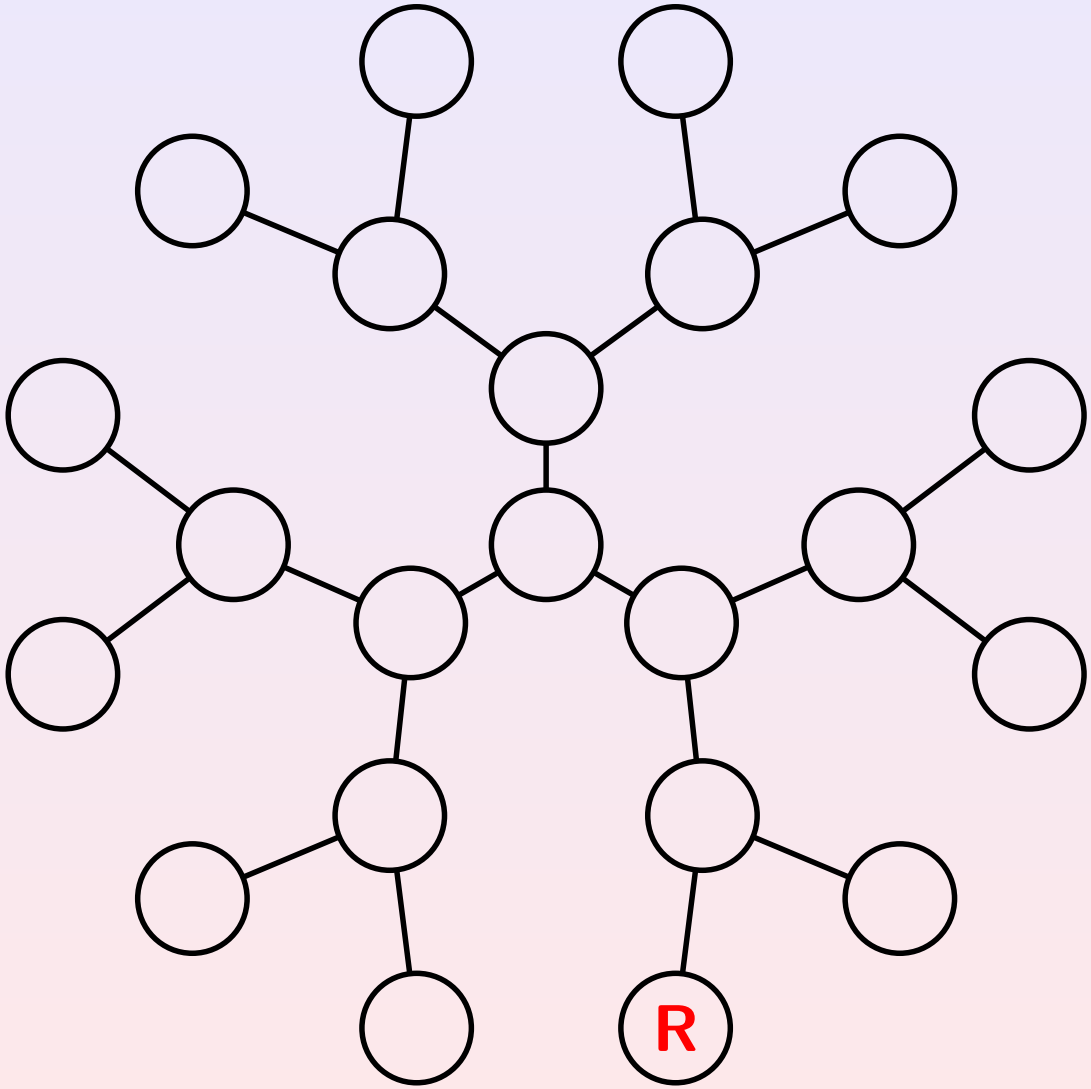
Applications: graph exploration (1)



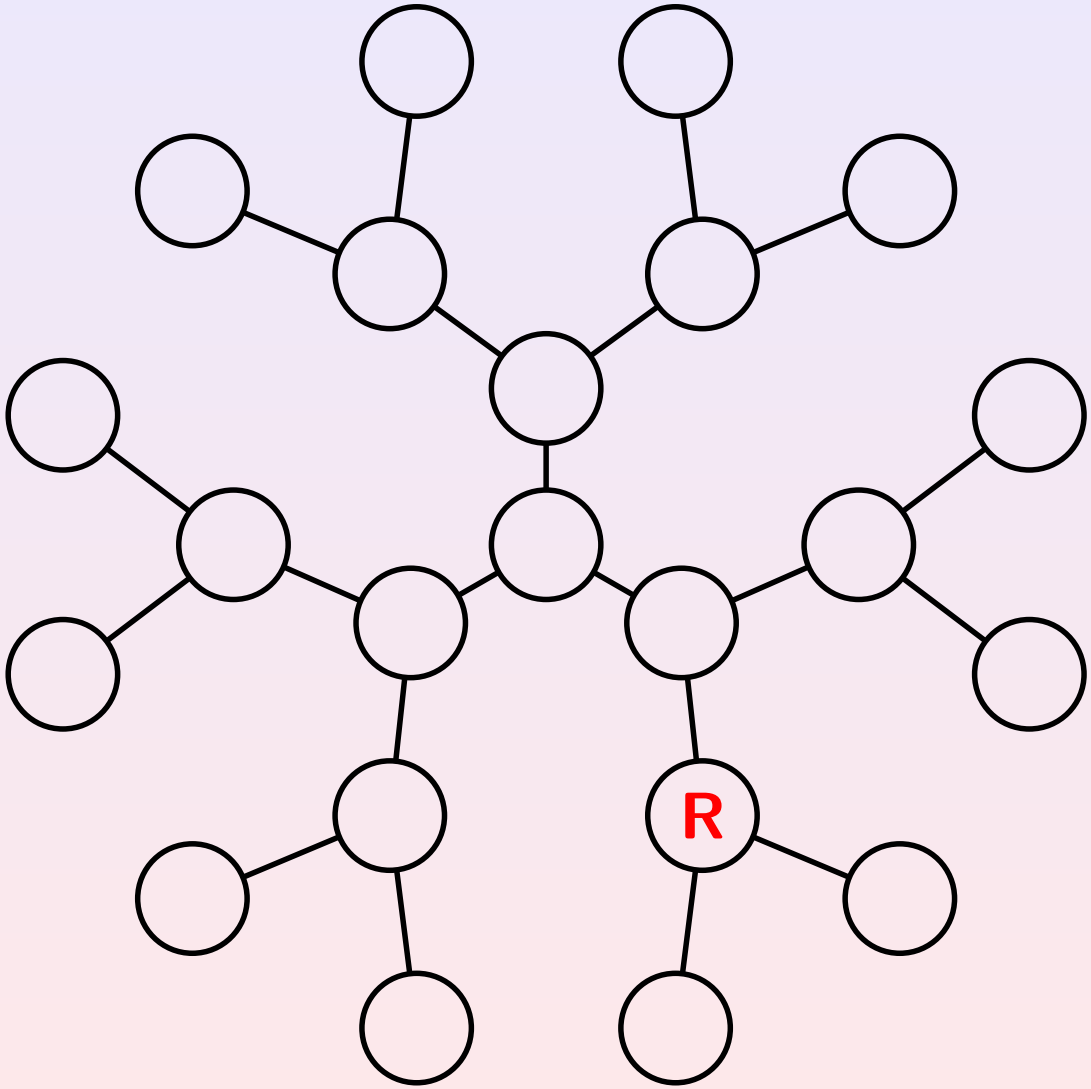
Applications: graph exploration (1)



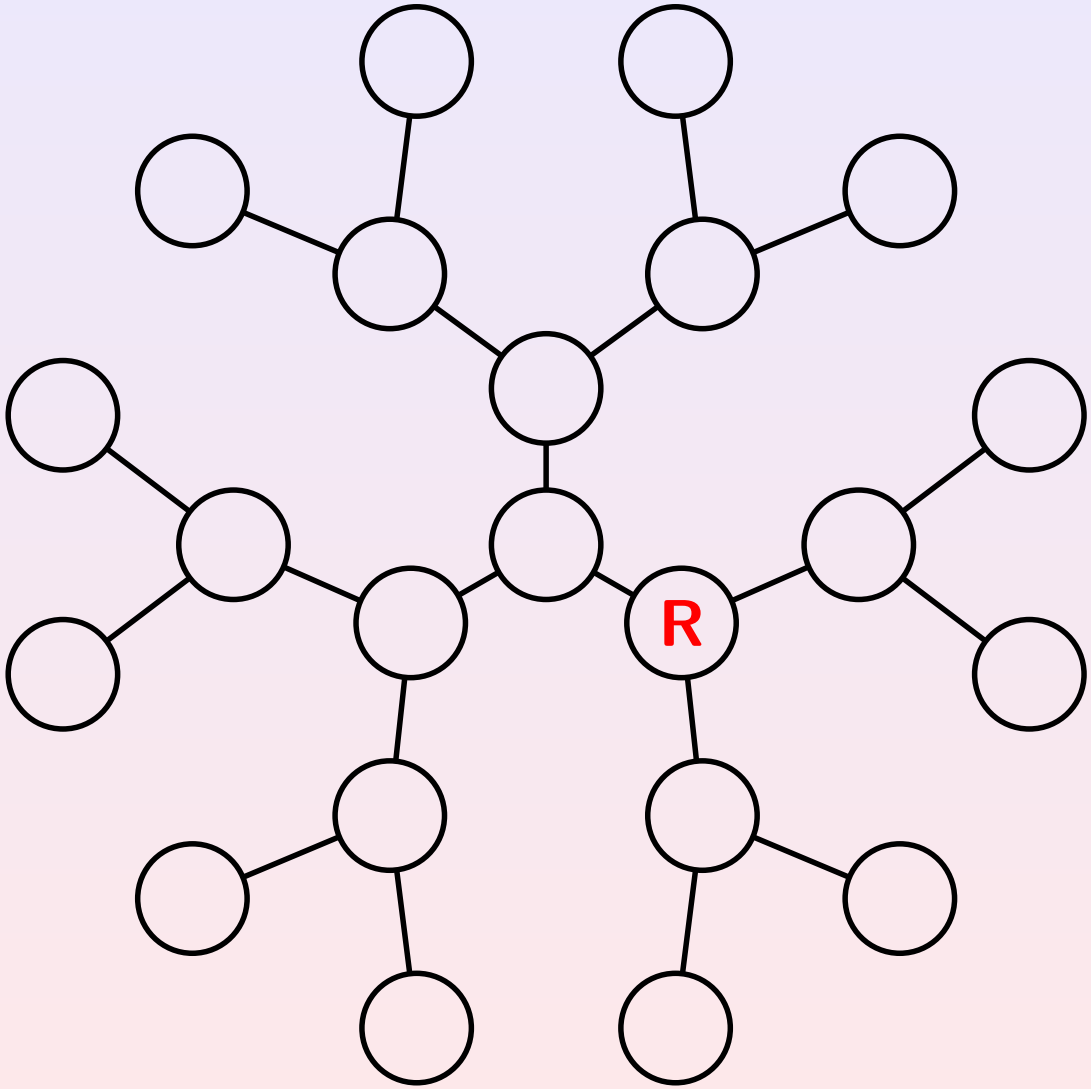
Applications: graph exploration (1)



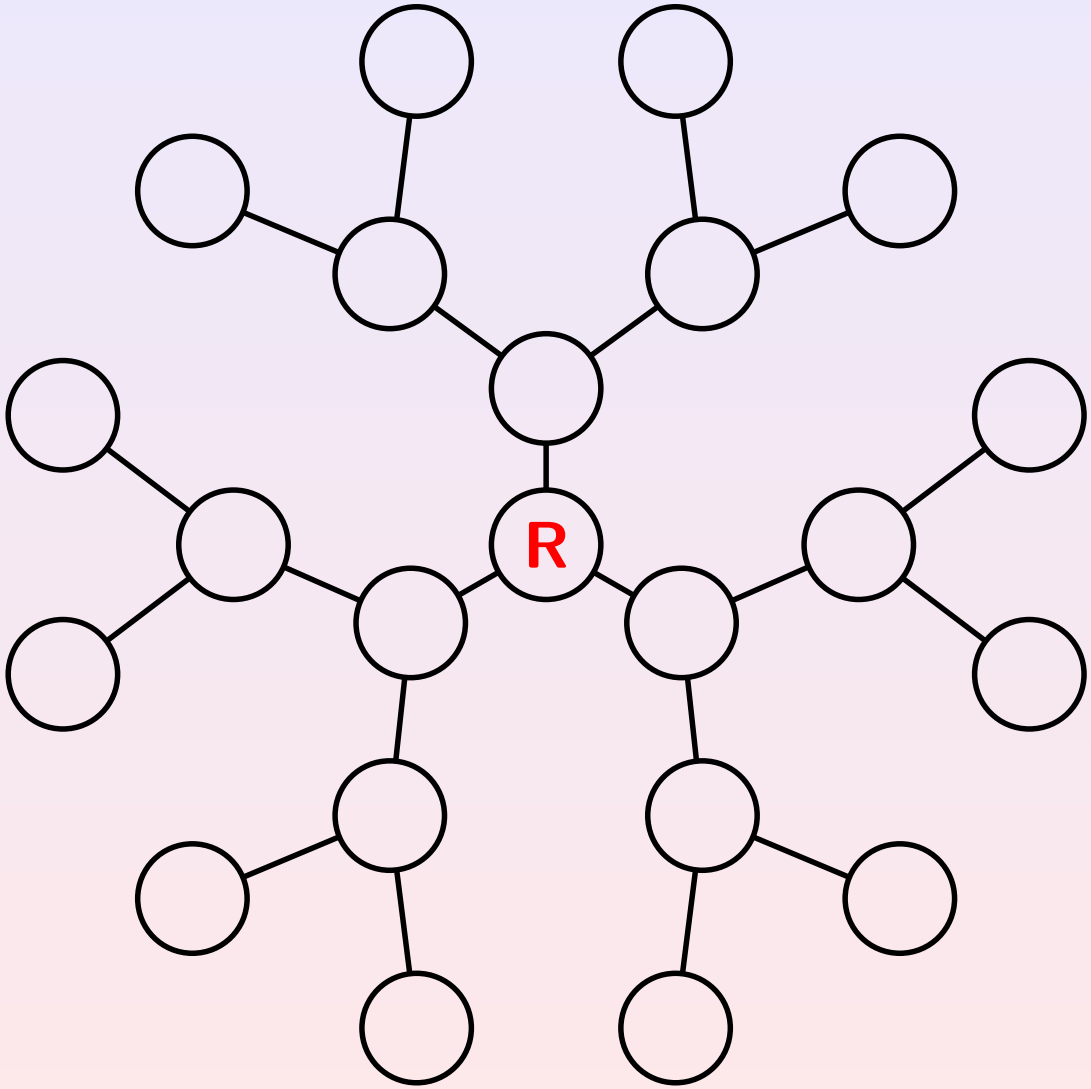
Applications: graph exploration (1)



