Leader Election I

Algorithmen für verteilte Systeme

Sebastian Forster

Universität Salzburg

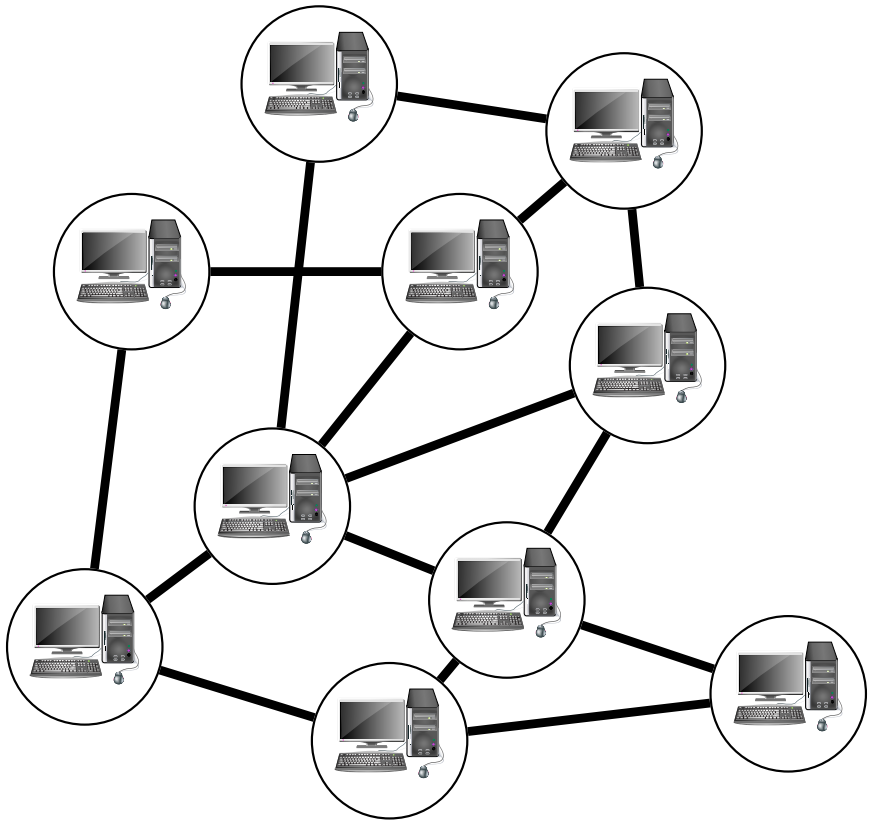
Prozessornetzwerke

Ausgangssituation: Mehrere gleichartige Prozessoren, die über Direktverbindungen kommunizieren

Prozessornetzwerke

Ausgangssituation: Mehrere gleichartige Prozessoren, die über Direktverbindungen kommunizieren

- Prozessornetzwerke werden als ungerichtete Graphen modelliert.
- Jedem Knoten des Graphen entspricht ein Prozessor mit lokalem Speicher und jeder Kante ein Kommunikationskanal.
- Lokale Sicht: Jeder Prozesser kennt nur seine direkten Nachbarn.



Netzwerke als Graphen

Definition

Ein **Graph** $G = (V, E)$ ist ein Paar bestehend aus einer Menge von Knoten V und einer Menge von Kanten E, wobei

- in einem **gerichteten** Graph jede Kante $e = (u, v)$ in (geordnetes) Paar zweier Knoten ist und
- in einem **ungerichteten** Graph jede Kante $e = \{u, v\}$ eine Menge zweier Knoten ist.

Anmerkung: Anstelle von $\{u, v\}$ schreiben wir dennoch oft (u, v)

Motivation Leader Election

Leader Election:

- Knoten eines Netzwerks einigen sich auf einen *Leader*
- Alle anderen heißen *Follower*

Motivation Leader Election

Leader Election:

- Knoten eines Netzwerks einigen sich auf einen *Leader*
- Alle anderen heißen *Follower*
- **Gesucht:** Algorithmus, der für jeden Knoten entscheidet, ob er der Leader ist oder ein Follower
- **Motivation:** Leader kann Koordinationsaufgaben übernehmen

Motivation Leader Election

Leader Election:

- Knoten eines Netzwerks einigen sich auf einen *Leader*
- Alle anderen heißen *Follower*
- **Gesucht:** Algorithmus, der für jeden Knoten entscheidet, ob er der Leader ist oder ein Follower
- **Motivation:** Leader kann Koordinationsaufgaben übernehmen

Symmetry Breaking:

- A priori eignet sich jeder Knoten als Leader
- Lösung ist nicht eindeutig, es gibt n verschiedene Lösungen

Motivation Leader Election

Leader Election:

- Knoten eines Netzwerks einigen sich auf einen *Leader*
- Alle anderen heißen *Follower*
- **Gesucht:** Algorithmus, der für jeden Knoten entscheidet, ob er der Leader ist oder ein Follower
- **Motivation:** Leader kann Koordinationsaufgaben übernehmen

Symmetry Breaking:

- A priori eignet sich jeder Knoten als Leader
- Lösung ist nicht eindeutig, es gibt n verschiedene Lösungen
- Schwierigkeit beim Finden einer Lösung ist die konsistente Entscheidung für eine der Lösungen

Kommunikationsmodell

- Kommunikationsnetzwerk durch ungerichteten Graph vorgegeben

Kommunikationsmodell

- Kommunikationsnetzwerk durch ungerichteten Graph vorgegeben
- Jeder Knoten kennt Anzahl seiner Nachbarn

Kommunikationsmodell

- Kommunikationsnetzwerk durch ungerichteten Graph vorgegeben
- Jeder Knoten kennt Anzahl seiner Nachbarn
- Jeder Knoten kann über nummerierte Ports beliebige Nachrichten an seine Nachbarn senden und von diesen empfangen

Kommunikationsmodell

- Kommunikationsnetzwerk durch ungerichteten Graph vorgegeben
- Jeder Knoten kennt Anzahl seiner Nachbarn
- Jeder Knoten kann über nummerierte Ports beliebige Nachrichten an seine Nachbarn senden und von diesen empfangen
- Synchron:
  - Rundenbasierte Kommunikation:

Kommunikationsmodell

- Kommunikationsnetzwerk durch ungerichteten Graph vorgegeben
- Jeder Knoten kennt Anzahl seiner Nachbarn
- Jeder Knoten kann über nummerierte Ports beliebige Nachrichten an seine Nachbarn senden und von diesen empfangen
- Synchron:
  - Rundenbasierte Kommunikation:
    1. Nachrichten von Nachbarn empfangen (entfällt in erster Runde)
    2. Interne Berechnungen
    3. Nachrichten an Nachbarn versenden

Kommunikationsmodell

- Kommunikationsnetzwerk durch ungerichteten Graph vorgegeben
- Jeder Knoten kennt Anzahl seiner Nachbarn
- Jeder Knoten kann über nummerierte Ports beliebige Nachrichten an seine Nachbarn senden und von diesen empfangen
- Synchron:
  - Rundenbasierte Kommunikation:
    1. Nachrichten von Nachbarn empfangen (entfällt in erster Runde)
    2. Interne Berechnungen
    3. Nachrichten an Nachbarn versenden
  - Beginn jeder Runde wird von globaler Uhr vorgegeben

Kommunikationsmodell

- Kommunikationsnetzwerk durch ungerichteten Graph vorgegeben
- Jeder Knoten kennt Anzahl seiner Nachbarn
- Jeder Knoten kann über nummerierte Ports beliebige Nachrichten an seine Nachbarn senden und von diesen empfangen
- Synchron:
  - Rundenbasierte Kommunikation:
    1. Nachrichten von Nachbarn empfangen (entfällt in erster Runde)
    2. Interne Berechnungen
    3. Nachrichten an Nachbarn versenden
  - Beginn jeder Runde wird von globaler Uhr vorgegeben
- Asynchron:
  - Event-basiert statt diskrete Zeitschritte

Kommunikationsmodell

- Kommunikationsnetzwerk durch ungerichteten Graph vorgegeben
- Jeder Knoten kennt Anzahl seiner Nachbarn
- Jeder Knoten kann über nummerierte Ports beliebige Nachrichten an seine Nachbarn senden und von diesen empfangen
- Synchron:
  - Rundenbasierte Kommunikation:
    1. Nachrichten von Nachbarn empfangen (entfällt in erster Runde)
    2. Interne Berechnungen
    3. Nachrichten an Nachbarn versenden
  - Beginn jeder Runde wird von globaler Uhr vorgegeben
- Asynchron:
  - Event-basiert statt diskrete Zeitschritte
  - Jeder Knoten startet initial zu einem beliebigen Zeitpunkt und wird darüberhinaus immer aktiv, wenn er eine Nachricht empfängt

Kommunikationsmodell

- Kommunikationsnetzwerk durch ungerichteten Graph vorgegeben
- Jeder Knoten kennt Anzahl seiner Nachbarn
- Jeder Knoten kann über nummerierte Ports beliebige Nachrichten an seine Nachbarn senden und von diesen empfangen
- Synchron:
  - Rundenbasierte Kommunikation:
    1. Nachrichten von Nachbarn empfangen (entfällt in erster Runde)
    2. Interne Berechnungen
    3. Nachrichten an Nachbarn versenden
  - Beginn jeder Runde wird von globaler Uhr vorgegeben
- Asynchron:
  - Event-basiert statt diskrete Zeitschritte
  - Jeder Knoten startet initial zu einem beliebigen Zeitpunkt und wird darüberhinaus immer aktiv, wenn er eine Nachricht empfängt
  - Jede Nachrichtenübermittlung benötigt endliche Zeit

Kommunikationsmodell

- Kommunikationsnetzwerk durch ungerichteten Graph vorgegeben
- Jeder Knoten kennt Anzahl seiner Nachbarn
- Jeder Knoten kann über nummerierte Ports beliebige Nachrichten an seine Nachbarn senden und von diesen empfangen
- Synchron:
  - Rundenbasierte Kommunikation:
    1. Nachrichten von Nachbarn empfangen (entfällt in erster Runde)
    2. Interne Berechnungen
    3. Nachrichten an Nachbarn versenden
  - Beginn jeder Runde wird von globaler Uhr vorgegeben
- Asynchron:
  - Event-basiert statt diskrete Zeitschritte
  - Jeder Knoten startet initial zu einem beliebigen Zeitpunkt und wird darüberhinaus immer aktiv, wenn er eine Nachricht empfängt
  - Jede Nachrichtenübermittlung benötigt endliche Zeit
  - Für Laufzeitanalyse: Delay von höchstens einer Zeiteinheit

Kommunikationsmodell

- Kommunikationsnetzwerk durch ungerichteten Graph vorgegeben
- Jeder Knoten kennt Anzahl seiner Nachbarn
- Jeder Knoten kann über nummerierte Ports beliebige Nachrichten an seine Nachbarn senden und von diesen empfangen
- Synchron:
  - Rundenbasierte Kommunikation:
    1. Nachrichten von Nachbarn empfangen (entfällt in erster Runde)
    2. Interne Berechnungen
    3. Nachrichten an Nachbarn versenden
  - Beginn jeder Runde wird von globaler Uhr vorgegeben
- Asynchron:
  - Event-basiert statt diskrete Zeitschritte
  - Jeder Knoten startet initial zu einem beliebigen Zeitpunkt und wird darüberhinaus immer aktiv, wenn er eine Nachricht empfängt
  - Jede Nachrichtenübermittlung benötigt endliche Zeit
  - Für Laufzeitanalyse: Delay von höchstens einer Zeiteinheit
- Komplexitätsmaße: Anzahl Zeiteinheiten (Runden) und Anzahl gesendeter Nachrichten (ohne interne Berechnungszeit)

Varianten

- Synchron vs. asynchron

Varianten

- Synchron vs. asynchron
- Anonym vs. identifizierbar

Varianten

- Synchron vs. asynchron
- Anonym vs. identifizierbar
  Identifizierbare Knoten v besitzen eindeutige ID (UID) $\mathrm{ID}(v)$
  IDs sind in der Regel Bit-Strings, oft der Länge $O(\log n)$
  Können als nichtnegative ganze Zahlen interpretiert werden

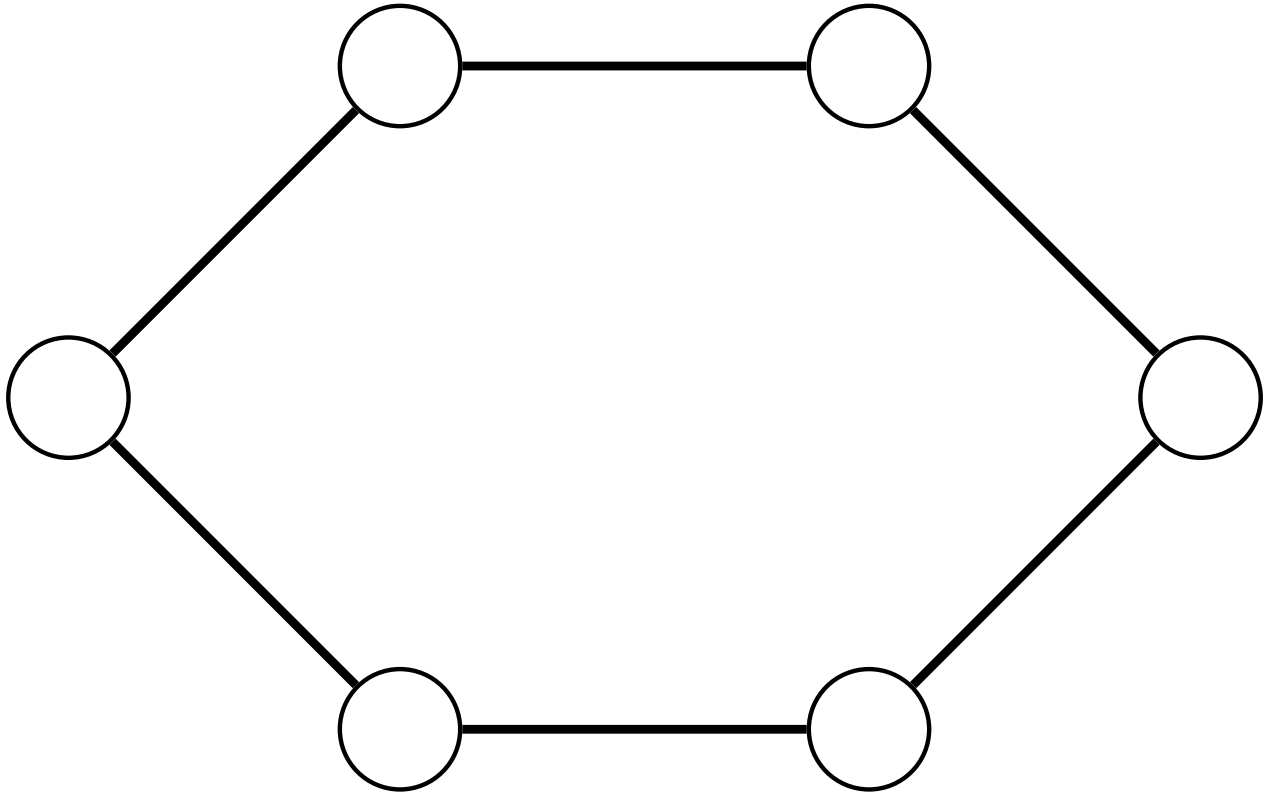
Varianten

- Synchron vs. asynchron
- Anonym vs. identifizierbar
  Identifizierbare Knoten v besitzen eindeutige ID (UID) $\mathrm{ID}(v)$
  IDs sind in der Regel Bit-Strings, oft der Länge $O(\log n)$
  Können als nichtnegative ganze Zahlen interpretiert werden
- Uniform vs. non-uniform

