Leader Election I

Distributed Algorithms

Sebastian Forster

University of Salzburg



This work is licensed under a Creative Commons Attribution 4.0 International License.

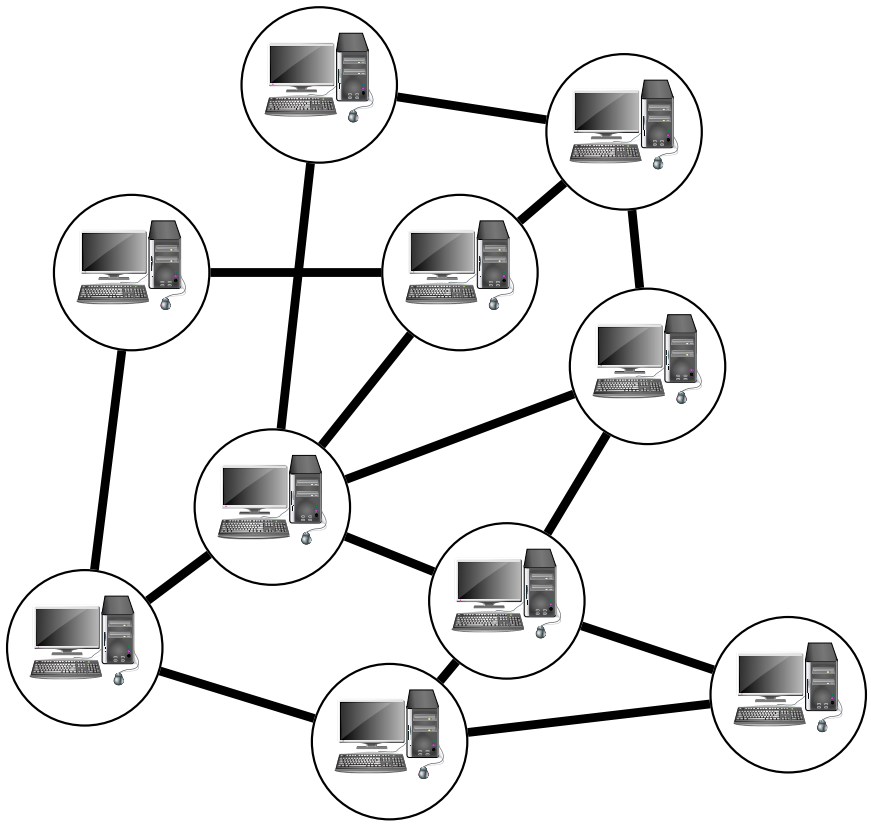
Processor Networks

Initial situation: Several identical processors communicating via direct connections

Processor Networks

Initial situation: Several identical processors communicating via direct connections

- Processor networks are modeled as undirected graphs.
- Each vertex of the graph corresponds to a processor with local memory, and each edge to a communication channel.
- Local view: Each processor knows only its direct neighbors.



Networks as Graphs

Definition

A **graph** $G = (V, E)$ is a pair consisting of a set of vertices V and a set of edges E, where

- in a **directed** graph, each edge $e = (u, v)$ is an (ordered) pair of vertices, and
- in an **undirected** graph, each edge $e = \{u, v\}$ is a set of two vertices.

Remark: Nevertheless, we will often write (u, v) instead of $\{u, v\}$

Motivation: Leader Election

Leader Election:

- The nodes of a network agree on a *leader*.
- All others are called *followers*.

Motivation: Leader Election

Leader Election:

- The nodes of a network agree on a *leader*.
- All others are called *followers*.
- **Goal:** An algorithm that decides for each node whether it is the leader or a follower.
- **Motivation:** The leader can take over coordination tasks.

Motivation: Leader Election

Leader Election:

- The nodes of a network agree on a *leader*.
- All others are called *followers*.
- **Goal:** An algorithm that decides for each node whether it is the leader or a follower.
- **Motivation:** The leader can take over coordination tasks.

Symmetry Breaking:

- A priori, every node is suitable as leader.
- The solution is not unique; there are n different solutions.

Motivation: Leader Election

Leader Election:

- The nodes of a network agree on a *leader*.
- All others are called *followers*.
- **Goal:** An algorithm that decides for each node whether it is the leader or a follower.
- **Motivation:** The leader can take over coordination tasks.

Symmetry Breaking:

- A priori, every node is suitable as leader.
- The solution is not unique; there are n different solutions.
- The difficulty in finding a solution is making a consistent choice among these solutions.

Communication Model

- The communication network is specified by an undirected graph.

Communication Model

- The communication network is specified by an undirected graph.
- Each node knows the number of its neighbors.

Communication Model

- The communication network is specified by an undirected graph.
- Each node knows the number of its neighbors.
- Each node can send arbitrary messages to, and receive arbitrary messages from, its neighbors via numbered ports.

Communication Model

- The communication network is specified by an undirected graph.
- Each node knows the number of its neighbors.
- Each node can send arbitrary messages to, and receive arbitrary messages from, its neighbors via numbered ports.
- Synchronous:
  - Round-based communication:

Communication Model

- The communication network is specified by an undirected graph.
- Each node knows the number of its neighbors.
- Each node can send arbitrary messages to, and receive arbitrary messages from, its neighbors via numbered ports.
- Synchronous:
  - Round-based communication:
    1. Receive messages from neighbors (omitted in the first round).
    2. Perform internal computations.
    3. Send messages to neighbors.

Communication Model

- The communication network is specified by an undirected graph.
- Each node knows the number of its neighbors.
- Each node can send arbitrary messages to, and receive arbitrary messages from, its neighbors via numbered ports.
- Synchronous:
  - Round-based communication:
    1. Receive messages from neighbors (omitted in the first round).
    2. Perform internal computations.
    3. Send messages to neighbors.
  - The beginning of each round is determined by a global clock.

Communication Model

- The communication network is specified by an undirected graph.
- Each node knows the number of its neighbors.
- Each node can send arbitrary messages to, and receive arbitrary messages from, its neighbors via numbered ports.
- Synchronous:
  - Round-based communication:
    1. Receive messages from neighbors (omitted in the first round).
    2. Perform internal computations.
    3. Send messages to neighbors.
  - The beginning of each round is determined by a global clock.
- Asynchronous:
  - Event-based rather than based on discrete time steps.

Communication Model

- The communication network is specified by an undirected graph.
- Each node knows the number of its neighbors.
- Each node can send arbitrary messages to, and receive arbitrary messages from, its neighbors via numbered ports.
- Synchronous:
  - Round-based communication:
    1. Receive messages from neighbors (omitted in the first round).
    2. Perform internal computations.
    3. Send messages to neighbors.
  - The beginning of each round is determined by a global clock.
- Asynchronous:
  - Event-based rather than based on discrete time steps.
  - Each node starts initially at an arbitrary point in time and thereafter becomes active whenever it receives a message.

Communication Model

- The communication network is specified by an undirected graph.
- Each node knows the number of its neighbors.
- Each node can send arbitrary messages to, and receive arbitrary messages from, its neighbors via numbered ports.
- Synchronous:
  - Round-based communication:
    1. Receive messages from neighbors (omitted in the first round).
    2. Perform internal computations.
    3. Send messages to neighbors.
  - The beginning of each round is determined by a global clock.
- Asynchronous:
  - Event-based rather than based on discrete time steps.
  - Each node starts initially at an arbitrary point in time and thereafter becomes active whenever it receives a message.
  - Each message transmission takes finite time.

Communication Model

- The communication network is specified by an undirected graph.
- Each node knows the number of its neighbors.
- Each node can send arbitrary messages to, and receive arbitrary messages from, its neighbors via numbered ports.
- Synchronous:
  - Round-based communication:
    1. Receive messages from neighbors (omitted in the first round).
    2. Perform internal computations.
    3. Send messages to neighbors.
  - The beginning of each round is determined by a global clock.
- Asynchronous:
  - Event-based rather than based on discrete time steps.
  - Each node starts initially at an arbitrary point in time and thereafter becomes active whenever it receives a message.
  - Each message transmission takes finite time.
  - For runtime analysis: delay of at most one time unit.

Communication Model

- The communication network is specified by an undirected graph.
- Each node knows the number of its neighbors.
- Each node can send arbitrary messages to, and receive arbitrary messages from, its neighbors via numbered ports.
- Synchronous:
  - Round-based communication:
    1. Receive messages from neighbors (omitted in the first round).
    2. Perform internal computations.
    3. Send messages to neighbors.
  - The beginning of each round is determined by a global clock.
- Asynchronous:
  - Event-based rather than based on discrete time steps.
  - Each node starts initially at an arbitrary point in time and thereafter becomes active whenever it receives a message.
  - Each message transmission takes finite time.
  - For runtime analysis: delay of at most one time unit.
- Complexity measures: number of time units (rounds) and number of messages sent (excluding internal computation time).

Variants

- Synchronous vs. asynchronous.

Variants

- Synchronous vs. asynchronous.
- Anonymous vs. identifiable.

Variants

- Synchronous vs. asynchronous.
- Anonymous vs. identifiable.
  Identifiable nodes v have a unique ID (UID) $\mathrm{ID}(v)$.
  IDs are usually bit strings, often of length $O(\log n)$.
  They can be interpreted as nonnegative integers.

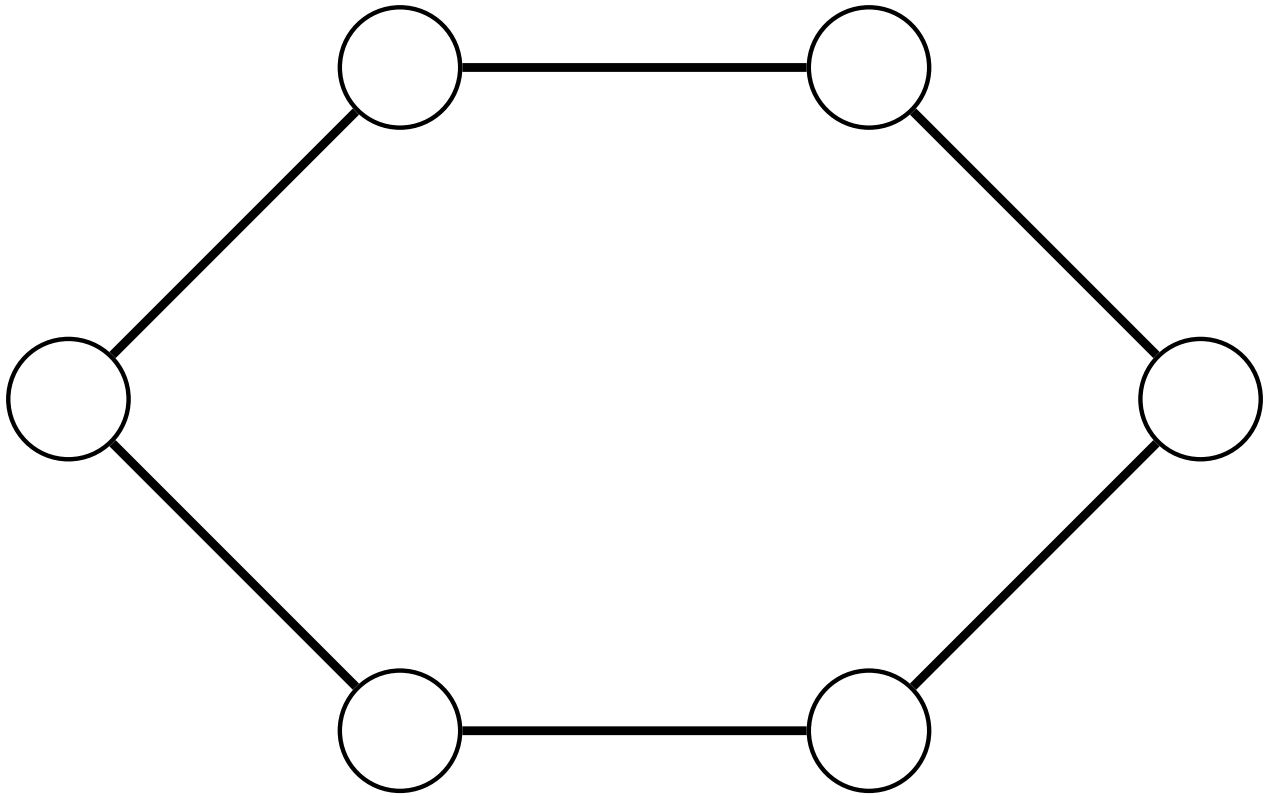
Variants

- Synchronous vs. asynchronous.
- Anonymous vs. identifiable.
  Identifiable nodes v have a unique ID (UID) $\mathrm{ID}(v)$.
  IDs are usually bit strings, often of length $O(\log n)$.
  They can be interpreted as nonnegative integers.
- Uniform vs. non-uniform.

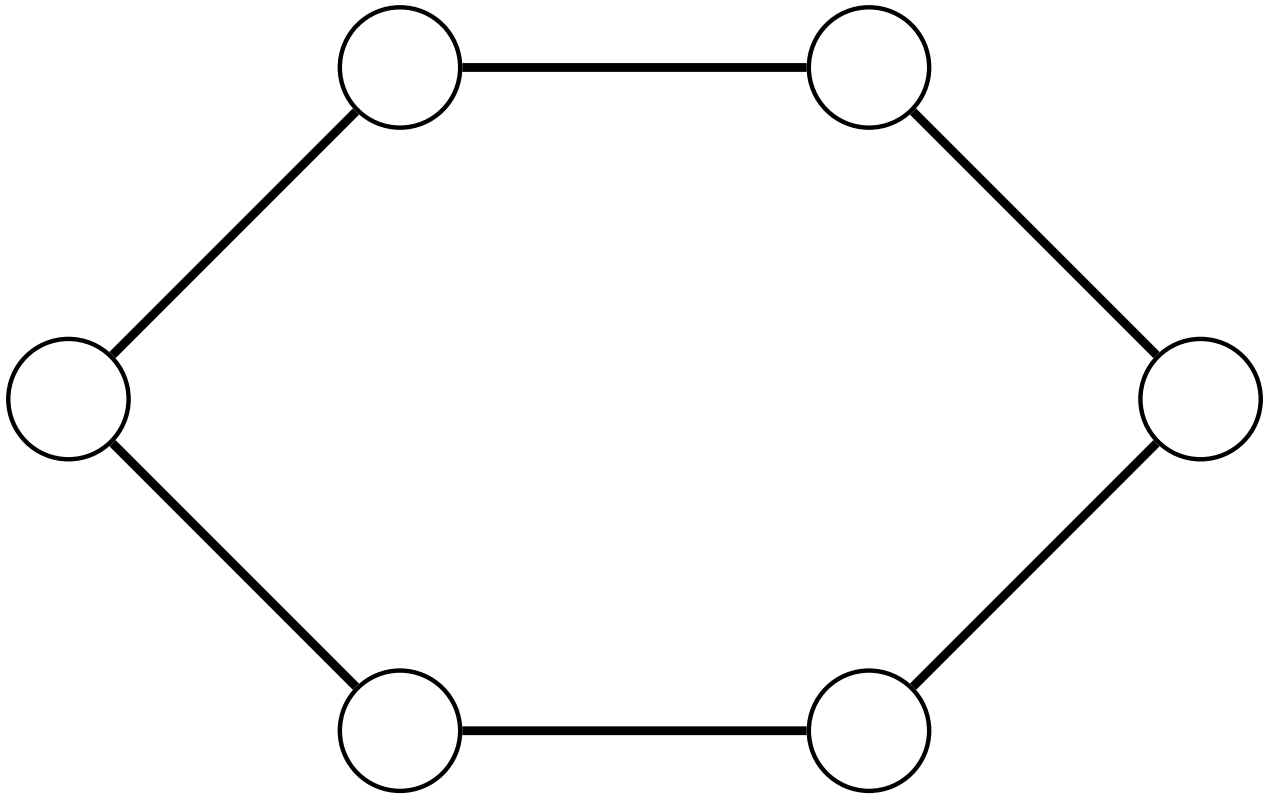
Variants

- Synchronous vs. asynchronous.
- Anonymous vs. identifiable.
  Identifiable nodes v have a unique ID (UID) $\mathrm{ID}(v)$.
  IDs are usually bit strings, often of length $O(\log n)$.
  They can be interpreted as nonnegative integers.
- Uniform vs. non-uniform.
  In uniform networks, the number of nodes n is not global knowledge.
  Weak non-uniformity: Sometimes an approximation $\hat{n}$ of the exact value n is sufficient.

Goal: Leader in the Ring



Goal: Leader in the Ring



Each node has two ports to neighboring nodes:

- “Clockwise”
- “Counterclockwise”

Leader Election with Identifiable Nodes

Idea

The node with the smallest ID becomes the leader.

Leader Election with Identifiable Nodes

Idea

The node with the smallest ID becomes the leader.

Question: Which algorithm can be used to determine this node?

Leader Election with Identifiable Nodes

Idea

The node with the smallest ID becomes the leader.

Question: Which algorithm can be used to determine this node?

Caution!

The smallest assigned ID is not known!

The problem would be trivial, for example, if there were always a node with ID 0.

Simple Clockwise Algorithm

Assumptions: synchronous, identifiable, non-uniform.

Simple Clockwise Algorithm

Assumptions: synchronous, identifiable, non-uniform.

Each node v executes the following algorithm:

Simple Clockwise Algorithm

Assumptions: synchronous, identifiable, non-uniform.

Each node v executes the following algorithm:

Round 1:

1. v sets $L_v := \mathrm{ID}(v)$ (local variable for the smallest ID seen so far)
2. v sends L_v to its clockwise neighbor

Simple Clockwise Algorithm

Assumptions: synchronous, identifiable, non-uniform.

Each node v executes the following algorithm:

Round 1:

1. v sets $L_v := \mathrm{ID}(v)$ (local variable for the smallest ID seen so far)
2. v sends L_v to its clockwise neighbor

Round $2 \leq r \leq n$:

1. **if** v *receives message* M **then**

Simple Clockwise Algorithm

Assumptions: synchronous, identifiable, non-uniform.

Each node v executes the following algorithm:

Round 1:

1 v sets $L_v := \mathrm{ID}(v)$ (local variable for the smallest ID seen so far)
2 v sends L_v to its clockwise neighbor

Round $2 \leq r \leq n$:

```
1 if v receives message M then
2     if M < L_v then
3         v sets L_v := M
4         v sends L_v to its clockwise neighbor
```

Simple Clockwise Algorithm

Assumptions: synchronous, identifiable, non-uniform.

Each node v executes the following algorithm:

Round 1:

1 v sets $L_v := \mathrm{ID}(v)$ (local variable for the smallest ID seen so far)
2 v sends L_v to its clockwise neighbor

Round $2 \leq r \leq n$:

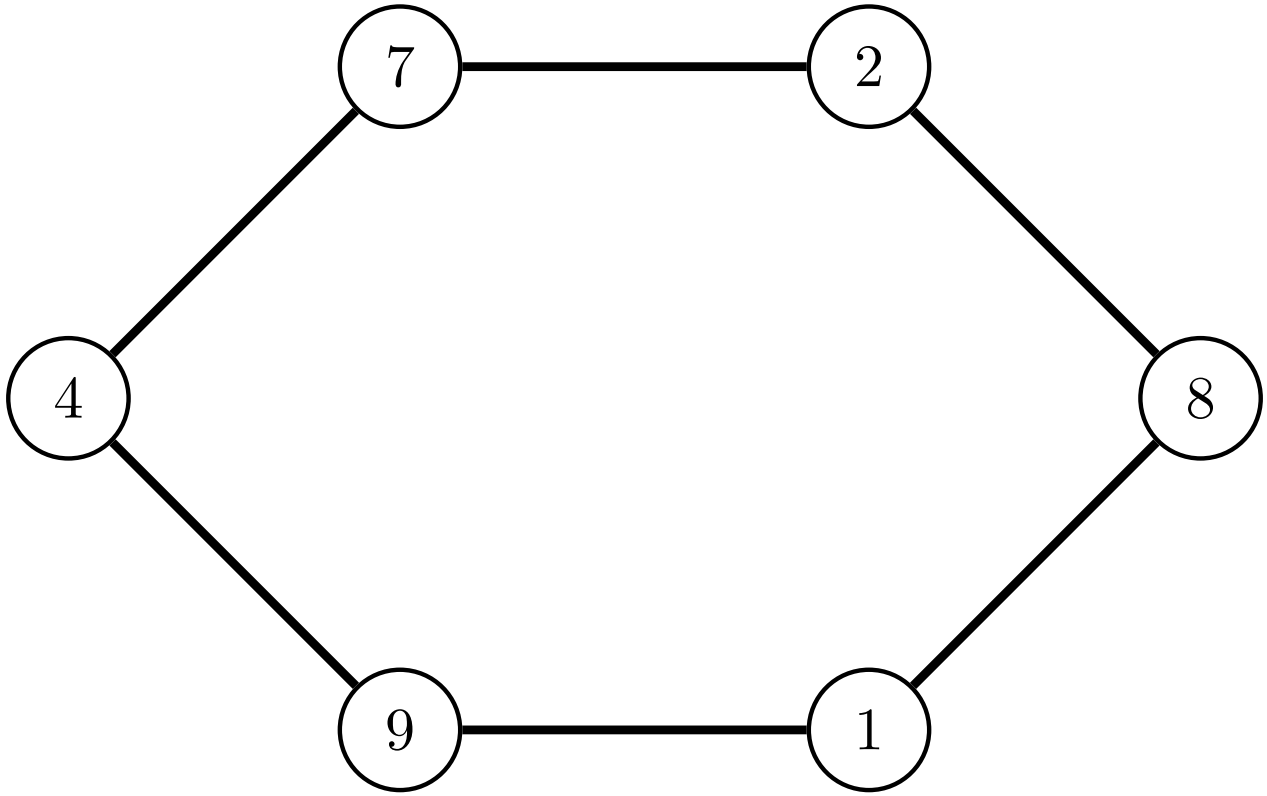
1 **if** v *receives message* M **then**
2 **if** $M < L_v$ **then**
3 v sets $L_v := M$
4 v sends L_v to its clockwise neighbor

Round $n + 1$:

1 **if** $L_v = \mathrm{ID}(v)$ **then**
2 v becomes the leader
3 **else**
4 v becomes a follower

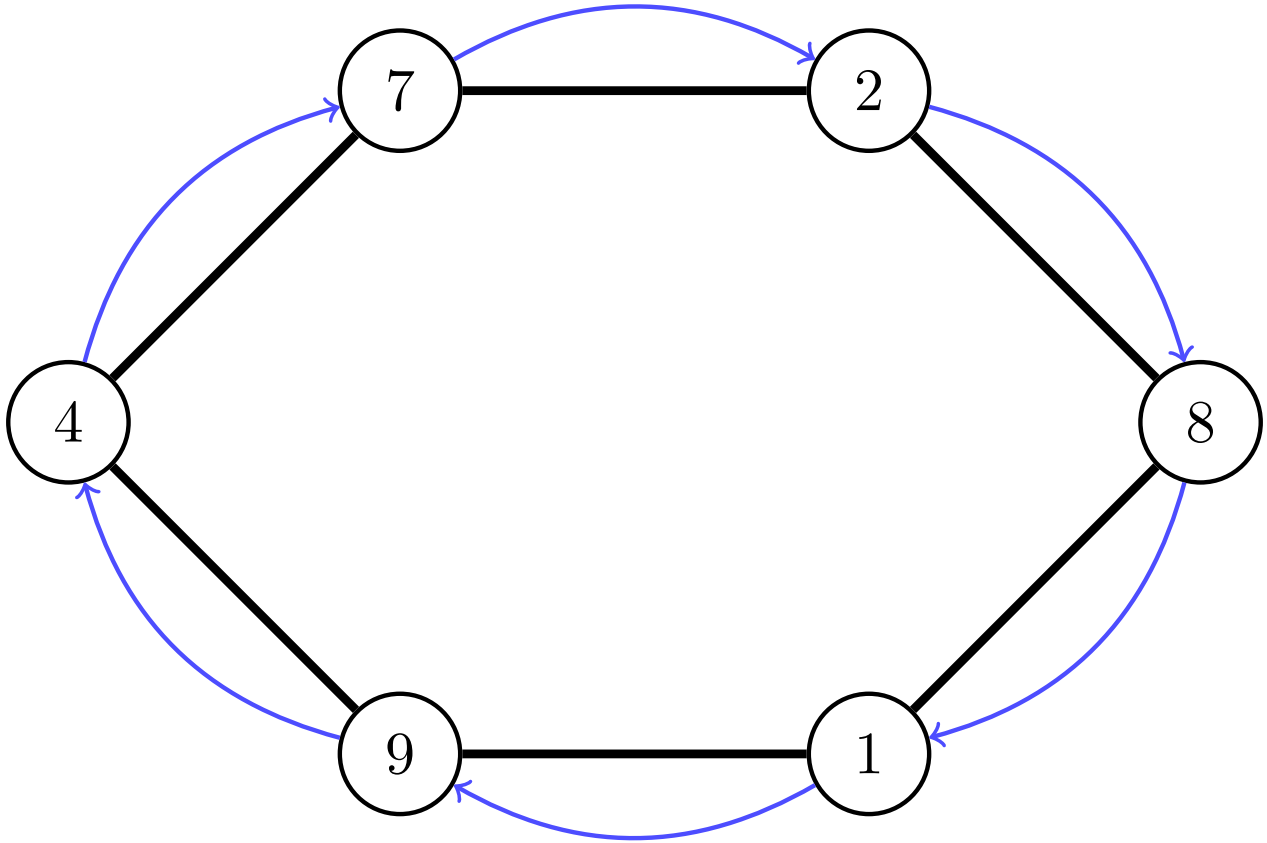
Example

Round 1:



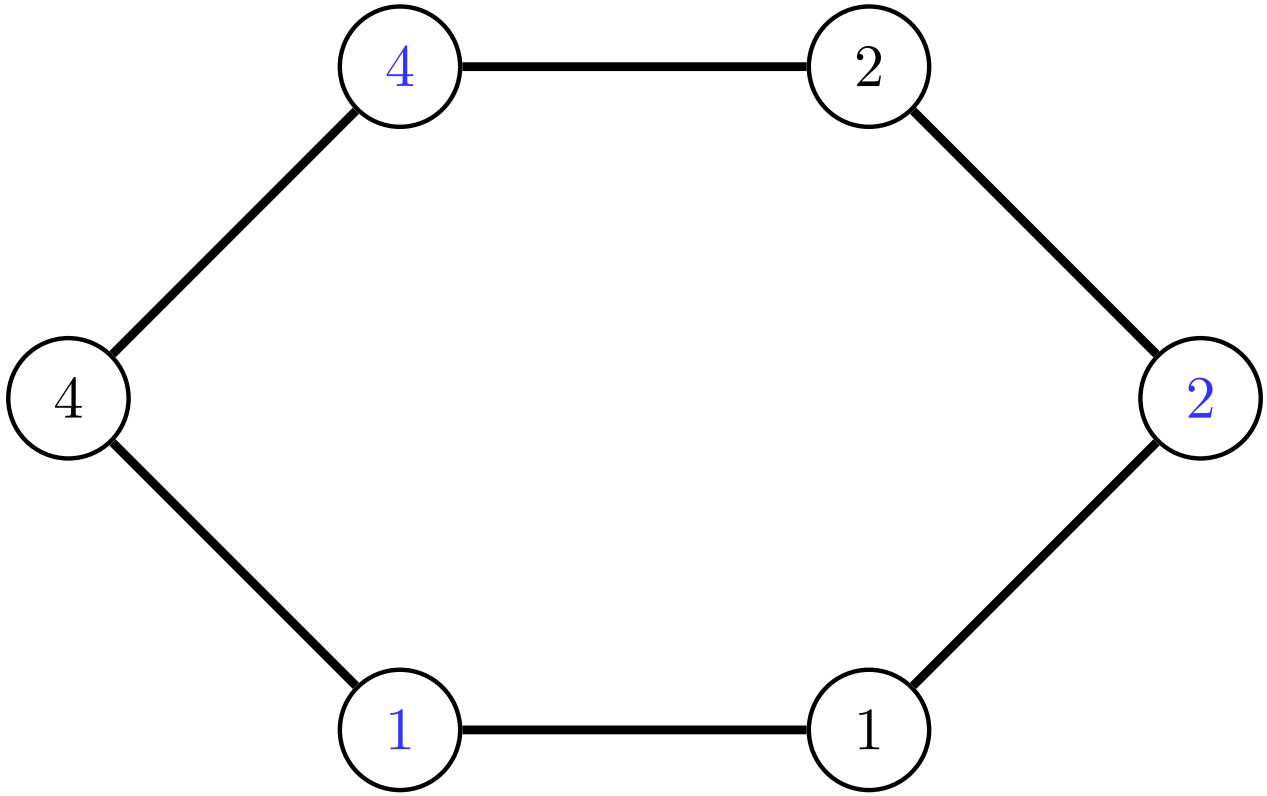
Example

Round 1:



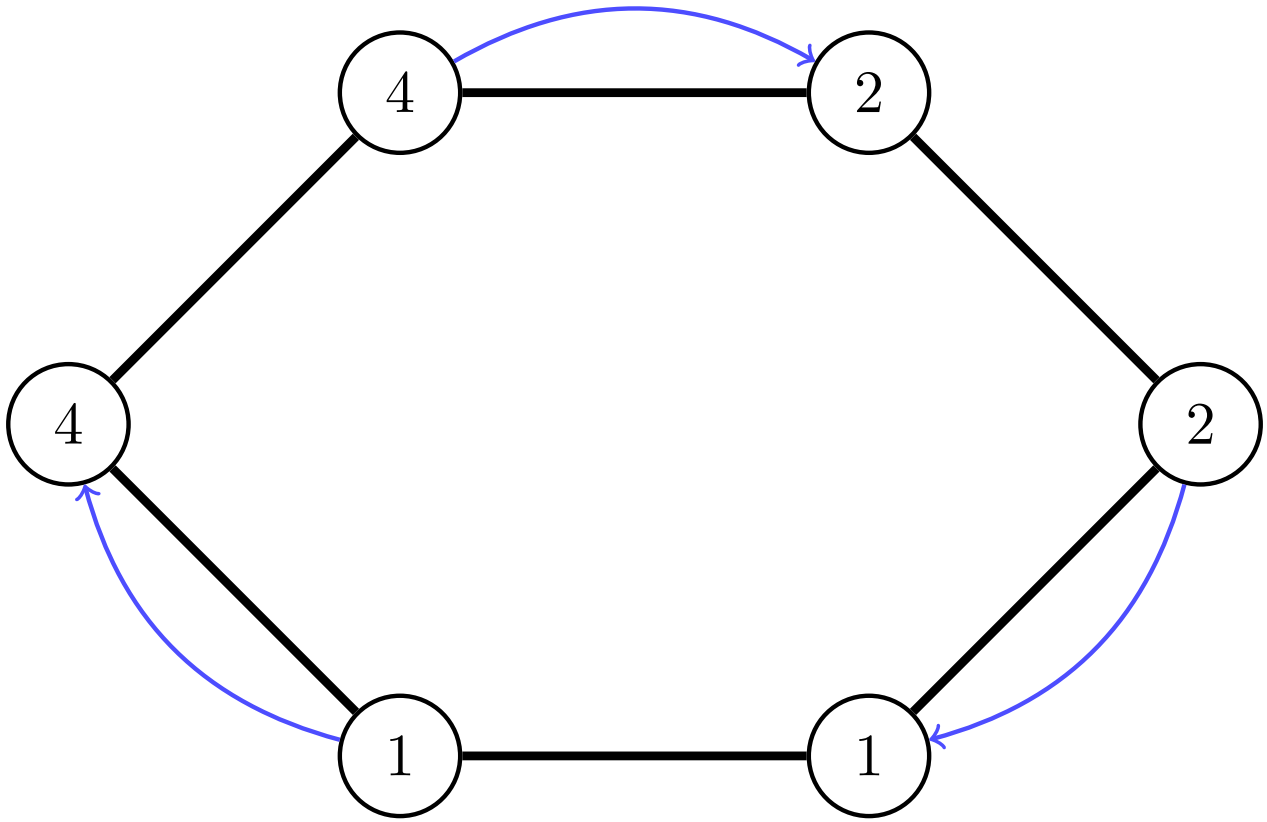
Example

Round 2:



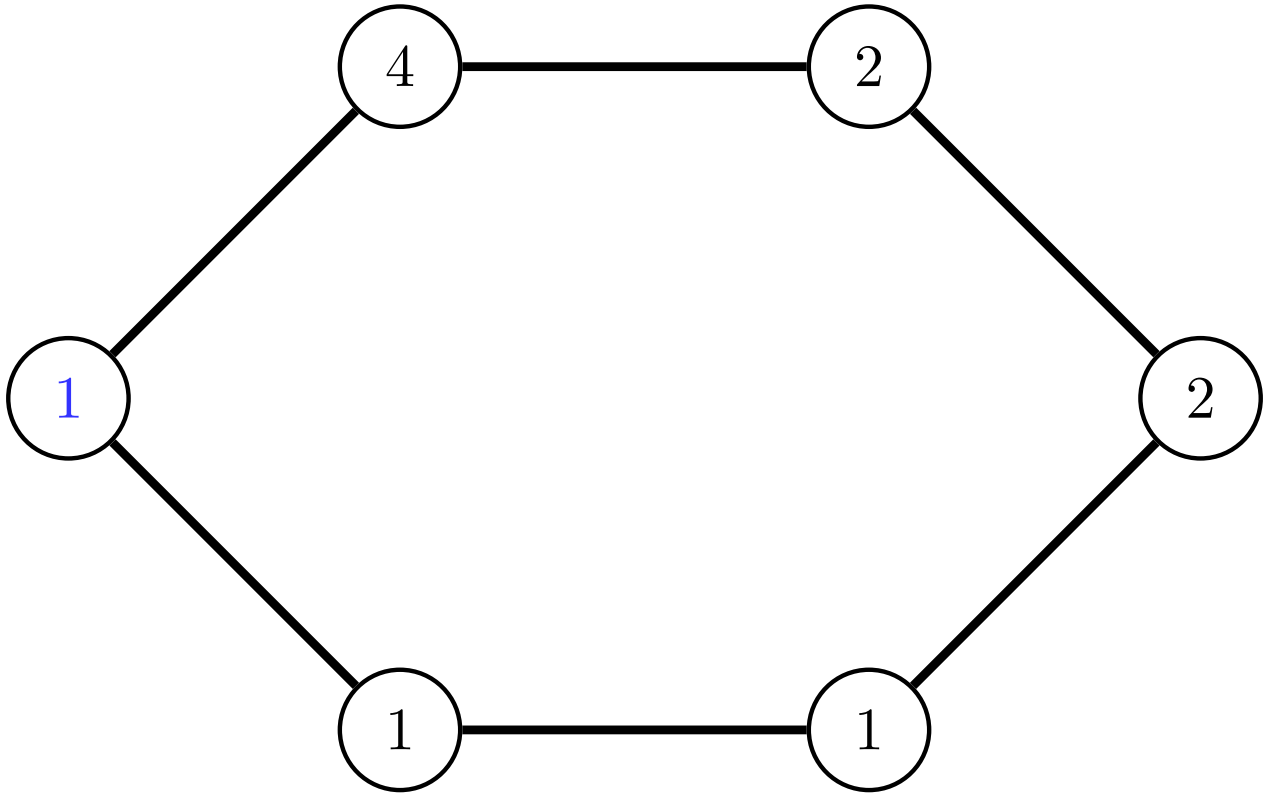
Example

Round 2:



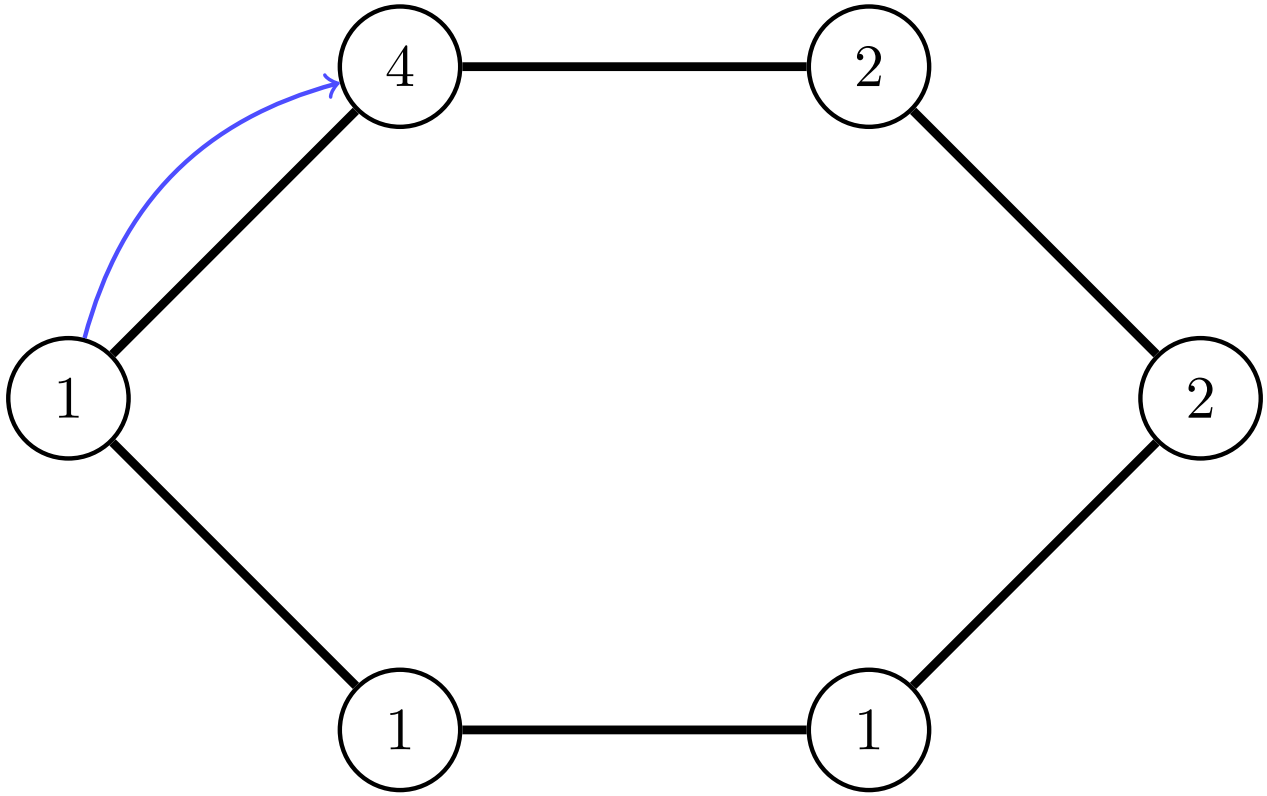
Example

Round 3:



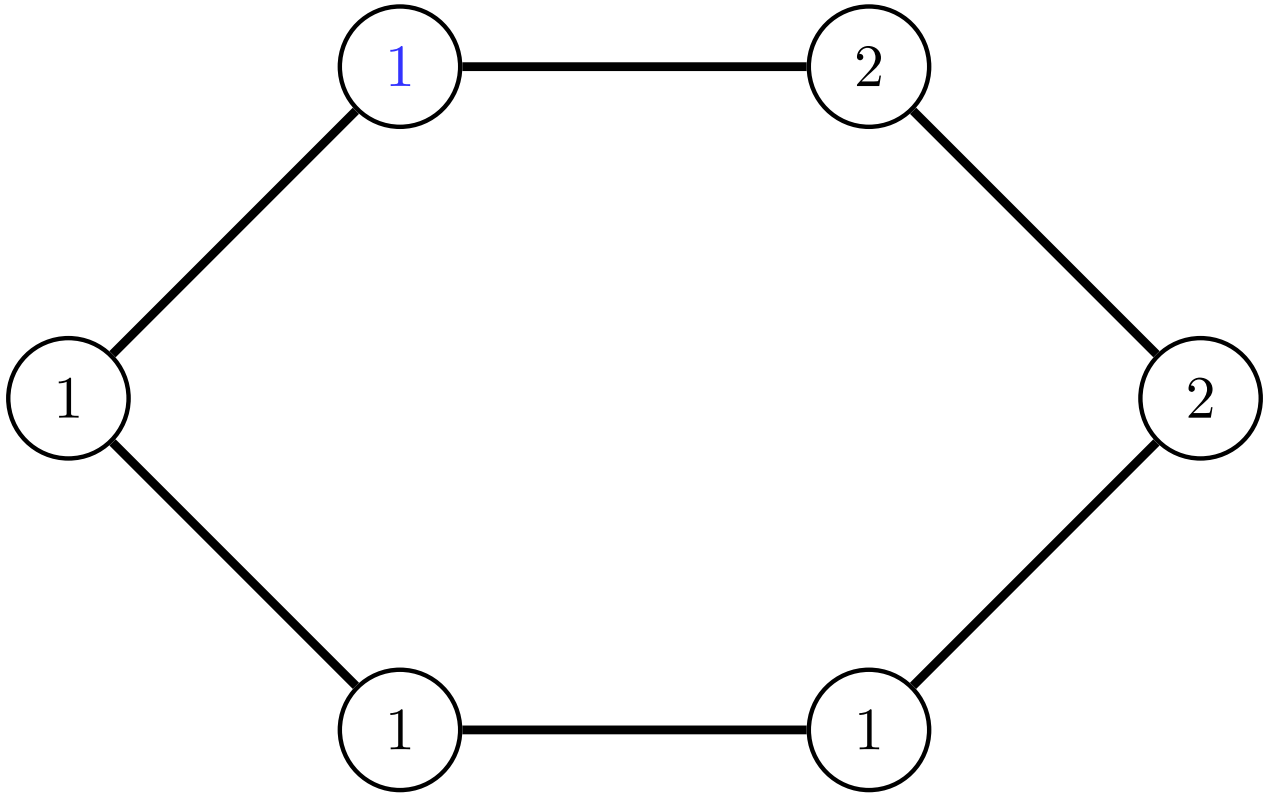
Example

Round 3:



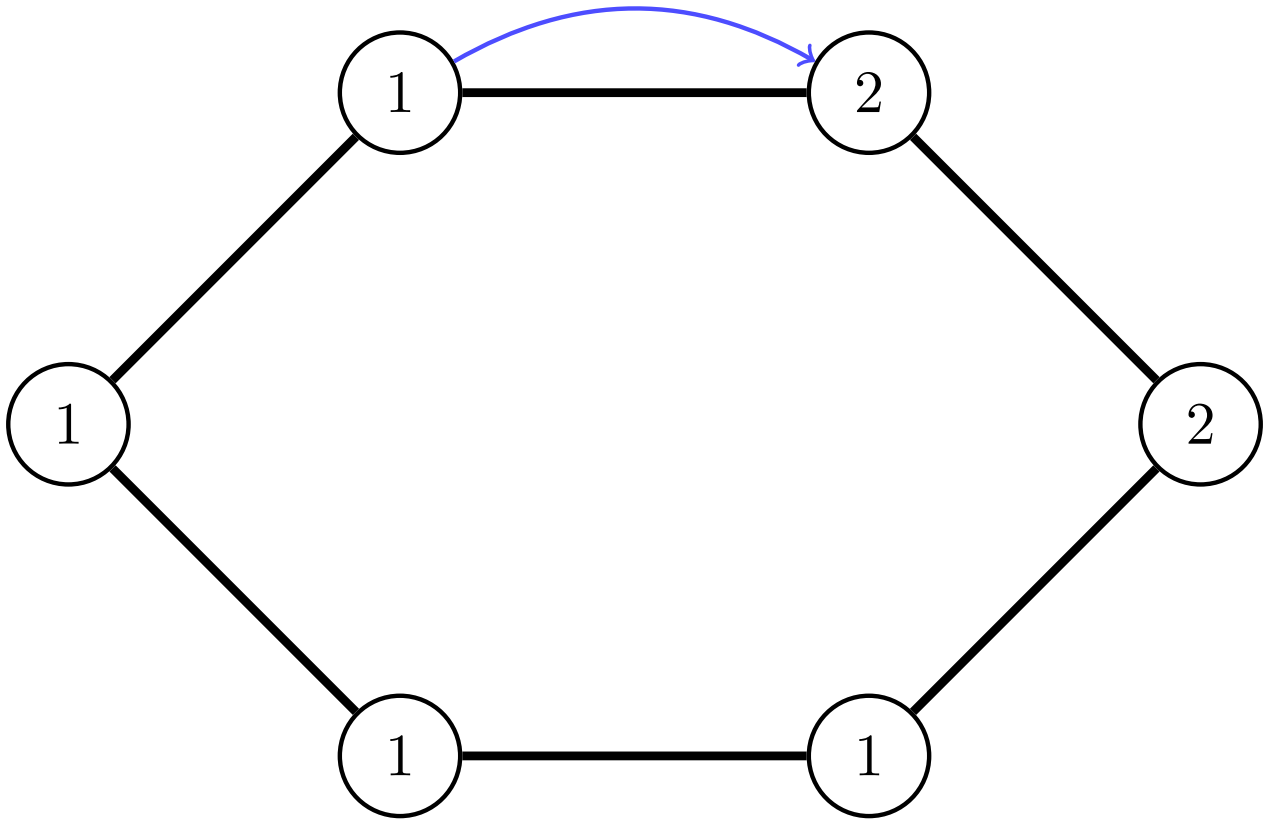
Example

Round 4:



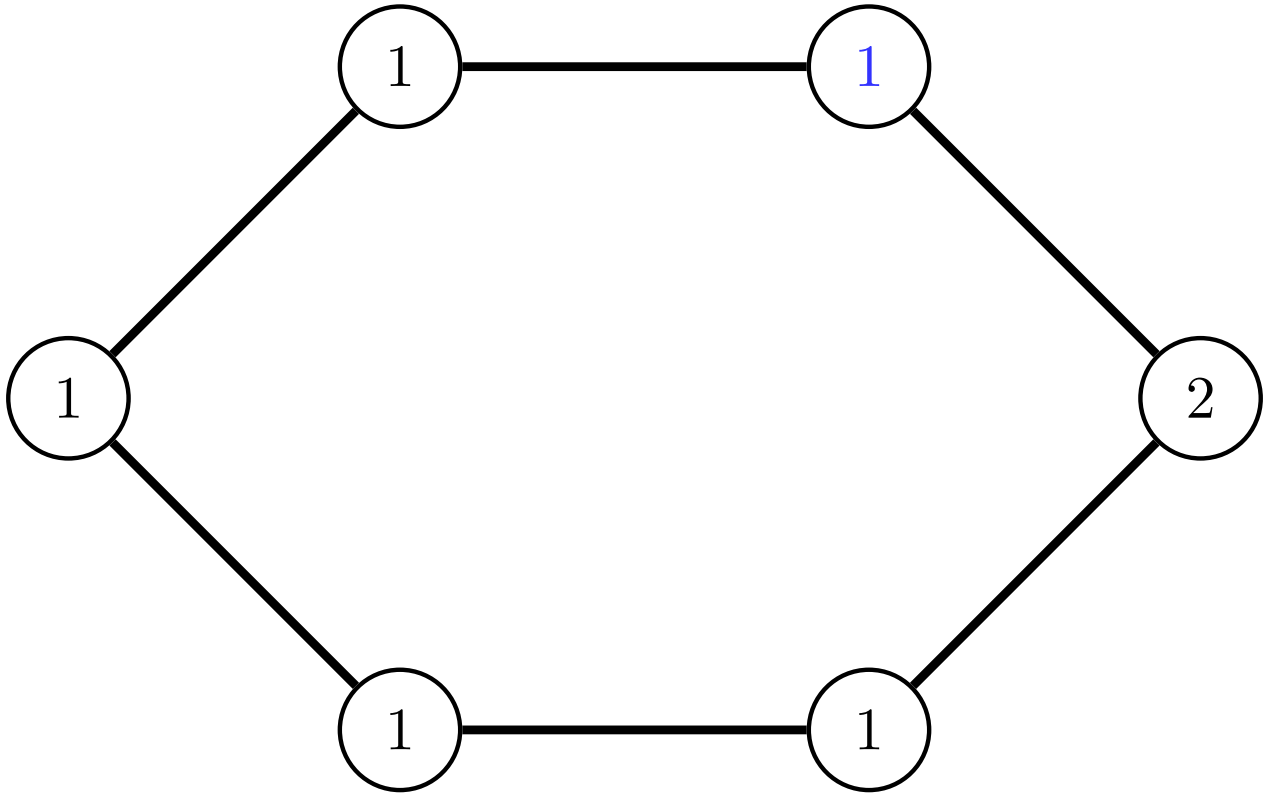
Example

Round 4:



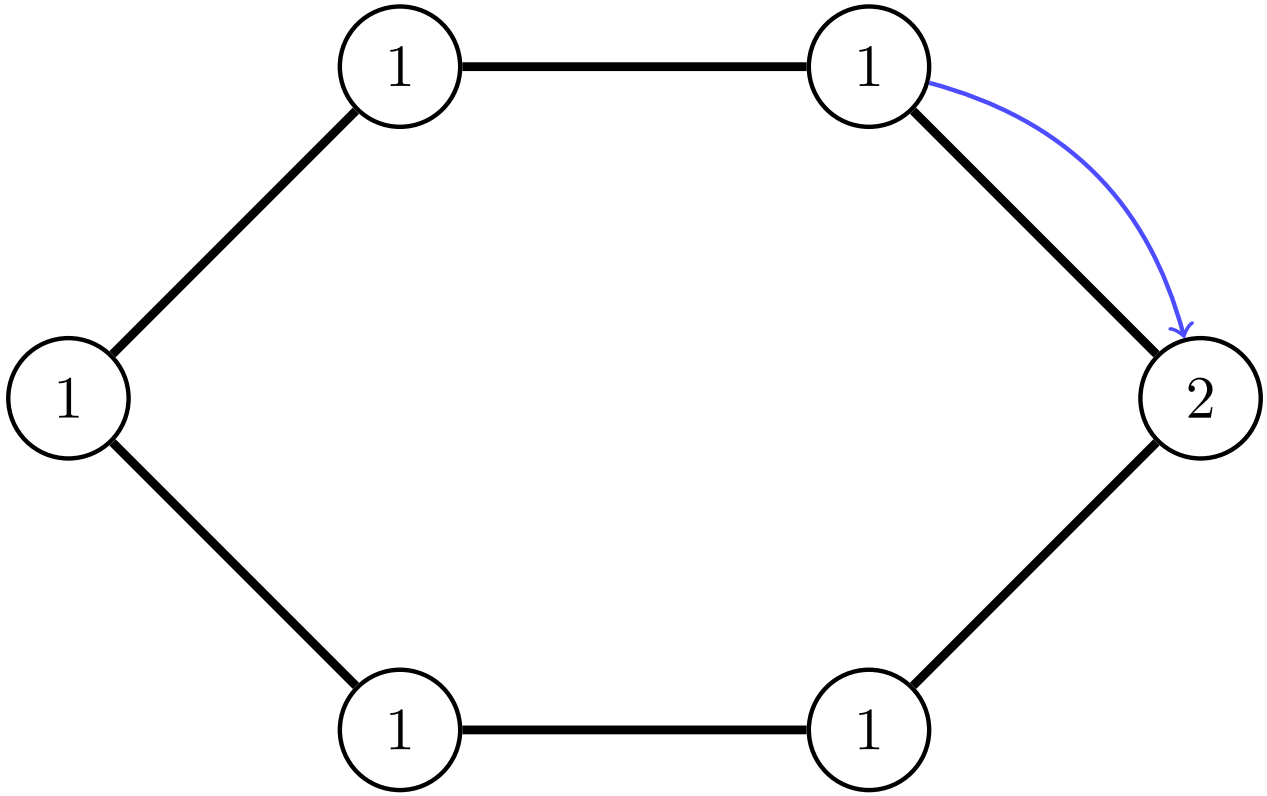
Example

Round 5:



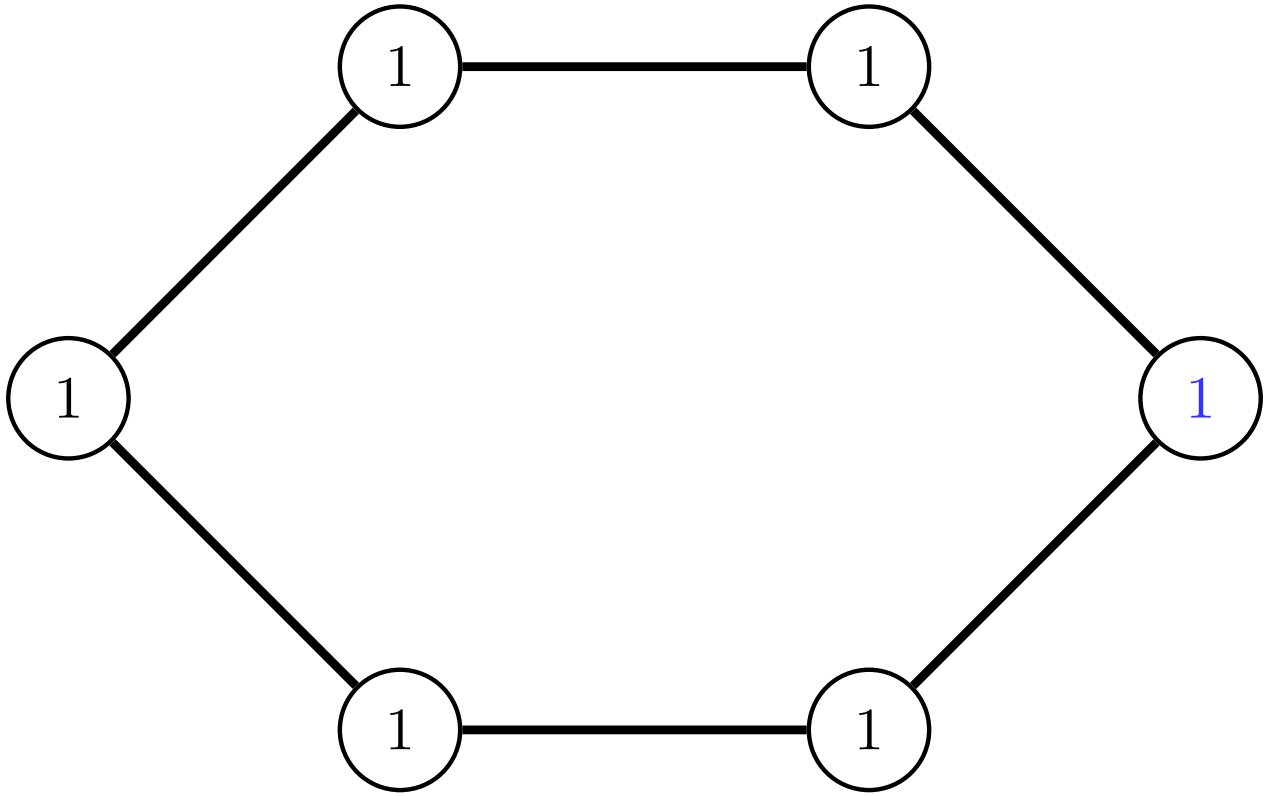
Example

Round 5:



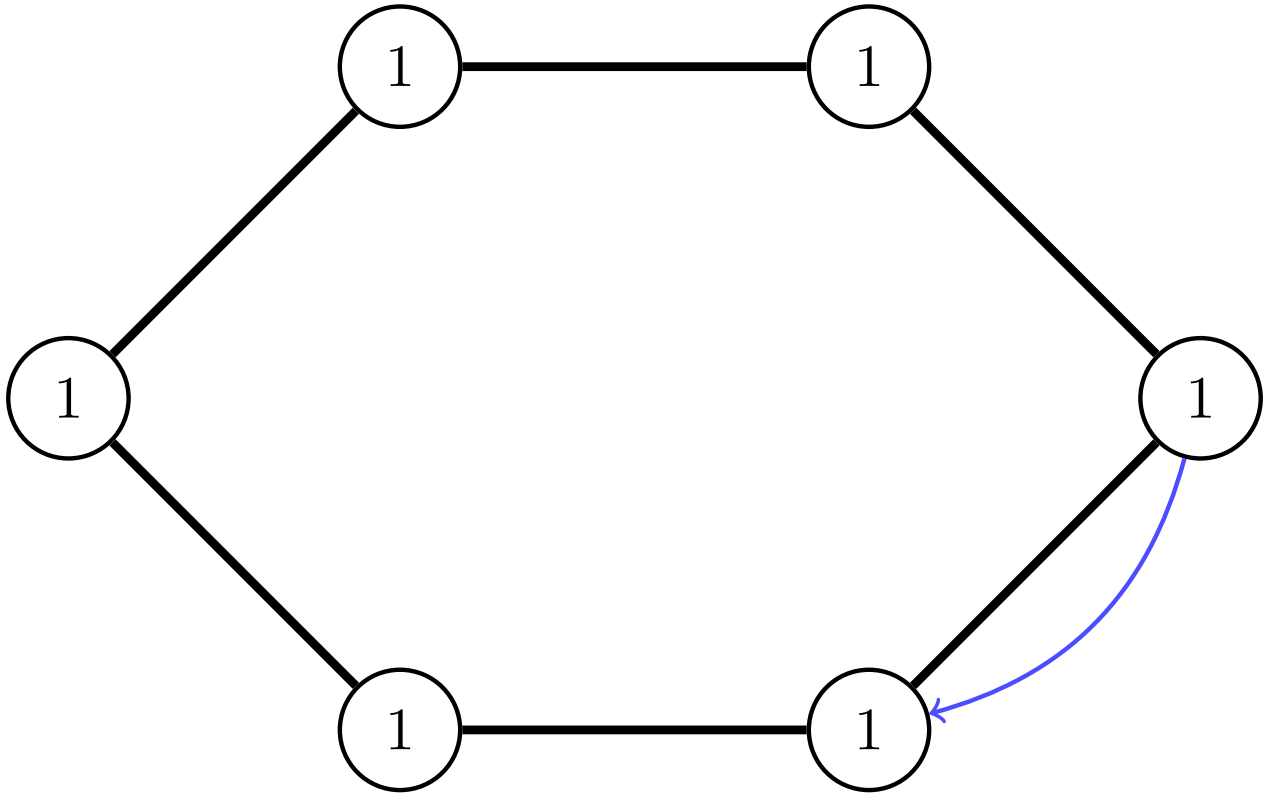
Example

Round 6:



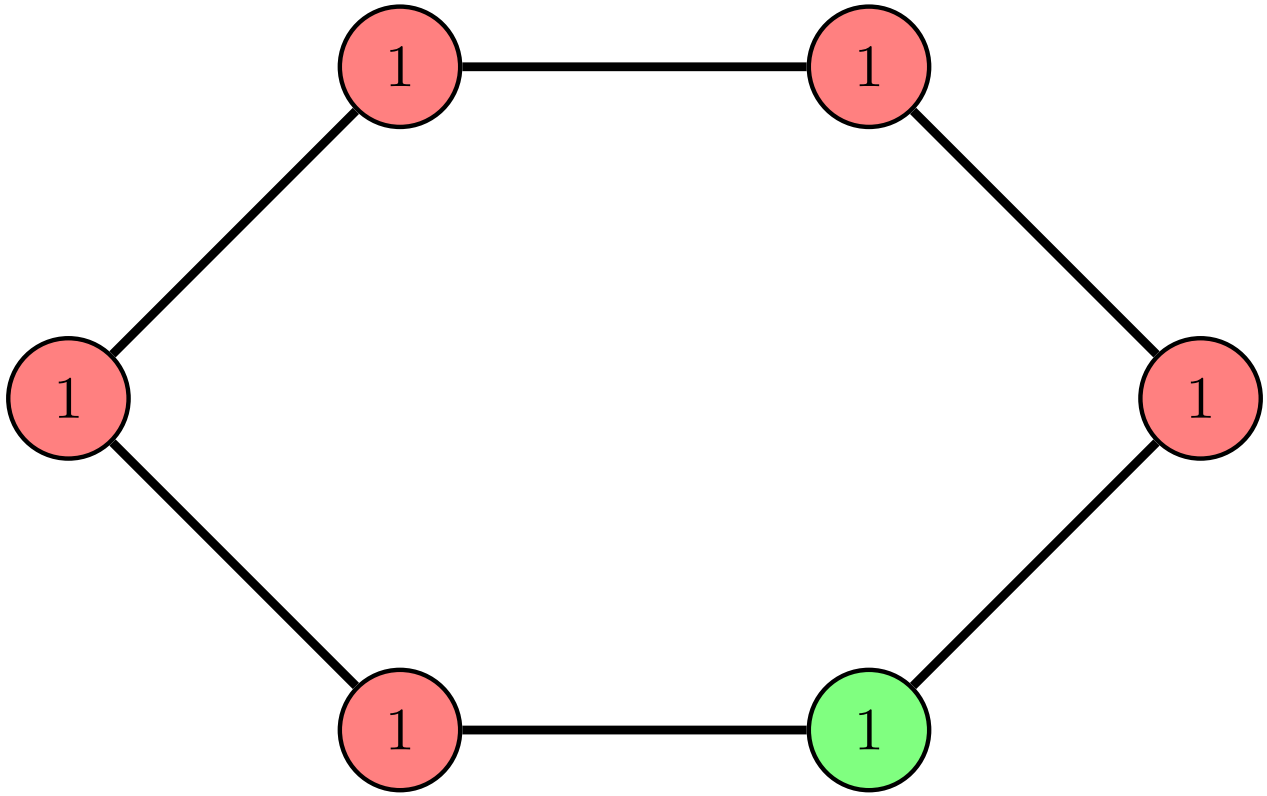
Example

Round 6:



Example

Round 7:



Runtime

Theorem

After $n + 1$ rounds, the simple clockwise algorithm selects the node with the smallest ID as leader and all other nodes as followers.

Observation: The smallest ID is forwarded in every round.

Runtime

Theorem

After $n + 1$ rounds, the simple clockwise algorithm selects the node with the smallest ID as leader and all other nodes as followers.

Observation: The smallest ID is forwarded in every round.

Proof:

- Let z be node with smallest ID; then $\mathrm{ID}(z) \leq \mathrm{ID}(v)$ for every node v.

Runtime

Theorem

After $n + 1$ rounds, the simple clockwise algorithm selects the node with the smallest ID as leader and all other nodes as followers.

Observation: The smallest ID is forwarded in every round.

Proof:

- Let z be node with smallest ID; then $\mathrm{ID}(z) \leq \mathrm{ID}(v)$ for every node v.
- For each $i \geq 0$, let v_i be the node reached when traversing i edges clockwise starting from z.

Runtime

Theorem

After $n + 1$ rounds, the simple clockwise algorithm selects the node with the smallest ID as leader and all other nodes as followers.

Observation: The smallest ID is forwarded in every round.

Proof:

- Let z be node with smallest ID; then $\mathrm{ID}(z) \leq \mathrm{ID}(v)$ for every node v.
- For each $i \geq 0$, let v_i be the node reached when traversing i edges clockwise starting from z.
- **Induction claim:** Node v_i sets L_{v_i} to $\mathrm{ID}(z)$ (at the latest) in round $i + 1$ (for every $0 \leq i \leq n$).

Runtime

Theorem

After $n + 1$ rounds, the simple clockwise algorithm selects the node with the smallest ID as leader and all other nodes as followers.

Observation: The smallest ID is forwarded in every round.

Proof:

- Let z be node with smallest ID; then $\mathrm{ID}(z) \leq \mathrm{ID}(v)$ for every node v.
- For each $i \geq 0$, let v_i be the node reached when traversing i edges clockwise starting from z.
- **Induction claim:** Node v_i sets L_{v_i} to $\mathrm{ID}(z)$ (at the latest) in round $i + 1$ (for every $0 \leq i \leq n$).
- **Consequence:**
  - Observation: Every value assigned to L_v corresponds to an ID in the network.
  - Let $v \neq z$. Then $v = v_i$ for some $i \leq n - 1$, and within $i + 1 \leq n$ rounds, v sets L_v to $\mathrm{ID}(z)$; from this round onward, $L_v = \mathrm{ID}(z)$.
  - At the end of round n, $L_v = \mathrm{ID}(z)$ holds for every node v.
  - Thus, in round $n + 1$, z becomes the leader and all other nodes become

Message Complexity

Theorem

The clockwise algorithm sends at most n^2 messages in total.

Message Complexity

Theorem

The clockwise algorithm sends at most n^2 messages in total.

Proof:

- Node v sends only after L_v has been initialized or decreased.

Message Complexity

Theorem

The clockwise algorithm sends at most n^2 messages in total.

Proof:

- Node v sends only after L_v has been initialized or decreased.
- Every value assigned to L_v corresponds to an ID in the network.

Message Complexity

Theorem

The clockwise algorithm sends at most n^2 messages in total.

Proof:

- Node v sends only after L_v has been initialized or decreased.
- Every value assigned to L_v corresponds to an ID in the network.
- Therefore, L_v can assume at most n different values during the algorithm.

Message Complexity

Theorem

The clockwise algorithm sends at most n^2 messages in total.

Proof:

- Node v sends only after L_v has been initialized or decreased.
- Every value assigned to L_v corresponds to an ID in the network.
- Therefore, L_v can assume at most n different values during the algorithm.
- Thus, every node sends at most n messages.

General Clockwise Algorithm [LeLann '77, Chang/Roberts '79]

Assumptions: asynchronous, identifiable, uniform.

General Clockwise Algorithm [LeLann '77, Chang/Roberts '79]

Assumptions: asynchronous, identifiable, uniform.

Two possible events: initialization or receipt of a message.

General Clockwise Algorithm [LeLann '77, Chang/Roberts '79]

Assumptions: asynchronous, identifiable, uniform.

Two possible events: initialization or receipt of a message.

Each node v executes the following algorithm:

1 **if** v *is initialized* **then**
2 $\quad$ v sets $L_v := \mathrm{ID}(v)$ (local variable for the smallest ID seen so far)
3 $\quad$ v sends L_v to its clockwise neighbor

General Clockwise Algorithm [LeLann '77, Chang/Roberts '79]

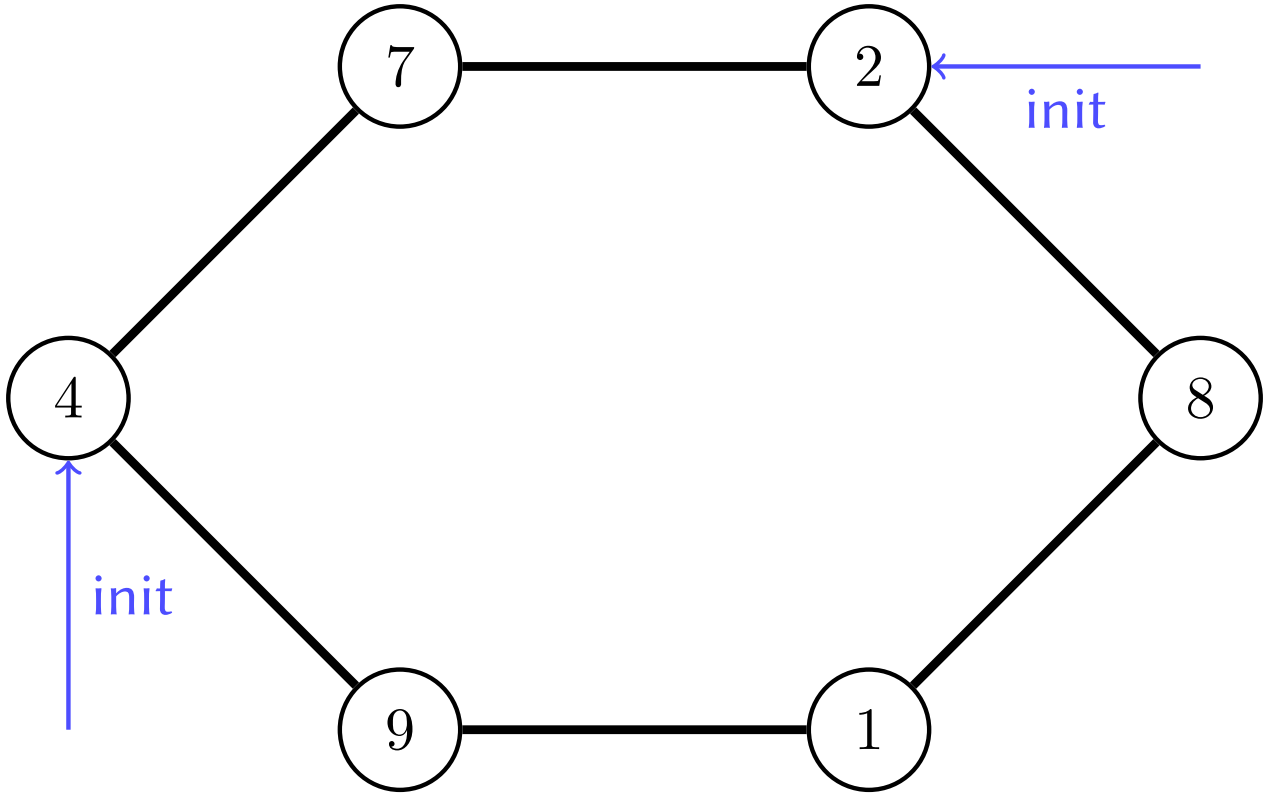
Assumptions: asynchronous, identifiable, uniform.

Two possible events: initialization or receipt of a message.

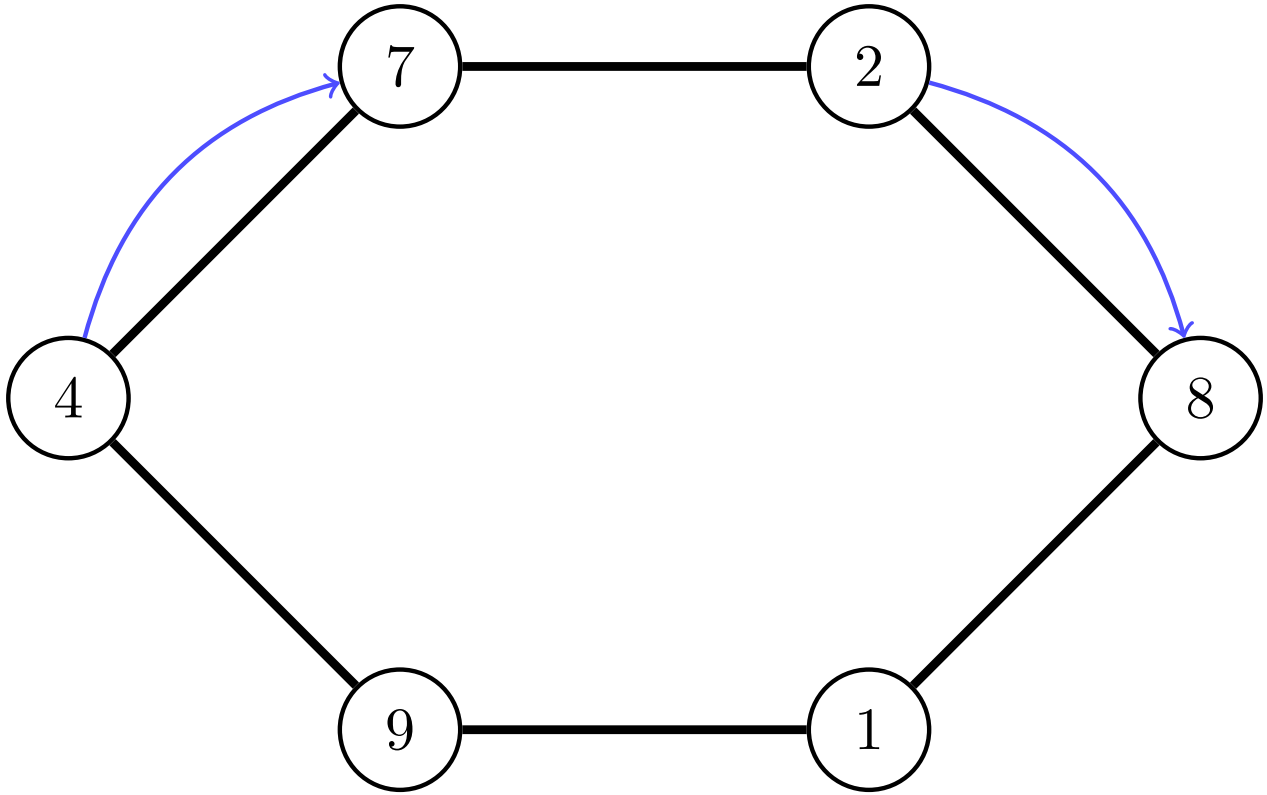
Each node v executes the following algorithm:

```
1  if v is initialized then
2      v sets L_v := ID(v) (local variable for the smallest ID seen so far)
3      v sends L_v to its clockwise neighbor
4  if v receives message M then
5      if v is uninitialized then initialize v
6      if M < L_v then
7          v sets L_v := M
8          v becomes a follower (if this has not already happened)
9          v sends L_v to its clockwise neighbor
10     if M = ID(v) then
11         v becomes the leader
```