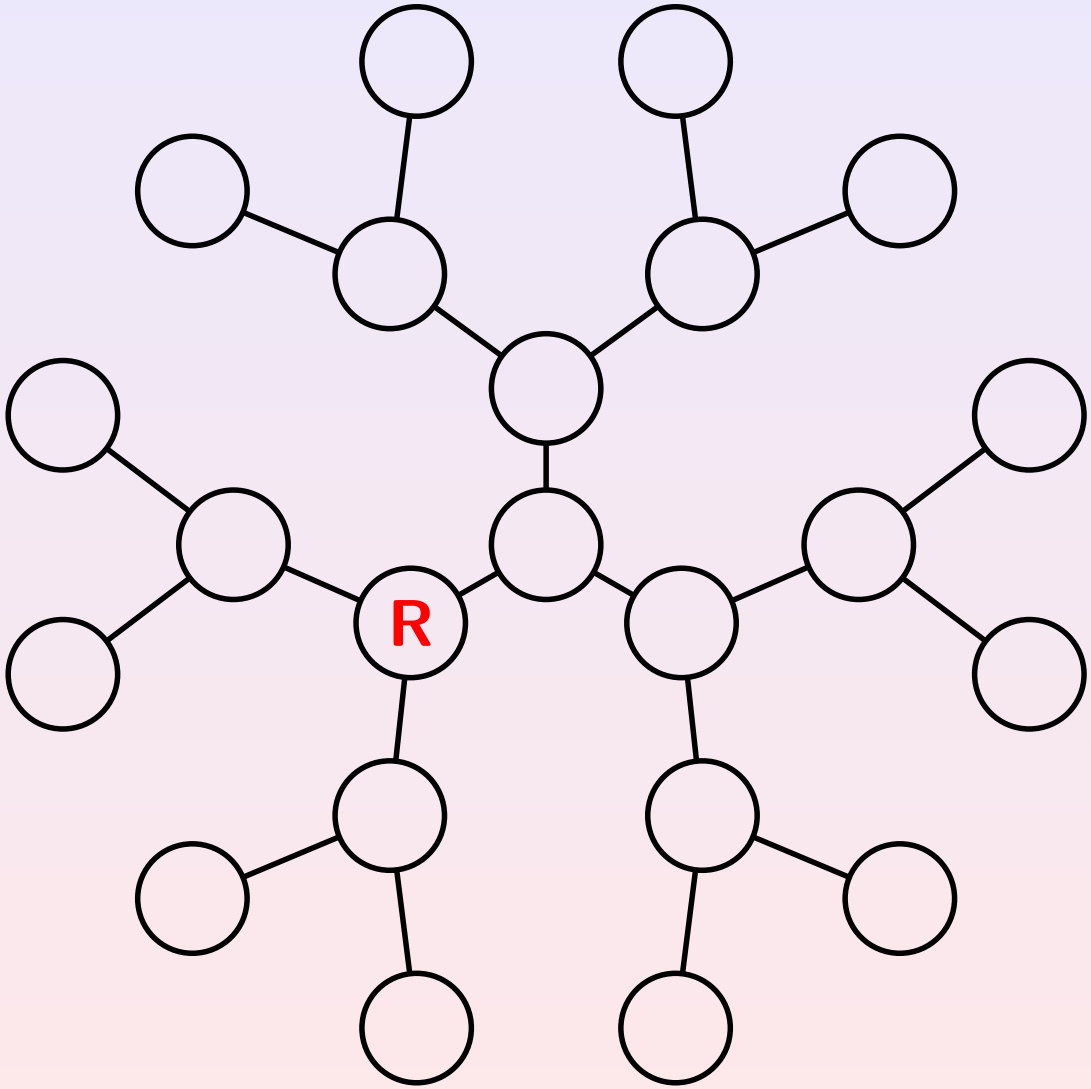
Applications: graph exploration (1)



Applications: graph exploration (1)



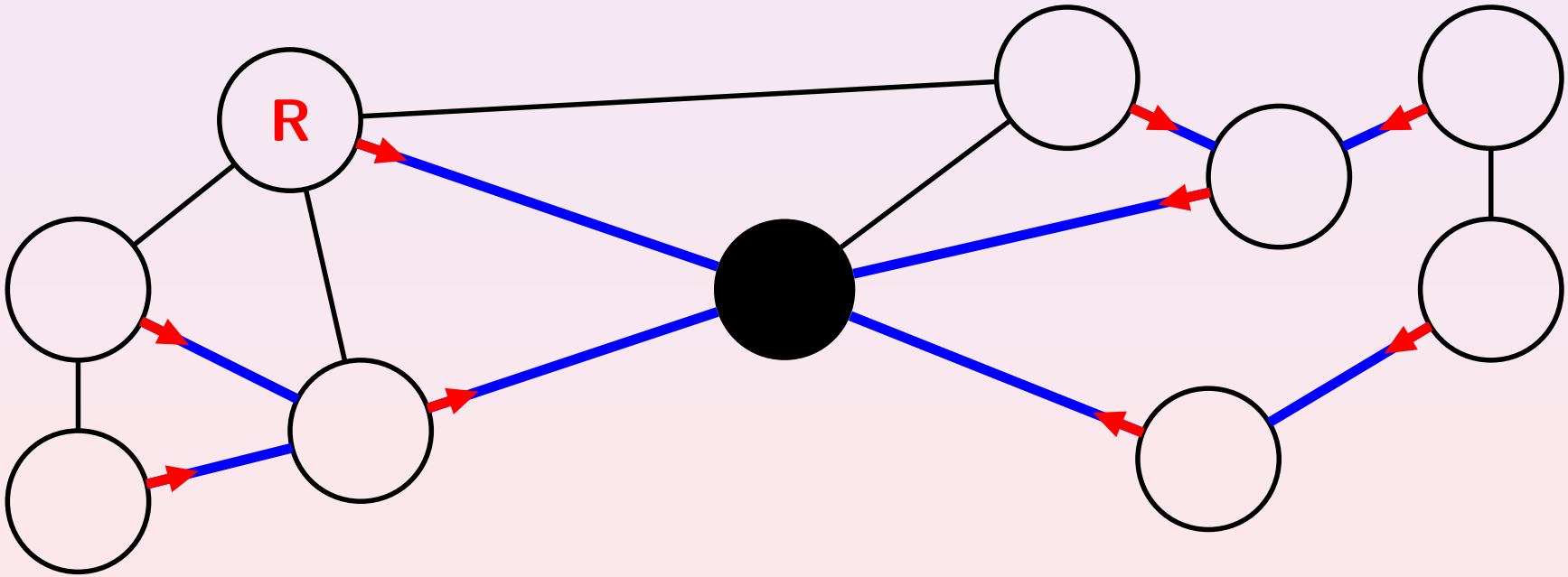
Applications: graph exploration (1)



Applications: graph exploration (2)

Goal

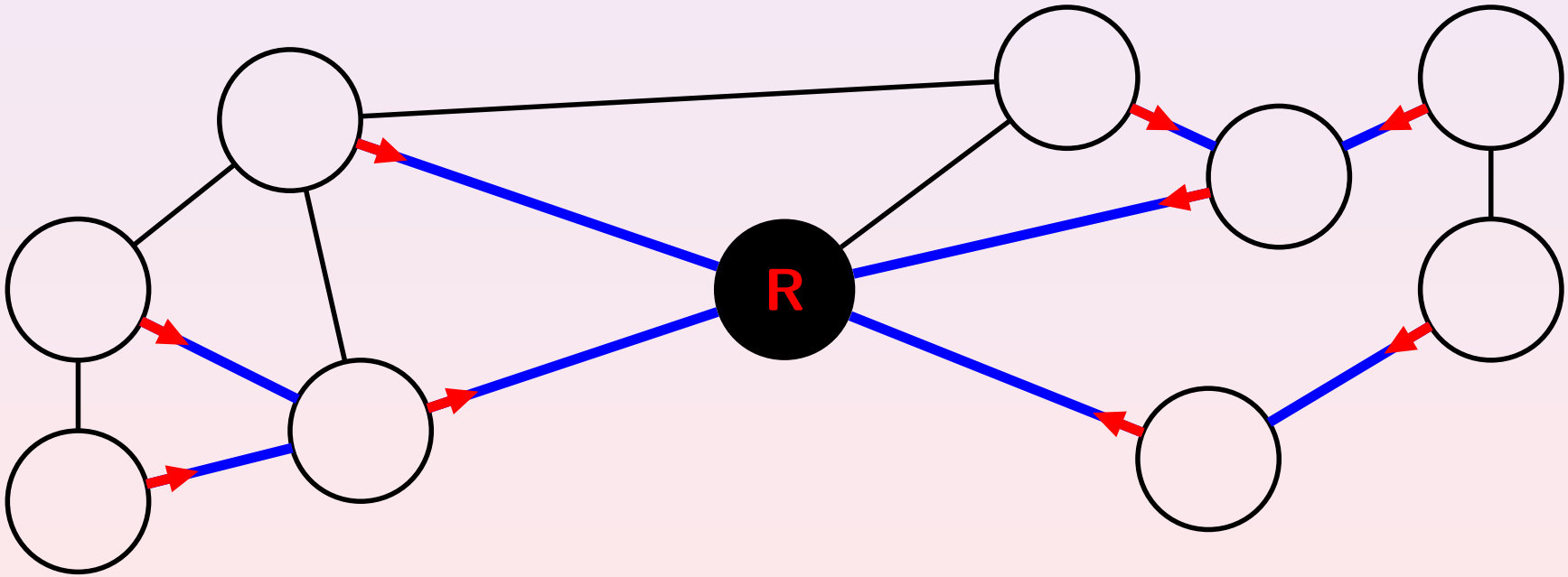
A mobile entity has to traverse every edge of an unknown anonymous graph.



Applications: graph exploration (2)

Goal

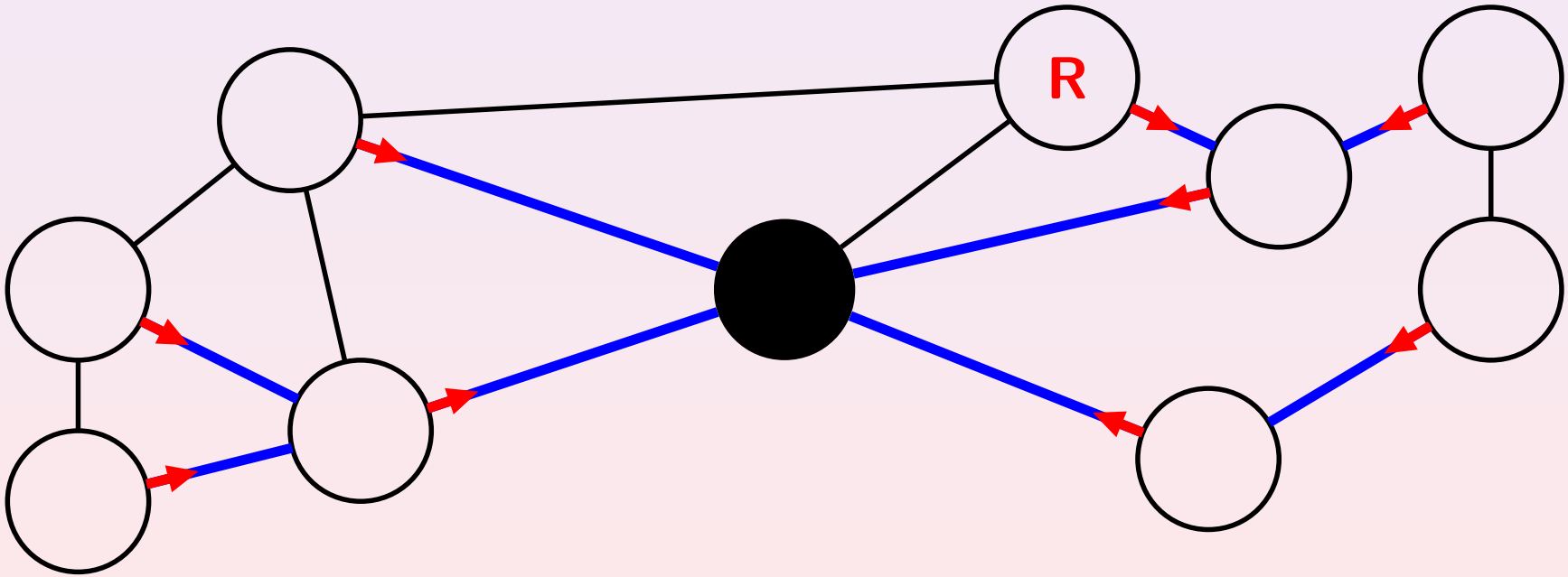
A mobile entity has to traverse every edge of an unknown anonymous graph.