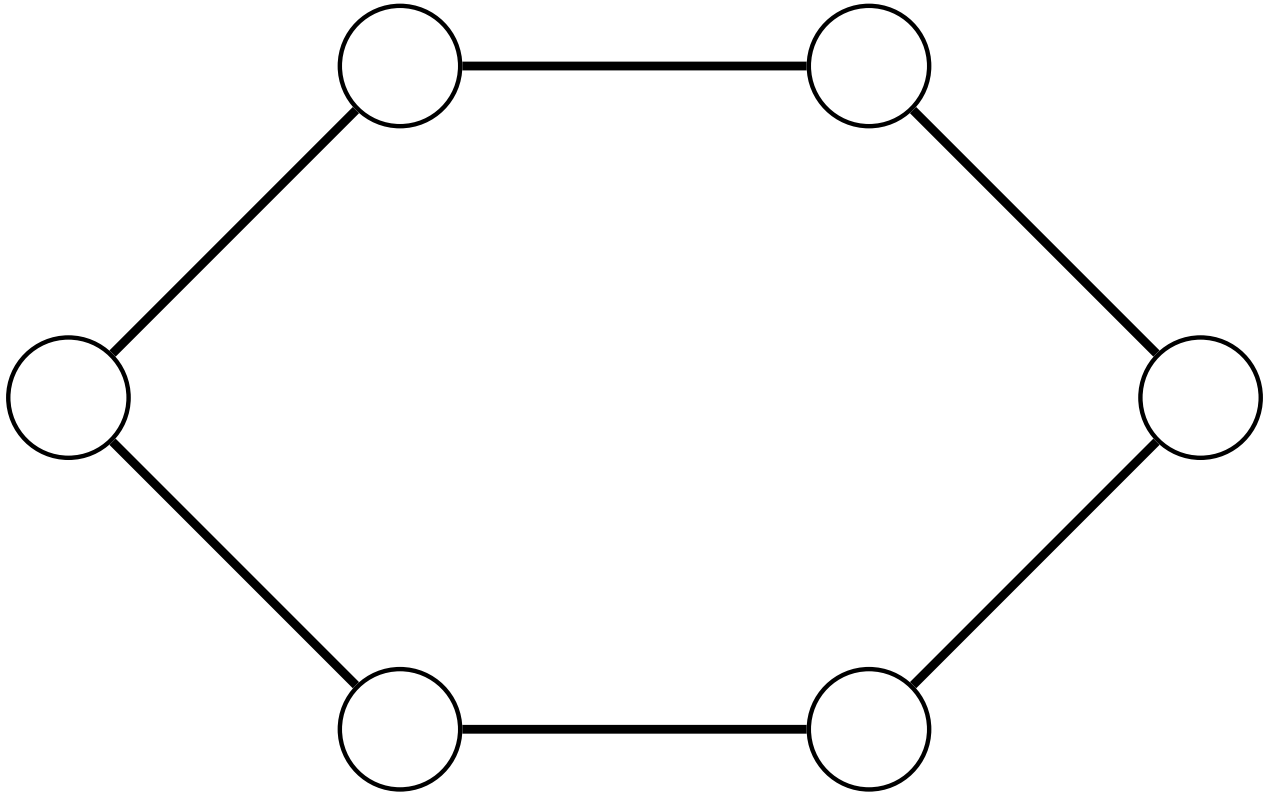
Varianten

- Synchron vs. asynchron
- Anonym vs. identifizierbar
  Identifizierbare Knoten v besitzen eindeutige ID (UID) $\mathrm{ID}(v)$
  IDs sind in der Regel Bit-Strings, oft der Länge $O(\log n)$
  Können als nichtnegative ganze Zahlen interpretiert werden
- Uniform vs. non-uniform
  In uniformen Netzwerken ist Anzahl der Knoten n kein globales Wissen
  Schwache Non-Uniformität: Manchmal ist Approximation $\hat{n}$ des exakten Werts n ausreichend

Ziel: Leader im Ring



Ziel: Leader im Ring



Jeder Knoten hat zwei Ports zu Nachbarknoten:

- „Im Uhrzeigersinn“ (Clockwise)
- „Gegen den Uhrzeigersinn“ (Counter-Clockwise)

Leader Election mit identifizierbaren Knoten

Idee

Knoten mit kleinster ID wird zum Leader

Leader Election mit identifizierbaren Knoten

Idee

Knoten mit kleinster ID wird zum Leader

Frage: Mit welchem Algorithmus kann dieser Knoten bestimmt werden?

Leader Election mit identifizierbaren Knoten

Idee

Knoten mit kleinster ID wird zum Leader

Frage: Mit welchem Algorithmus kann dieser Knoten bestimmt werden?

Achtung!

Die kleinste vergebene ID ist nicht bekannt!

Problem wäre z. B. trivial, wenn es immer einen Knoten mit ID 0 gäbe

Einfacher Clockwise Algorithmus

Annahmen: synchron, identifizierbar, non-uniform

Einfacher Clockwise Algorithmus

Annahmen: synchron, identifizierbar, non-uniform

Jeder Knoten v führt folgenden Algorithmus aus:

Einfacher Clockwise Algorithmus

Annahmen: synchron, identifizierbar, non-uniform

Jeder Knoten v führt folgenden Algorithmus aus:

Runde 1:

1. v setzt $L_v := \mathrm{ID}(v)$ (lokale Variable für kleinste bisher gesehene ID)
2. v sendet L_v an Nachbar im Uhrzeigersinn

Einfacher Clockwise Algorithmus

Annahmen: synchron, identifizierbar, non-uniform

Jeder Knoten v führt folgenden Algorithmus aus:

Runde 1:

1 v setzt $L_v := \mathrm{ID}(v)$ (lokale Variable für kleinste bisher gesehene ID)

2 v sendet L_v an Nachbar im Uhrzeigersinn

Runde $2 \leq r \leq n$:

1 **if** v *empfängt Nachricht* M **then**

Einfacher Clockwise Algorithmus

Annahmen: synchron, identifizierbar, non-uniform

Jeder Knoten v führt folgenden Algorithmus aus:

Runde 1:

1 v setzt $L_v := \mathrm{ID}(v)$ (lokale Variable für kleinste bisher gesehene ID)
2 v sendet L_v an Nachbar im Uhrzeigersinn

Runde $2 \leq r \leq n$:

```
1 if v empfängt Nachricht M then
2     if M < L_v then
3         v setzt L_v := M
4         v sendet L_v an Nachbar im Uhrzeigersinn
```

Einfacher Clockwise Algorithmus

Annahmen: synchron, identifizierbar, non-uniform

Jeder Knoten v führt folgenden Algorithmus aus:

Runde 1:

1 v setzt $L_v := \mathrm{ID}(v)$ (lokale Variable für kleinste bisher gesehene ID)
2 v sendet L_v an Nachbar im Uhrzeigersinn

Runde $2 \leq r \leq n$:

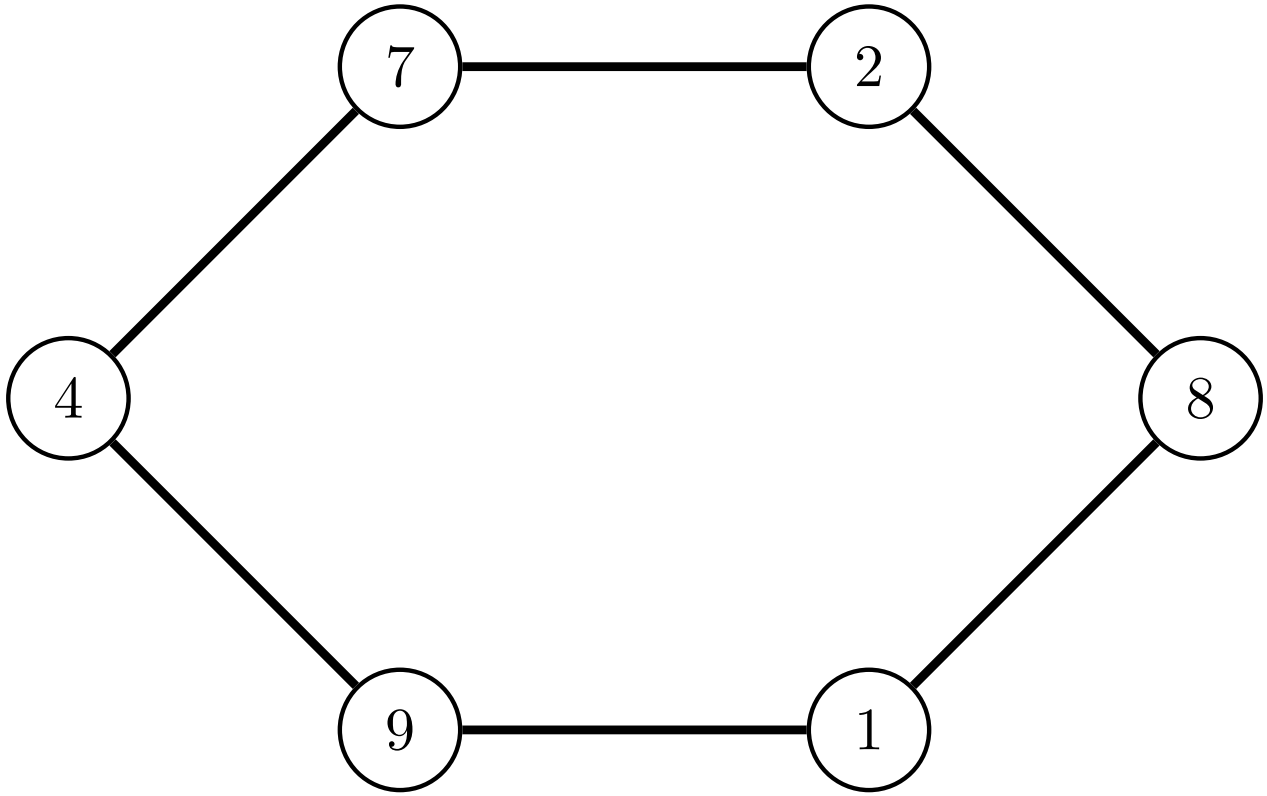
1 **if** v *empfängt Nachricht* M **then**
2 **if** $M < L_v$ **then**
3 v setzt $L_v := M$
4 v sendet L_v an Nachbar im Uhrzeigersinn

Runde $n + 1$:

1 **if** $L_v = \mathrm{ID}(v)$ **then**
2 v wird zum Leader
3 **else**
4 v wird zum Follower

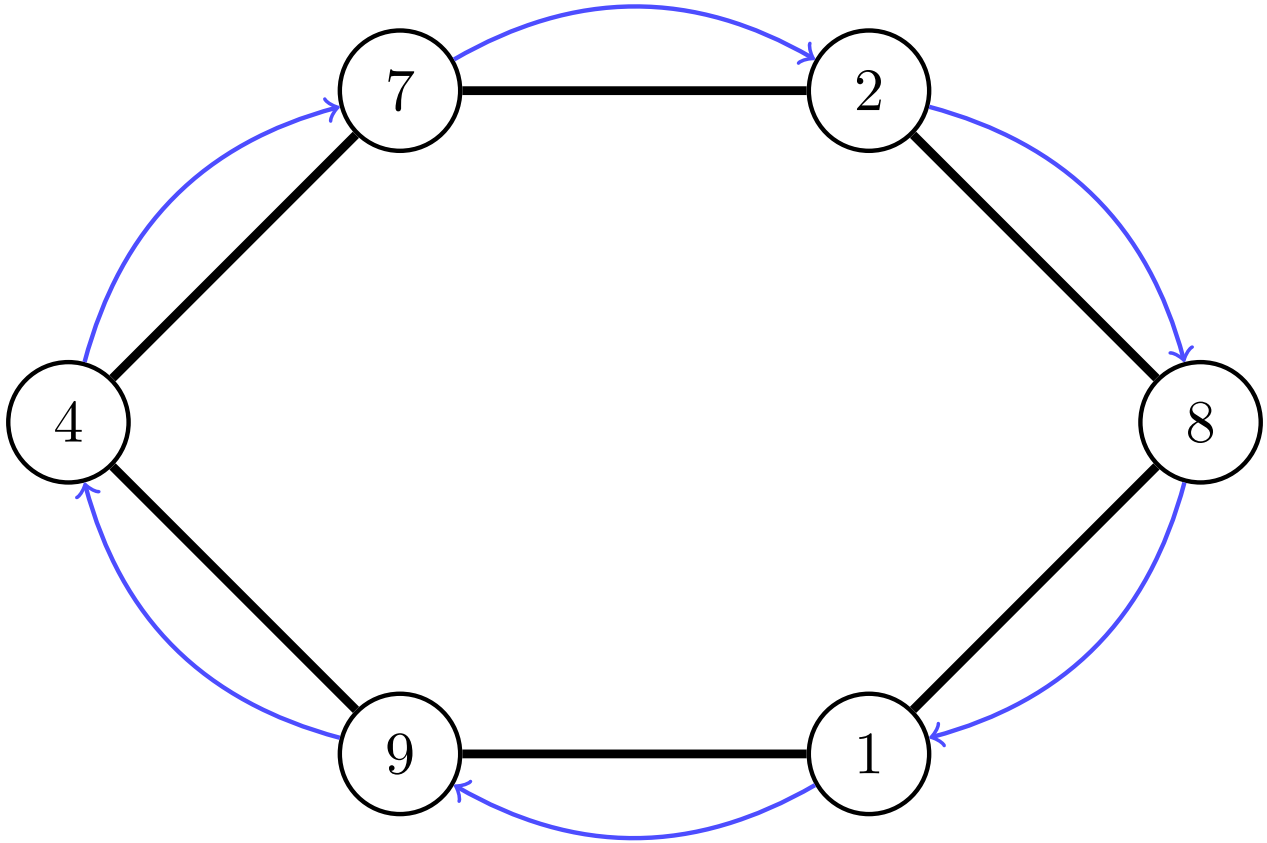
Beispiel

Runde 1:



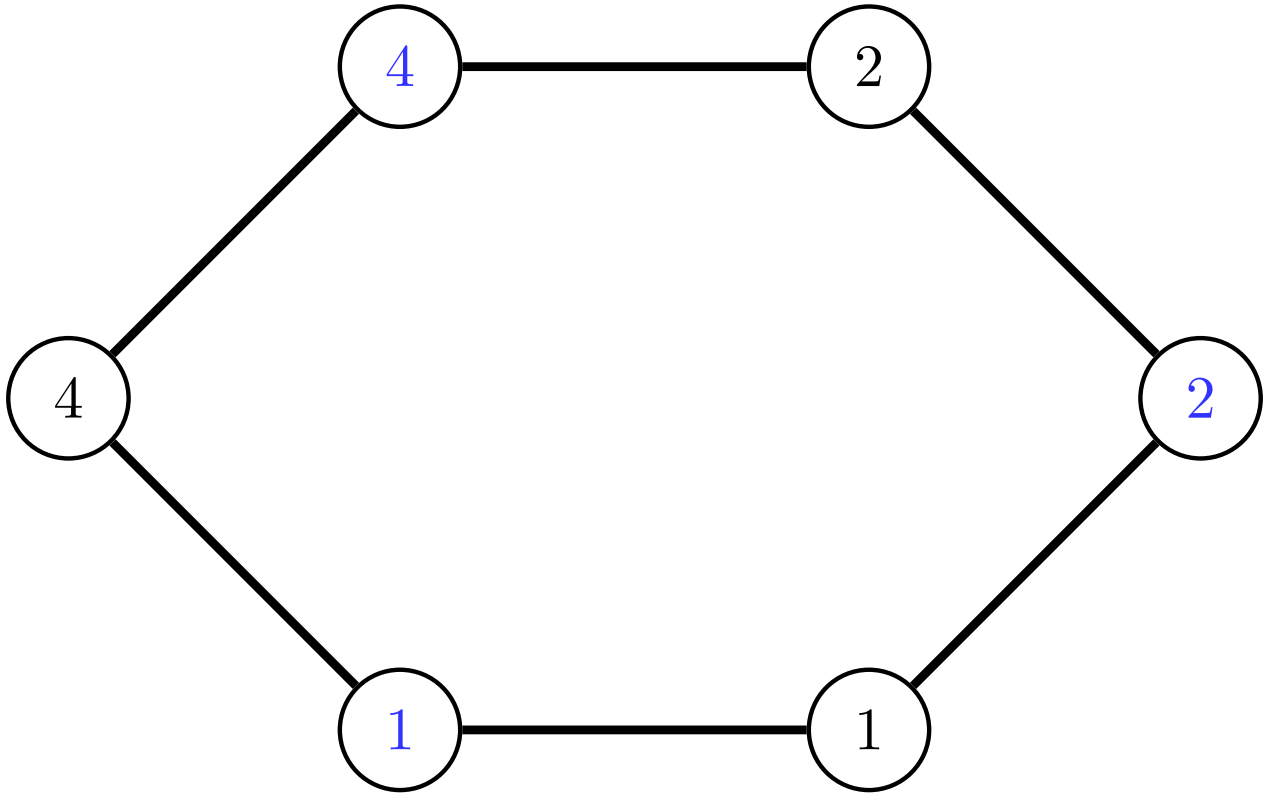
Beispiel

Runde 1:



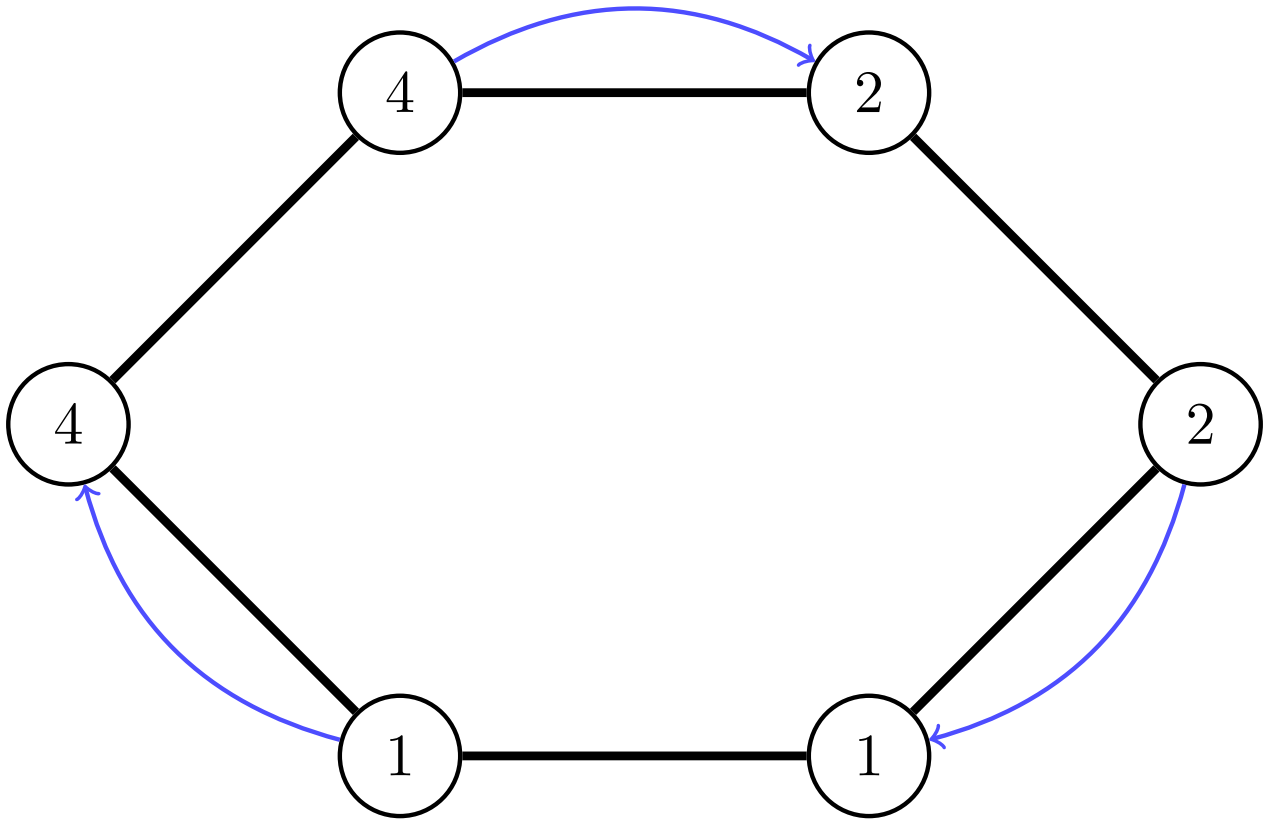
Beispiel

Runde 2:



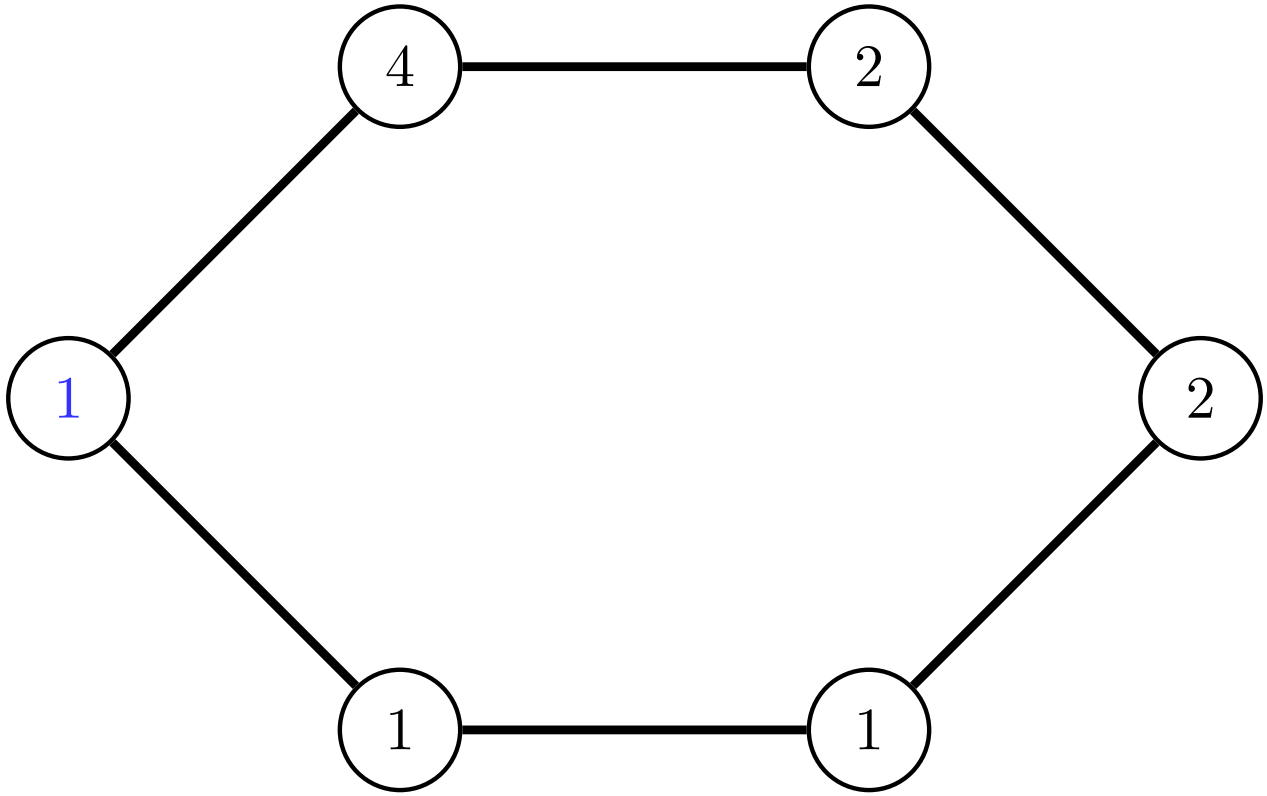
Beispiel

Runde 2:



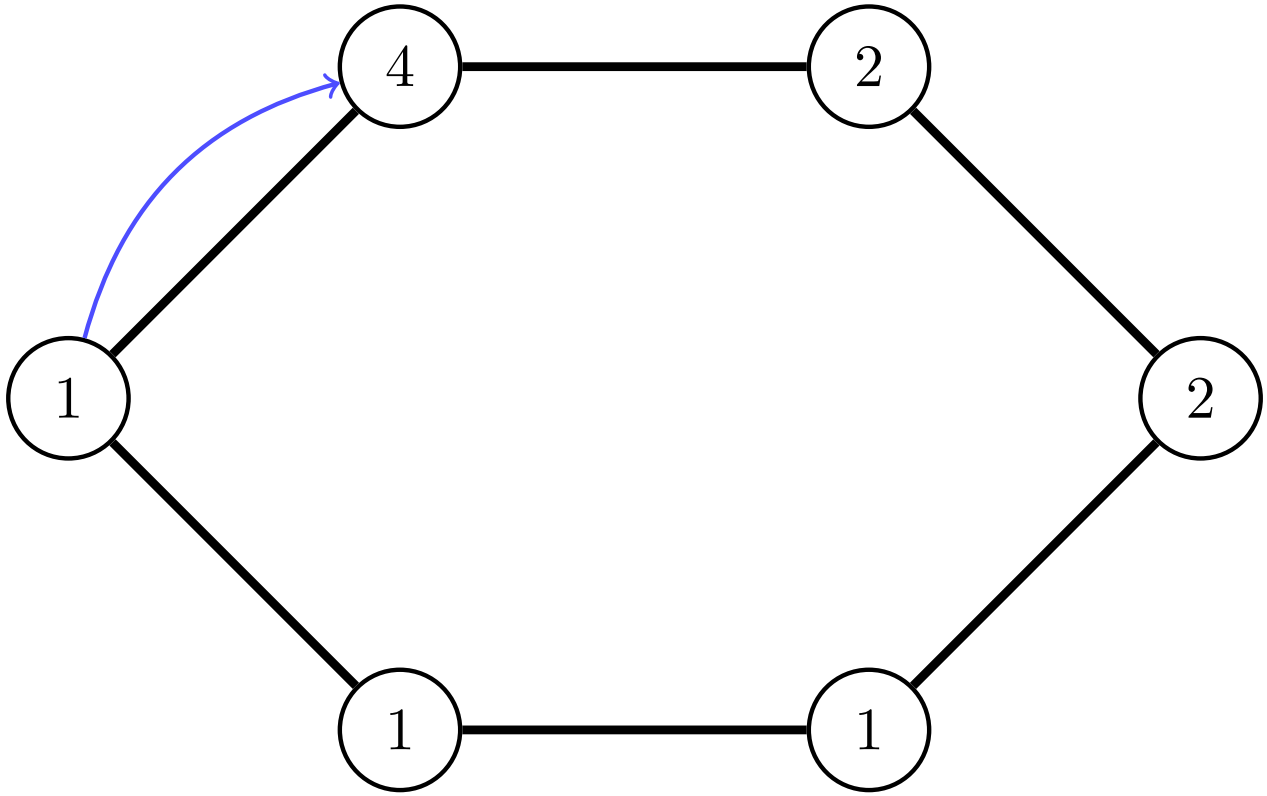
Beispiel

Runde 3:



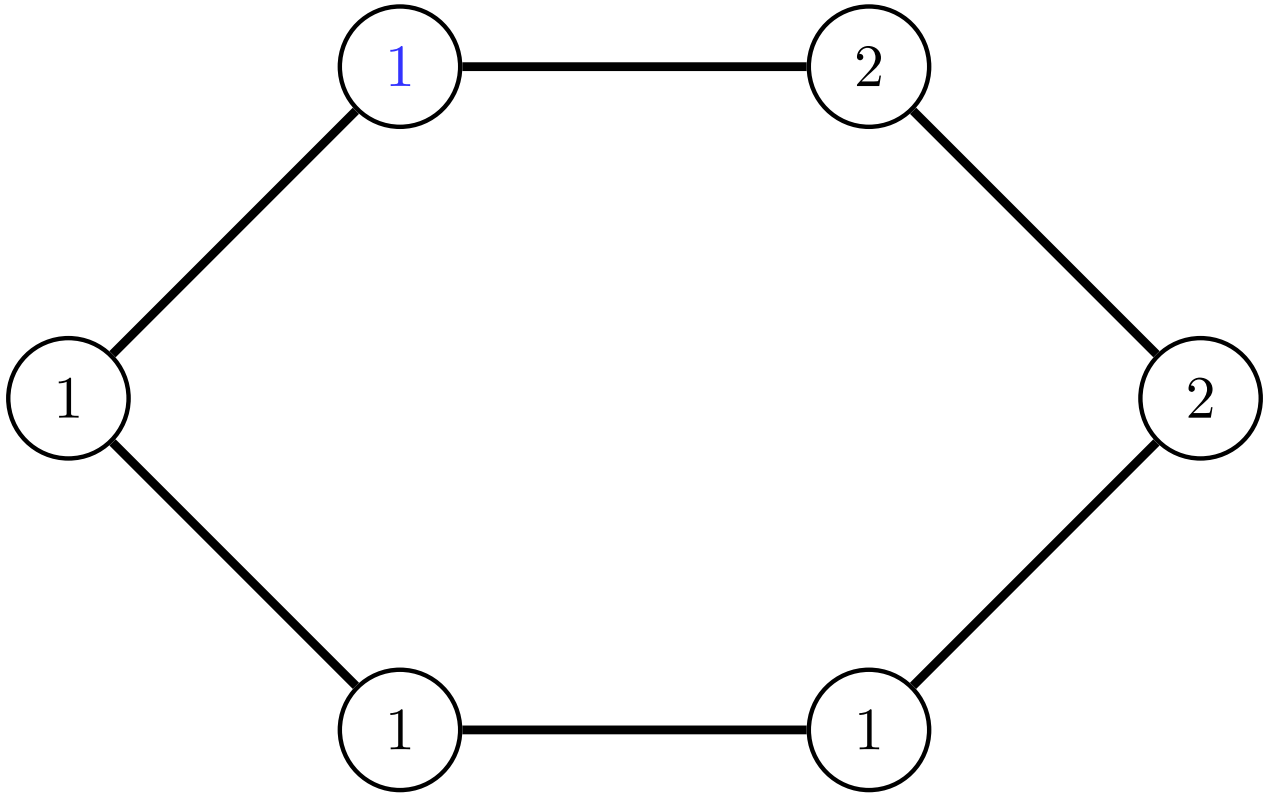
Beispiel

Runde 3:



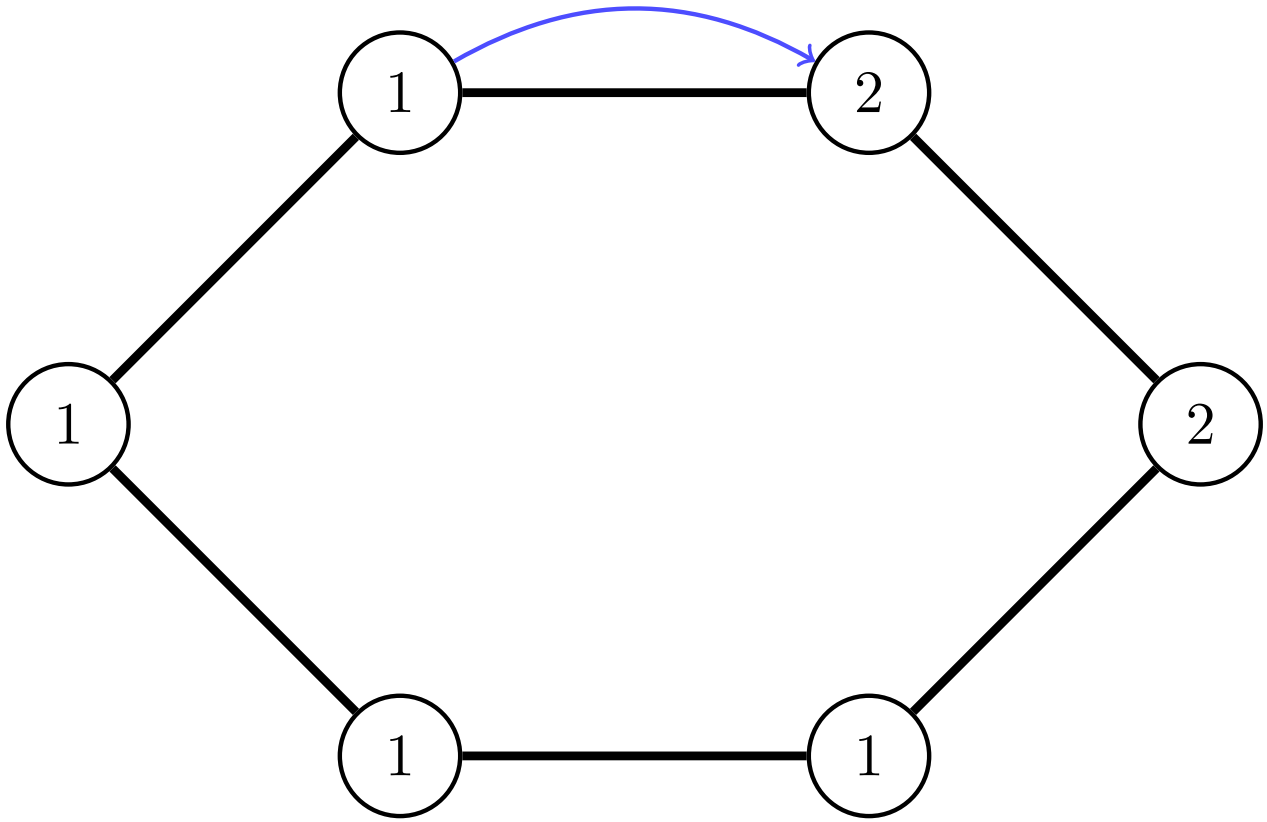
Beispiel

Runde 4:



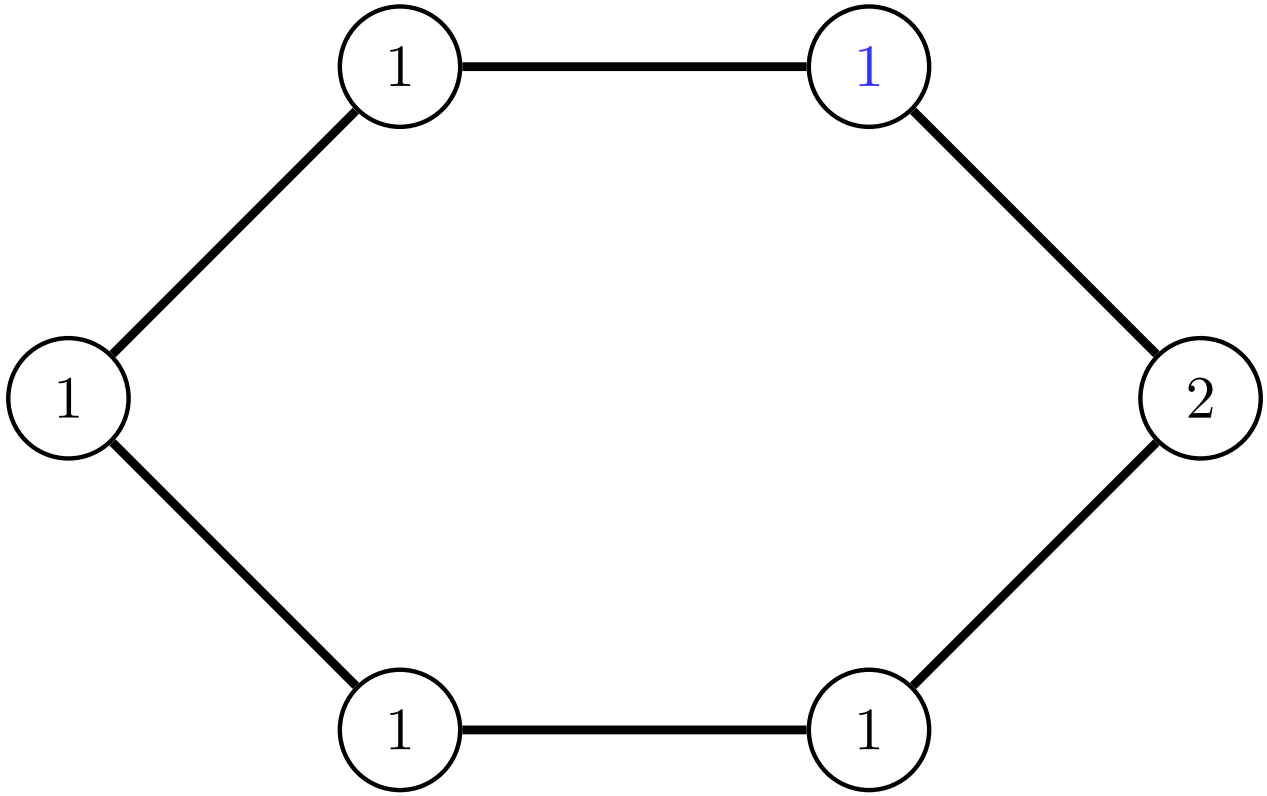
Beispiel

Runde 4:



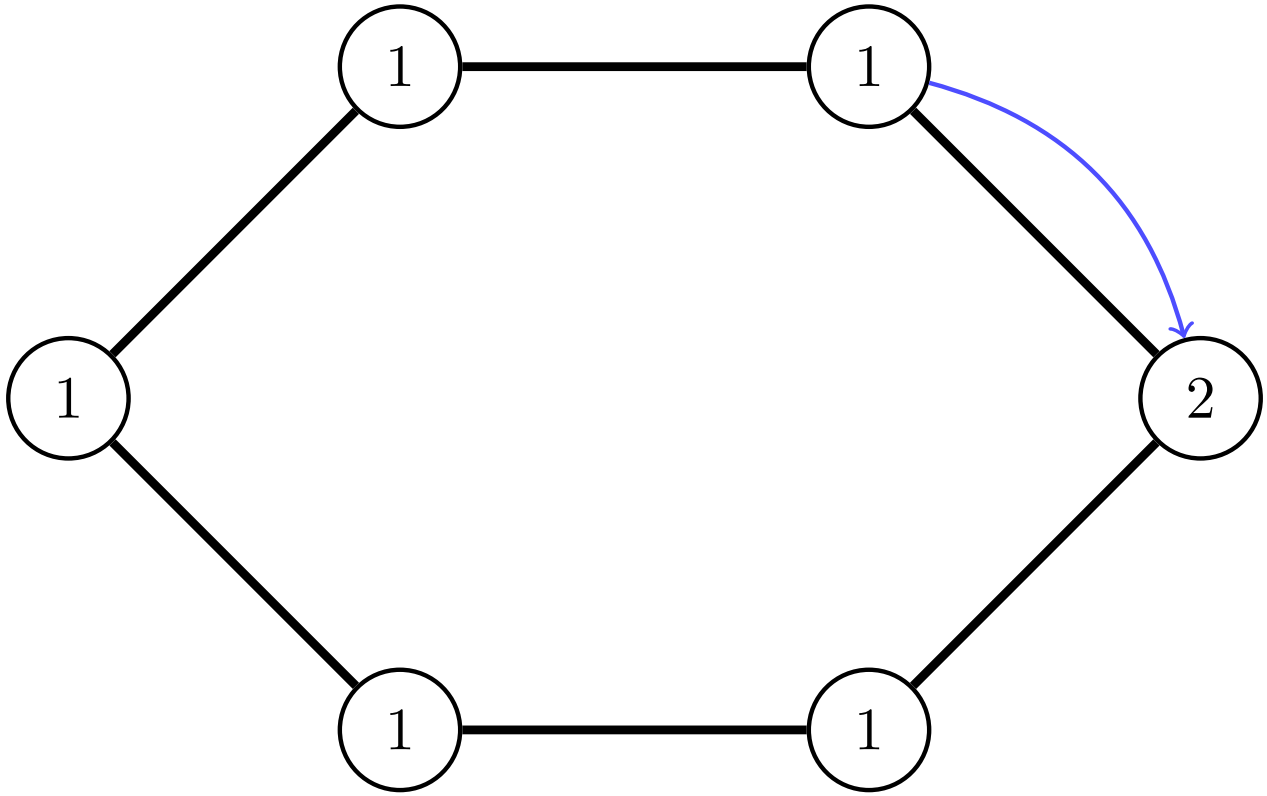
Beispiel

Runde 5:



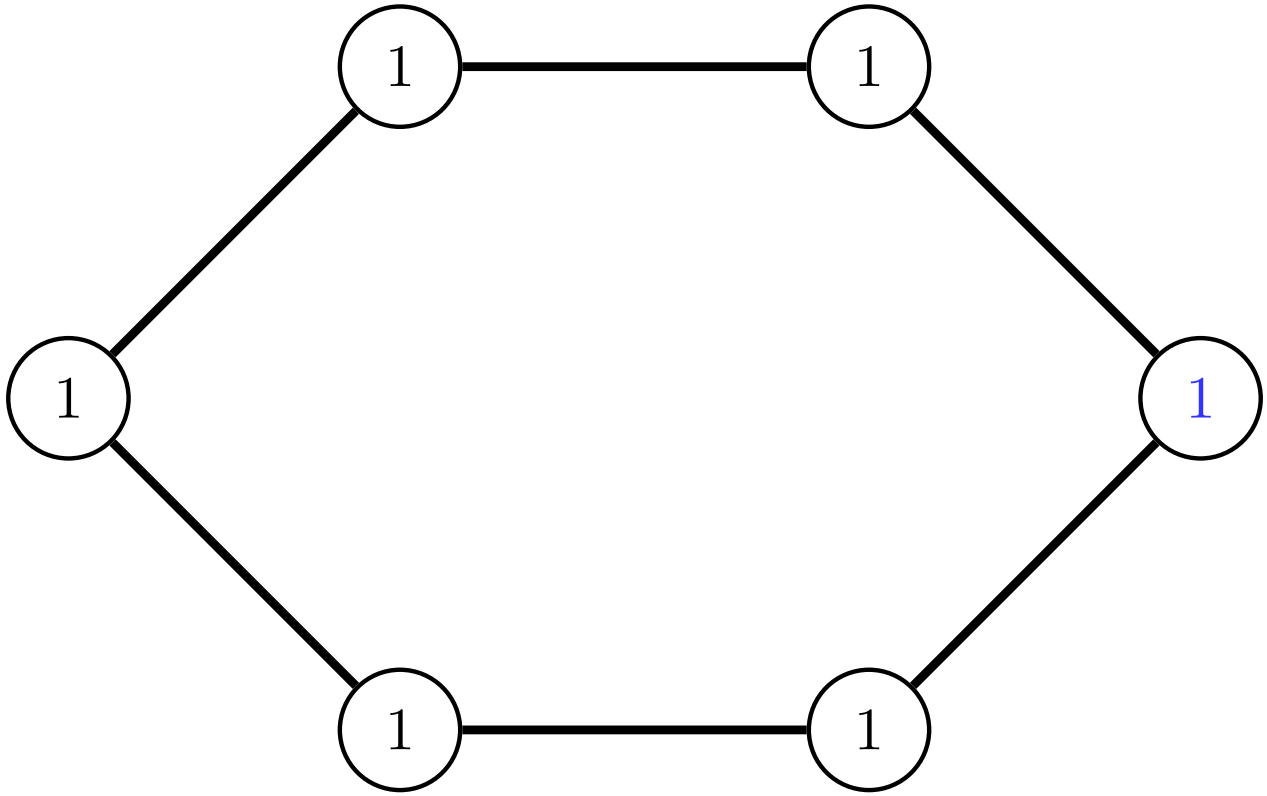
Beispiel

Runde 5:



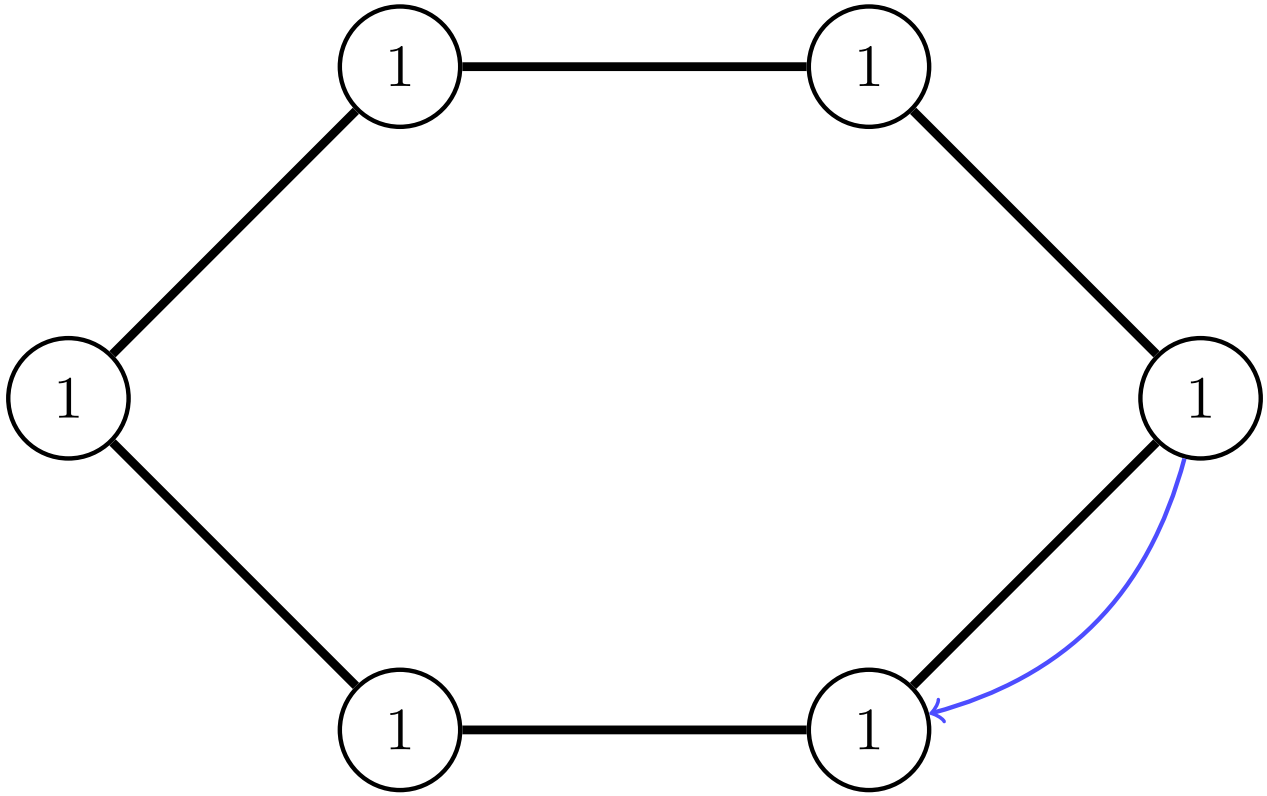
Beispiel

Runde 6:



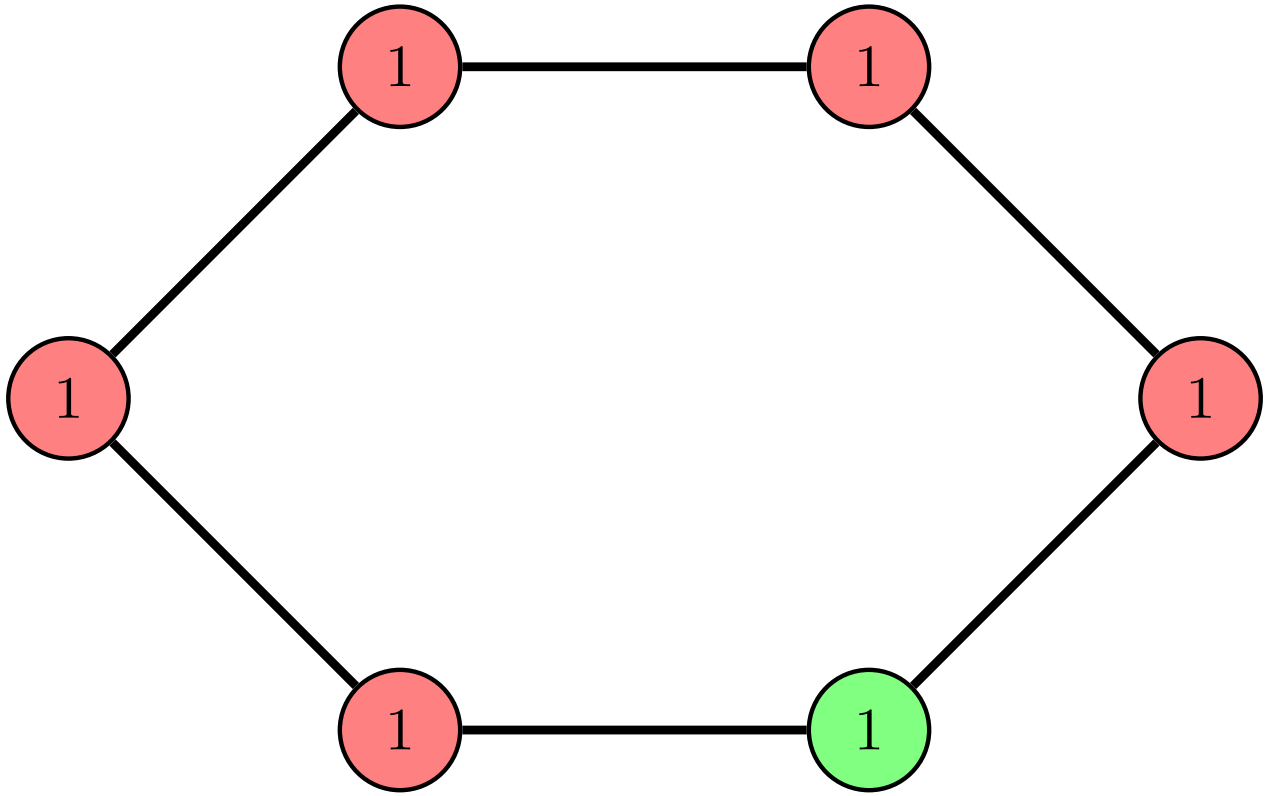
Beispiel

Runde 6:



Beispiel

Runde 7:



Laufzeit

Theorem

Nach $n + 1$ Runden bestimmt der einfache Clockwise Algorithmus den Knoten mit der kleinsten ID zum Leader und alle anderen zu Followern.

Beobachtung: Kleinste ID wird in jeder Runde weitergeleitet

Laufzeit

Theorem

Nach $n + 1$ Runden bestimmt der einfache Clockwise Algorithmus den Knoten mit der kleinsten ID zum Leader und alle anderen zu Followern.

Beobachtung: Kleinste ID wird in jeder Runde weitergeleitet

Beweis:

- Sei z Knoten mit kleinster ID; es gilt: $\mathrm{ID}(z) \leq \mathrm{ID}(v)$ für jeden Knoten v

Laufzeit

Theorem

Nach $n + 1$ Runden bestimmt der einfache Clockwise Algorithmus den Knoten mit der kleinsten ID zum Leader und alle anderen zu Followern.

Beobachtung: Kleinste ID wird in jeder Runde weitergeleitet

Beweis:

- Sei z Knoten mit kleinster ID; es gilt: $\mathrm{ID}(z) \leq \mathrm{ID}(v)$ für jeden Knoten v
- Für jedes $i \geq 0$, sei v_i der Knoten, der von z aus nach Traversieren von i Kanten im Uhrzeigersinn erreicht wird.

Laufzeit

Theorem

Nach $n + 1$ Runden bestimmt der einfache Clockwise Algorithmus den Knoten mit der kleinsten ID zum Leader und alle anderen zu Followern.

Beobachtung: Kleinste ID wird in jeder Runde weitergeleitet

Beweis:

- Sei z Knoten mit kleinster ID; es gilt: $\mathrm{ID}(z) \leq \mathrm{ID}(v)$ für jeden Knoten v
- Für jedes $i \geq 0$, sei v_i der Knoten, der von z aus nach Traversieren von i Kanten im Uhrzeigersinn erreicht wird.
- **Induktionsaussage:** Der Knoten v_i empfängt die Nachricht $\mathrm{ID}(z)$ (spätestens) in Runde $i + 1$ (für jedes $0 \leq i \leq n$).

Laufzeit

Theorem

Nach $n+1$ Runden bestimmt der einfache Clockwise Algorithmus den Knoten mit der kleinsten ID zum Leader und alle anderen zu Followern.

Beobachtung: Kleinste ID wird in jeder Runde weitergeleitet

Beweis:

- Sei z Knoten mit kleinster ID; es gilt: $\mathrm{ID}(z) \leq \mathrm{ID}(v)$ für jeden Knoten v
- Für jedes $i \geq 0$, sei v_i der Knoten, der von z aus nach Traversieren von i Kanten im Uhrzeigersinn erreicht wird.
- **Induktionsaussage:** Der Knoten v_i empfängt die Nachricht $\mathrm{ID}(z)$ (spätestens) in Runde $i+1$ (für jedes $0 \leq i \leq n$).
- **Konsequenz:**
  - Beobachtung: Jeder von L_v angenommene Wert entspricht einer ID im Netzwerk
  - Sei $v \neq z$. Dann ist $v = v_i$ für ein $i \leq n-1$ und innerhalb von $i+1 \leq n$ Runden empfängt v_i Nachricht $\mathrm{ID}(z)$ und ab dieser Runde gilt $L_v = \mathrm{ID}(z)$
  - Am Ende von Runde n gilt $L_v = \mathrm{ID}(z)$ für jeden Knoten v
  - Somit: In Runde $n+1$ wird z zum Leader alle anderen zu Followern

Nachrichtenkomplexität

Theorem

Der Clockwise Algorithmus versendet insgesamt höchstens n^2 Nachrichten.

Nachrichtenkomplexität

Theorem

Der Clockwise Algorithmus versendet insgesamt höchstens n^2 Nachrichten.

Beweis:

- Knoten v sendet nur, nachdem L_v initialisiert oder verringert wurde.

Nachrichtenkomplexität

Theorem

Der Clockwise Algorithmus versendet insgesamt höchstens n^2 Nachrichten.

Beweis:

- Knoten v sendet nur, nachdem L_v initialisiert oder verringert wurde.
- Jeder von L_v angenommene Wert entspricht einer ID im Netzwerk.

Nachrichtenkomplexität

Theorem

Der Clockwise Algorithmus versendet insgesamt höchstens n^2 Nachrichten.

Beweis:

- Knoten v sendet nur, nachdem L_v initialisiert oder verringert wurde.
- Jeder von L_v angenommene Wert entspricht einer ID im Netzwerk.
- Daher kann L_v während des Algorithmus höchstens n verschiedene Werte annehmen.

Nachrichtenkomplexität

Theorem

Der Clockwise Algorithmus versendet insgesamt höchstens n^2 Nachrichten.

Beweis:

- Knoten v sendet nur, nachdem L_v initialisiert oder verringert wurde.
- Jeder von L_v angenommene Wert entspricht einer ID im Netzwerk.
- Daher kann L_v während des Algorithmus höchstens n verschiedene Werte annehmen.
- Somit sendet jeder Knoten höchstens n Nachrichten.

Allgemeiner Clockwise Algorithmus [LeLann '77, Chang/Roberts '79]

Annahmen: asynchron, identifizierbar, uniform

Allgemeiner Clockwise Algorithmus [LeLann '77, Chang/Roberts '79]

Annahmen: asynchron, identifizierbar, uniform

Zwei mögliche Events: Initialisierung oder Empfang einer Nachricht

Allgemeiner Clockwise Algorithmus [LeLann '77, Chang/Roberts '79]

Annahmen: asynchron, identifizierbar, uniform

Zwei mögliche Events: Initialisierung oder Empfang einer Nachricht

Jeder Knoten v führt folgenden Algorithmus aus:

1 **if** v *wird initialisiert* **then**
2 $\quad$ v setzt $L_v := \mathrm{ID}(v)$ (lokale Variable für kleinste bisher gesehene ID)
3 $\quad$ v sendet L_v an Nachbar im Uhrzeigersinn

Allgemeiner Clockwise Algorithmus [LeLann '77, Chang/Roberts '79]

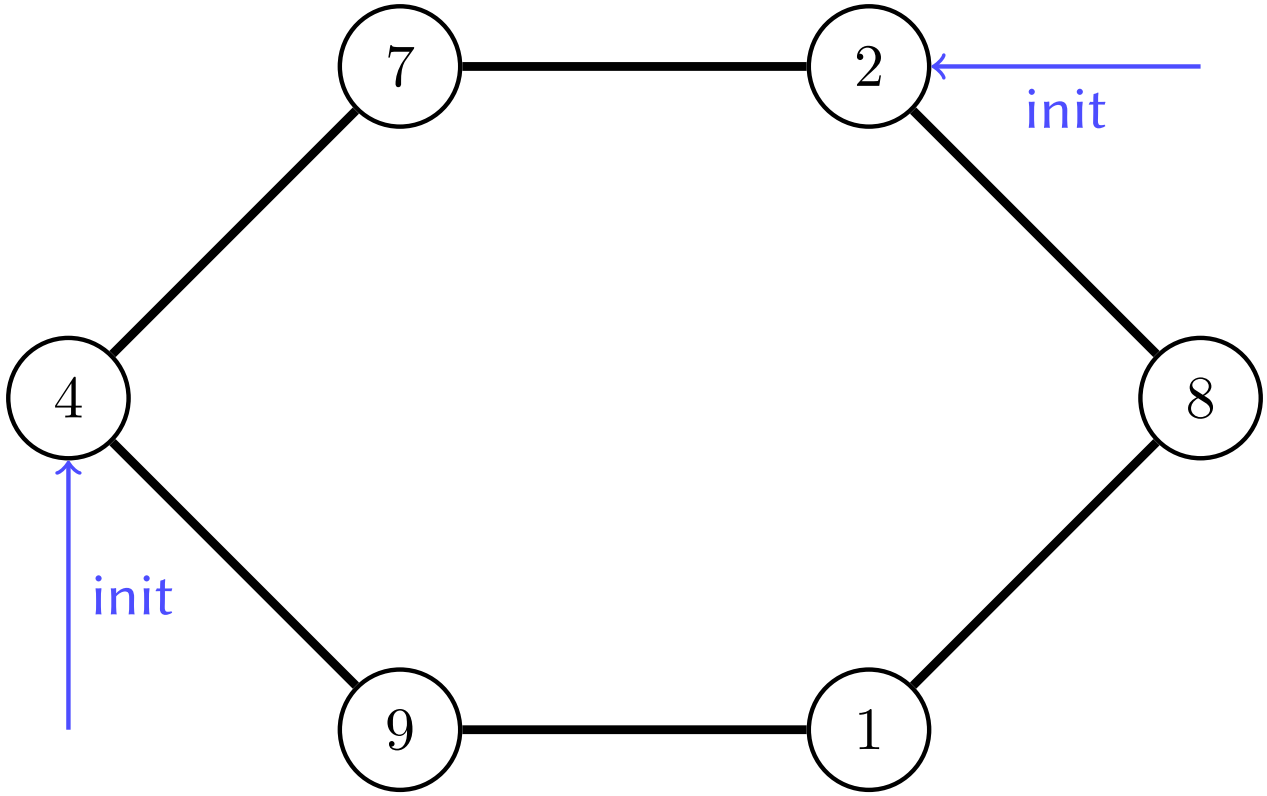
Annahmen: asynchron, identifizierbar, uniform