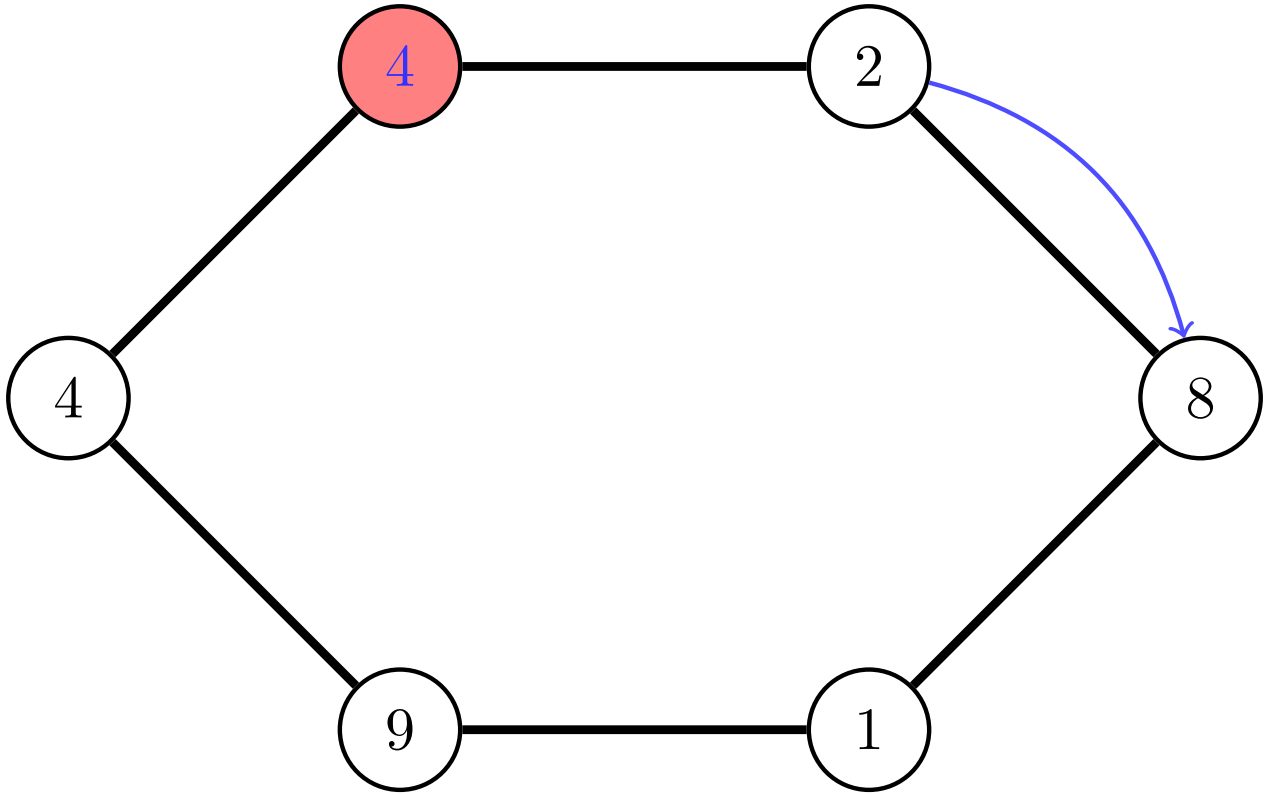
Example



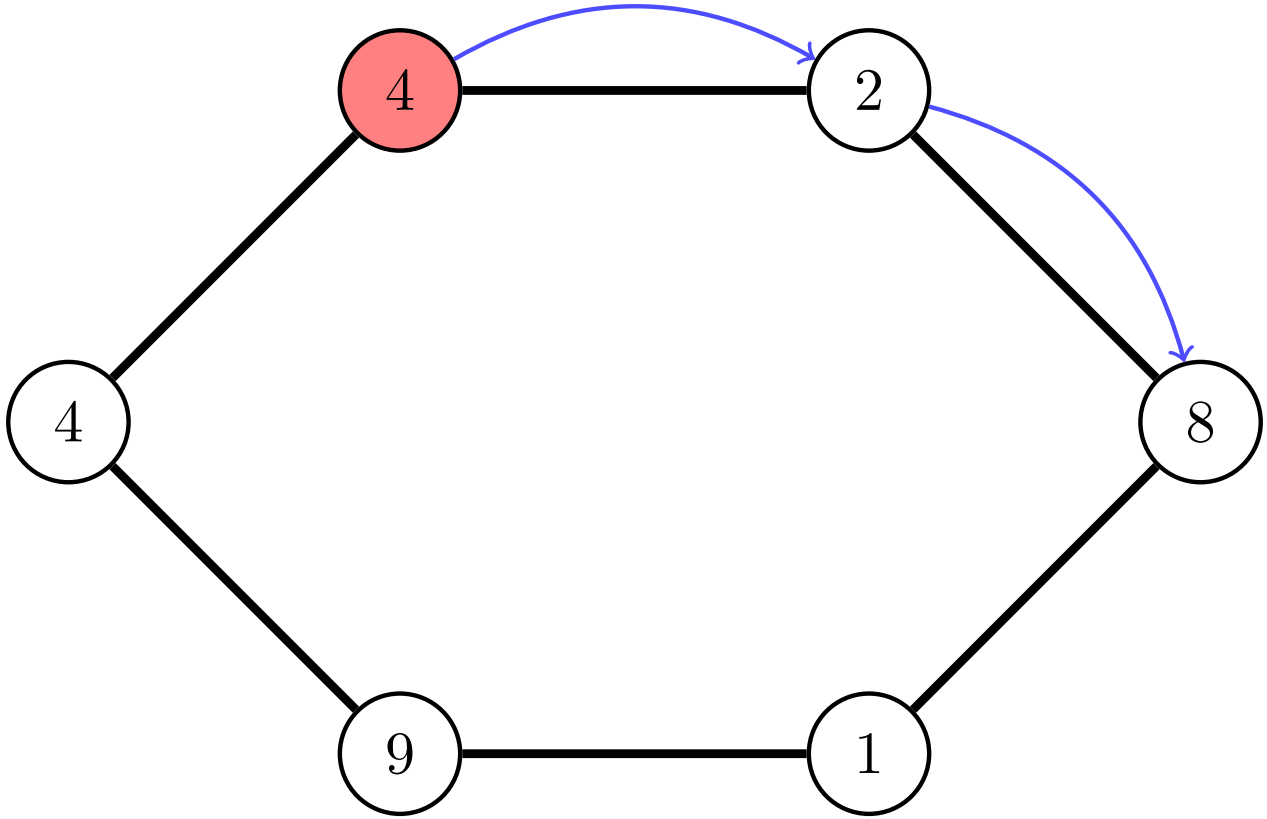
Example



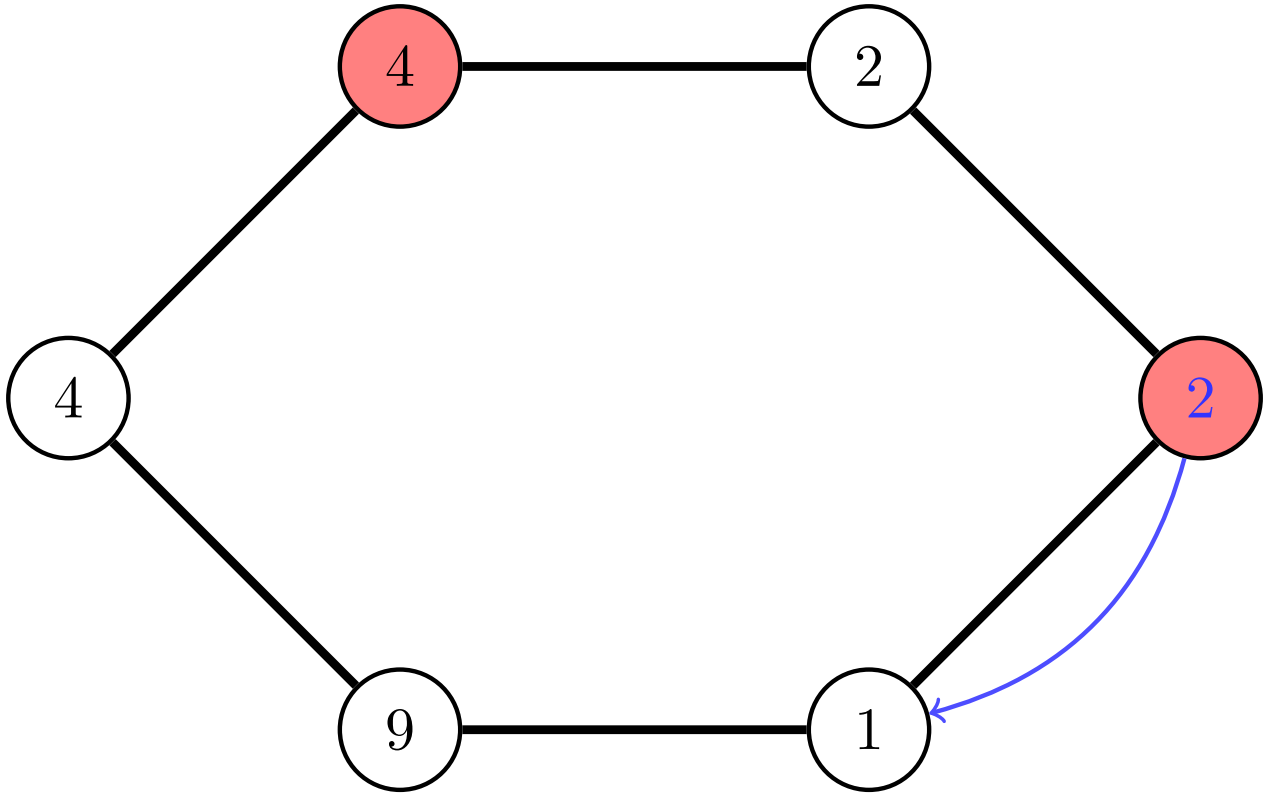
Example



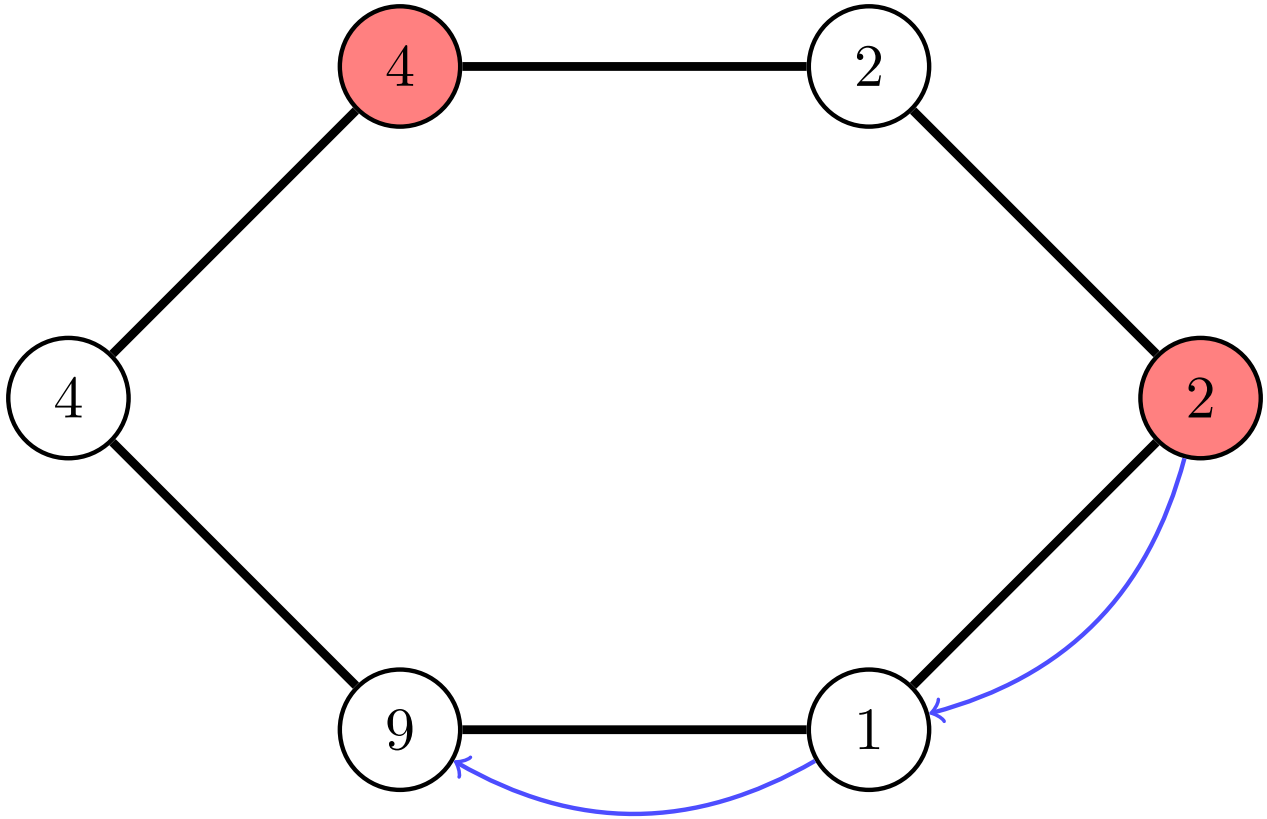
Example



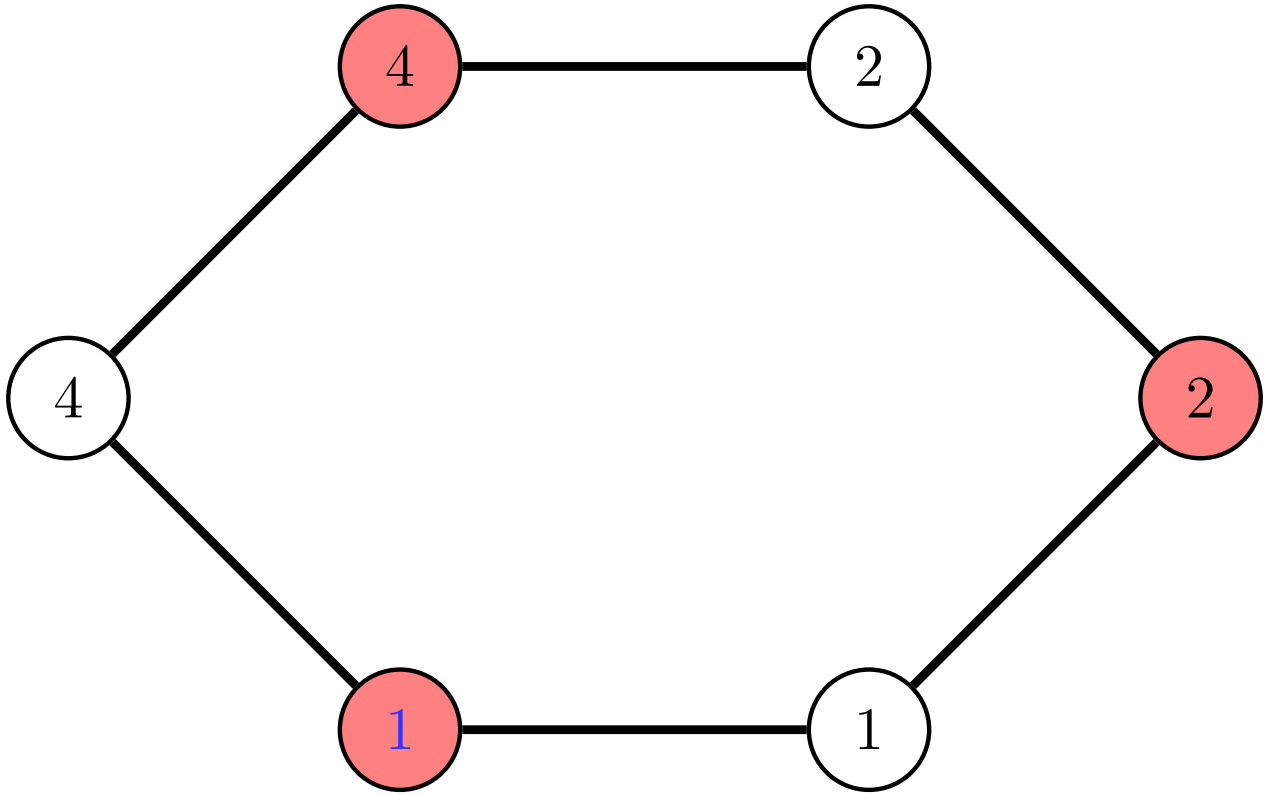
Example



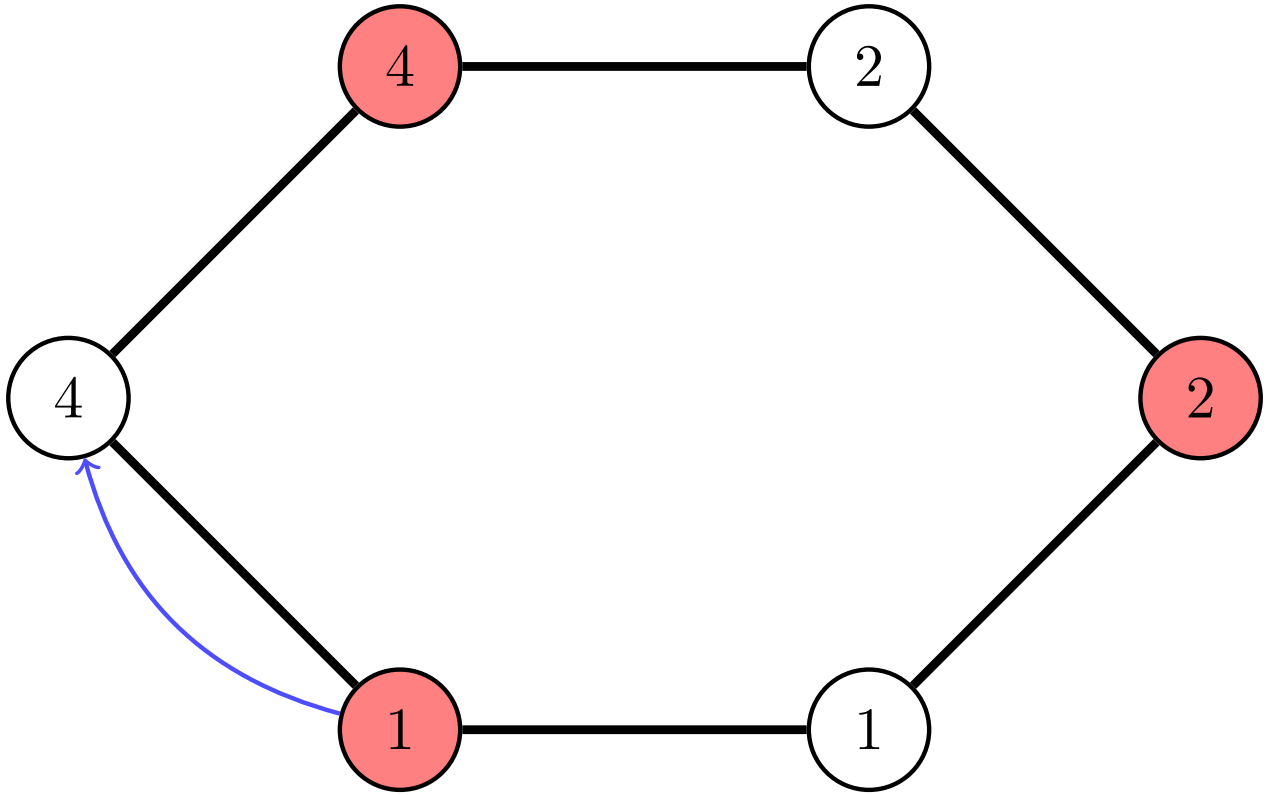
Example



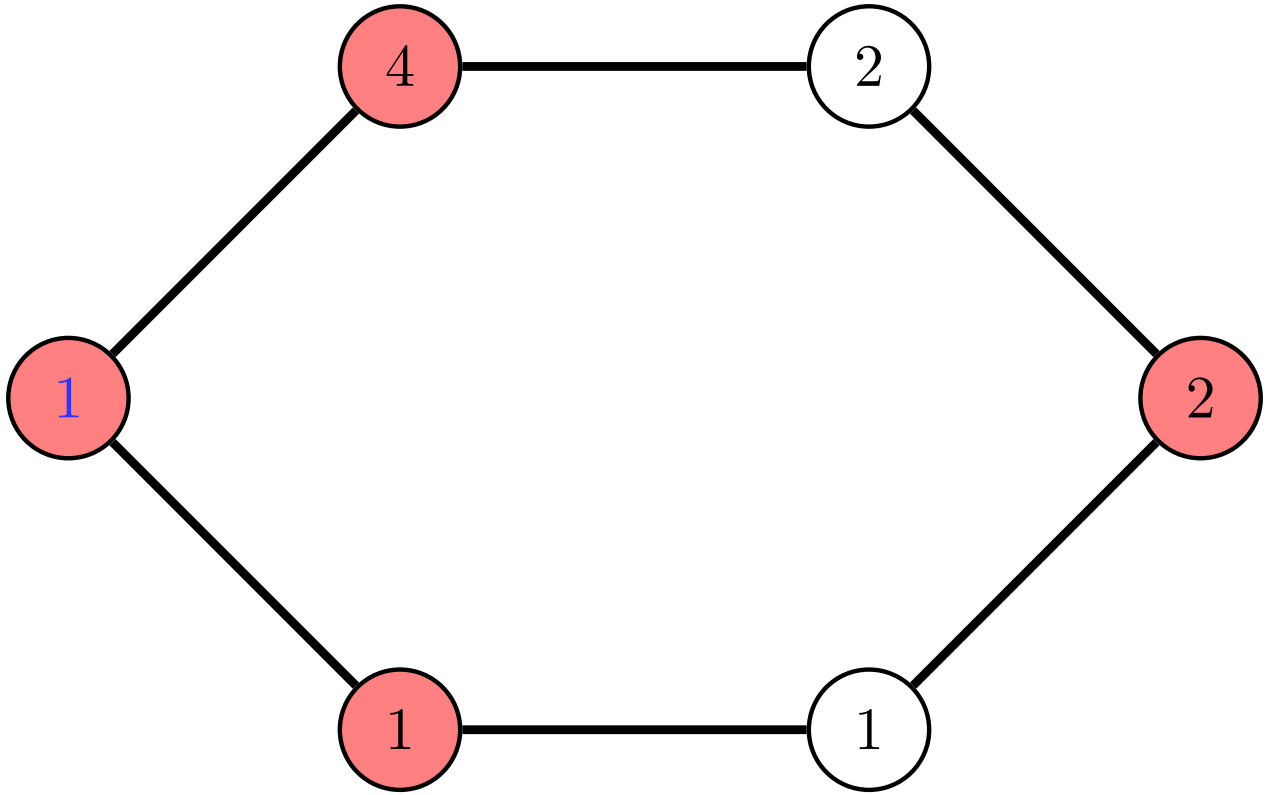
Example



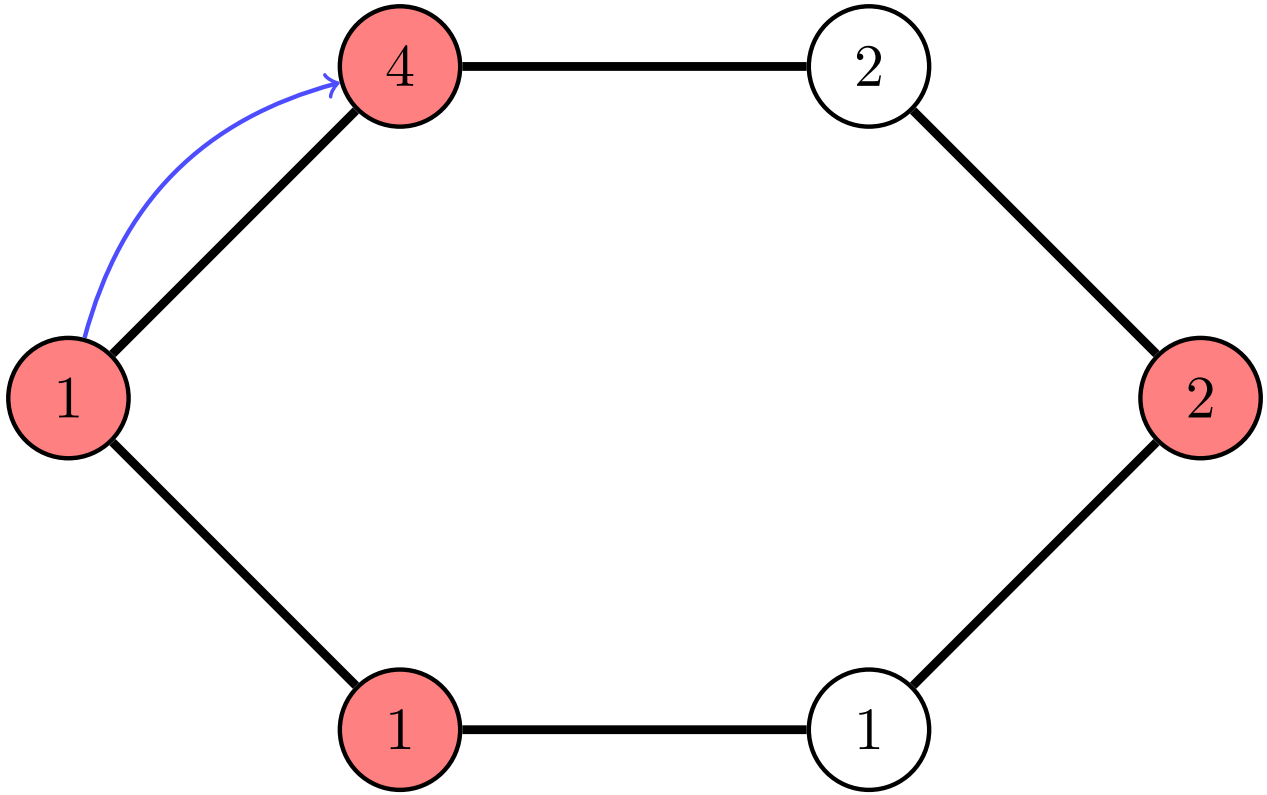
Example



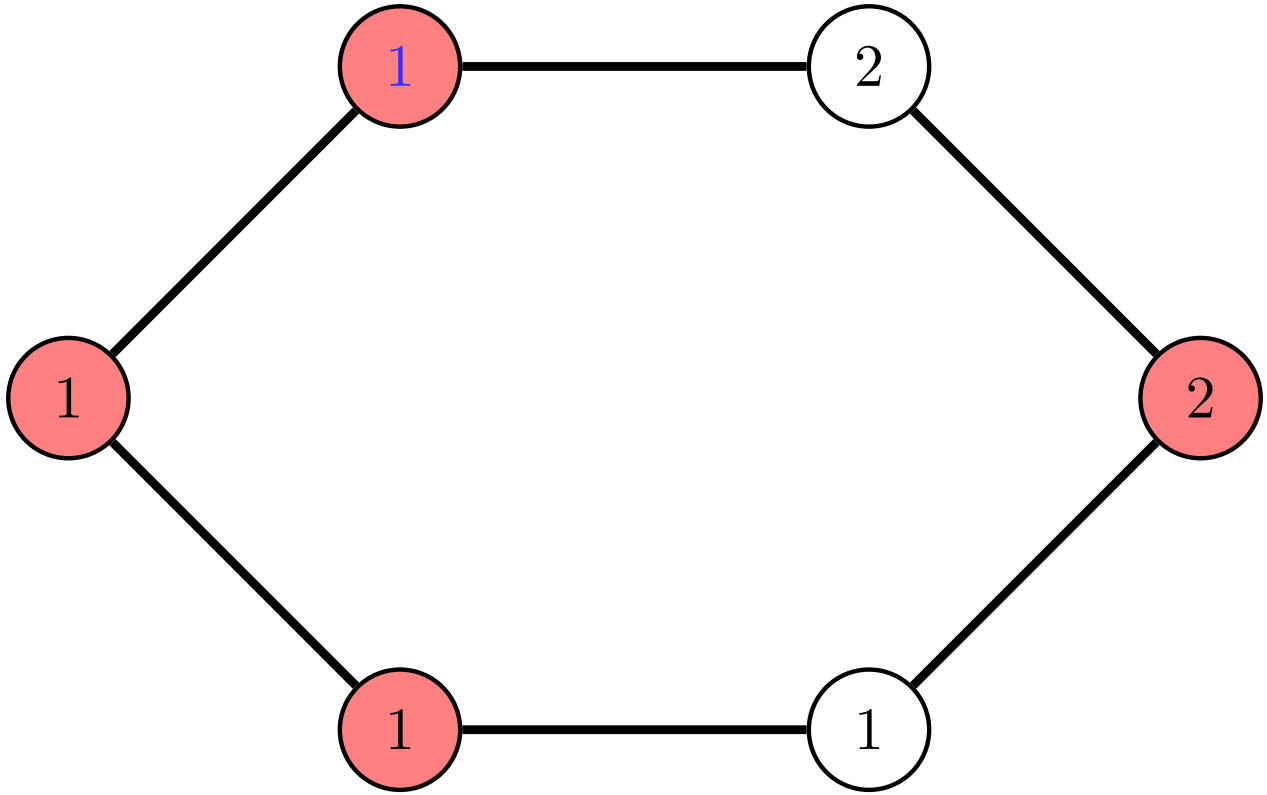
Example



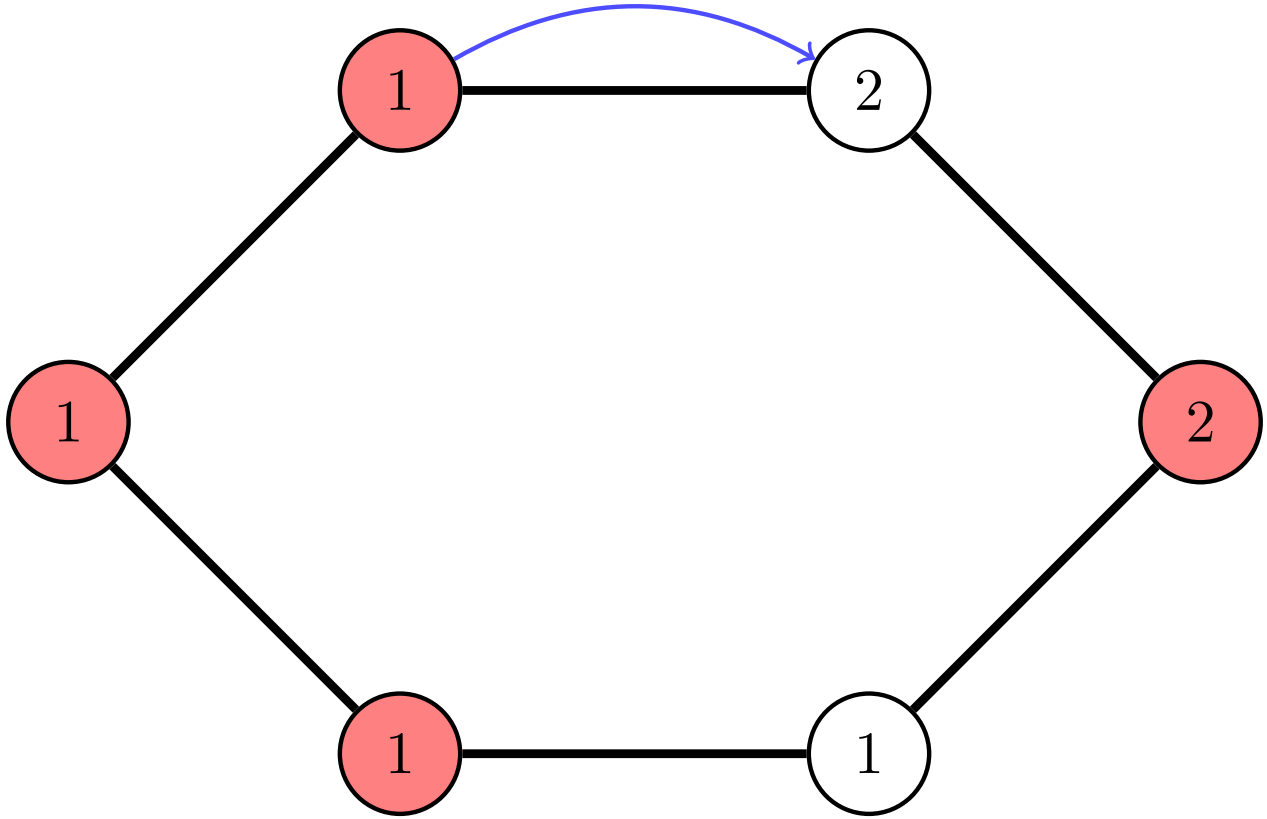
Example



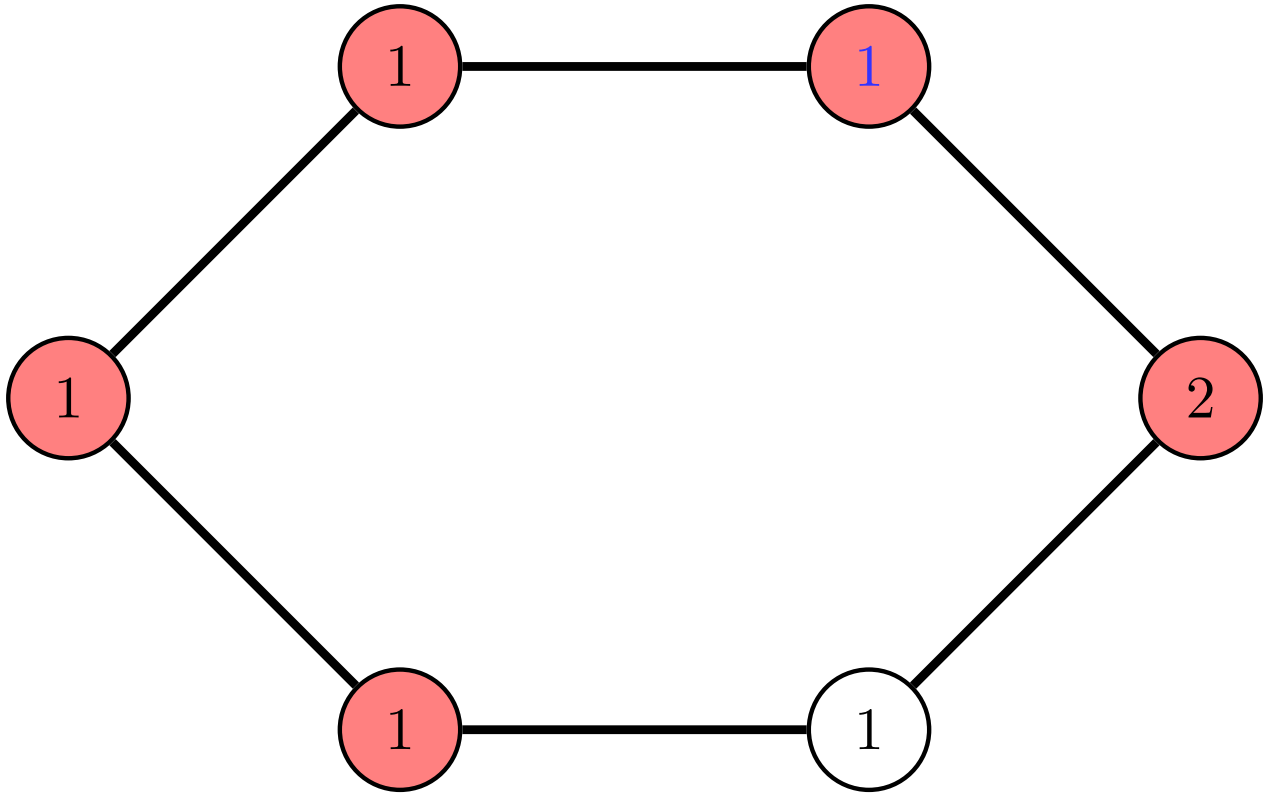
Example



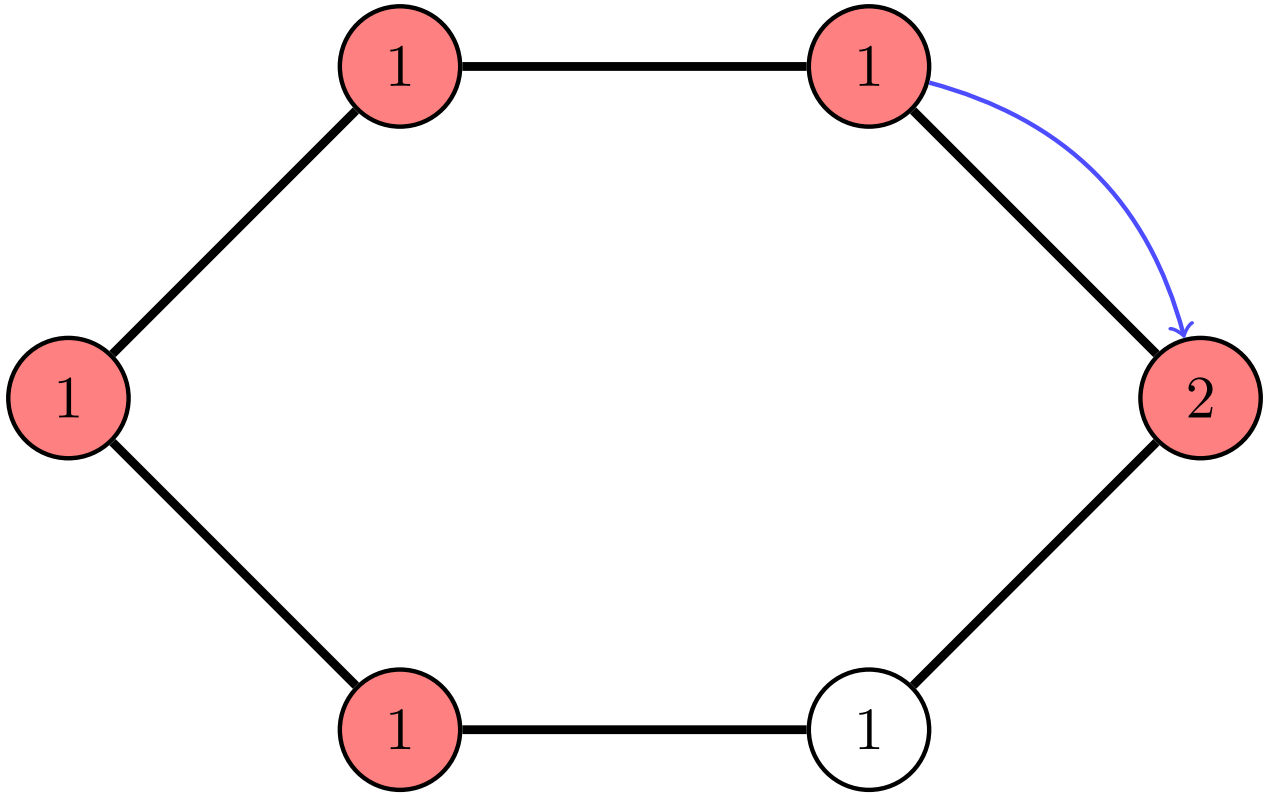
Example



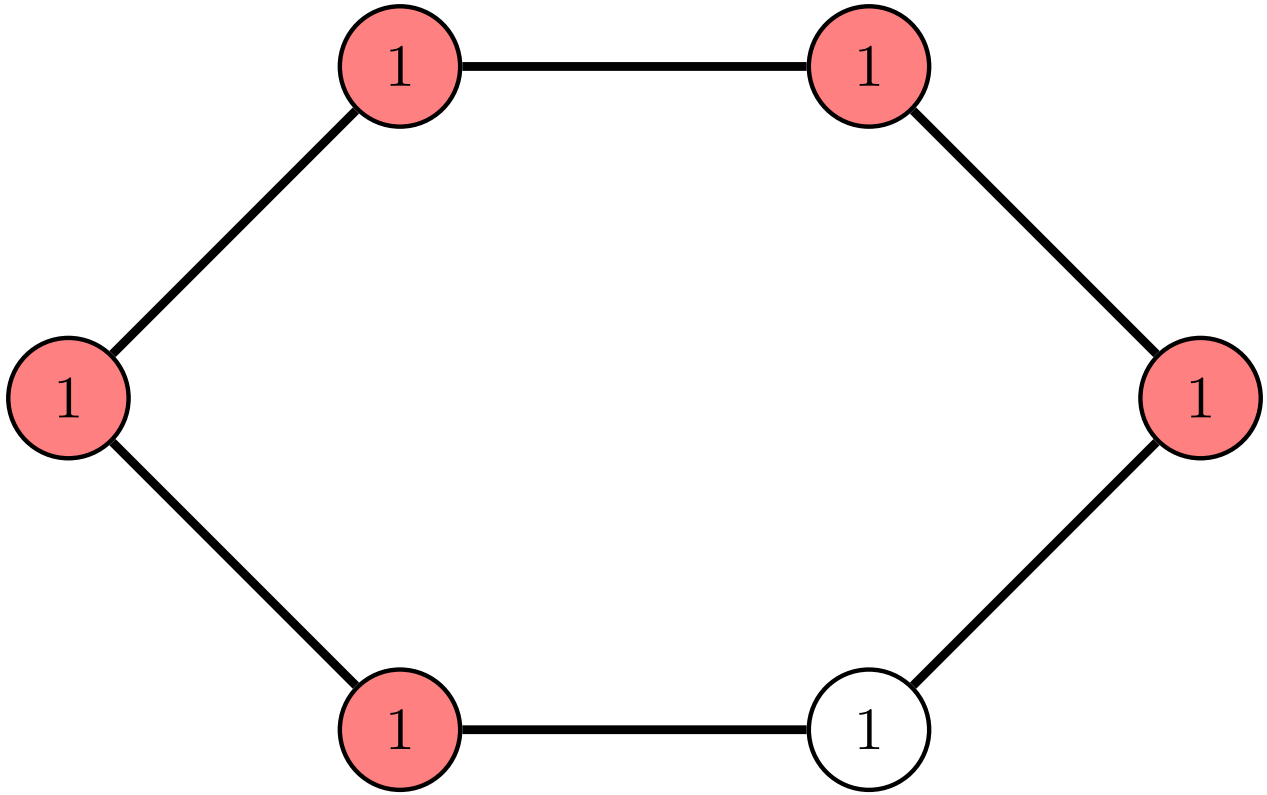
Example



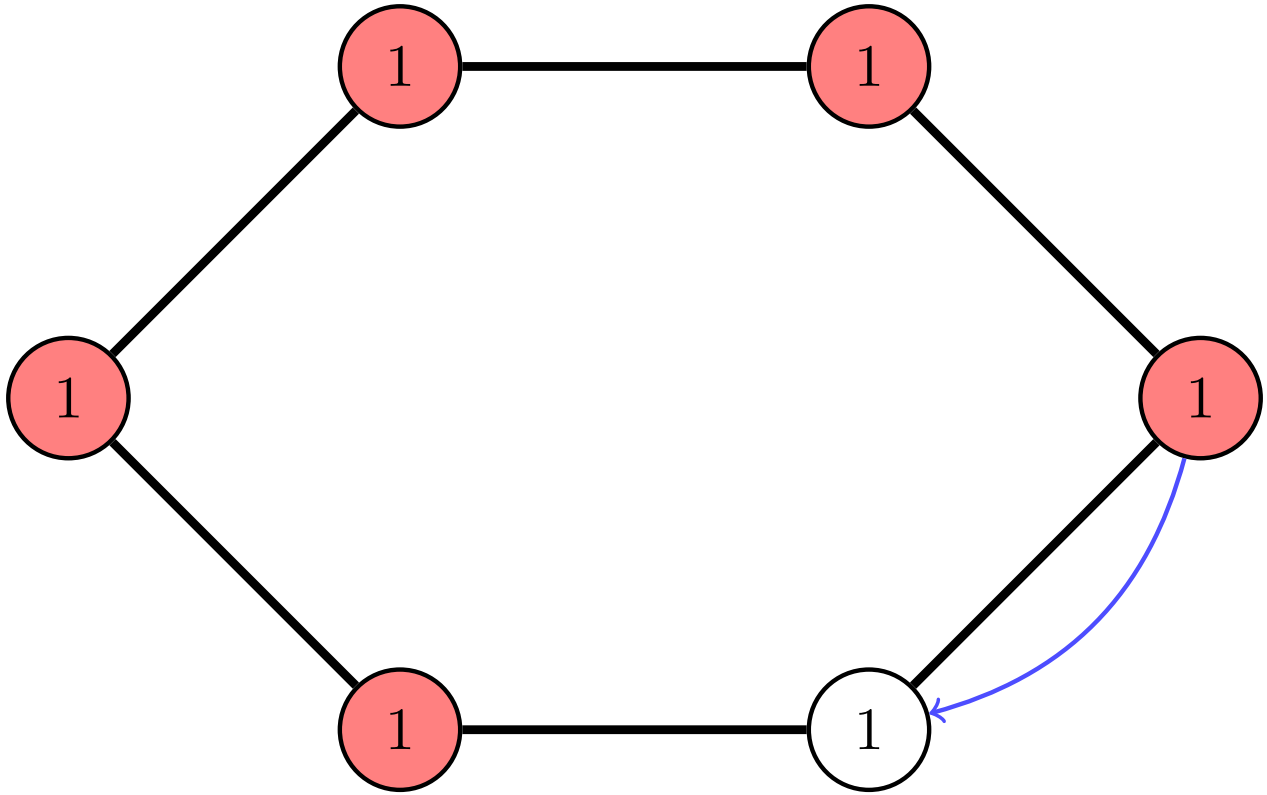
Example



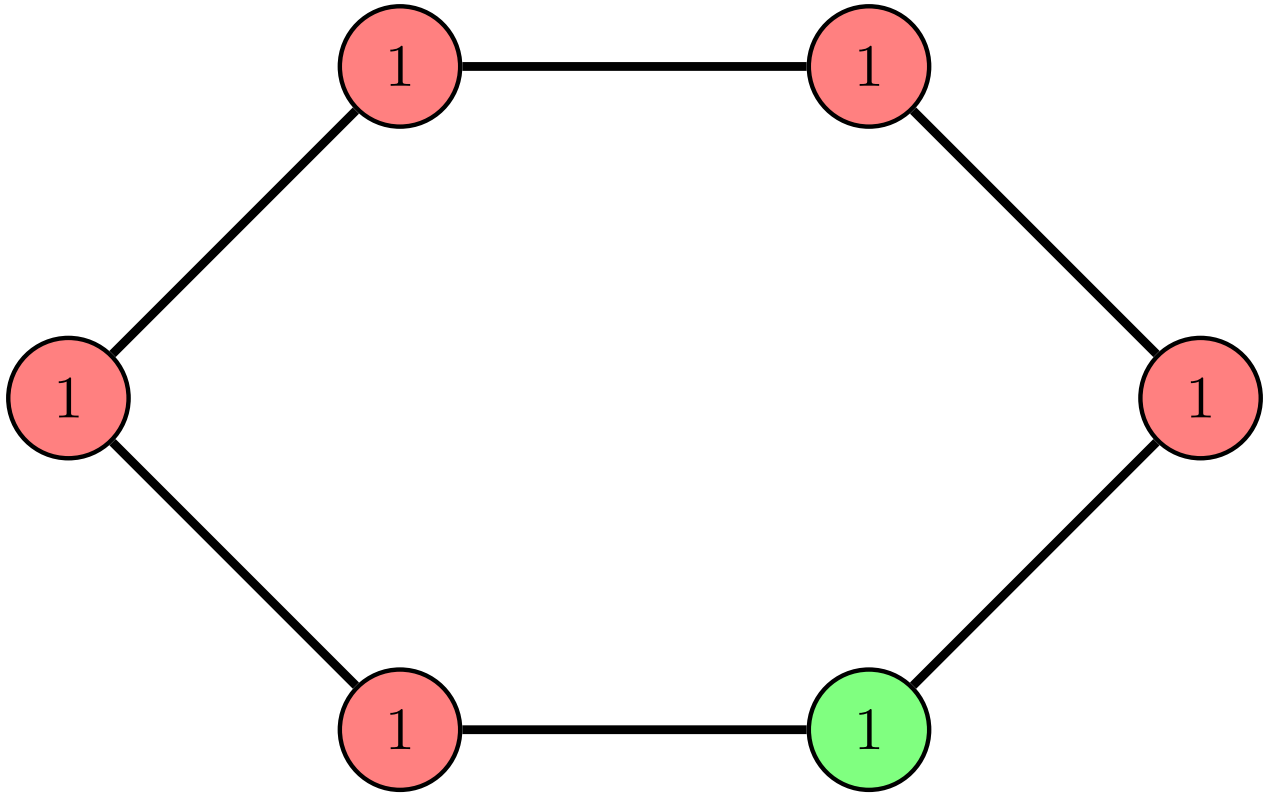
Example



Example



Example



Analysis

Theorem

After at most $2n - 1$ time units, the asynchronous clockwise algorithm selects the node with the smallest ID as leader and all other nodes as followers, provided that the transmission of each message takes at most one time unit.

Analysis

Theorem

After at most $2n - 1$ time units, the asynchronous clockwise algorithm selects the node with the smallest ID as leader and all other nodes as followers, provided that the transmission of each message takes at most one time unit.

Proof idea:

- Time measurement starts with the first node to be initialized.

Analysis

Theorem

After at most $2n - 1$ time units, the asynchronous clockwise algorithm selects the node with the smallest ID as leader and all other nodes as followers, provided that the transmission of each message takes at most one time unit.