Applications: graph exploration (2)

Goal

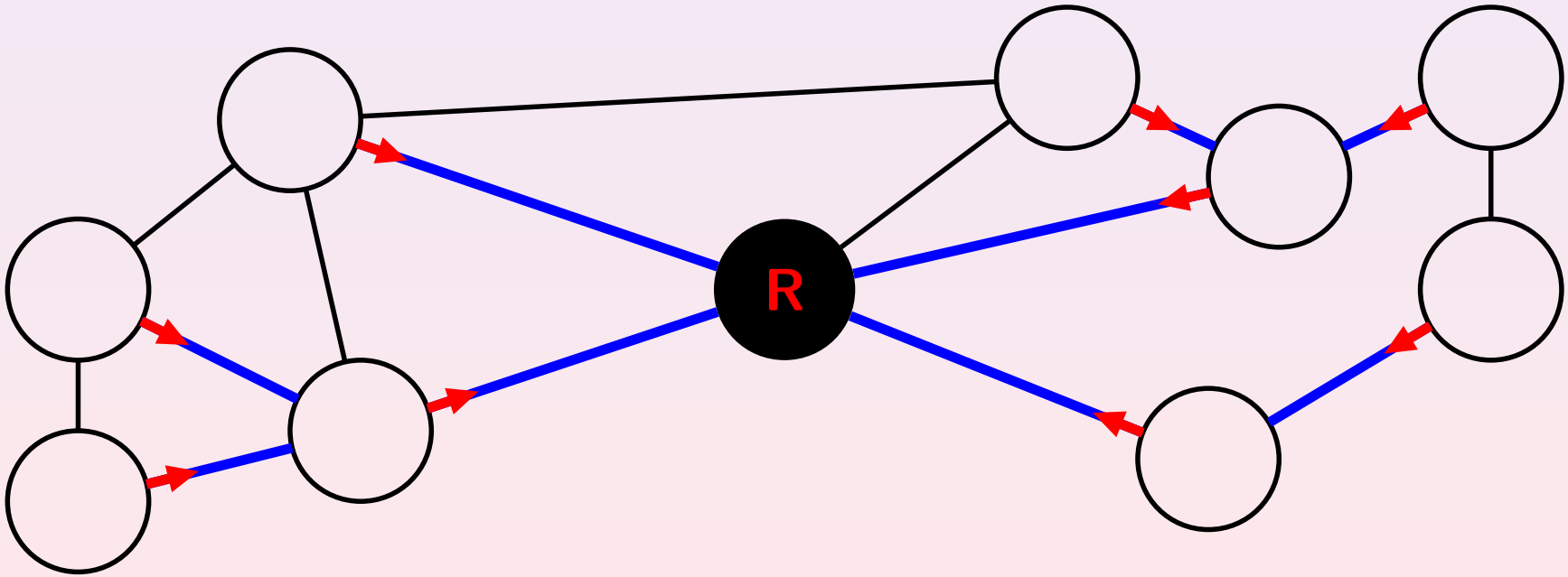
A mobile entity has to traverse every edge of an unknown anonymous graph.



Applications: graph exploration (2)

Goal

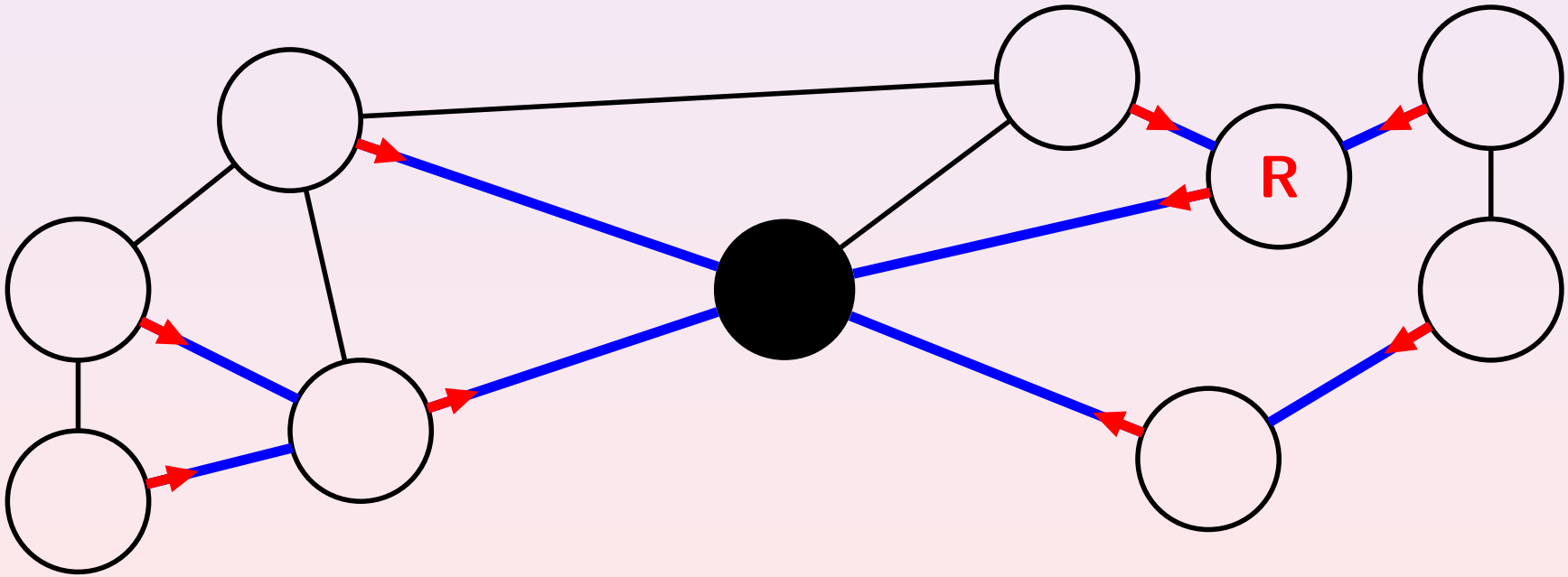
A mobile entity has to traverse every edge of an unknown anonymous graph.



Applications: graph exploration (2)

Goal

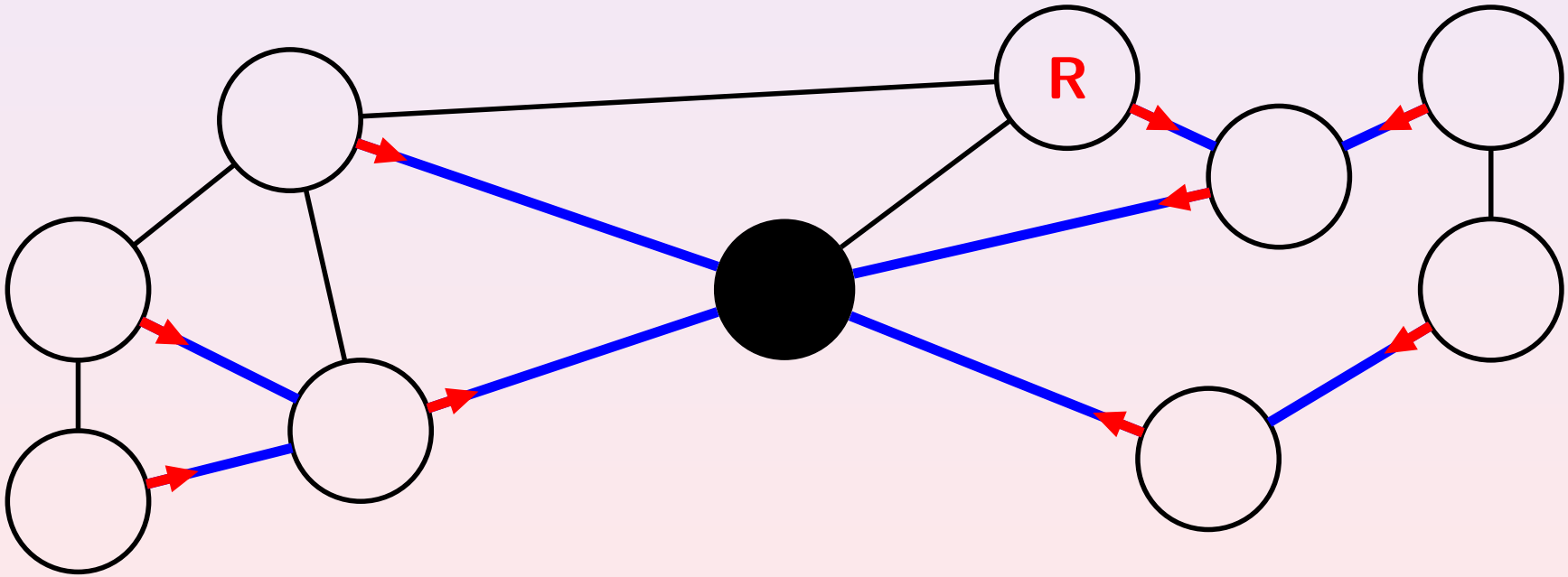
A mobile entity has to traverse every edge of an unknown anonymous graph.



Applications: graph exploration (2)

Goal

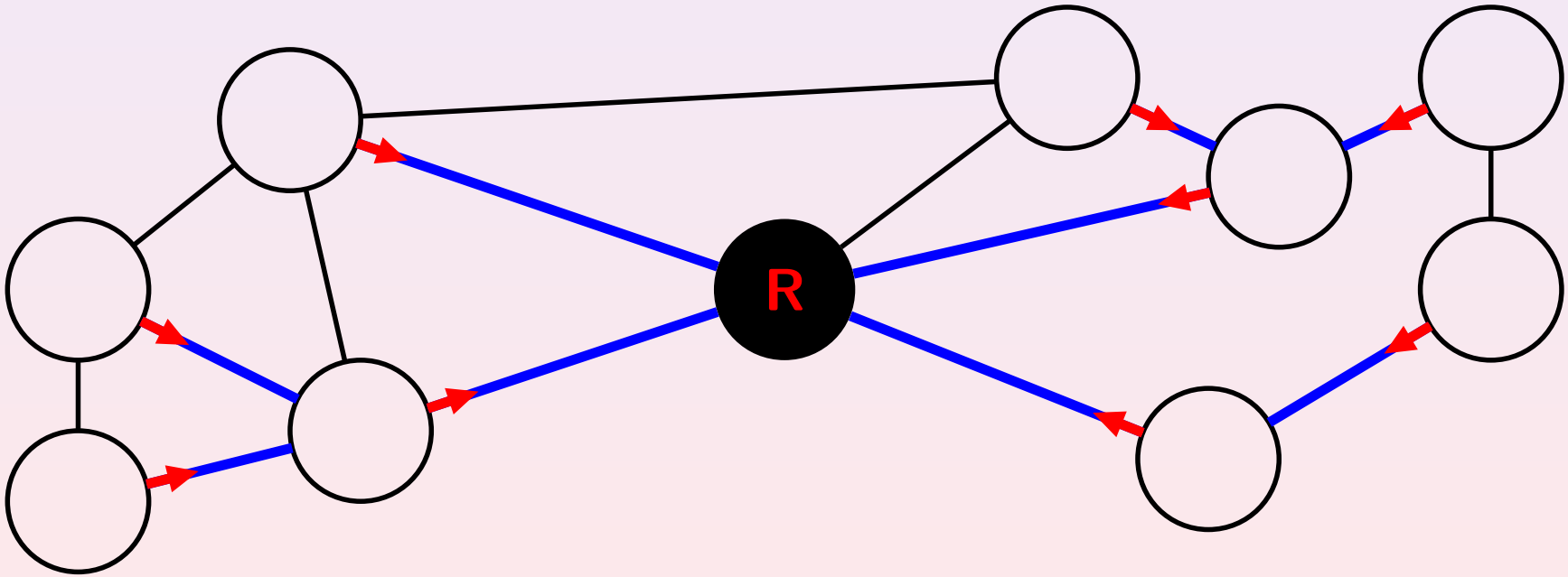
A mobile entity has to traverse every edge of an unknown anonymous graph.



Applications: graph exploration (2)

Goal

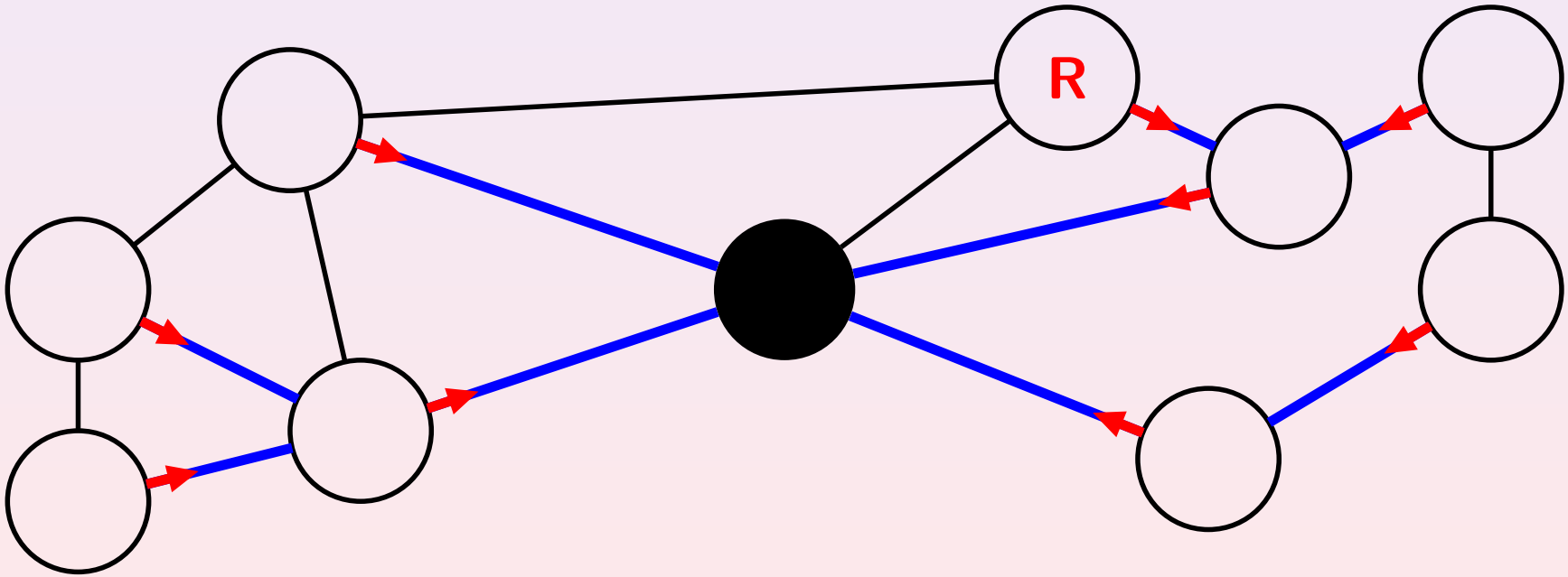
A mobile entity has to traverse every edge of an unknown anonymous graph.