Zwei mögliche Events: Initialisierung oder Empfang einer Nachricht

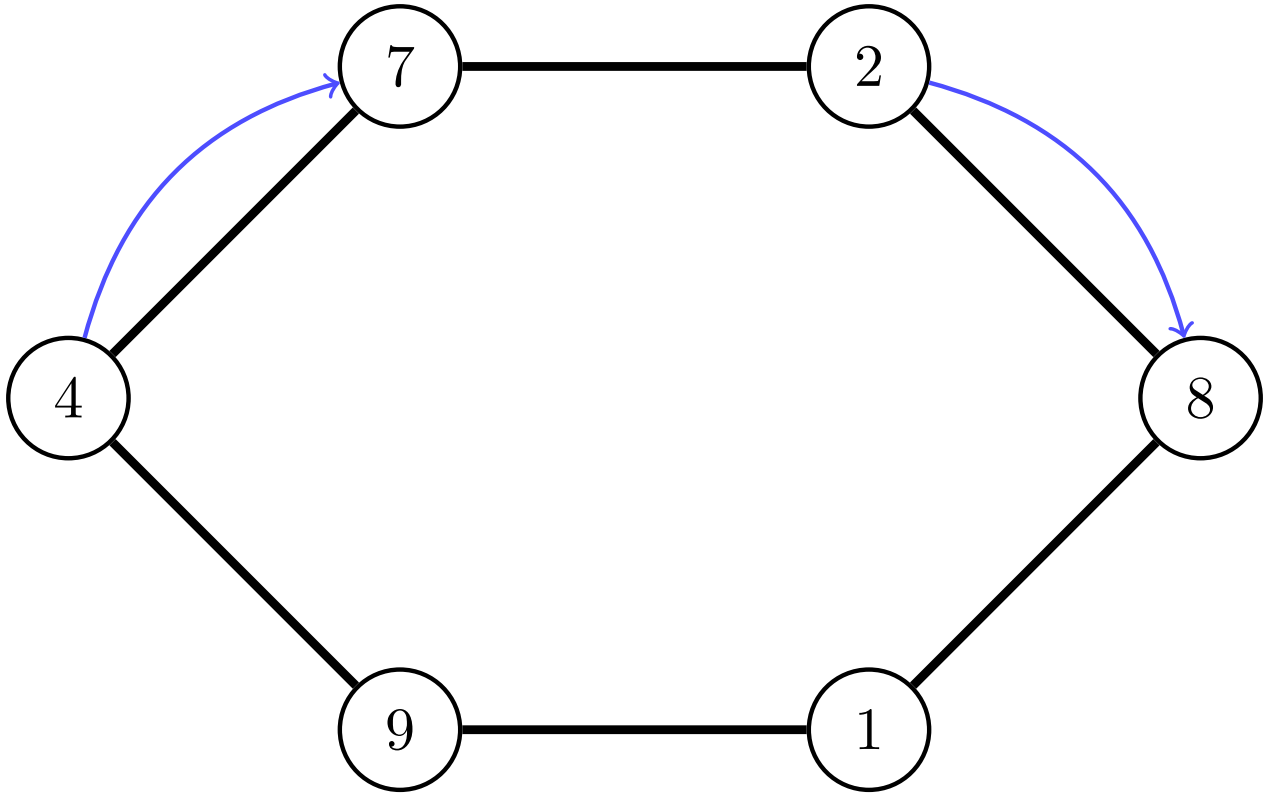
Jeder Knoten v führt folgenden Algorithmus aus:

1 **if** v *wird initialisiert* **then**
2 v setzt $L_v := \mathrm{ID}(v)$ (lokale Variable für kleinste bisher gesehene ID)
3 v sendet L_v an Nachbar im Uhrzeigersinn
4 **if** v *empfängt Nachricht* M **then**
5 **if** v *uninitialisiert* **then** initialisiere v
6 **if** $M < L_v$ **then**
7 v setzt $L_v := M$
8 v wird zum Follower (sofern nicht bereits vorher geschehen)
9 v sendet L_v an Nachbar im Uhrzeigersinn
10 **if** $M = \mathrm{ID}(v)$ **then**
11 v wird zum Leader