Proof idea:

- Time measurement starts with the first node to be initialized.
- Then at most $n - 1$ time units elapse before node z with the smallest ID receives a message and becomes active for the first time.

Analysis

Theorem

After at most $2n - 1$ time units, the asynchronous clockwise algorithm selects the node with the smallest ID as leader and all other nodes as followers, provided that the transmission of each message takes at most one time unit.

Proof idea:

- Time measurement starts with the first node to be initialized.
- Then at most $n - 1$ time units elapse before node z with the smallest ID receives a message and becomes active for the first time.
- Subsequently: After at most n additional time units, every node has received $\mathrm{ID}(z)$ and made its decision.

Analysis

Theorem

After at most $2n-1$ time units, the asynchronous clockwise algorithm selects the node with the smallest ID as leader and all other nodes as followers, provided that the transmission of each message takes at most one time unit.

Proof idea:

- Time measurement starts with the first node to be initialized.
- Then at most $n-1$ time units elapse before node z with the smallest ID receives a message and becomes active for the first time.
- Subsequently: After at most n additional time units, every node has received $\mathrm{ID}(z)$ and made its decision.

Message complexity: $O(n^2)$
The same argument as for the synchronous algorithm applies.

Optimal Message Complexity

Question

Can a leader be selected with significantly fewer than $\Theta(n^2)$ messages?