Applications: graph exploration (2)

Goal

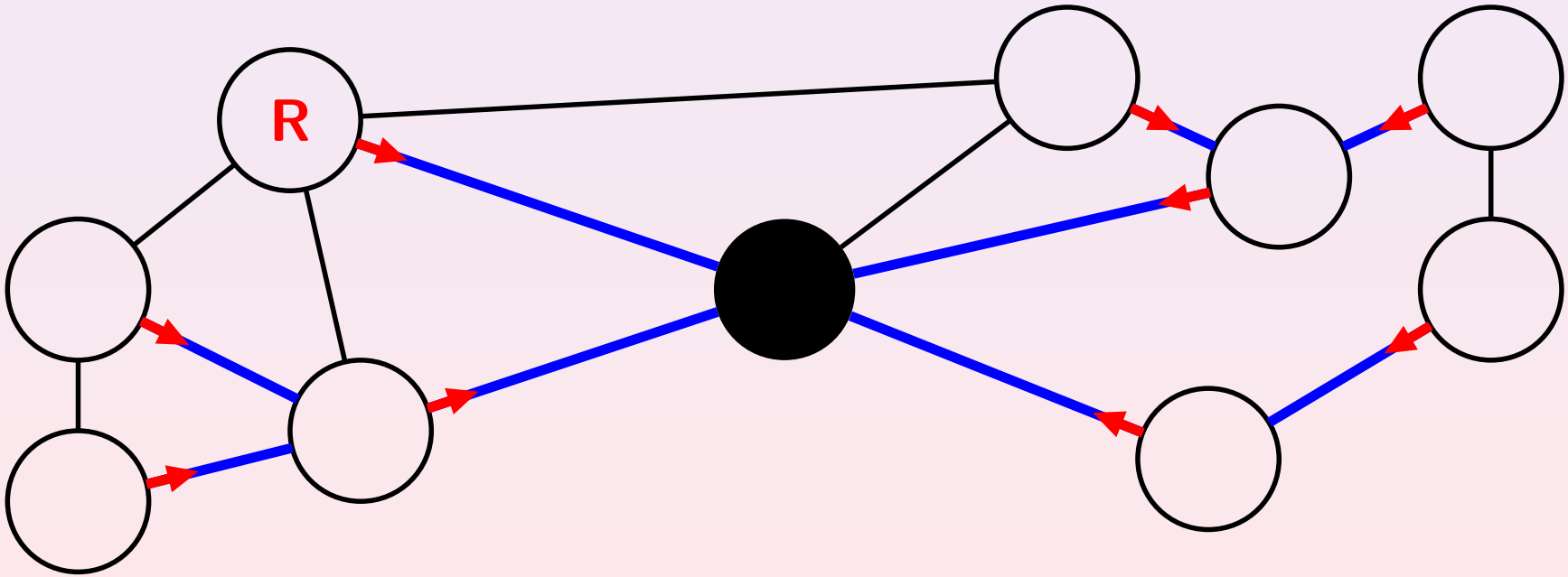
A mobile entity has to traverse every edge of an unknown anonymous graph.



Applications: graph exploration (2)

Goal

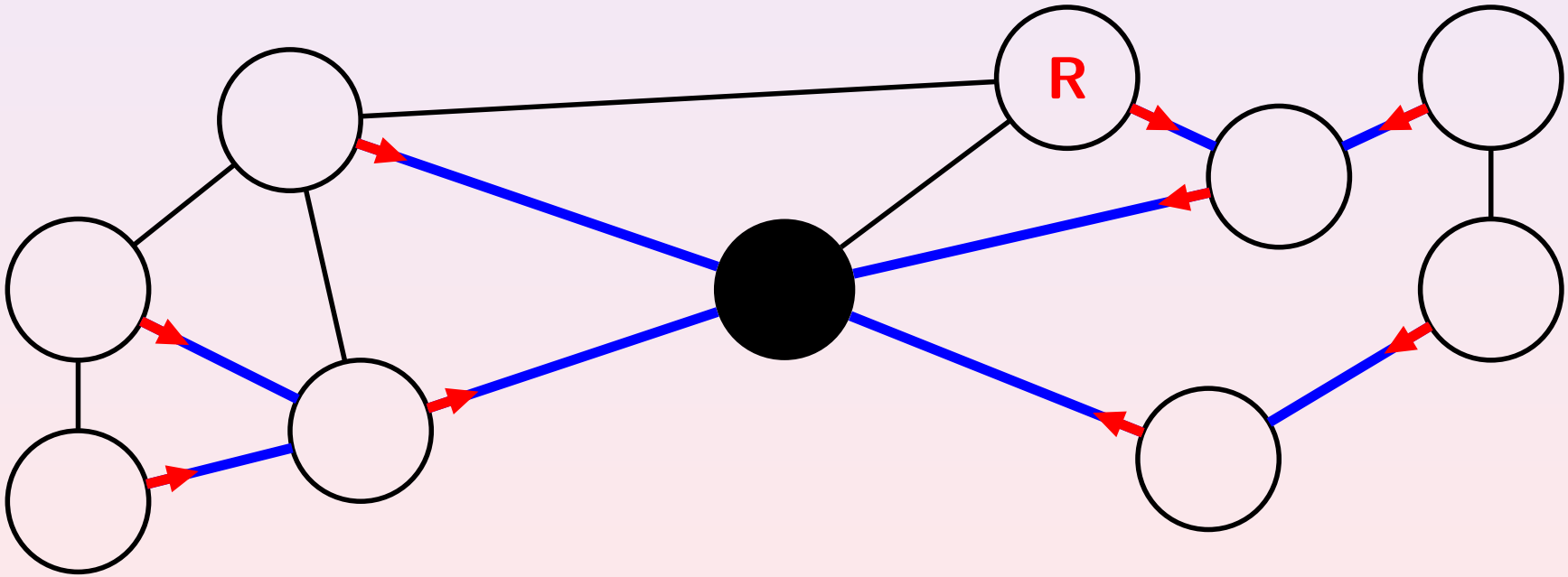
A mobile entity has to traverse every edge of an unknown anonymous graph.



Applications: graph exploration (2)

Goal

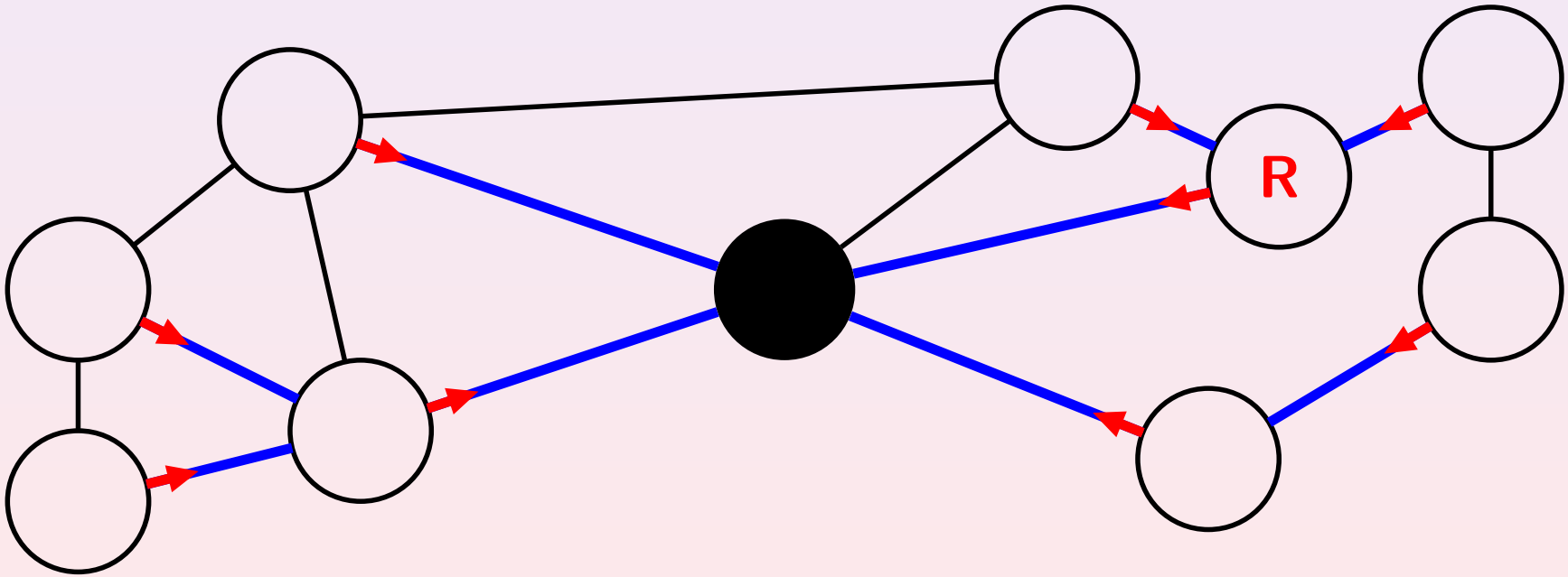
A mobile entity has to traverse every edge of an unknown anonymous graph.



Applications: graph exploration (2)

Goal

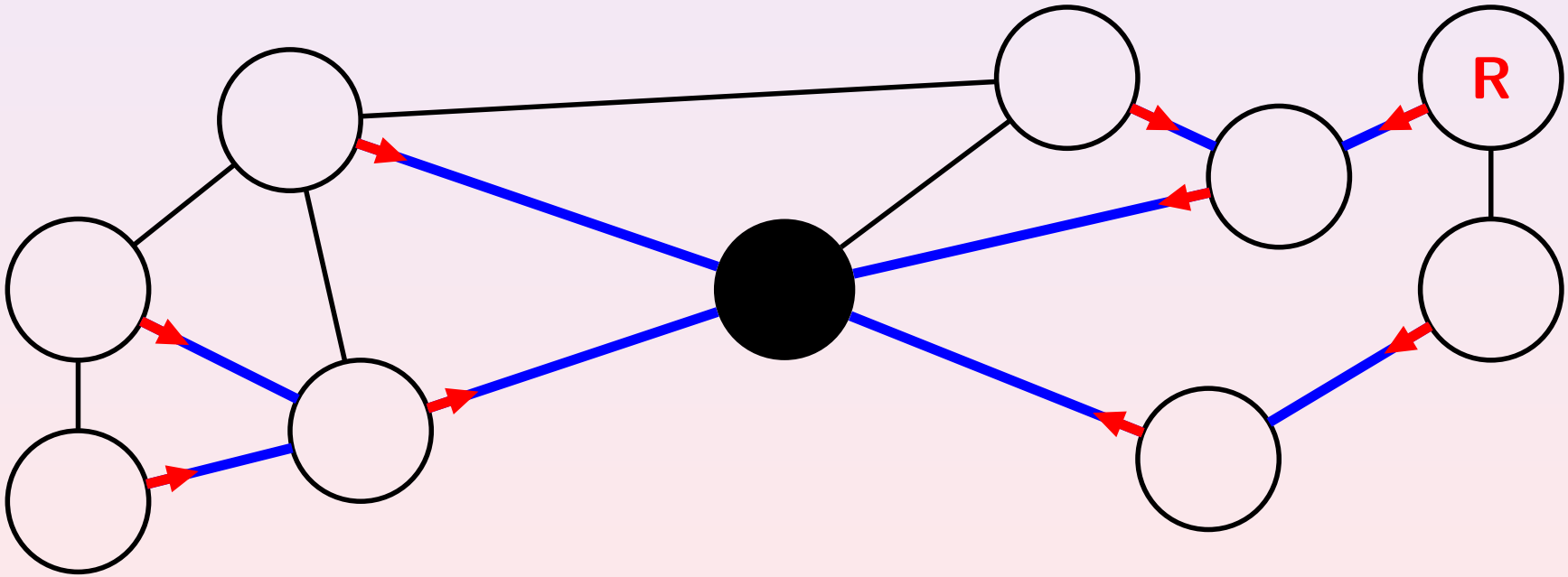
A mobile entity has to traverse every edge of an unknown anonymous graph.



Applications: graph exploration (2)

Goal

A mobile entity has to traverse every edge of an unknown anonymous graph.



Naive upper bounds

Trivial observations

- Maximum: $M_{up}(T) \leq \lceil \log \Delta \rceil$, $\Delta =$ maximum degree
- Sum: $S_{up}(T) \leq \sum_v \lceil \log \deg(v) \rceil \longrightarrow O(n \log \Delta)$ bits

The bound on $M_{up}(T)$ is tight.