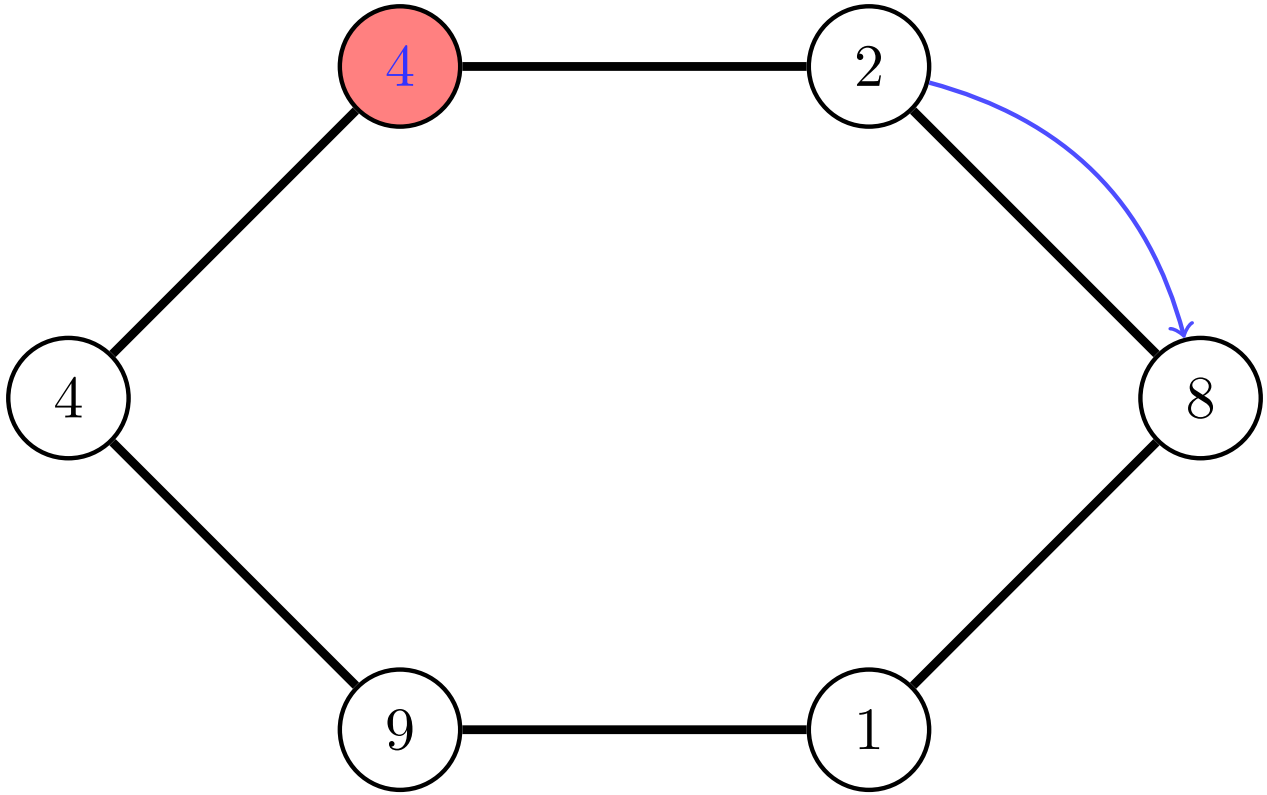
Beispiel



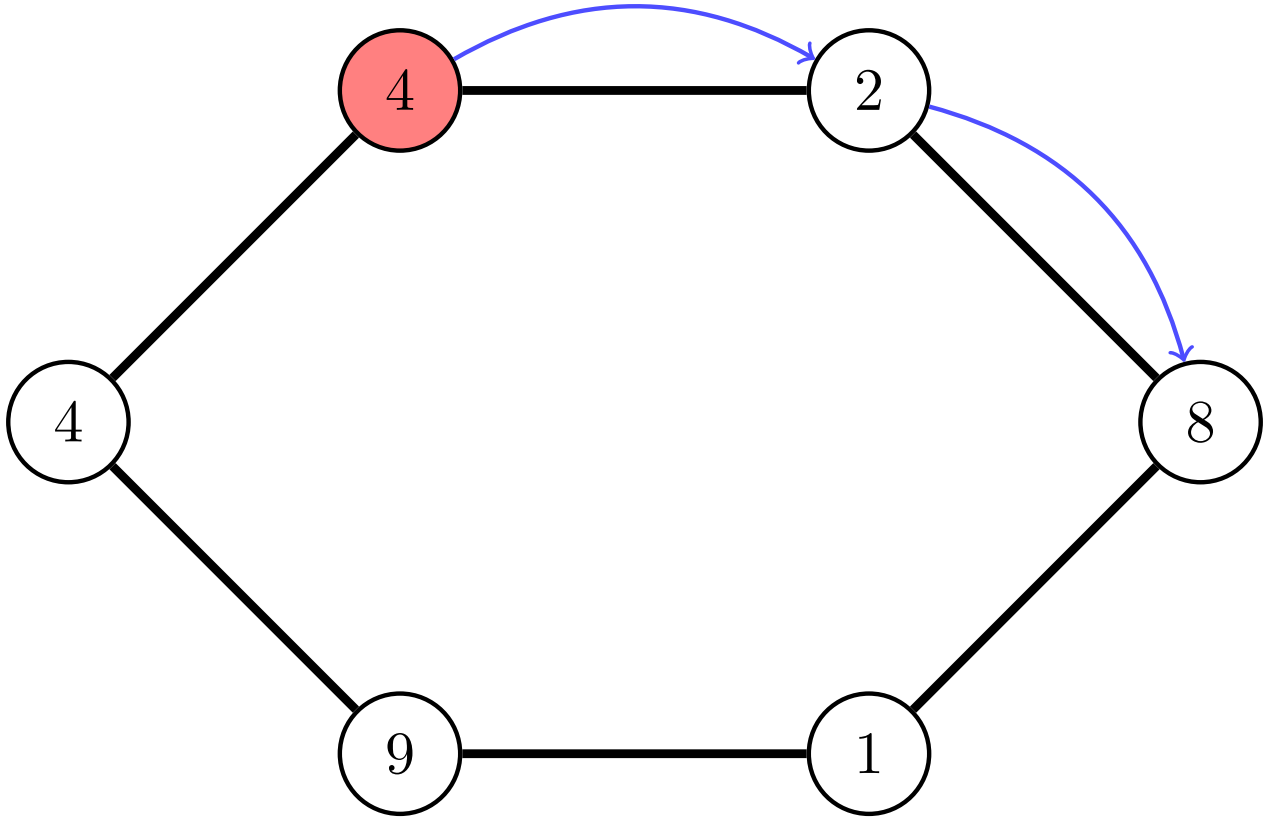
Beispiel



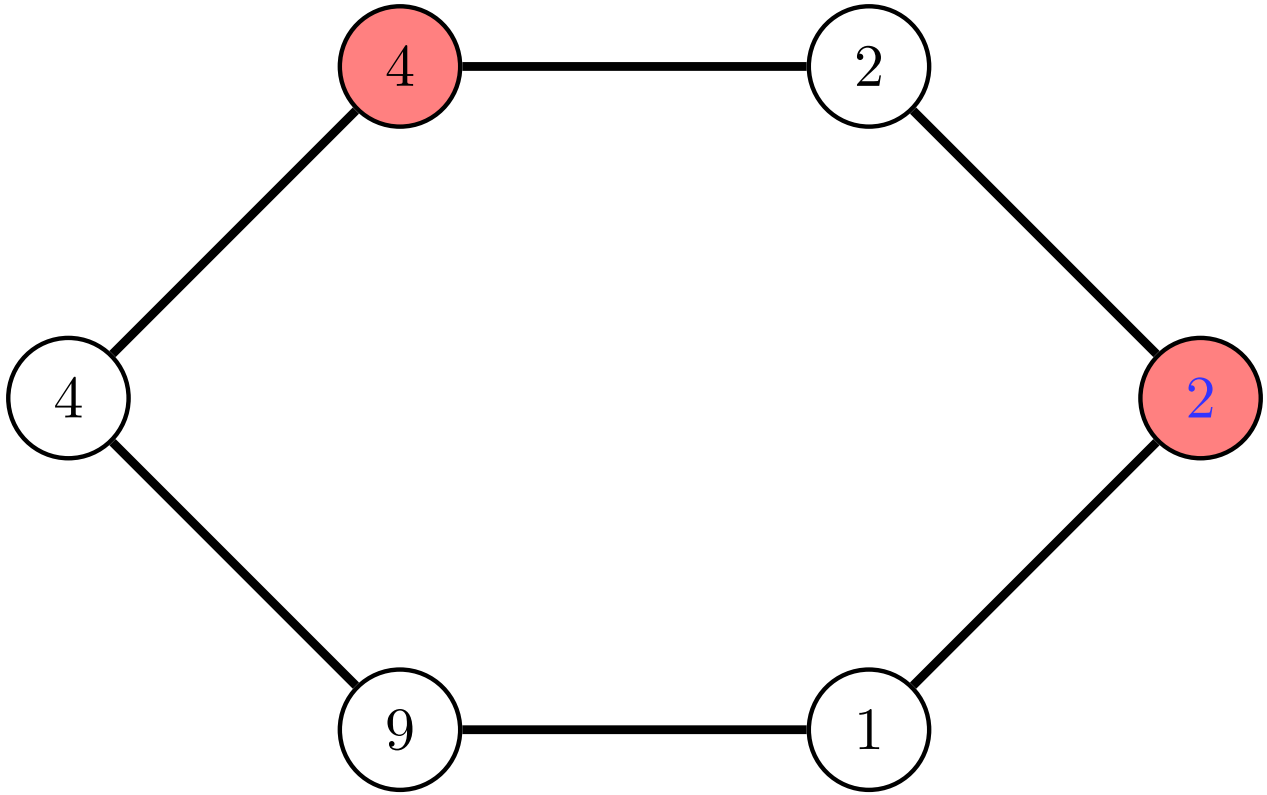
Beispiel



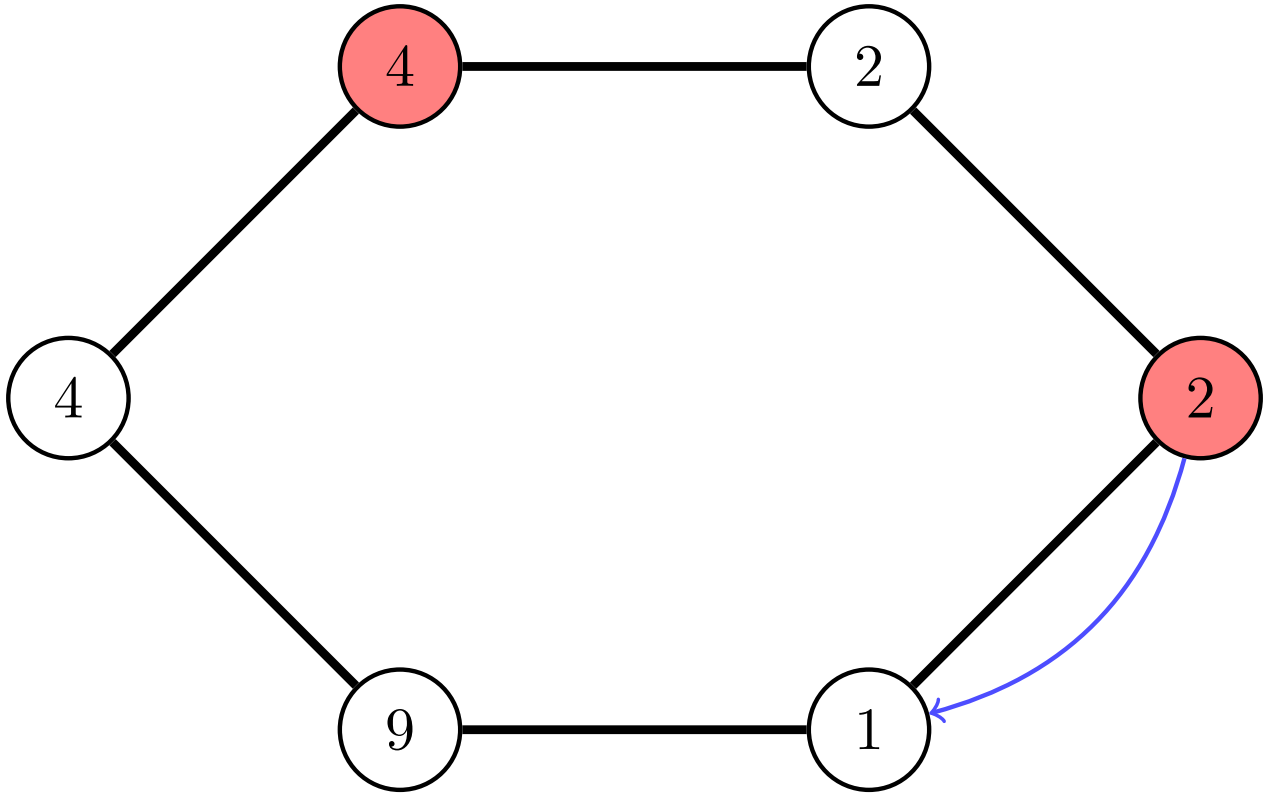
Beispiel



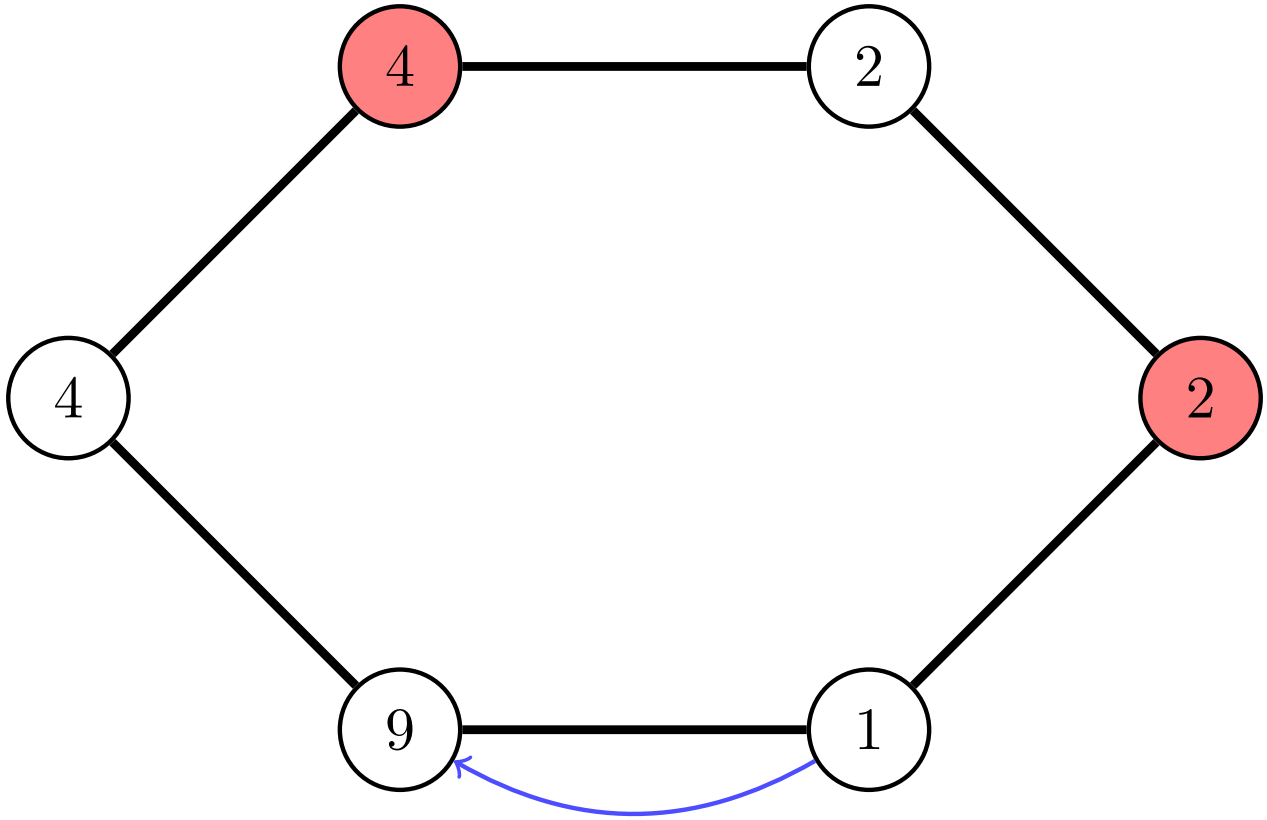
Beispiel



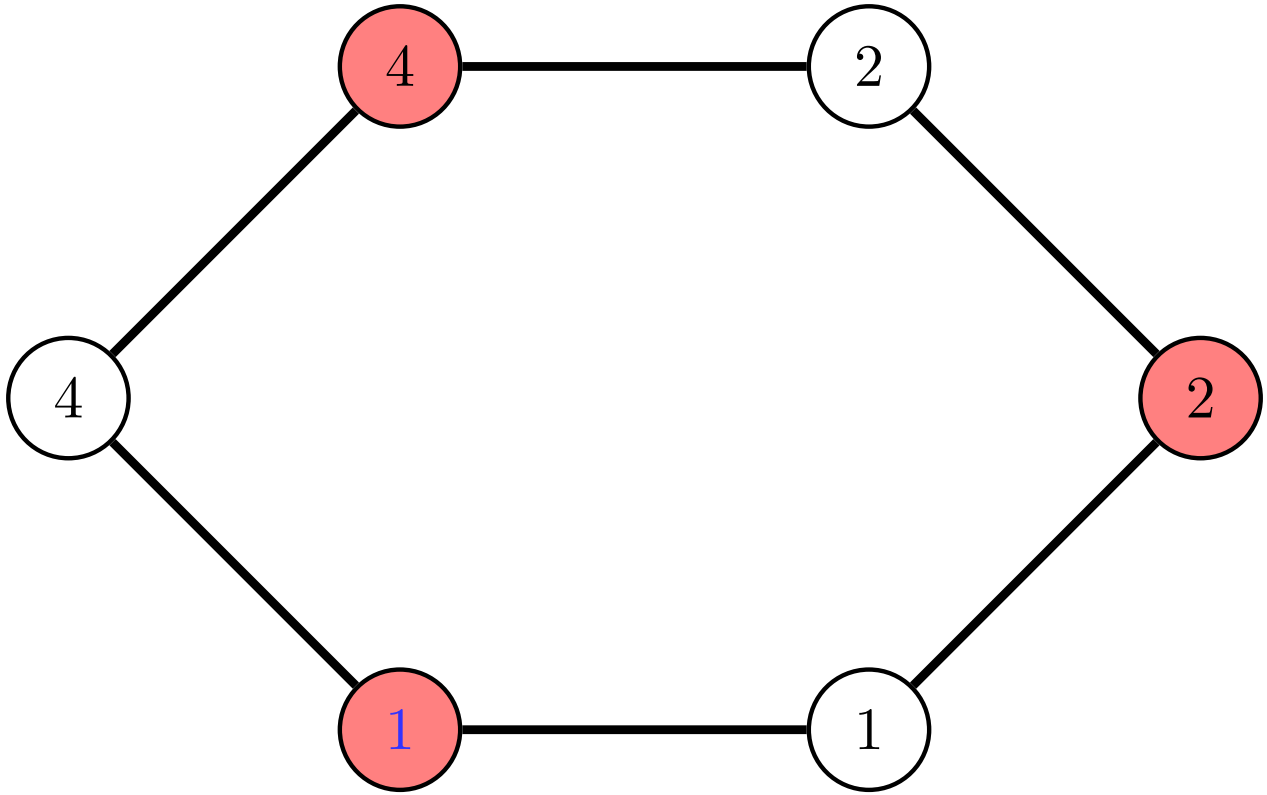
Beispiel



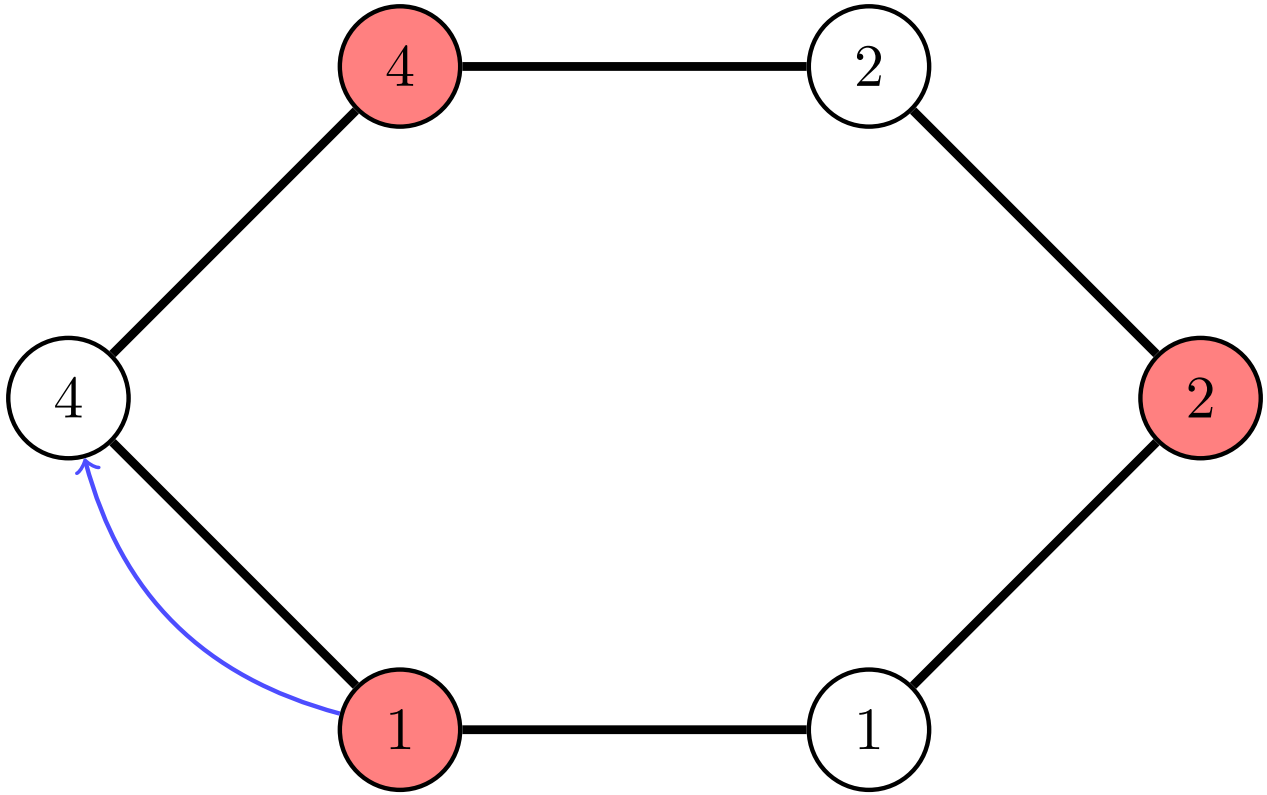