Optimal Message Complexity

Question

Can a leader be selected with significantly fewer than $\Theta(n^2)$ messages?

"Act without action."

(Laozi)



Optimal Message Complexity

Question

Can a leader be selected with significantly fewer than $\Theta(n^2)$ messages?

"Act without action."

(Laozi)

Idea: Sending *no* message must be interpretable as consent.



Waiting-Time Algorithm [Attiya/Welch '98]

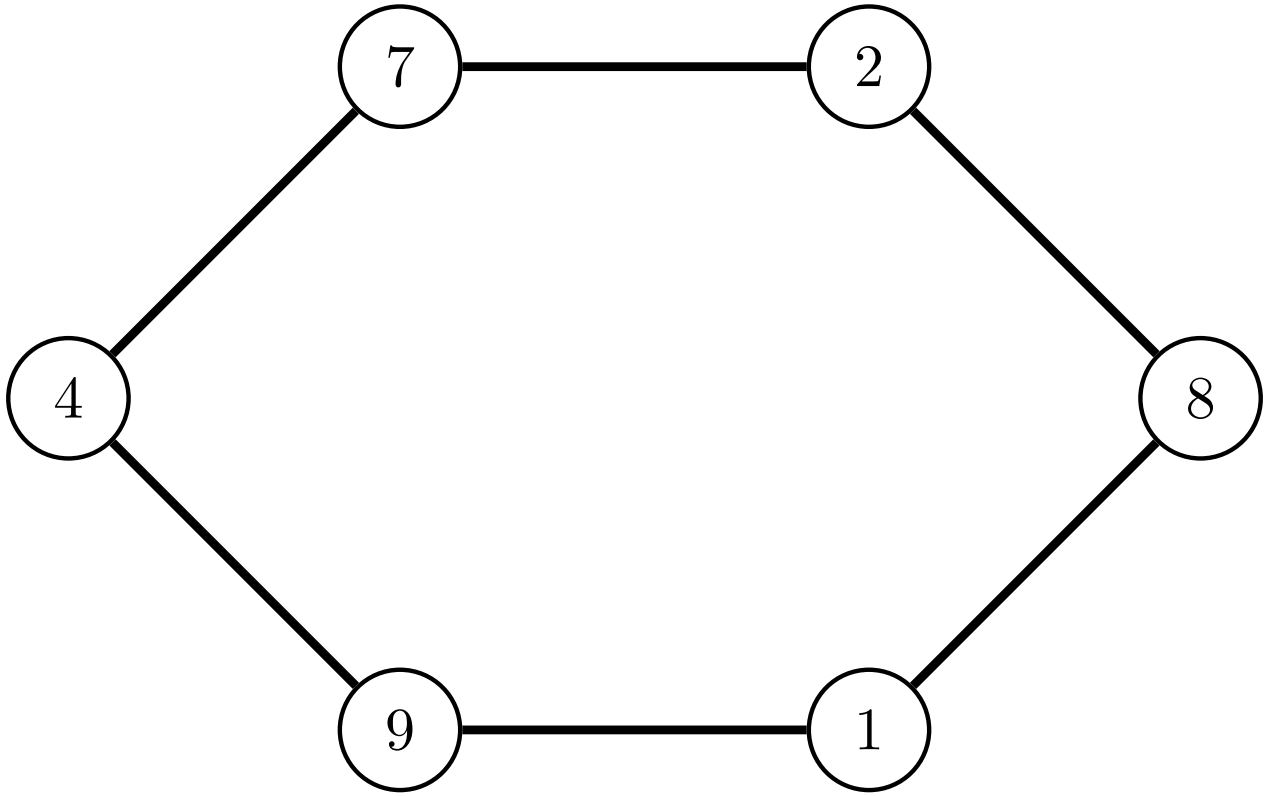
Assumptions: synchronous, identifiable, non-uniform.

Each node v executes the following algorithm in every round:

1 **if** *a leader announcement is received and no leader has been selected yet* **then**
2 Become a follower
3 Forward the leader announcement clockwise
4 **if** $\#rounds = \mathrm{ID}(v) * n + 1$ *and* v *is not yet a follower* **then**
5 Become the leader
6 Send a leader announcement to the clockwise neighbor

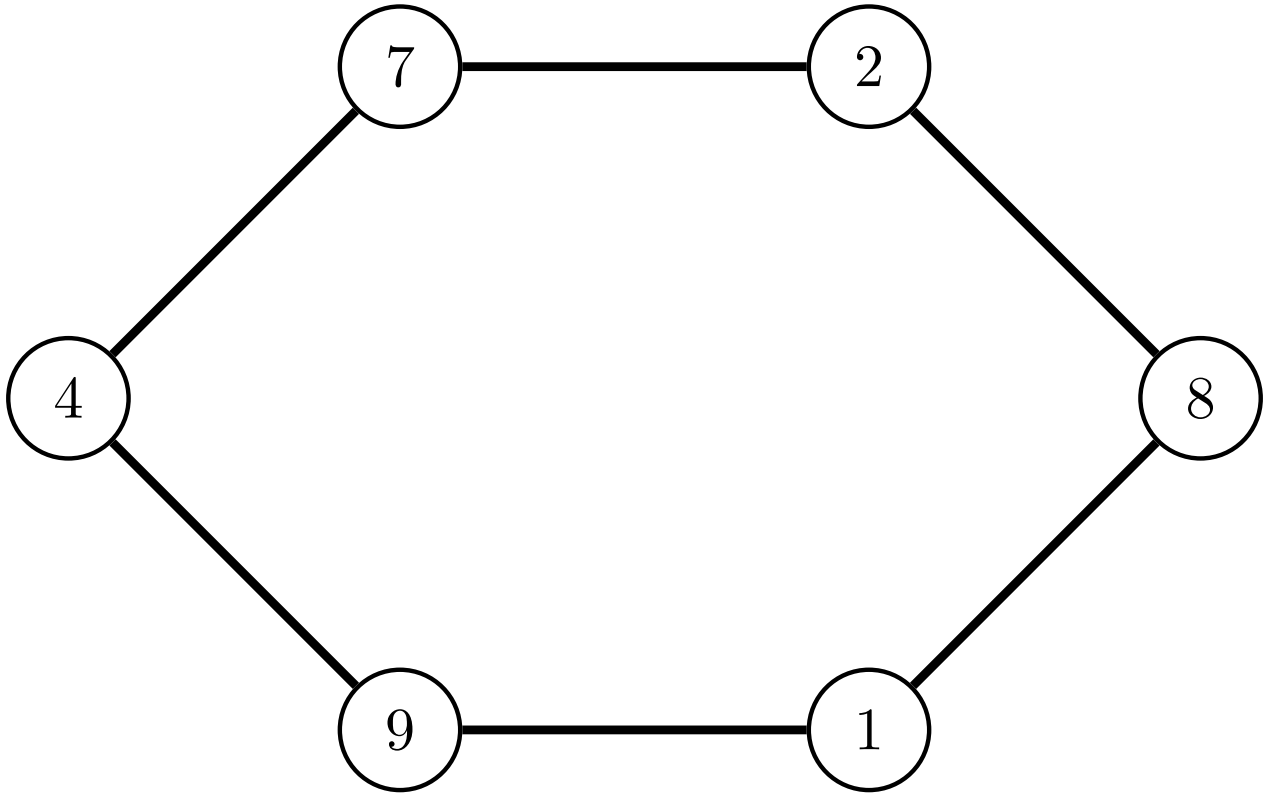
Example

Round 1:



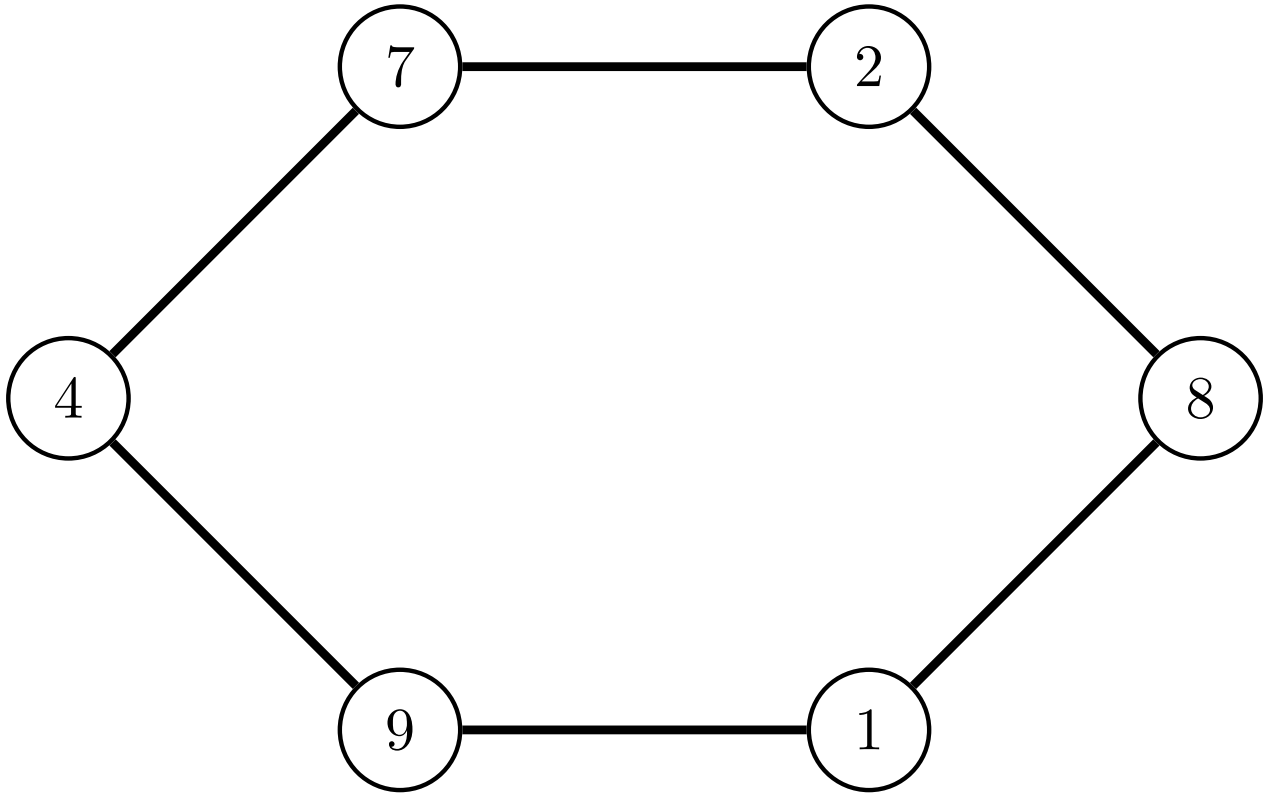
Example

Round 2:



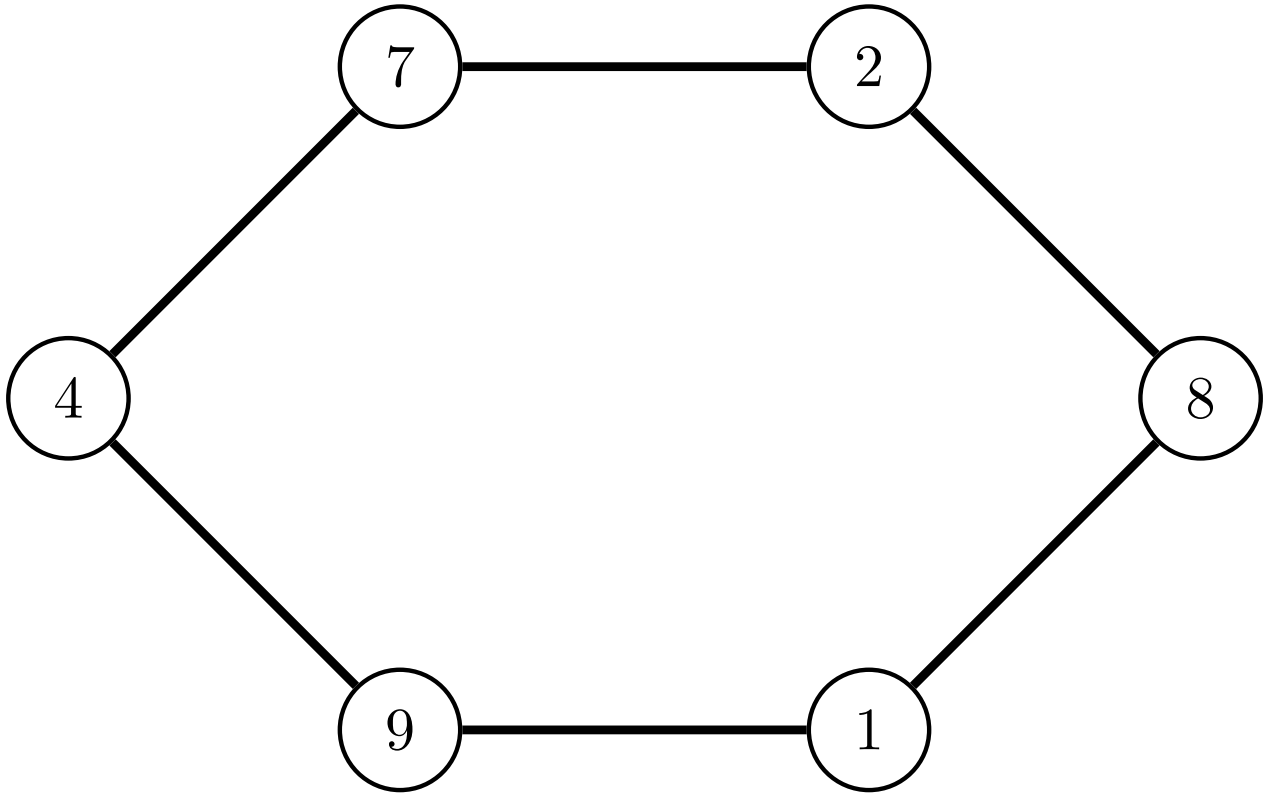
Example

Round 3:



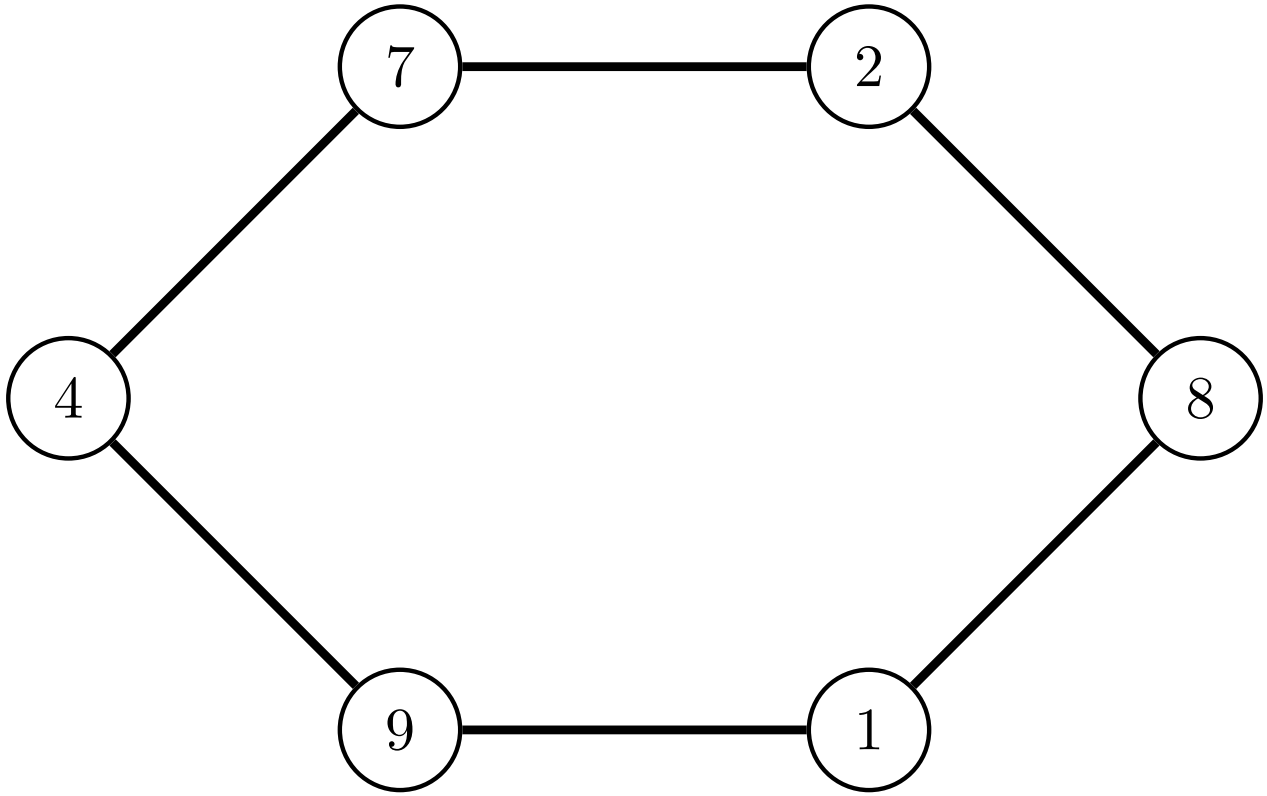
Example

Round 4:



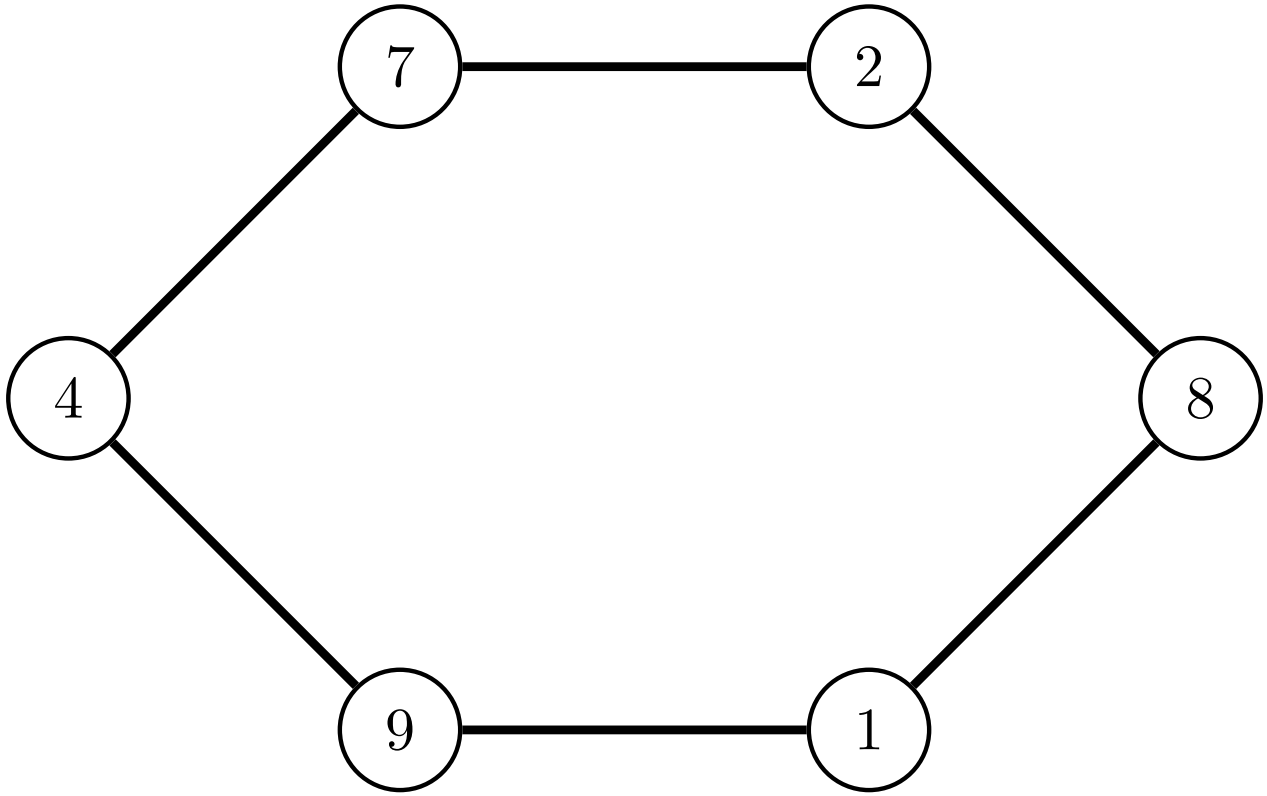
Example

Round 5:



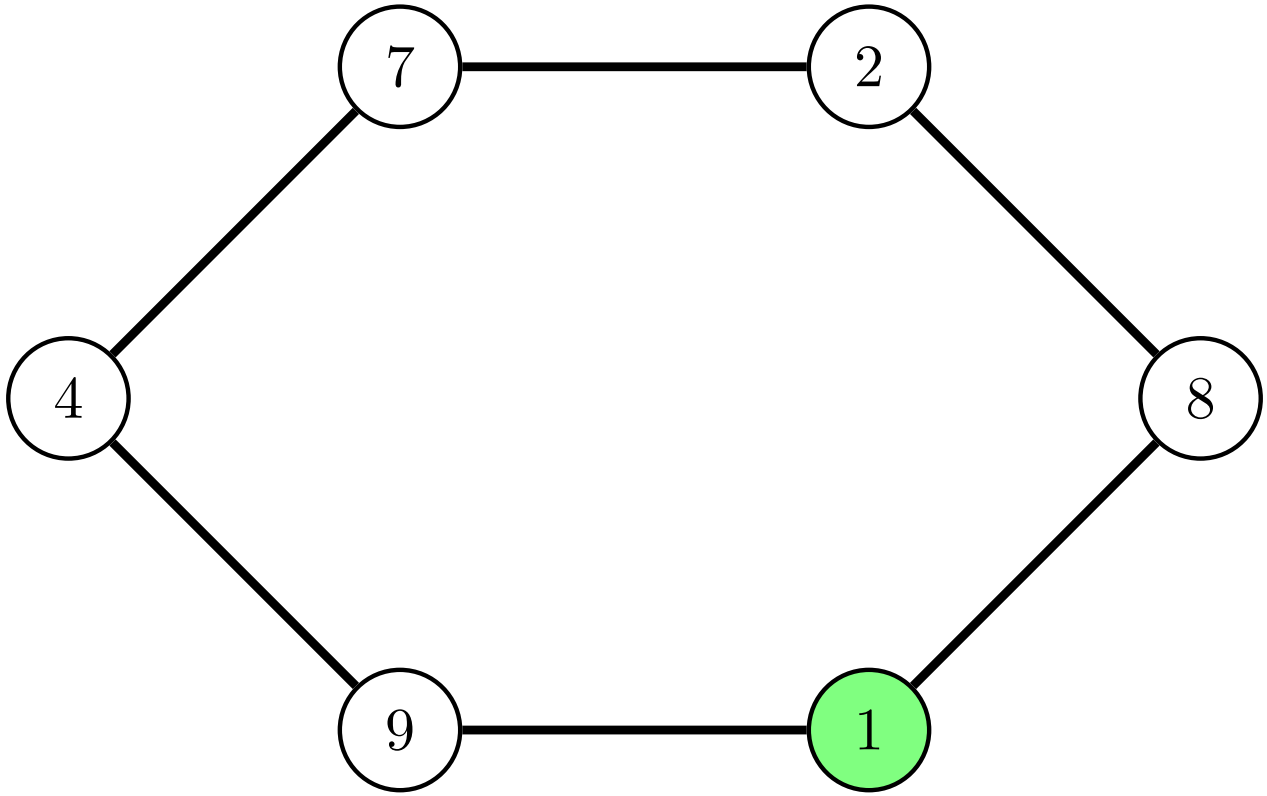
Example

Round 6:



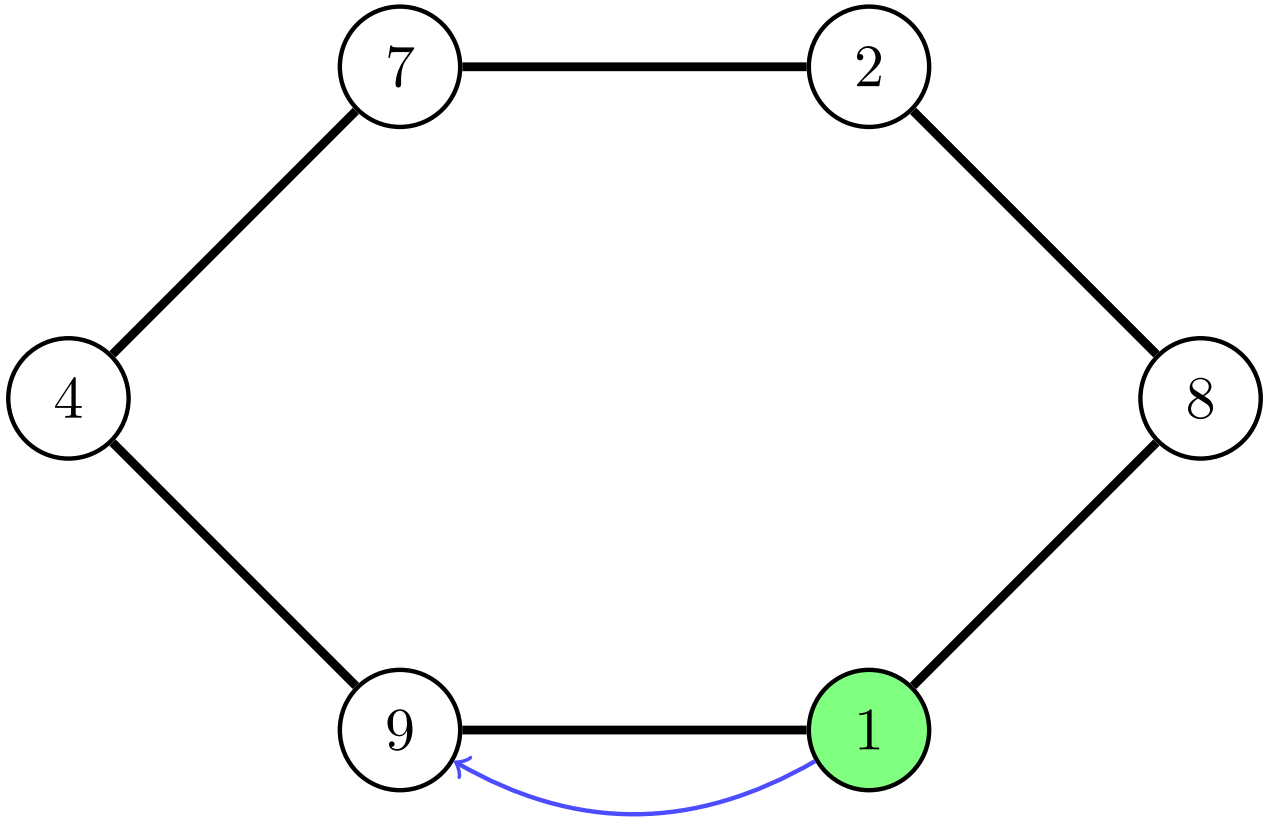
Example

Round 7:



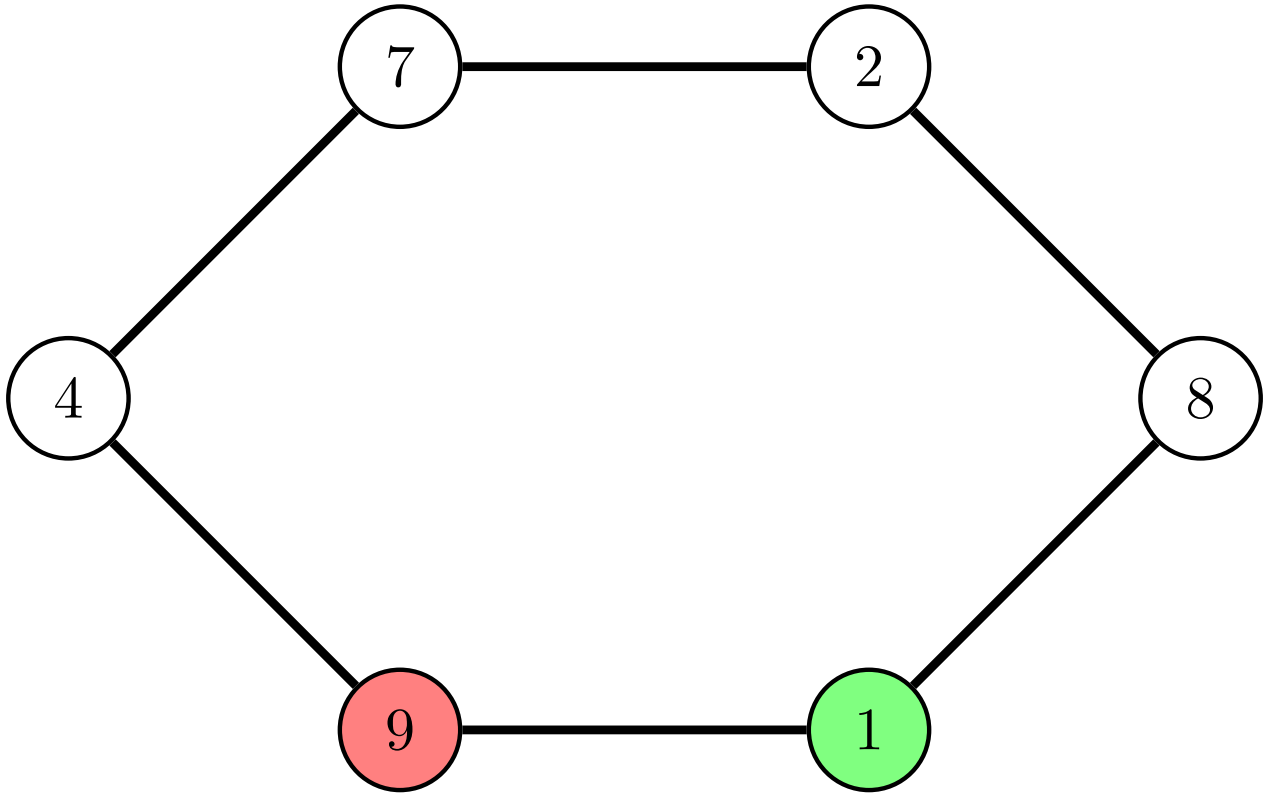
Example

Round 7:



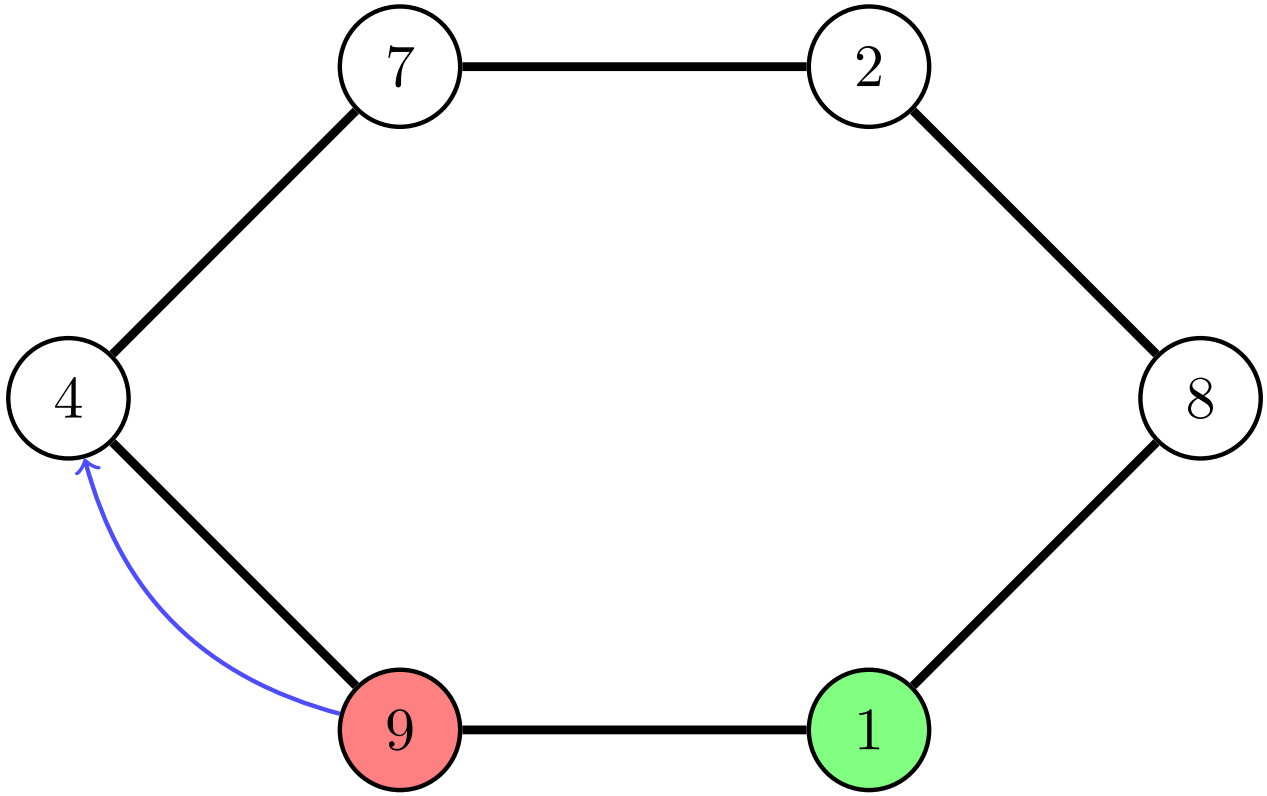
Example

Round 8:



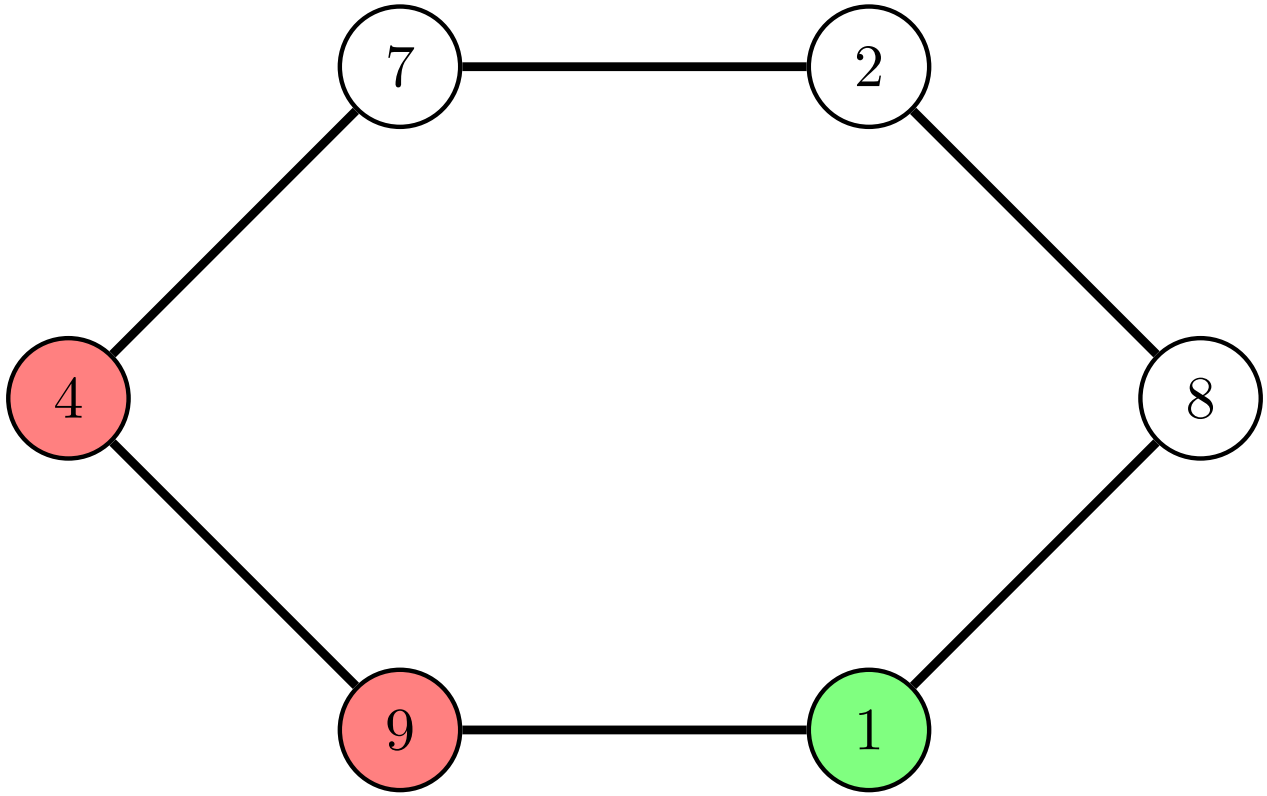
Example

Round 8:



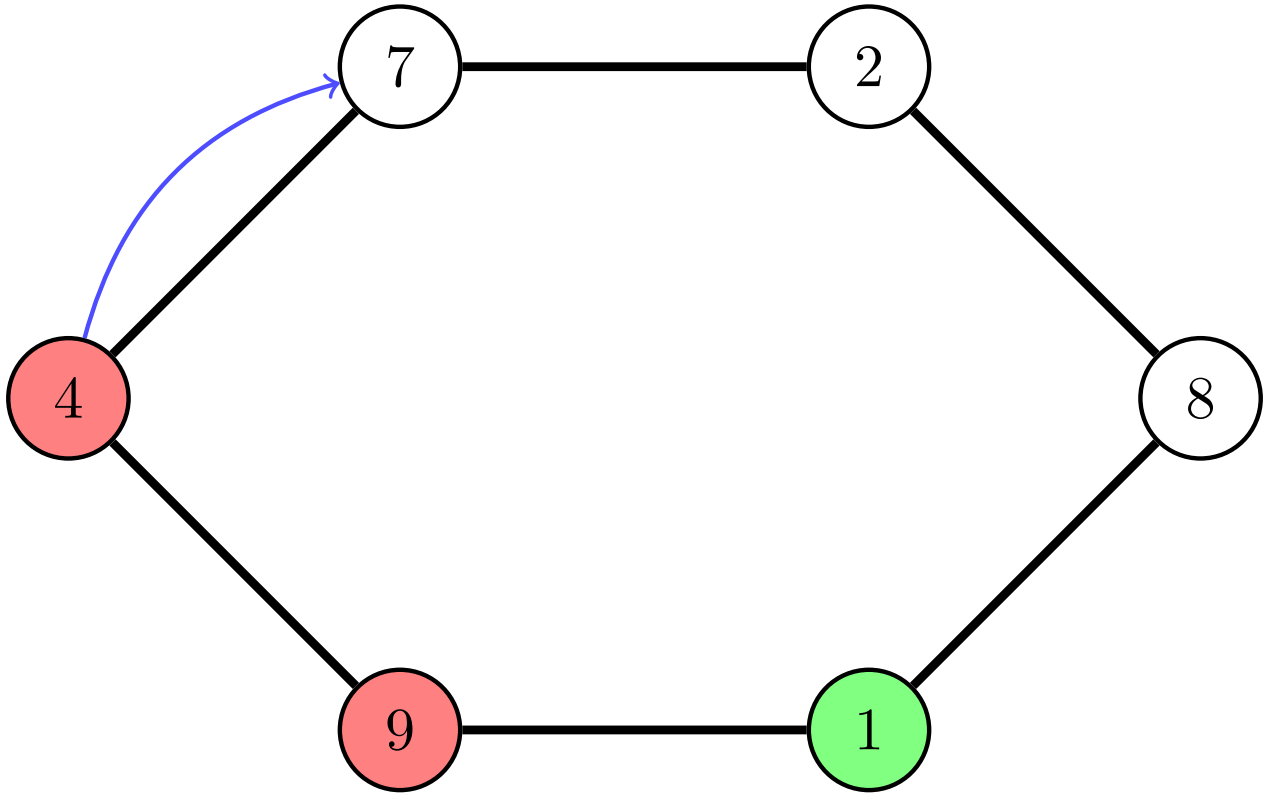
Example

Round 9:



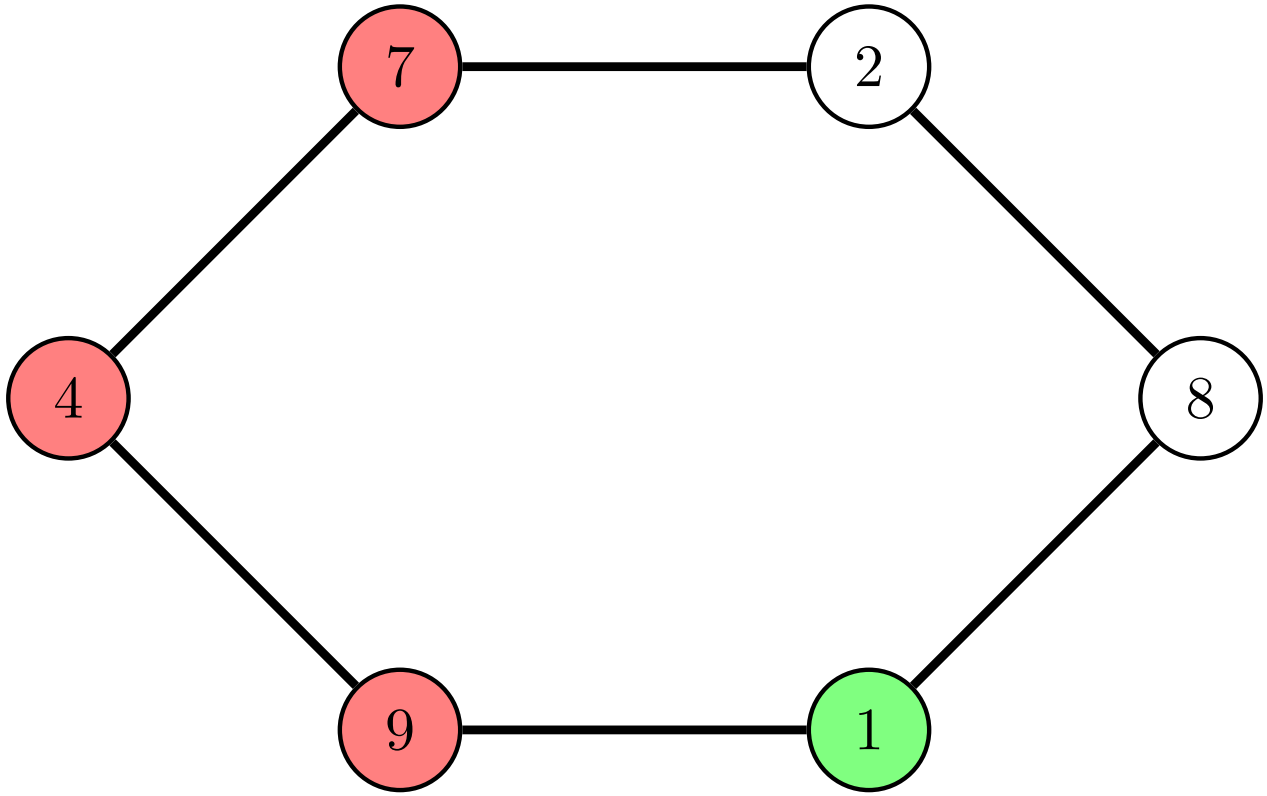
Example

Round 9:



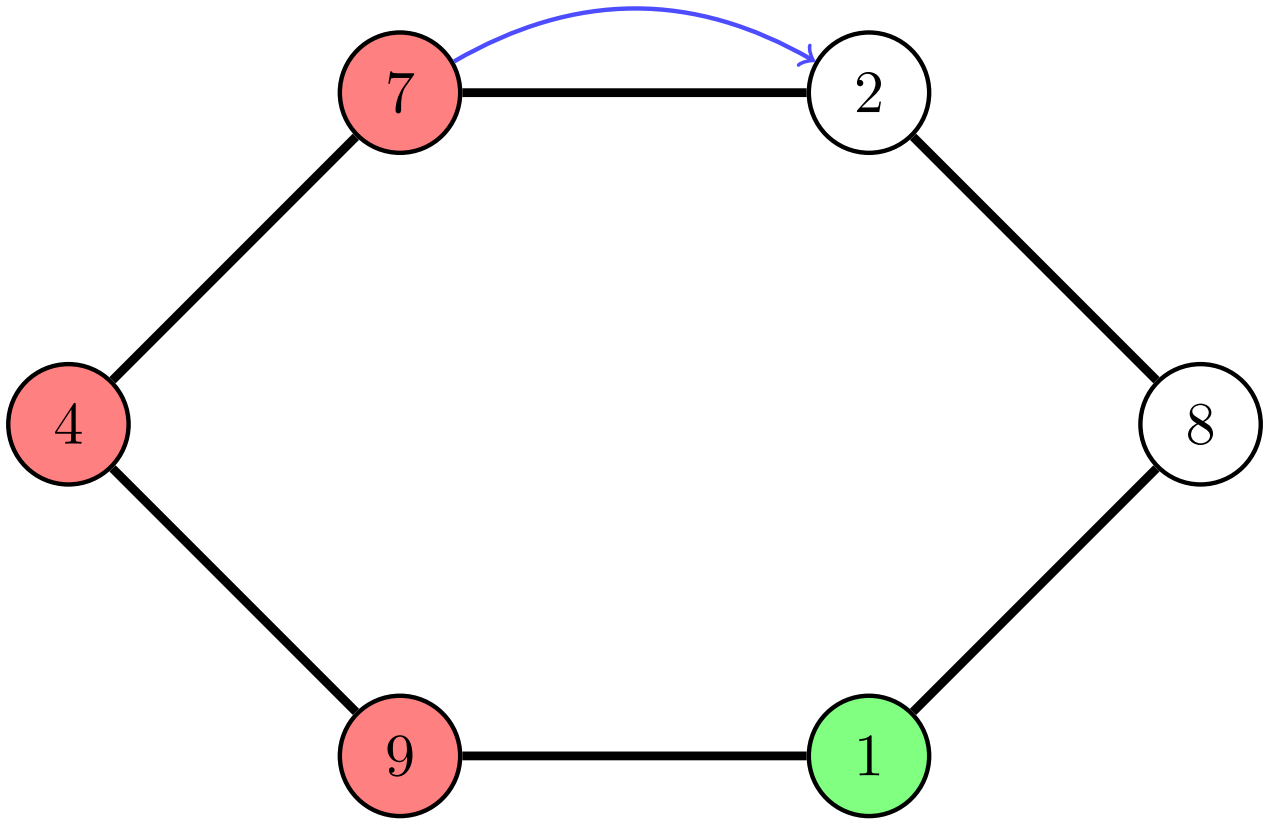
Example

Round 10:



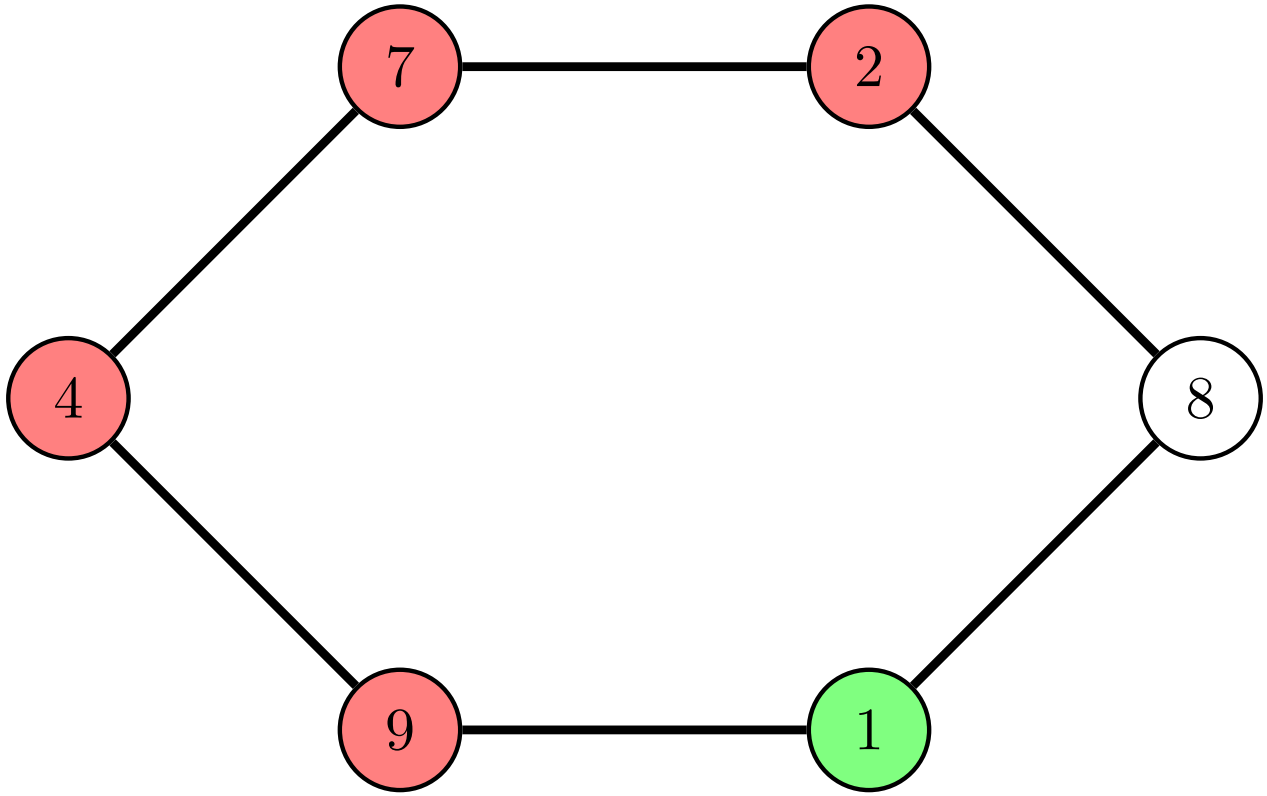
Example

Round 10:



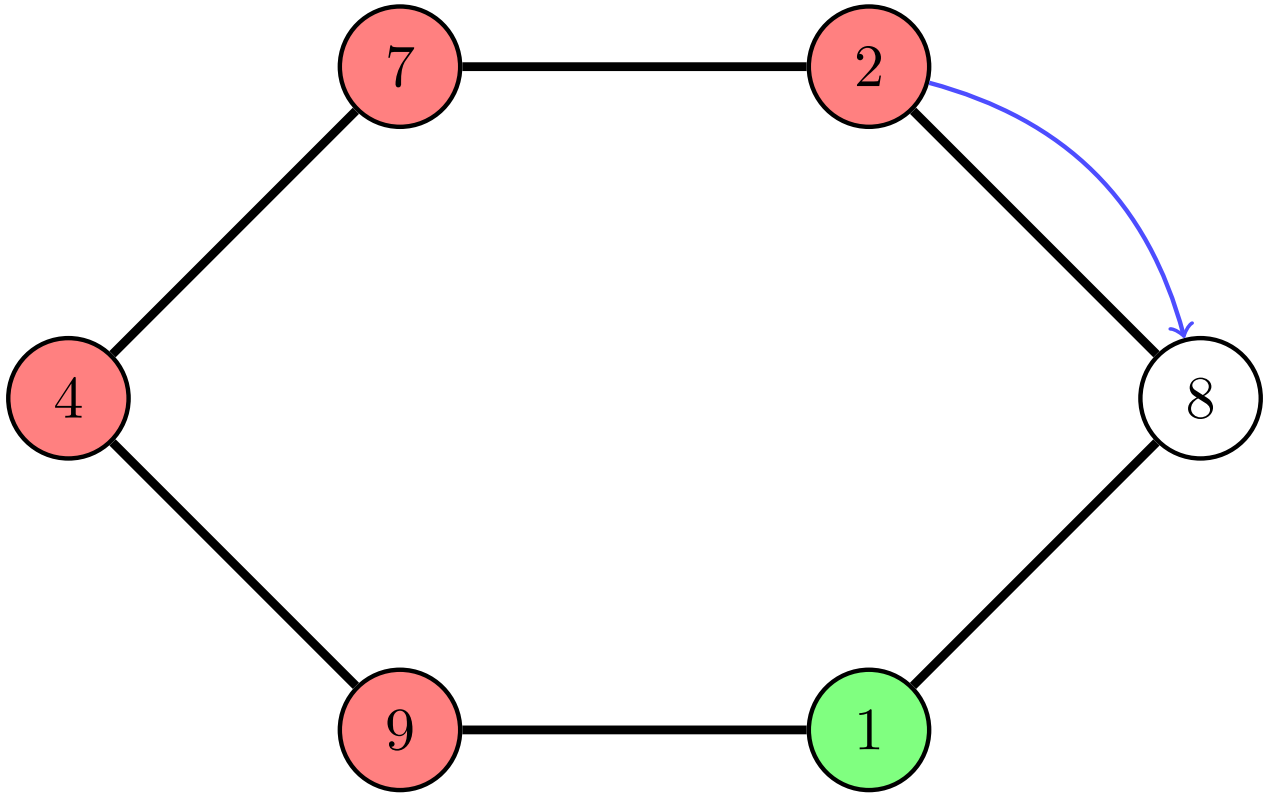
Example

Round 11:



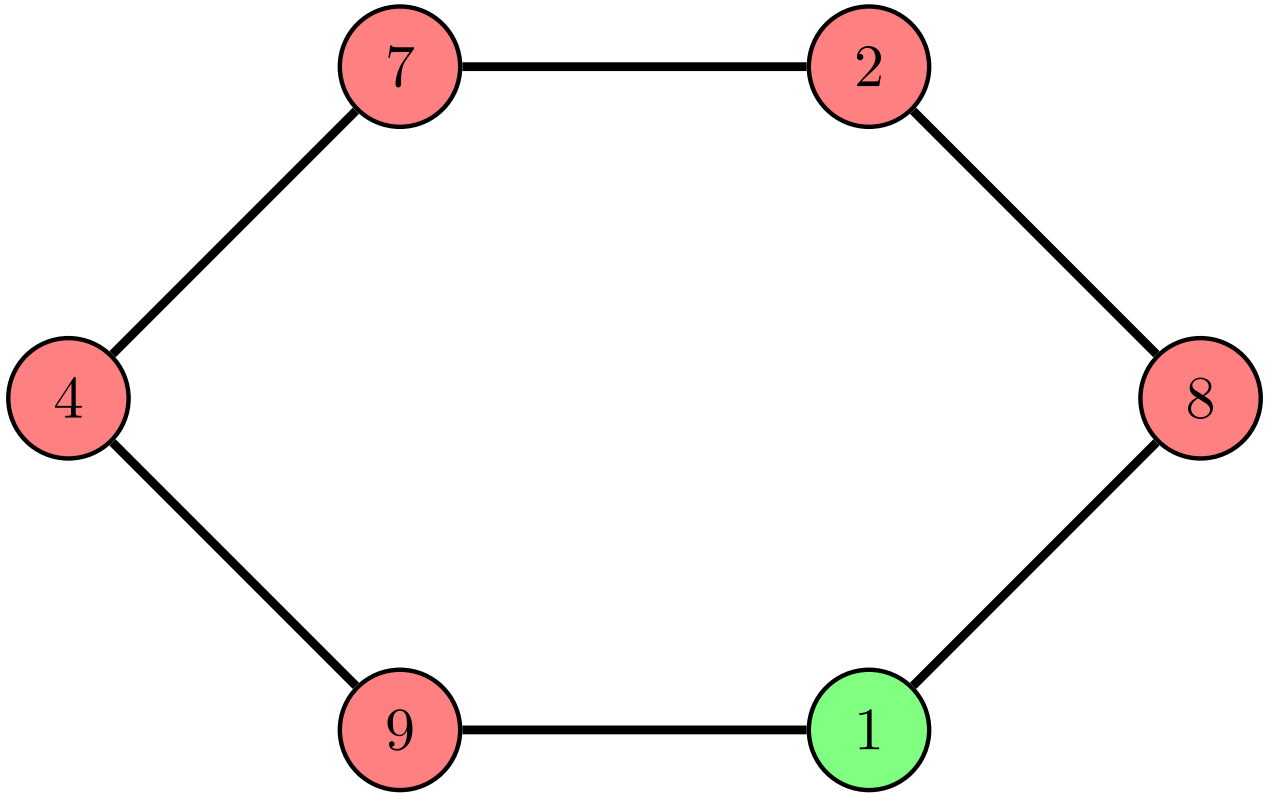
Example

Round 11:



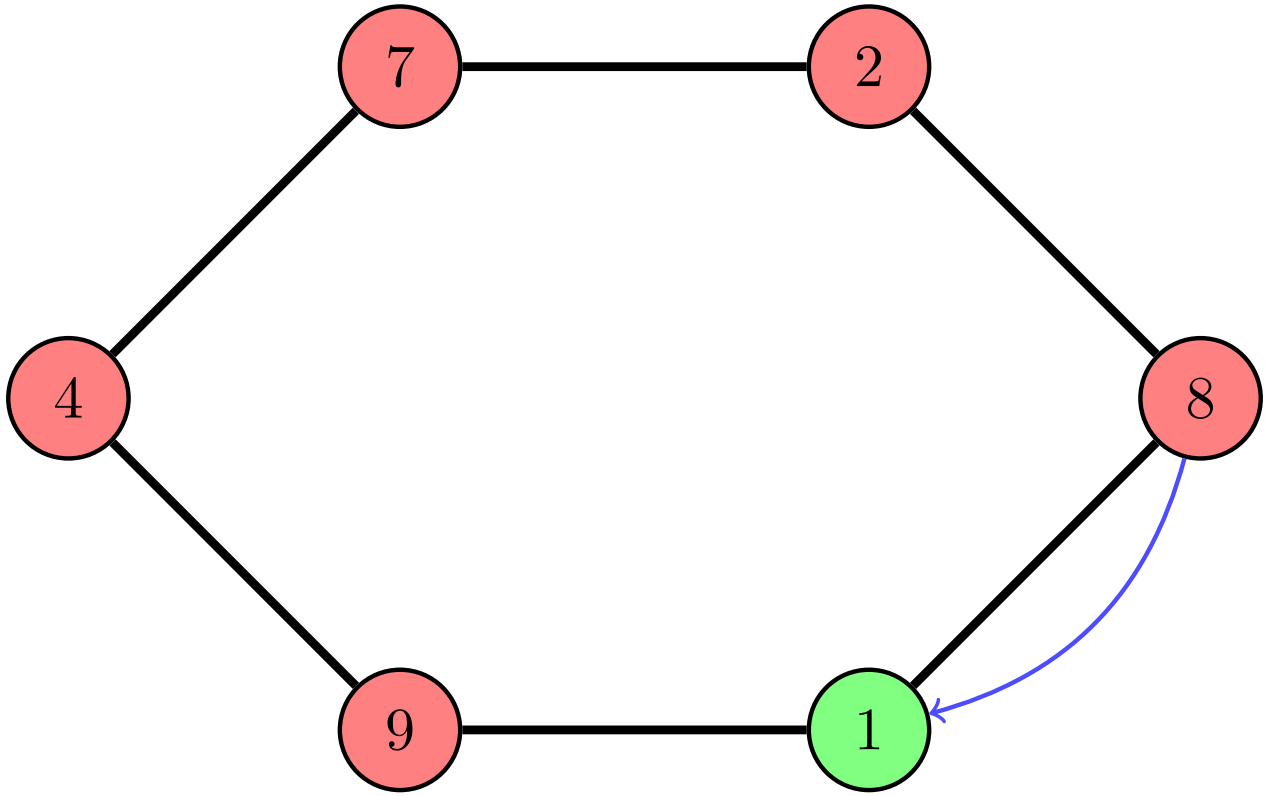
Example

Round 12:



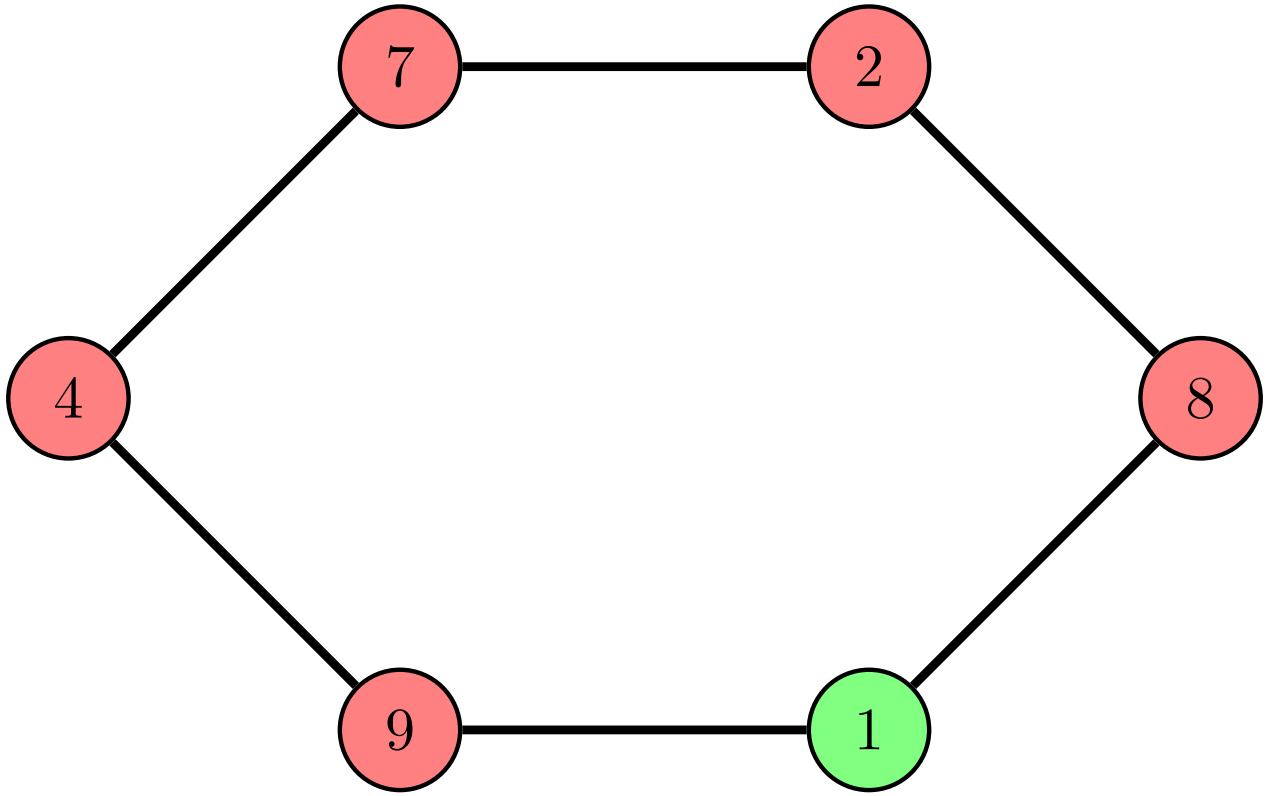
Example

Round 12:



Example

Round 13:



Analysis

- In every n rounds, at most one node decides to become the leader.

Analysis

- In every n rounds, at most one node decides to become the leader.
- Therefore, the leader must inform all other nodes within n rounds for the algorithm to be correct.

Analysis

- In every n rounds, at most one node decides to become the leader.
- Therefore, the leader must inform all other nodes within n rounds for the algorithm to be correct.
- This is guaranteed by forwarding the message around the ring.

Analysis

- In every n rounds, at most one node decides to become the leader.
- Therefore, the leader must inform all other nodes within n rounds for the algorithm to be correct.
- This is guaranteed by forwarding the message around the ring.
- Number of messages: $O(n)$.

Analysis

- In every n rounds, at most one node decides to become the leader.
- Therefore, the leader must inform all other nodes within n rounds for the algorithm to be correct.
- This is guaranteed by forwarding the message around the ring.
- Number of messages: $O(n)$.
- Number of rounds: $\Theta(n \cdot \min_v \mathrm{ID}(v))$.

Analysis

- In every n rounds, at most one node decides to become the leader.
- Therefore, the leader must inform all other nodes within n rounds for the algorithm to be correct.
- This is guaranteed by forwarding the message around the ring.
- Number of messages: $O(n)$.
- Number of rounds: $\Theta(n \cdot \min_v \mathrm{ID}(v))$.
  Example: For IDs in the range from 0 to $2n$, the smallest ID could have value n. In that case, already $\Theta(n^2)$ rounds are required.

Anonymous Networks

Anonymous: Nodes have no IDs.

Anonymous Networks

Anonymous: Nodes have no IDs.

Remark: After leader election, assigning IDs is relatively easy.

Anonymous Networks

Anonymous: Nodes have no IDs.

Remark: After leader election, assigning IDs is relatively easy.

Theorem ([Angluin '80])

Leader election in anonymous networks is impossible with deterministic algorithms, even in a synchronous ring.

Anonymous Networks

Anonymous: Nodes have no IDs.

Remark: After leader election, assigning IDs is relatively easy.

Theorem ([Angluin '80])

Leader election in anonymous networks is impossible with deterministic algorithms, even in a synchronous ring.

Key issue: How are deterministic algorithms formally defined?

Anonymous Networks

Anonymous: Nodes have no IDs.

Remark: After leader election, assigning IDs is relatively easy.

Theorem ([Angluin '80])

Leader election in anonymous networks is impossible with deterministic algorithms, even in a synchronous ring.

Key issue: How are deterministic algorithms formally defined?

- A function f is applied to each node v:

Anonymous Networks

Anonymous: Nodes have no IDs.

Remark: After leader election, assigning IDs is relatively easy.

Theorem ([Angluin '80])

Leader election in anonymous networks is impossible with deterministic algorithms, even in a synchronous ring.

Key issue: How are deterministic algorithms formally defined?

- A function f is applied to each node v:
  - **Input:** Number of nodes n, number of ports of v, complete history of v (the record of all messages received so far by a node, including their arrival times and input ports).

Anonymous Networks

Anonymous: Nodes have no IDs.

Remark: After leader election, assigning IDs is relatively easy.

Theorem ([Angluin '80])

Leader election in anonymous networks is impossible with deterministic algorithms, even in a synchronous ring.

Key issue: How are deterministic algorithms formally defined?

- A function f is applied to each node v:
  - **Input:** Number of nodes n, number of ports of v, complete history of v (the record of all messages received so far by a node, including their arrival times and input ports).
  - **Output:** Messages to be sent by the node through its output ports and, if applicable, the decision whether v becomes leader or follower.

Anonymous Networks

Anonymous: Nodes have no IDs.

Remark: After leader election, assigning IDs is relatively easy.

Theorem ([Angluin '80])

Leader election in anonymous networks is impossible with deterministic algorithms, even in a synchronous ring.

Key issue: How are deterministic algorithms formally defined?

- A function f is applied to each node v:
  - **Input:** Number of nodes n, number of ports of v, complete history of v (the record of all messages received so far by a node, including their arrival times and input ports).
  - **Output:** Messages to be sent by the node through its output ports and, if applicable, the decision whether v becomes leader or follower.
- The function solves the problem if, for every network, after finitely many applications of f, one node has been designated as leader and all others as followers.

Impossibility Proof

Induction claim

For every deterministic algorithm, in every round, the messages received clockwise and counterclockwise are the same for all nodes.

Impossibility Proof

Induction claim

For every deterministic algorithm, in every round, the messages received clockwise and counterclockwise are the same for all nodes.

If the induction claim holds:

- By the induction claim, all nodes have the same history in every round.

Impossibility Proof

Induction claim

For every deterministic algorithm, in every round, the messages received clockwise and counterclockwise are the same for all nodes.

If the induction claim holds:

- By the induction claim, all nodes have the same history in every round.
- Since the number of ports is also the same for every node, the input to function f is the same for all nodes in every round.

Impossibility Proof

Induction claim

For every deterministic algorithm, in every round, the messages received clockwise and counterclockwise are the same for all nodes.

If the induction claim holds:

- By the induction claim, all nodes have the same history in every round.
- Since the number of ports is also the same for every node, the input to function f is the same for all nodes in every round.
- Therefore, the output of function f is also the same for all nodes in every round.

Impossibility Proof

Induction claim

For every deterministic algorithm, in every round, the messages received clockwise and counterclockwise are the same for all nodes.

If the induction claim holds:

- By the induction claim, all nodes have the same history in every round.
- Since the number of ports is also the same for every node, the input to function f is the same for all nodes in every round.
- Therefore, the output of function f is also the same for all nodes in every round.
- Thus, in every round, f either makes all nodes leaders, makes all nodes followers, or makes no decision for any node.

Impossibility Proof

Induction claim

For every deterministic algorithm, in every round, the messages received clockwise and counterclockwise are the same for all nodes.

If the induction claim holds:

- By the induction claim, all nodes have the same history in every round.
- Since the number of ports is also the same for every node, the input to function f is the same for all nodes in every round.
- Therefore, the output of function f is also the same for all nodes in every round.
- Thus, in every round, f either makes all nodes leaders, makes all nodes followers, or makes no decision for any node.
- Therefore, f either produces an incorrect result or does not terminate.

Inductive Proof

Induction claim

For every deterministic algorithm, in every round, the messages received clockwise and counterclockwise are the same for all nodes.

Inductive Proof

Induction claim

For every deterministic algorithm, in every round, the messages received clockwise and counterclockwise are the same for all nodes.

Base case: This holds trivially because no messages have yet been received in the first round.

Inductive Proof

Induction claim

For every deterministic algorithm, in every round, the messages received clockwise and counterclockwise are the same for all nodes.

Base case: This holds trivially because no messages have yet been received in the first round.

Inductive step:

- Since the induction claim holds for all previous rounds, every node has the same history.

Inductive Proof

Induction claim

For every deterministic algorithm, in every round, the messages received clockwise and counterclockwise are the same for all nodes.

Base case: This holds trivially because no messages have yet been received in the first round.

Inductive step:

- Since the induction claim holds for all previous rounds, every node has the same history.
- Since the number of ports is also the same for every node, the input to function f is the same for all nodes.

Inductive Proof

Induction claim

For every deterministic algorithm, in every round, the messages received clockwise and counterclockwise are the same for all nodes.

Base case: This holds trivially because no messages have yet been received in the first round.

Inductive step:

- Since the induction claim holds for all previous rounds, every node has the same history.
- Since the number of ports is also the same for every node, the input to function f is the same for all nodes.
- Therefore, the output of function f is also the same for all nodes.

Inductive Proof

Induction claim

For every deterministic algorithm, in every round, the messages received clockwise and counterclockwise are the same for all nodes.

Base case: This holds trivially because no messages have yet been received in the first round.

Inductive step:

- Since the induction claim holds for all previous rounds, every node has the same history.
- Since the number of ports is also the same for every node, the input to function f is the same for all nodes.
- Therefore, the output of function f is also the same for all nodes.
- Thus, all nodes send the same message clockwise and counterclockwise, respectively.

Inductive Proof

Induction claim

For every deterministic algorithm, in every round, the messages received clockwise and counterclockwise are the same for all nodes.

Base case: This holds trivially because no messages have yet been received in the first round.

Inductive step:

- Since the induction claim holds for all previous rounds, every node has the same history.
- Since the number of ports is also the same for every node, the input to function f is the same for all nodes.
- Therefore, the output of function f is also the same for all nodes.
- Thus, all nodes send the same message clockwise and counterclockwise, respectively.
- Due to the ring topology, all nodes receive the same message clockwise and counterclockwise, respectively.

Summary

Leader Election in a Ring:

1. Anonymous rings: impossible for deterministic algorithms.

Summary

Leader Election in a Ring:

1. Anonymous rings: impossible for deterministic algorithms.
   Next lecture: Randomized algorithm.

Summary

Leader Election in a Ring:

1. Anonymous rings: impossible for deterministic algorithms.
   Next lecture: Randomized algorithm.
2. Clockwise algorithm:
   - Deterministic synchronous/asynchronous algorithm for uniform/non-uniform rings.
   - $O(n)$ rounds, $O(n^2)$ messages.

Summary

Leader Election in a Ring:

1. Anonymous rings: impossible for deterministic algorithms.
   Next lecture: Randomized algorithm.
2. Clockwise algorithm:
   - Deterministic synchronous/asynchronous algorithm for uniform/non-uniform rings.
   - $O(n)$ rounds, $O(n^2)$ messages.
3. Waiting-time algorithm:
   - Deterministic synchronous algorithm for non-uniform rings.
   - $O(n \cdot \min_v \mathrm{ID}(v))$ rounds, $O(n)$ messages.

Summary

Leader Election in a Ring:

1. Anonymous rings: impossible for deterministic algorithms.
   Next lecture: Randomized algorithm.
2. Clockwise algorithm:
   - Deterministic synchronous/asynchronous algorithm for uniform/non-uniform rings.
   - $O(n)$ rounds, $O(n^2)$ messages.
3. Waiting-time algorithm:
   - Deterministic synchronous algorithm for non-uniform rings.
   - $O(n \cdot \min_v \mathrm{ID}(v))$ rounds, $O(n)$ messages.
   - Question: Is efficiency in both measures possible?

Summary

Leader Election in a Ring:

1. Anonymous rings: impossible for deterministic algorithms.
   Next lecture: Randomized algorithm.
2. Clockwise algorithm:
   - Deterministic synchronous/asynchronous algorithm for uniform/non-uniform rings.
   - $O(n)$ rounds, $O(n^2)$ messages.
3. Waiting-time algorithm:
   - Deterministic synchronous algorithm for non-uniform rings.
   - $O(n \cdot \min_v \mathrm{ID}(v))$ rounds, $O(n)$ messages.
   - Question: Is efficiency in both measures possible?

   Next lecture: $O(n)$ rounds, $O(n \log n)$ messages.

References

The content of this lecture is partly based on lectures by Robert Elsässer and Stefan Schmid.

Literature:

- Dana Angluin. "Local and Global Properties in Networks of Processors (Extended Abstract)." In: *Proc. of the Symposium on Theory of Computing (STOC)*. 1980, pp. 82–93
- Hagit Attiya, Jennifer Welch (2004) *Distributed Computing*, Chapter 3, Wiley.
- Ernest Chang, Rosemary Roberts. "An improved algorithm for decentralized extrema-finding in circular configurations of processes." *Communications of the ACM* 22(5): 281–283 (1979)
- Gérard Le Lann. "Distributed Systems – Towards a Formal Approach." In: *Proc. of the IFIP Congress*. 1977, pp. 155–160
- Nancy A. Lynch (1996) *Distributed Algorithms*, Chapter 3, Morgan Kaufmann.