Naive upper bounds

Trivial observations

- Maximum: $M_{up}(T) \leq \lceil \log \Delta \rceil$, $\Delta =$ maximum degree
- Sum: $S_{up}(T) \leq \sum_v \lceil \log \deg(v) \rceil \longrightarrow O(n \log \Delta)$ bits

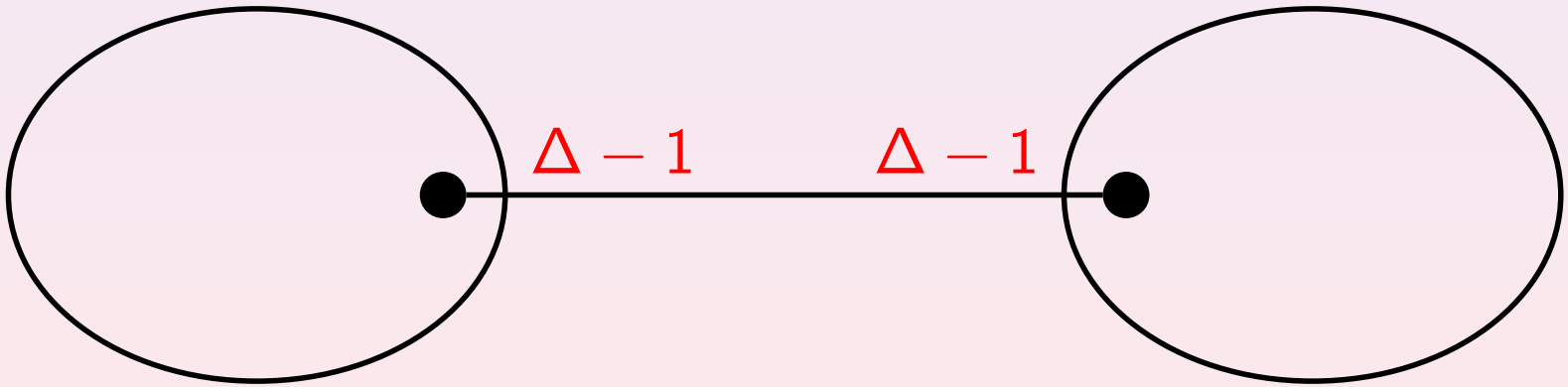
The bound on $M_{up}(T)$ is tight.

Naive upper bounds

Trivial observations

- Maximum: $M_{up}(T) \leq \lceil \log \Delta \rceil$, $\Delta =$ maximum degree
- Sum: $S_{up}(T) \leq \sum_v \lceil \log \deg(v) \rceil \longrightarrow O(n \log \Delta)$ bits

The bound on $M_{up}(T)$ is tight.



Optimal algorithm

Remark

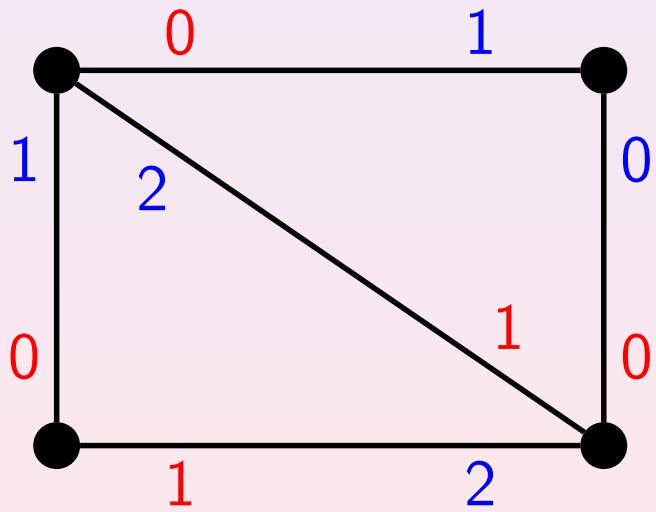
There is a polynomial time algorithm that given a graph G and a port numbering, construct a spanning tree T for G minimizing $S_{up}(T)$.

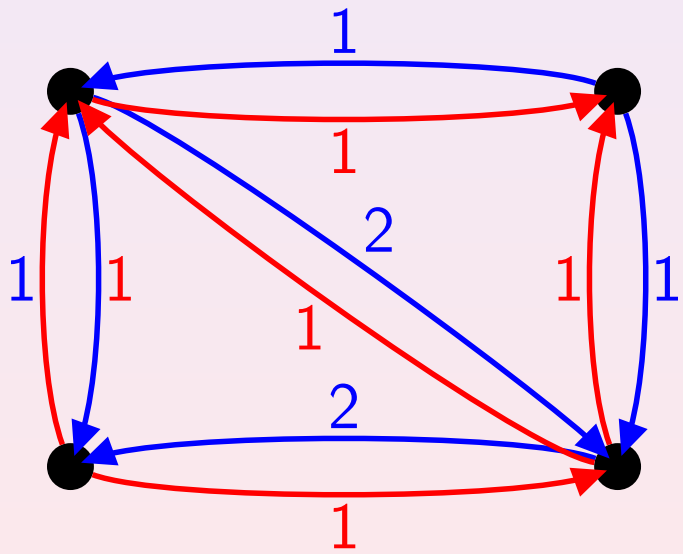
Use a Directed Minimum Spanning Tree algorithm.

Optimal algorithm

Remark

There is a polynomial time algorithm that given a graph G and a port numbering, construct a spanning tree T for G minimizing $S_{up}(T)$.



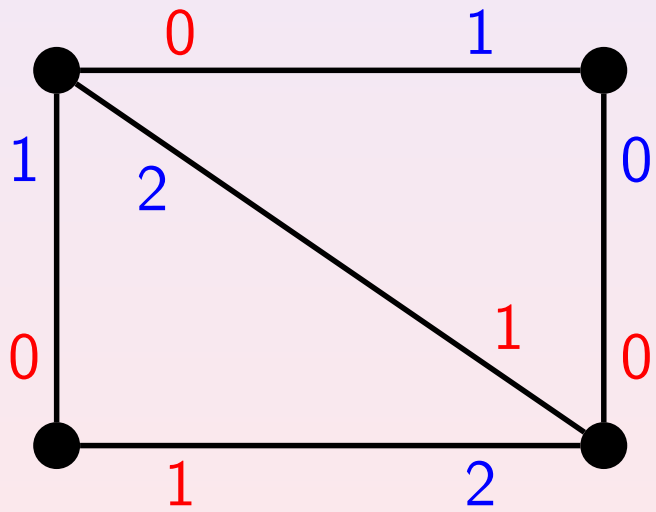


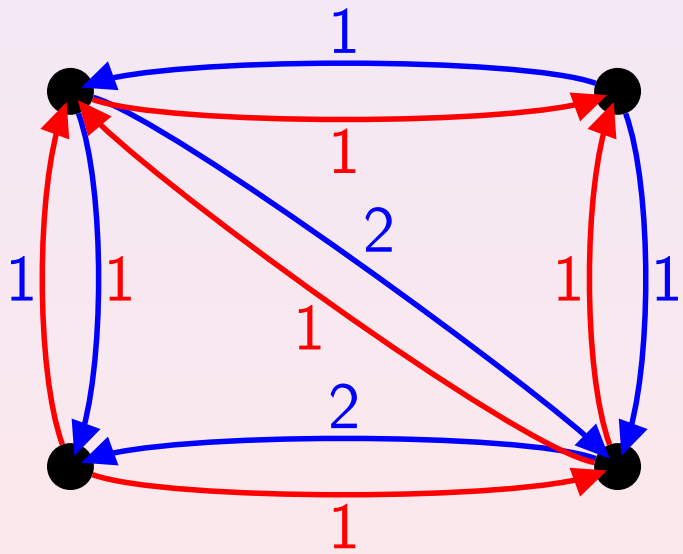
Use a Directed Minimum Spanning Tree algorithm.

Optimal algorithm

Remark

There is a polynomial time algorithm that given a graph G and a port numbering, construct a spanning tree T for G minimizing $S_{up}(T)$.





Use a Directed Minimum Spanning Tree algorithm.

Our results (1)

Theorem 1

In complete graphs with arbitrary labeling:

- There exists T such that $S_{up}(T) = O(n)$

Theorem 2

In arbitrary graphs with symmetric port assignments:

- There exists T such that $S_{up}(T) = O(n)$

Our results (1)

Theorem 1

In complete graphs with arbitrary labeling:

- There exists T such that $S_{up}(T) = O(n)$

Theorem 2

In arbitrary graphs with symmetric port assignments:

- There exists T such that $S_{up}(T) = O(n)$

Our results (2)

Conjecture

In arbitrary graphs with arbitrary labeling:

- There exists T such that $S_{up}(T) = O(n)$

Theorem 3

In arbitrary graphs with arbitrary labeling:

- There exists T such that $S_{up}(T) = O(n \log \log n)$

Our results (2)

Conjecture

In arbitrary graphs with arbitrary labeling:

- There exists T such that $S_{up}(T) = O(n)$

Theorem 3

In arbitrary graphs with arbitrary labeling:

- There exists T such that $S_{up}(T) = O(n \log \log n)$

Outline

Algorithm for complete graphs

Main idea

- Start with isolated vertices, the initial trees.
- Merge trees to finally obtain a spanning tree.

Description of phase k

Algorithm for complete graphs

Main idea

- Start with isolated vertices, the initial trees.
- Merge trees to finally obtain a spanning tree.

Description of phase k

1. Identify the collection of small trees for the phase: $\mathcal{T}_{\text{small}}(k) = \{T \mid \text{size}(T) < 2^k\}$.
2. For each tree $T \in \mathcal{T}_{\text{small}}(k)$, select the edge $e(T)$ of minimum weight going from the root outside T.
3. Merge the trees with the edges $e(T)$.
4. Erase one edge on each formed cycle to obtain trees.

Algorithm for complete graphs

Main idea

- Start with isolated vertices, the initial trees.
- Merge trees to finally obtain a spanning tree.

Description of phase k

1. Identify the collection of small trees for the phase: $\mathcal{T}_{\text{small}}(k) = \{T \mid \text{size}(T) < 2^k\}$.
2. For each tree $T \in \mathcal{T}_{\text{small}}(k)$, select the edge $e(T)$ of minimum weight going from the root outside T.
3. Merge the trees with the edges $e(T)$.
4. Erase one edge on each formed cycle to obtain trees.

Algorithm for complete graphs

Main idea

- Start with isolated vertices, the initial trees.
- Merge trees to finally obtain a spanning tree.

Description of phase k

1. Identify the collection of small trees for the phase: $\mathcal{T}_{\text{small}}(k) = \{T \mid \text{size}(T) < 2^k\}$.
2. For each tree $T \in \mathcal{T}_{\text{small}}(k)$, select the edge $e(T)$ of minimum weight going from the root outside T.
3. Merge the trees with the edges $e(T)$.
4. Erase one edge on each formed cycle to obtain trees.

Algorithm for complete graphs

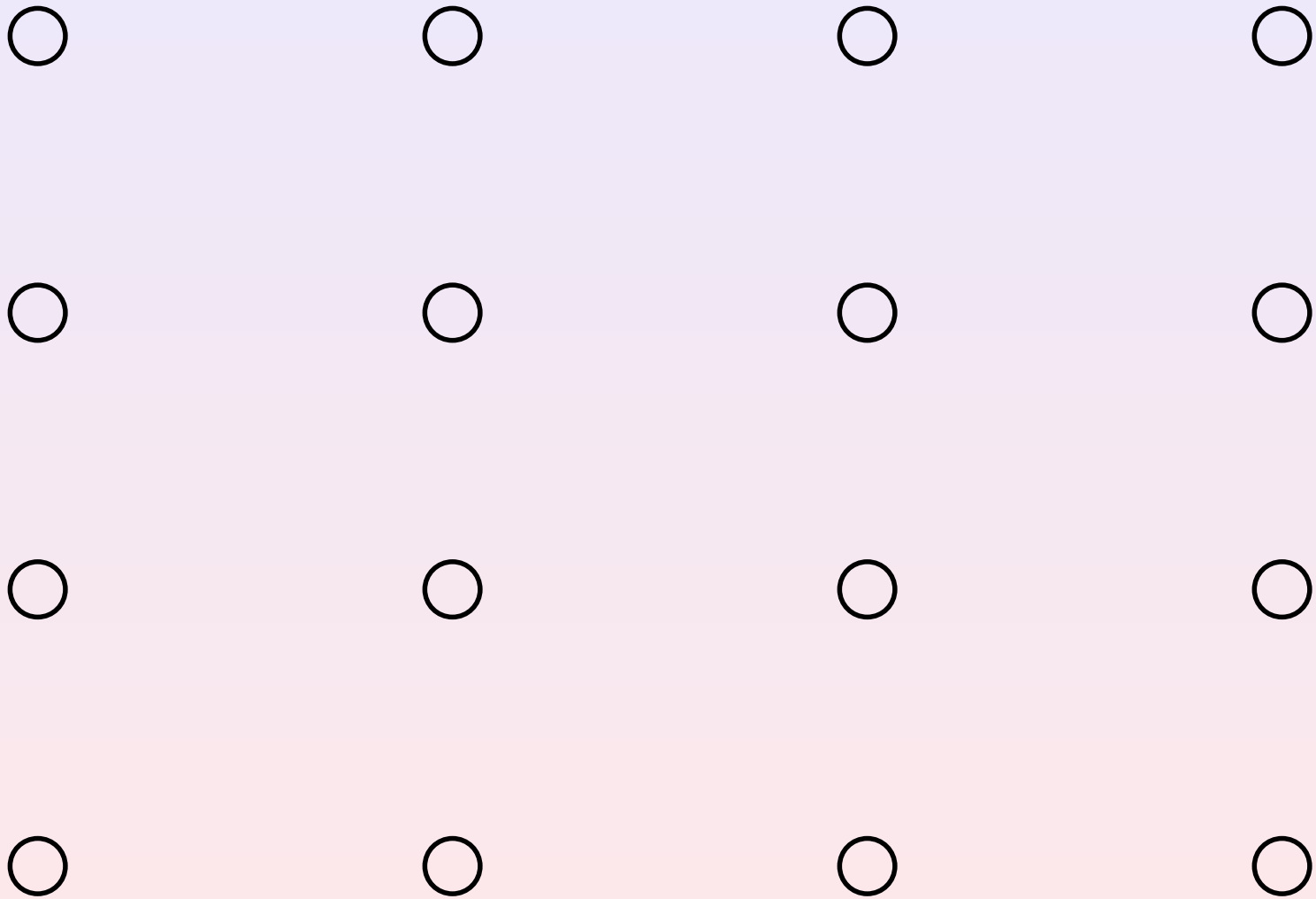
Main idea

- Start with isolated vertices, the initial trees.
- Merge trees to finally obtain a spanning tree.

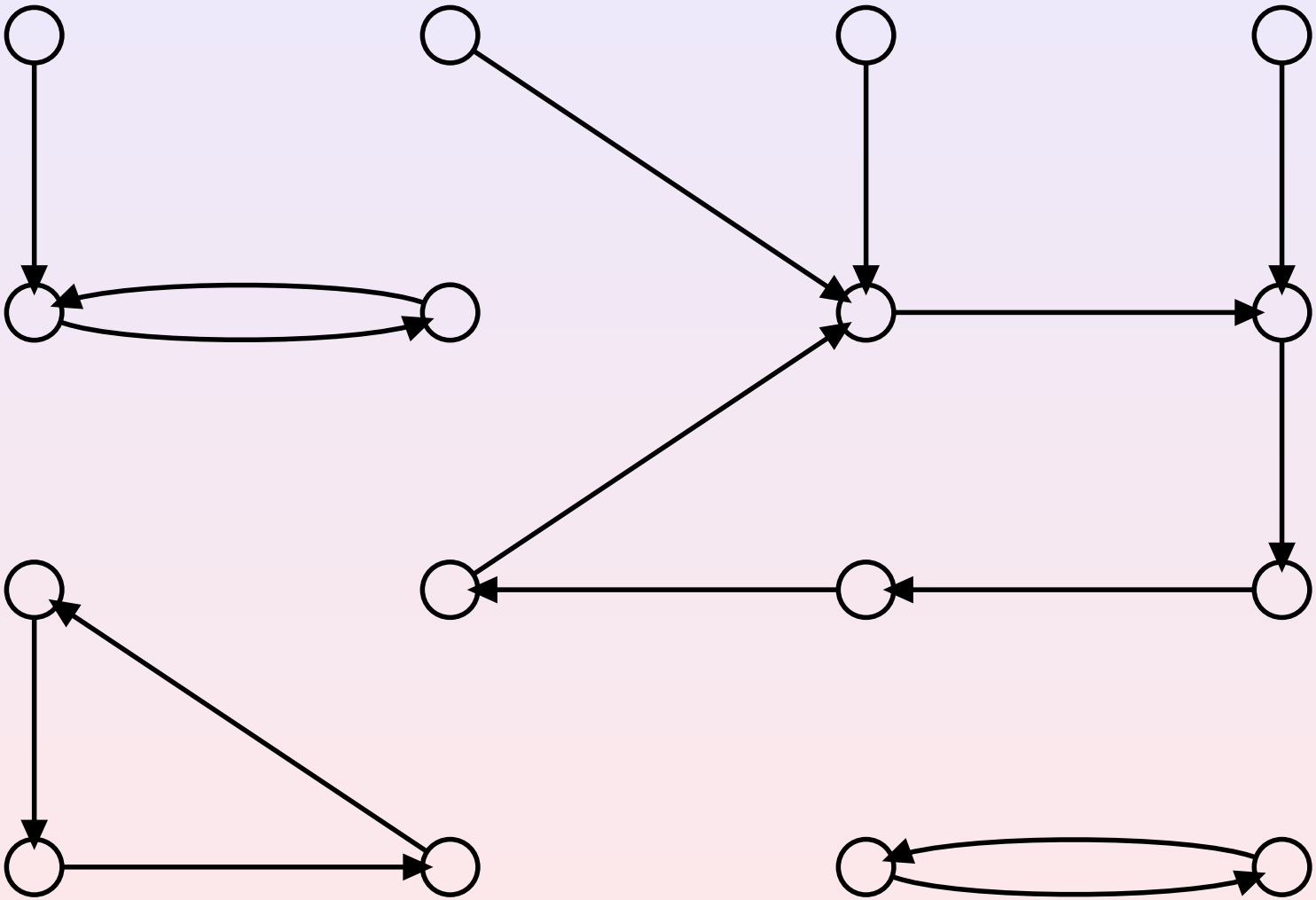
Description of phase k

1. Identify the collection of small trees for the phase: $\mathcal{T}_{\text{small}}(k) = \{T \mid \text{size}(T) < 2^k\}$.
2. For each tree $T \in \mathcal{T}_{\text{small}}(k)$, select the edge $e(T)$ of minimum weight going from the root outside T.
3. Merge the trees with the edges $e(T)$.
4. Erase one edge on each formed cycle to obtain trees.

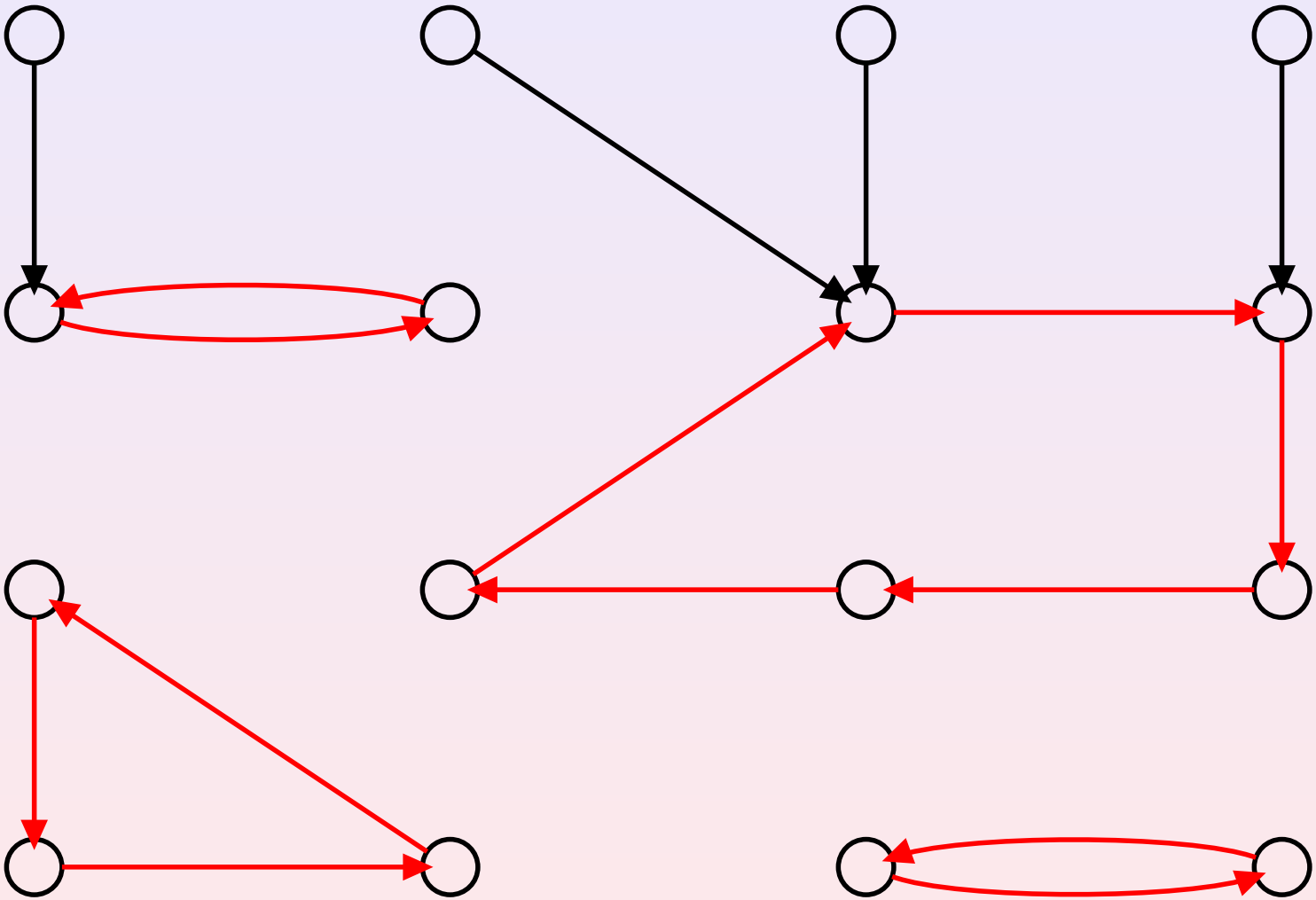
Example



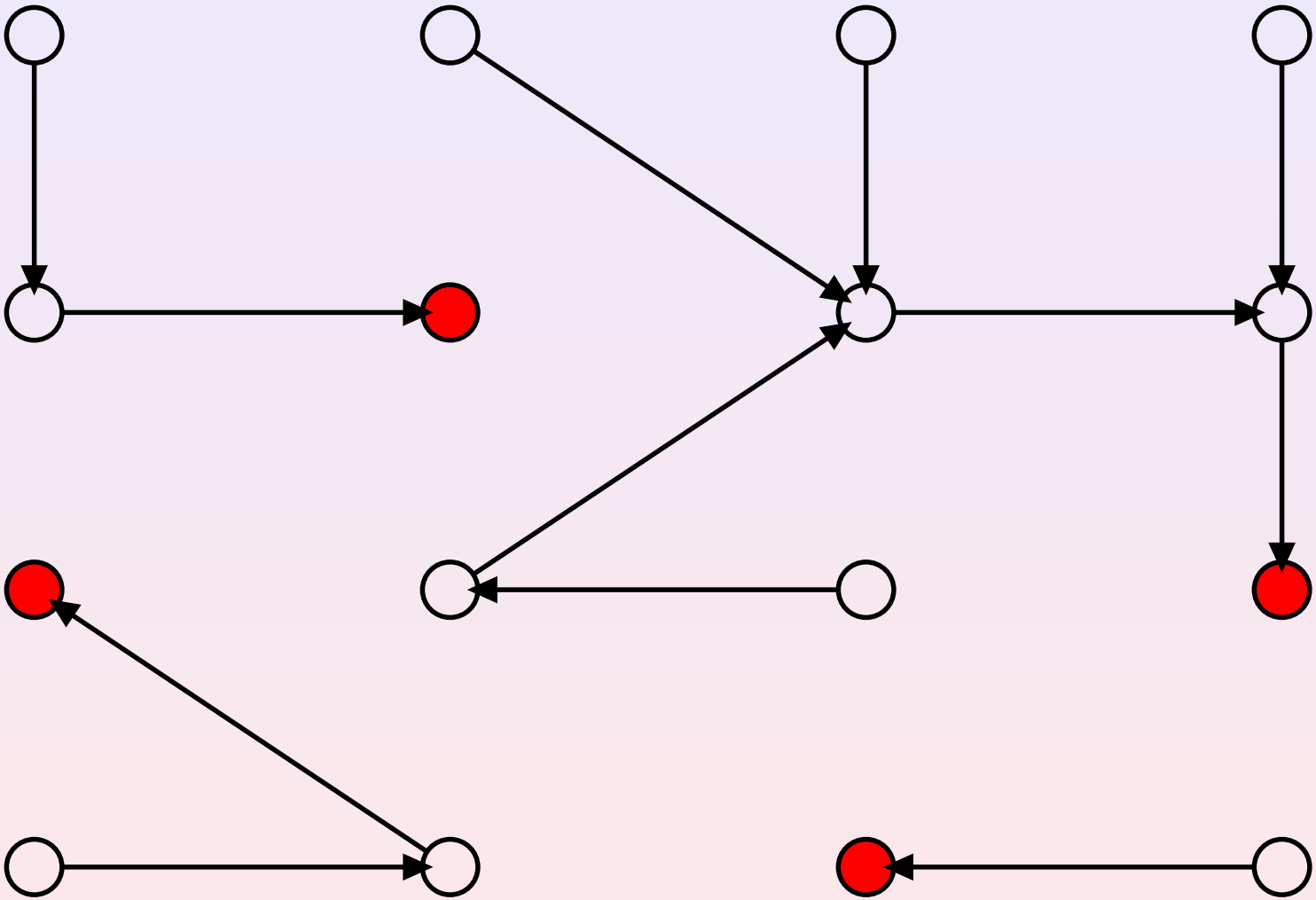
Example



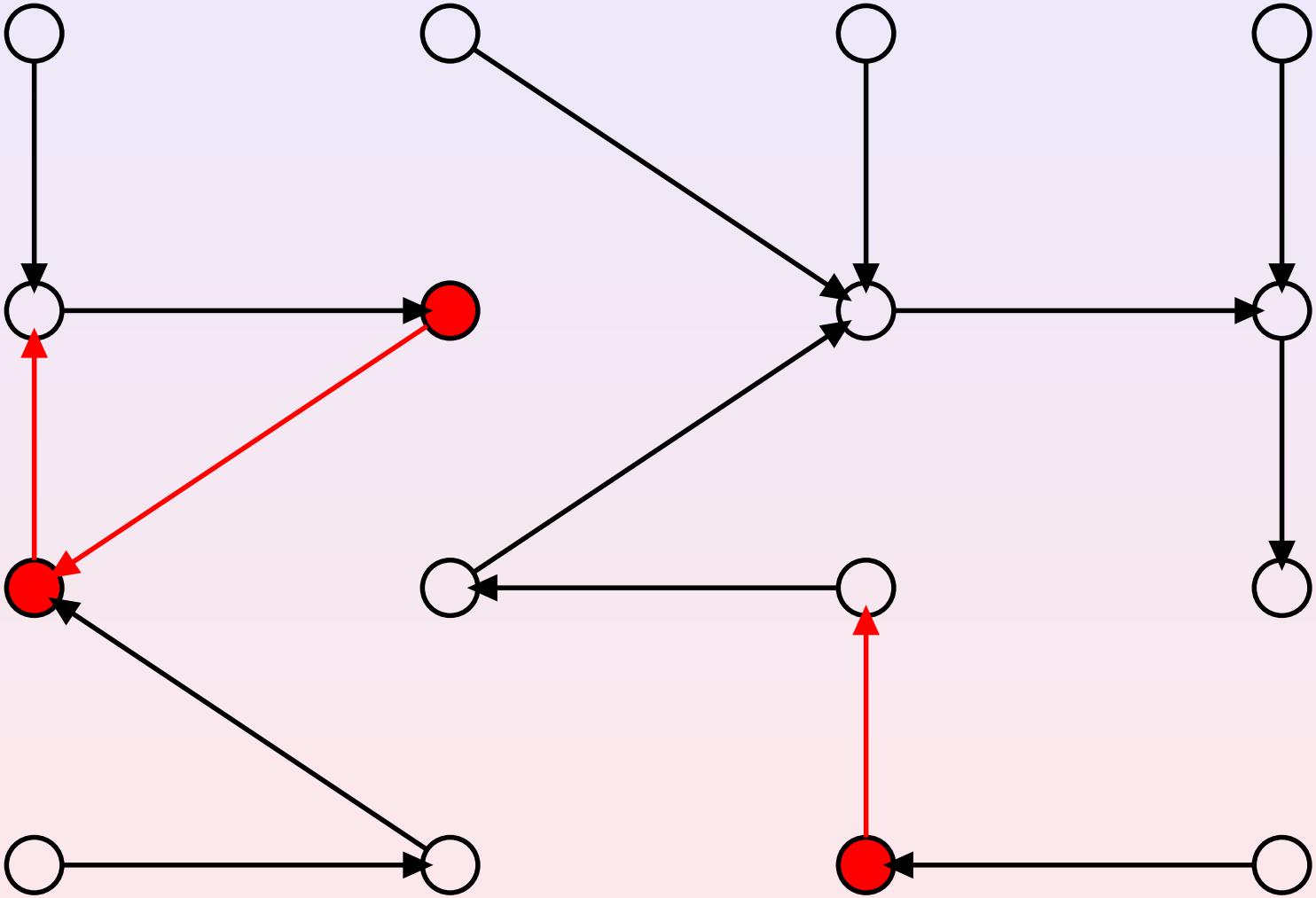
Example



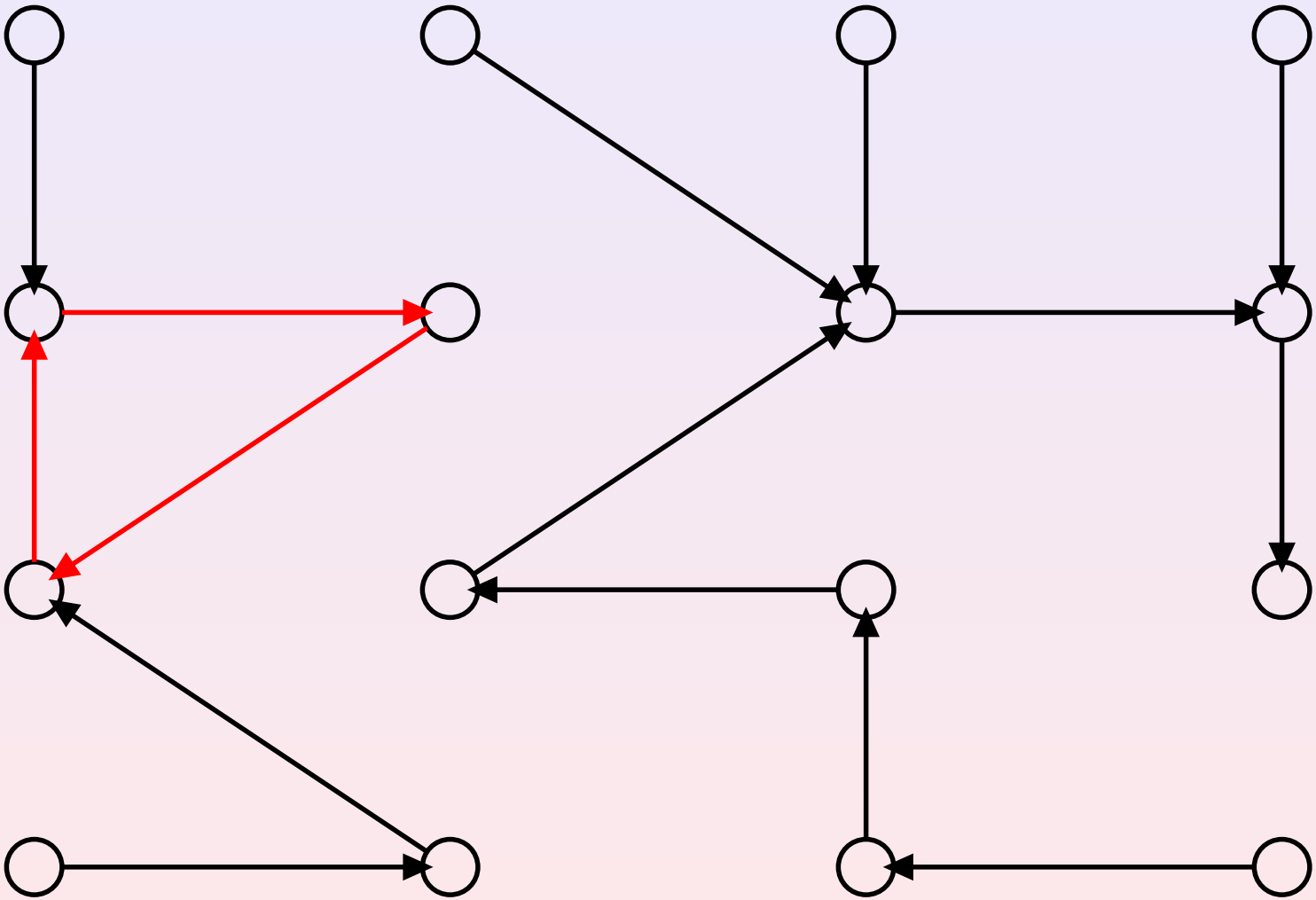
Example



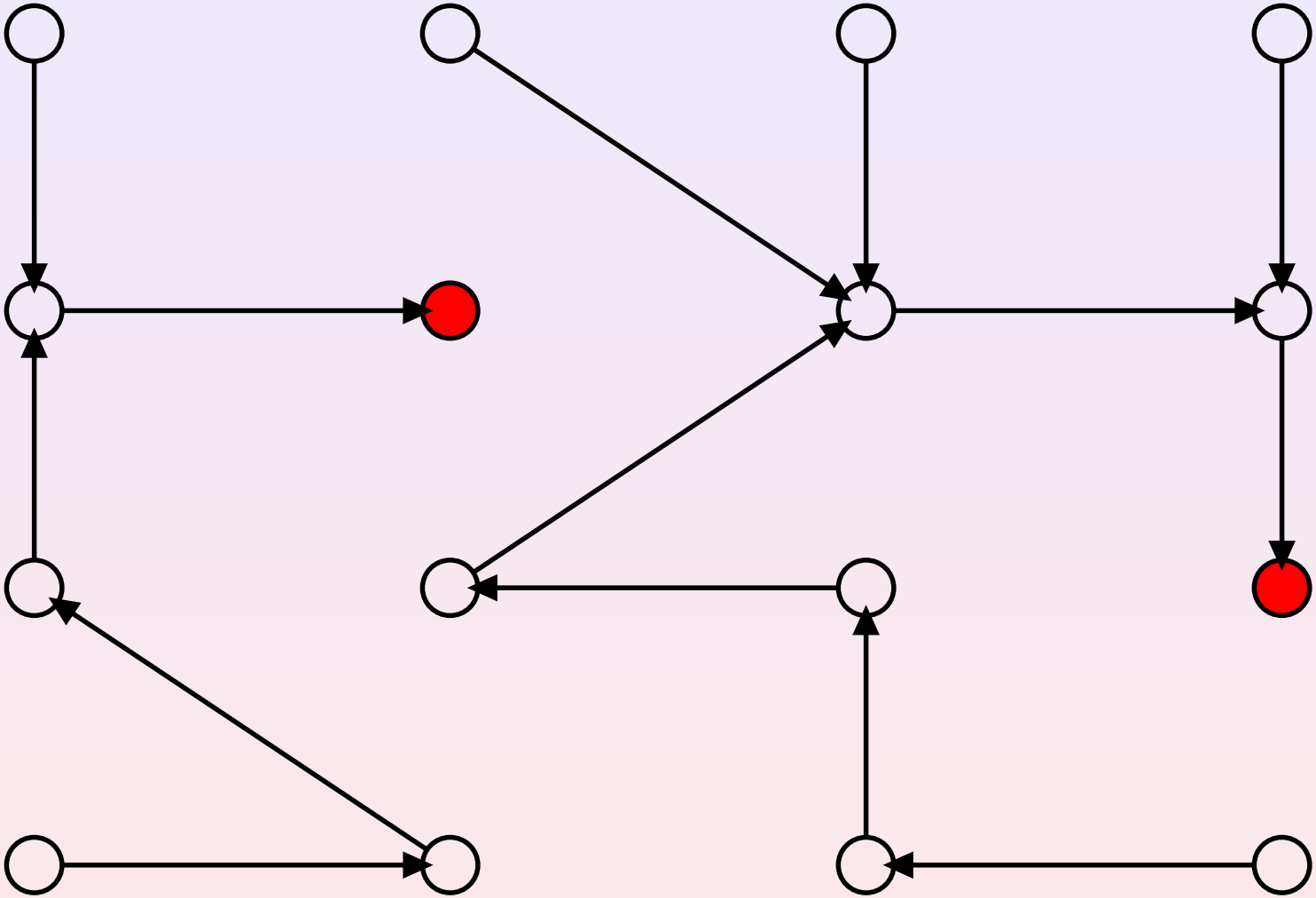
Example



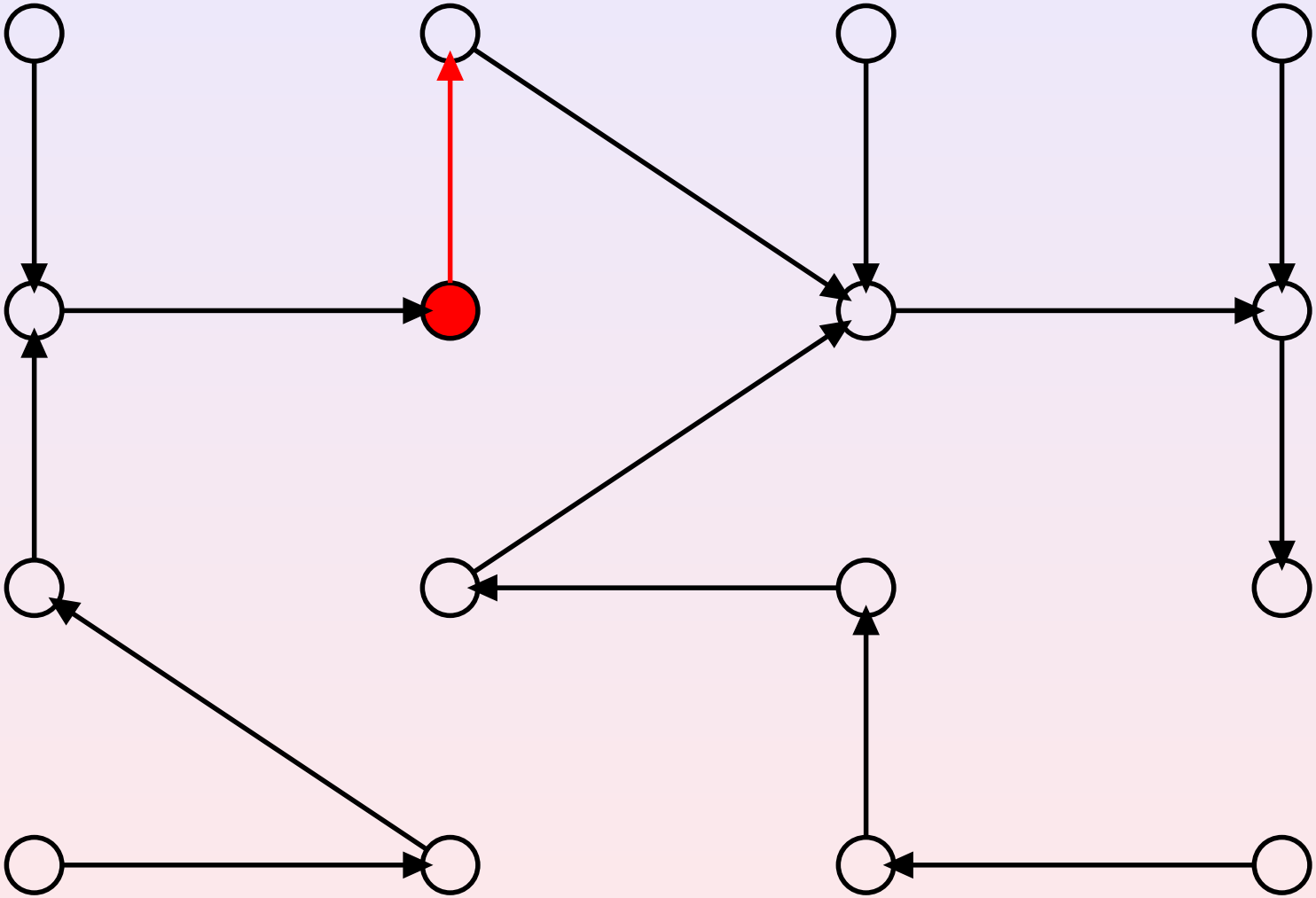
Example



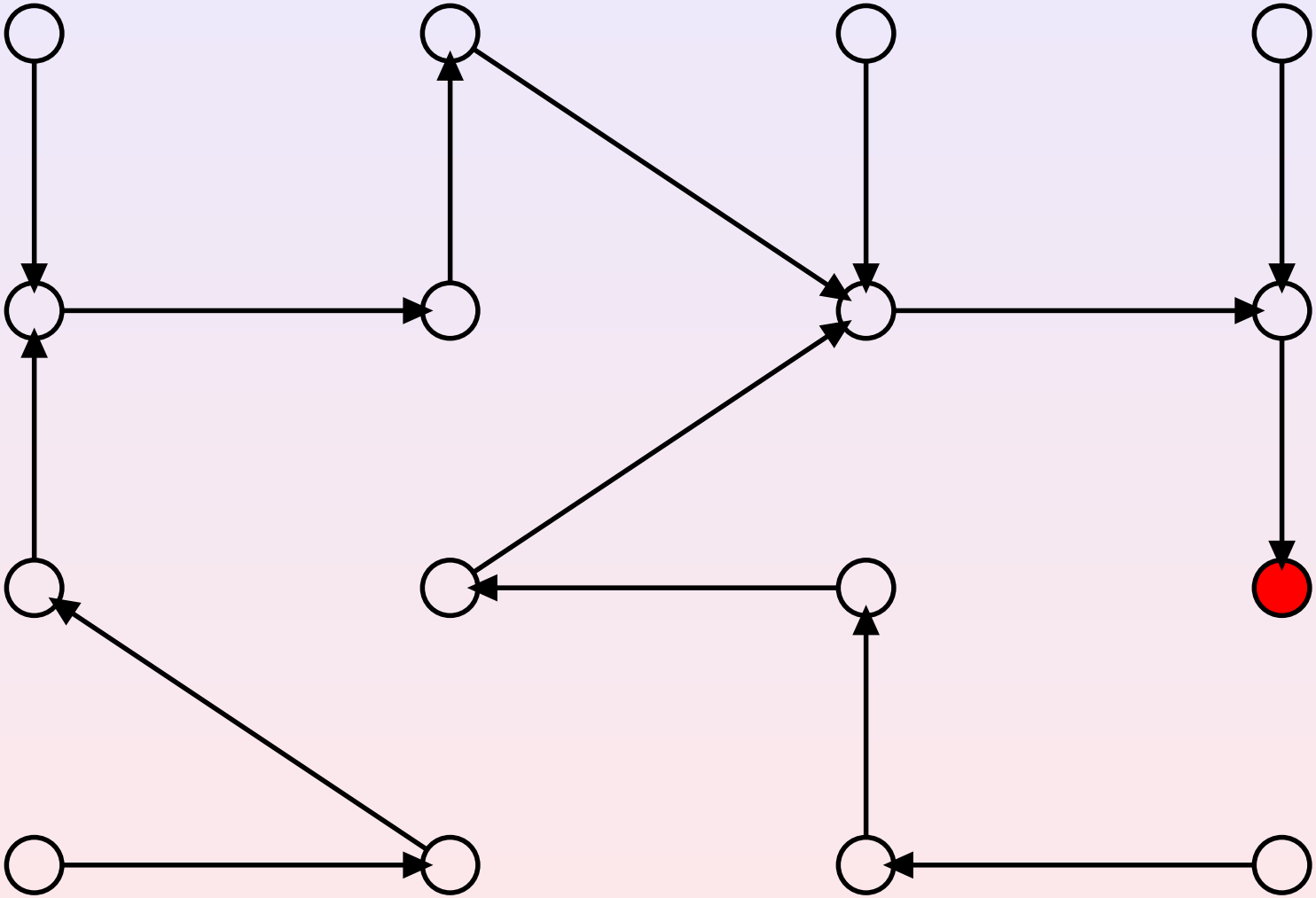
Example



Example



Example



Analysis (1)

Claim

1. The trees of a given phase form a partition of $V(G)$;
2. At the beginning of phase 1, all trees are of size $1 \geq 2^0$;
3. At the beginning of phase k, all trees are of size $\geq 2^{k-1}$;
4. There are $m_k \leq n/2^{k-1}$ trees at the beginning of phase k.
5. The number of phases is at most $\lceil \log n \rceil$.

Analysis (1)

Claim

1. The trees of a given phase form a partition of $V(G)$;
2. At the beginning of phase 1, all trees are of size $1 \geq 2^0$;
3. At the beginning of phase k, all trees are of size $\geq 2^{k-1}$;
4. There are $m_k \leq n/2^{k-1}$ trees at the beginning of phase k.
5. The number of phases is at most $\lceil \log n \rceil$.

Analysis (1)

Claim

1. The trees of a given phase form a partition of $V(G)$;
2. At the beginning of phase 1, all trees are of size $1 \geq 2^0$;
3. At the beginning of phase k, all trees are of size $\geq 2^{k-1}$;
4. There are $m_k \leq n/2^{k-1}$ trees at the beginning of phase k.
5. The number of phases is at most $\lceil \log n \rceil$.

Analysis (1)

Claim

1. The trees of a given phase form a partition of $V(G)$;
2. At the beginning of phase 1, all trees are of size $1 \geq 2^0$;
3. At the beginning of phase k, all trees are of size $\geq 2^{k-1}$;
4. There are $m_k \leq n/2^{k-1}$ trees at the beginning of phase k.
5. The number of phases is at most $\lceil \log n \rceil$.

Analysis (1)

Claim

1. The trees of a given phase form a partition of $V(G)$;
2. At the beginning of phase 1, all trees are of size $1 \geq 2^0$;
3. At the beginning of phase k, all trees are of size $\geq 2^{k-1}$;
4. There are $m_k \leq n/2^{k-1}$ trees at the beginning of phase k.
5. The number of phases is at most $\lceil \log n \rceil$.

Analysis (2)

Observation

- Consider the root r of a small tree T of x nodes.
- There are at most $x - 1$ outgoing edges from r leading to nodes inside T.
- The port number of $e(T)$ is at most $x - 1$.

Cost of the algorithm

Analysis (2)

Observation

- Consider the root r of a small tree T of x nodes.
- There are at most $x-1$ outgoing edges from r leading to nodes inside T.
- The port number of $e(T)$ is at most $x-1$.

Cost of the algorithm

Analysis (2)

Observation

- Consider the root r of a small tree T of x nodes.
- There are at most $x-1$ outgoing edges from r leading to nodes inside T.
- The port number of $e(T)$ is at most $x-1$.

Cost of the algorithm

Analysis (2)

Observation

- Consider the root r of a small tree T of x nodes.
- There are at most $x-1$ outgoing edges from r leading to nodes inside T.
- The port number of $e(T)$ is at most $x-1$.

Cost of the algorithm

Analysis (2)

Observation

- Consider the root r of a small tree T of x nodes.
- There are at most $x-1$ outgoing edges from r leading to nodes inside T.
- The port number of $e(T)$ is at most $x-1$.

Cost of the algorithm

- At the beginning of phase 1: no cost
- During phase k:
- Total cost at most $\sum_{k\geq 1} kn/2^{k-1} \leq 4n = O(n)$

Analysis (2)

Observation

- Consider the root r of a small tree T of x nodes.
- There are at most $x-1$ outgoing edges from r leading to nodes inside T.
- The port number of $e(T)$ is at most $x-1$.

Cost of the algorithm

- At the beginning of phase 1: no cost
- During phase k:
  - at most $n/2^{k-1}$ new edges
  - at most k bits per edge
- Total cost at most $\sum_{k\geq 1} kn/2^{k-1} \leq 4n = O(n)$

Analysis (2)

Observation

- Consider the root r of a small tree T of x nodes.
- There are at most $x-1$ outgoing edges from r leading to nodes inside T.
- The port number of $e(T)$ is at most $x-1$.

Cost of the algorithm

- At the beginning of phase 1: no cost
- During phase k:
  - at most $n/2^{k-1}$ new edges
  - at most k bits per edge
- Total cost at most $\sum_{k\geq 1} kn/2^{k-1} \leq 4n = O(n)$

Analysis (2)

Observation

- Consider the root r of a small tree T of x nodes.
- There are at most $x-1$ outgoing edges from r leading to nodes inside T.
- The port number of $e(T)$ is at most $x-1$.

Cost of the algorithm

- At the beginning of phase 1: no cost
- During phase k:
  - at most $n/2^{k-1}$ new edges
  - at most k bits per edge
- Total cost at most $\sum_{k\geq 1} kn/2^{k-1} \leq 4n = O(n)$

Analysis (2)

Observation

- Consider the root r of a small tree T of x nodes.
- There are at most $x-1$ outgoing edges from r leading to nodes inside T.
- The port number of $e(T)$ is at most $x-1$.

Cost of the algorithm

- At the beginning of phase 1: no cost
- During phase k:
  - at most $n/2^{k-1}$ new edges
  - at most k bits per edge
- Total cost at most $\sum_{k\geq 1} kn/2^{k-1} \leq 4n = O(n)$

Arbitrary graphs with symmetric port assignments

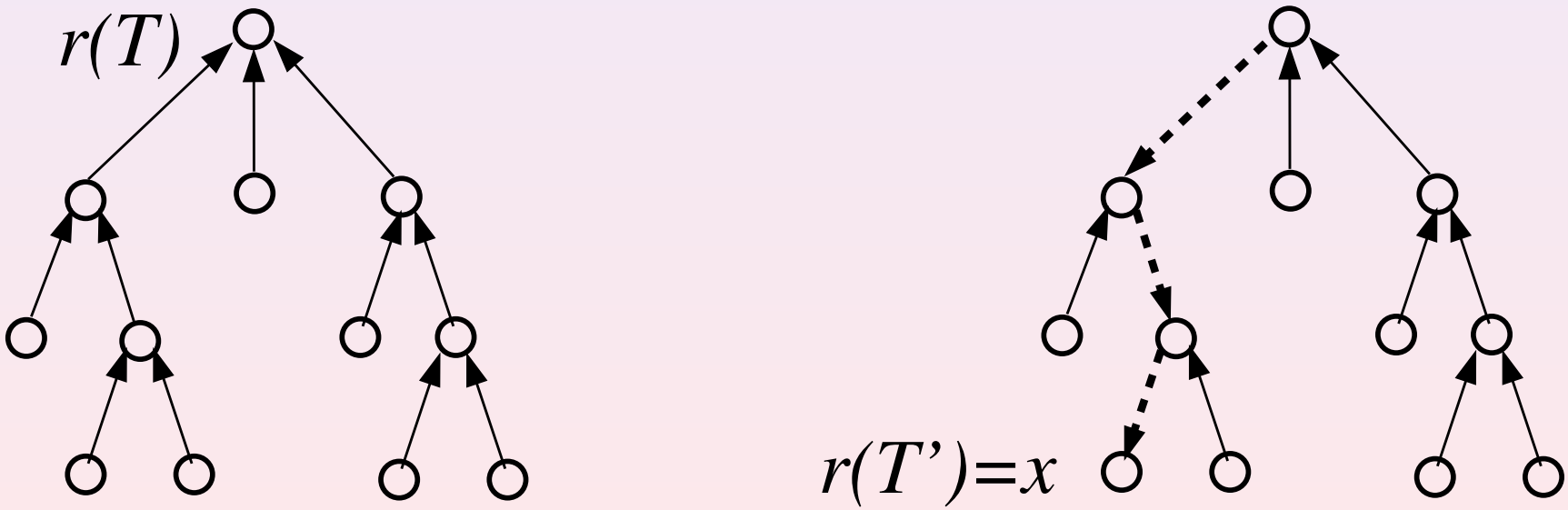
Remark

- The root of a small tree may have no edges going outside the tree.
- But some node of the tree has.

Arbitrary graphs with symmetric port assignments

Remark

- The root of a small tree may have no edges going outside the tree.
- But some node of the tree has.



Outline

Arbitrary graphs with arbitrary labeling

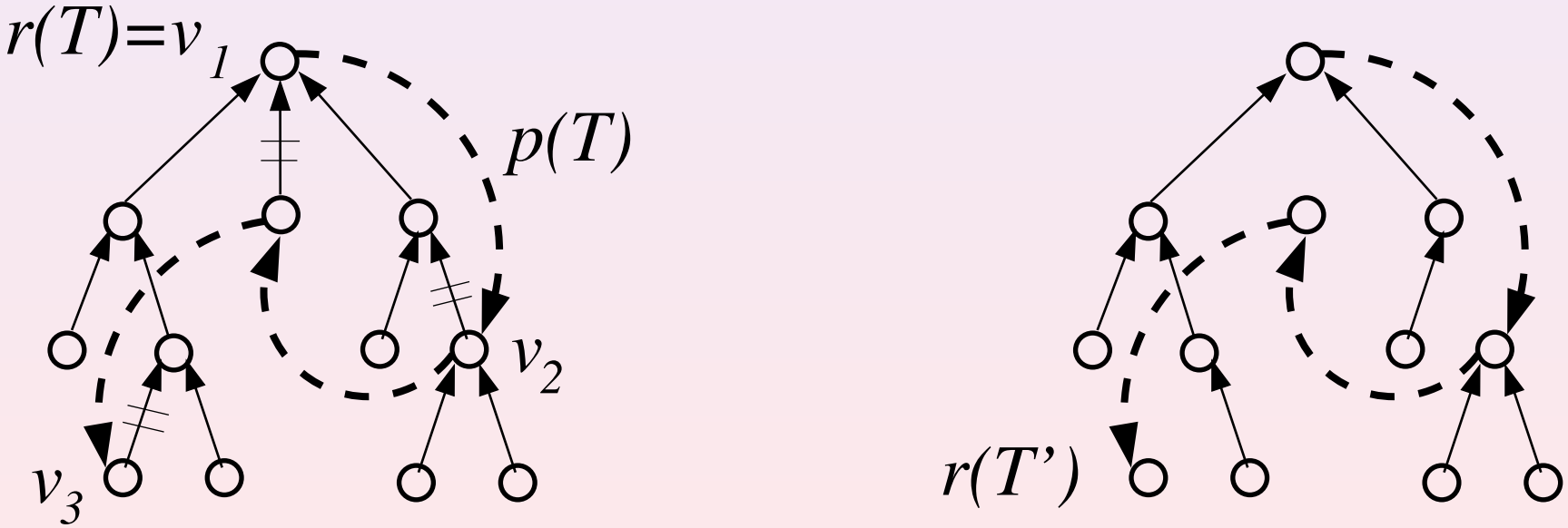
Main ideas

- Consider the nodes with edges going outside the tree.
- Choose the closest to the root according to the distance in hops in the tree.

Arbitrary graphs with arbitrary labeling

Main ideas

- Consider the nodes with edges going outside the tree.
- Choose the closest to the root according to the distance in hops in the tree.



Outline

Conclusion

Open problem

- Lower bound
  - $S_{up} = \Omega(n)$
- Upper bounds
  - $S_{up} = O(n \log \Delta)$
  - $S_{up} = O(n \log \log n)$

Close the gap.

For further information

An extended version of this paper will appear in the Proceedings of IWDC 2005.

Conclusion

Open problem

- Lower bound
  - $S_{up} = \Omega(n)$
- Upper bounds
  - $S_{up} = O(n \log \Delta)$
  - $S_{up} = O(n \log \log n)$

Close the gap.

For further information

An extended version of this paper will appear in the Proceedings of IWDC 2005.