Beispiel



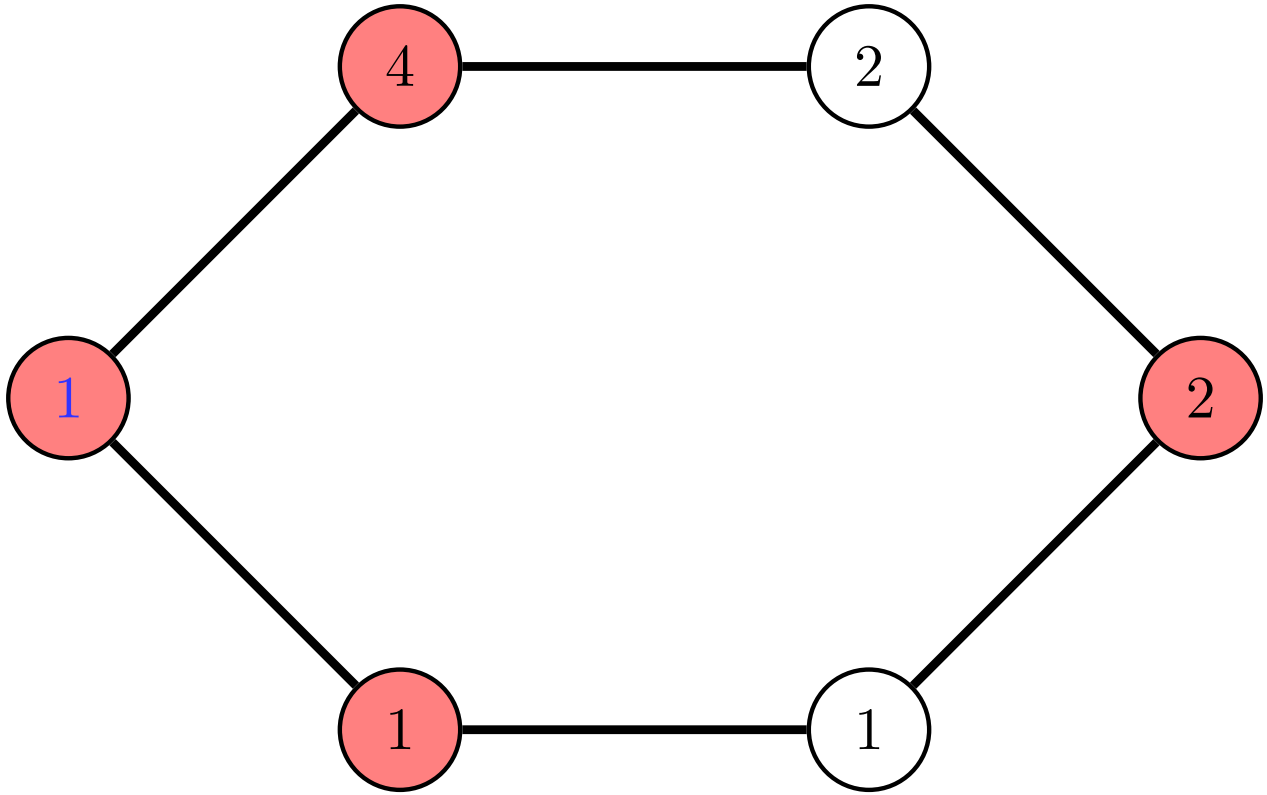
Beispiel



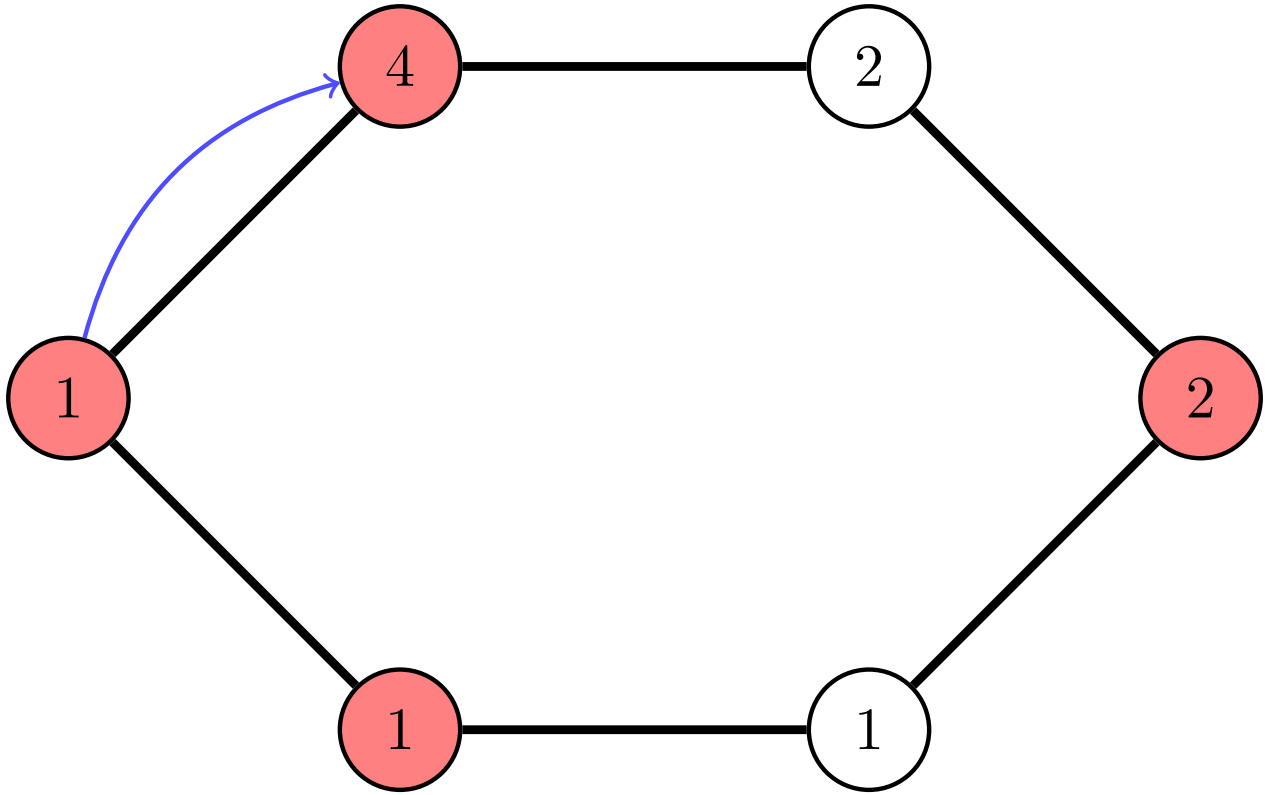
Beispiel



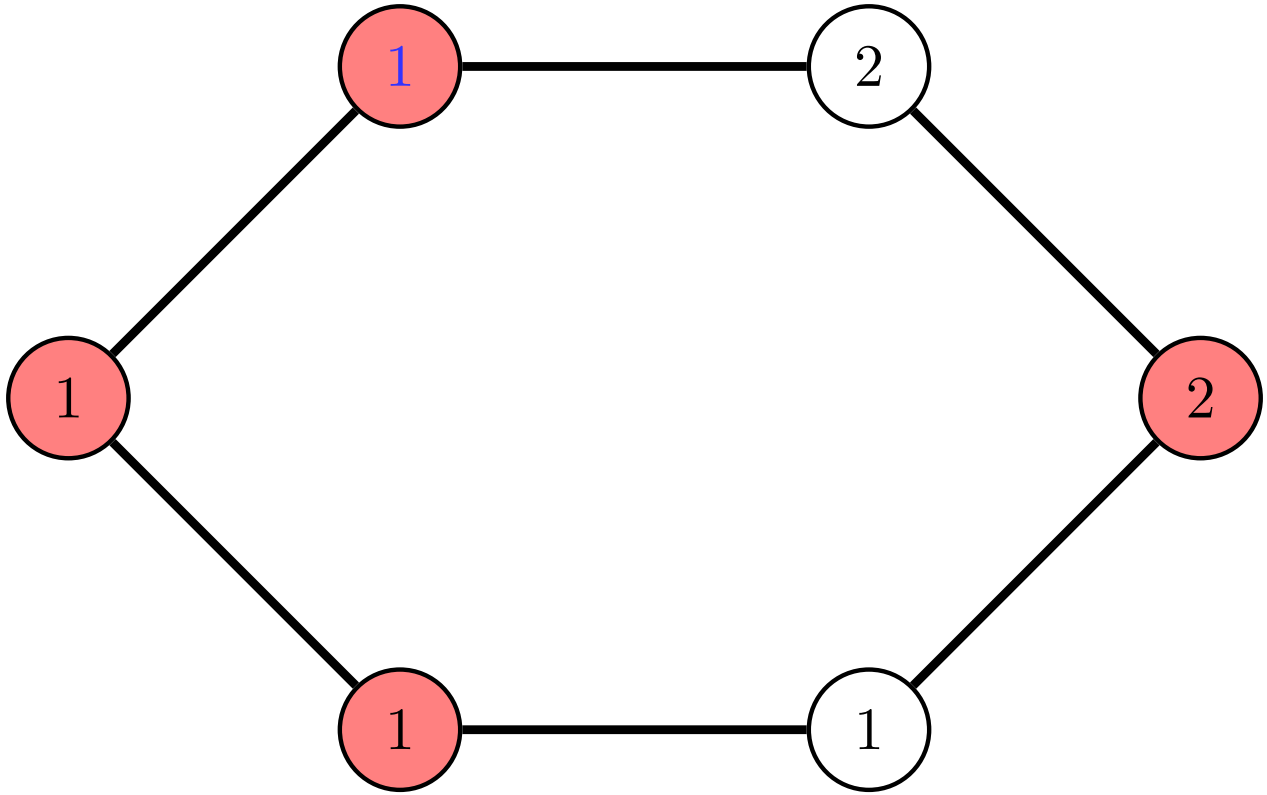
Beispiel



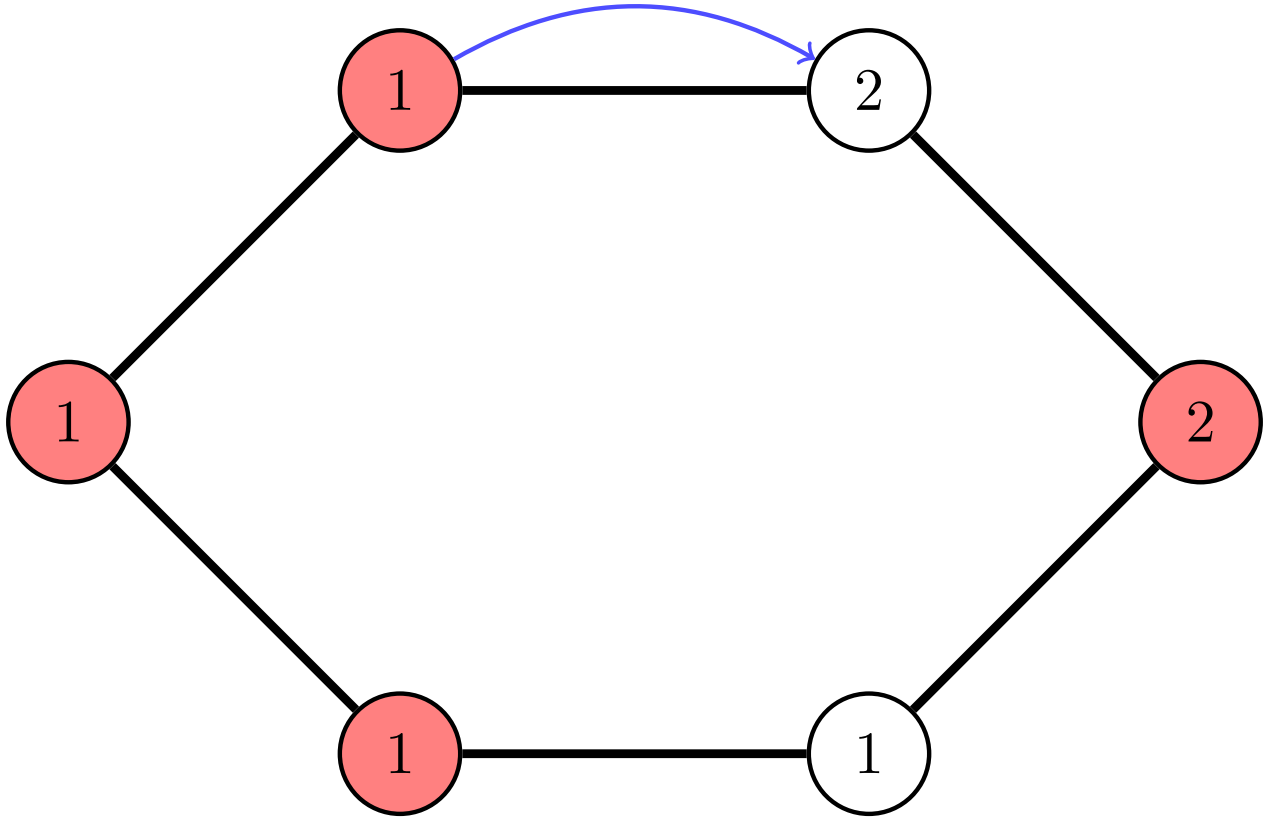
Beispiel



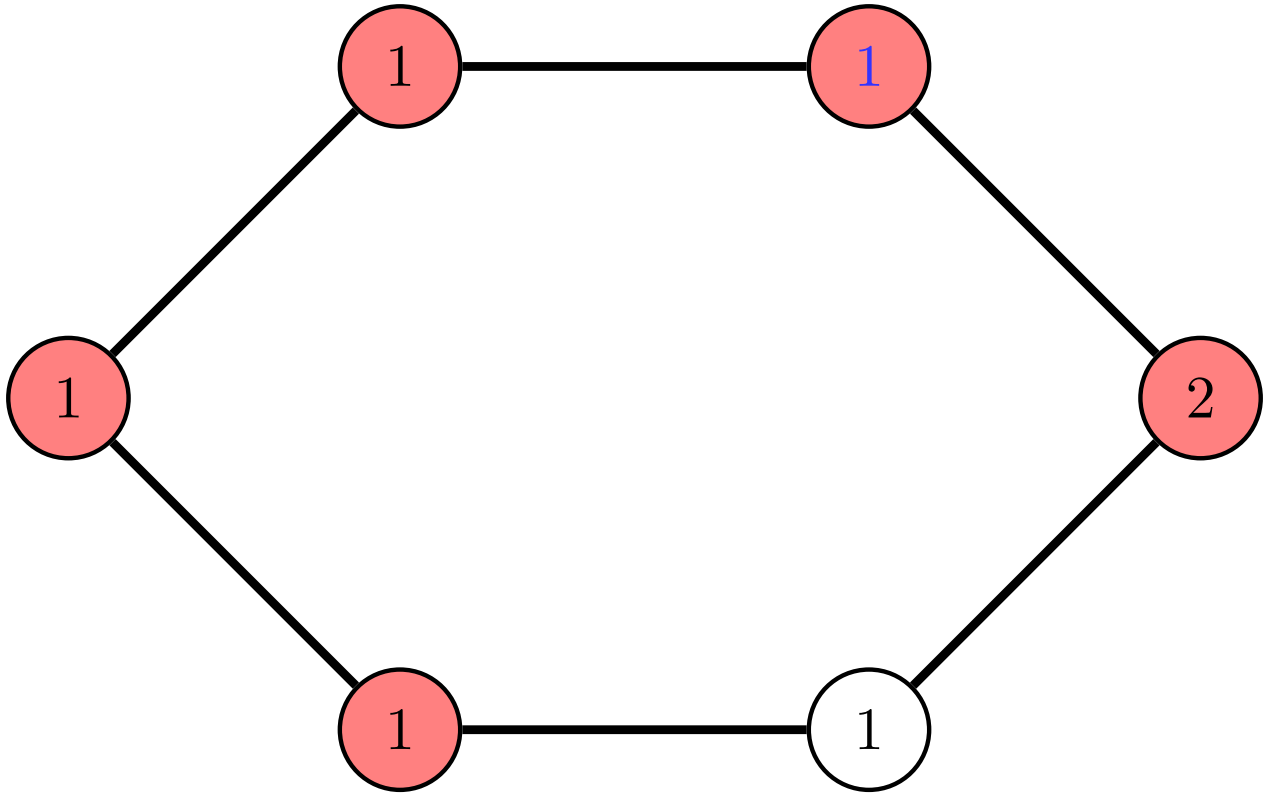
Beispiel



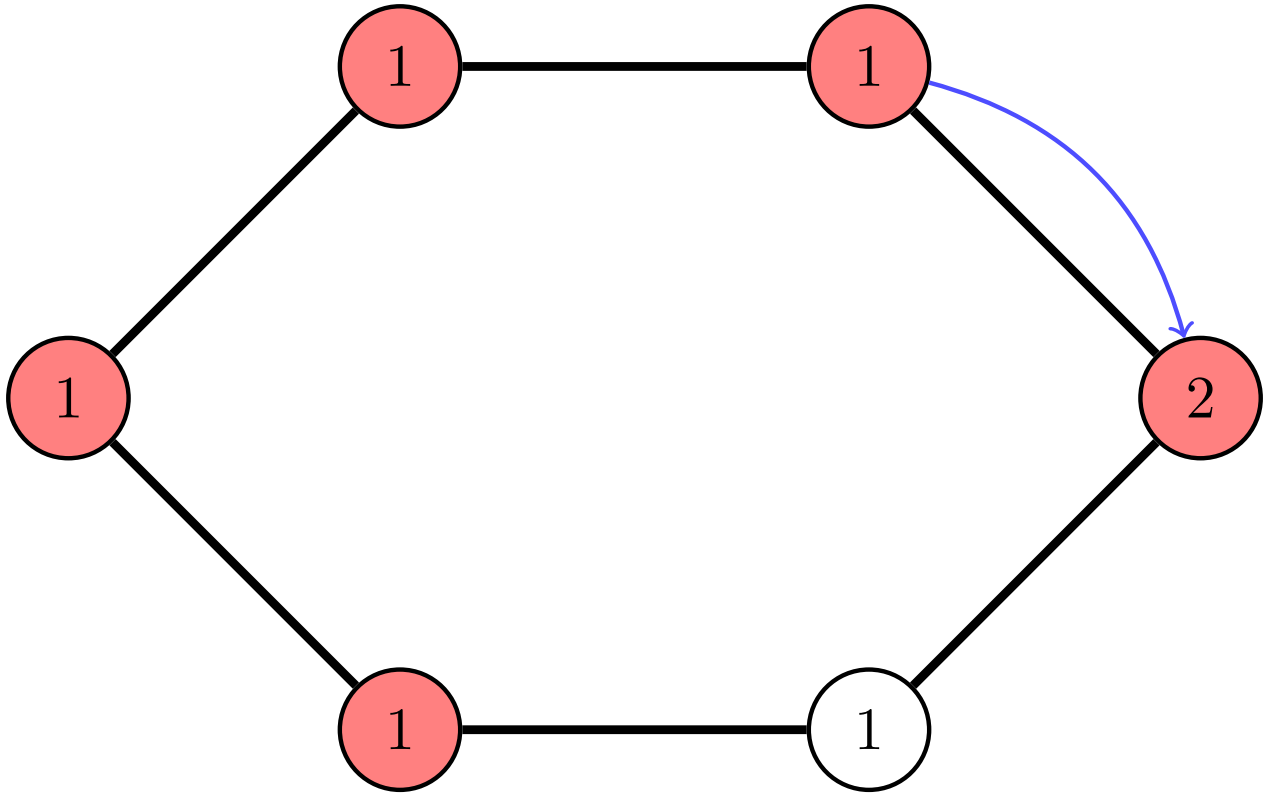
Beispiel



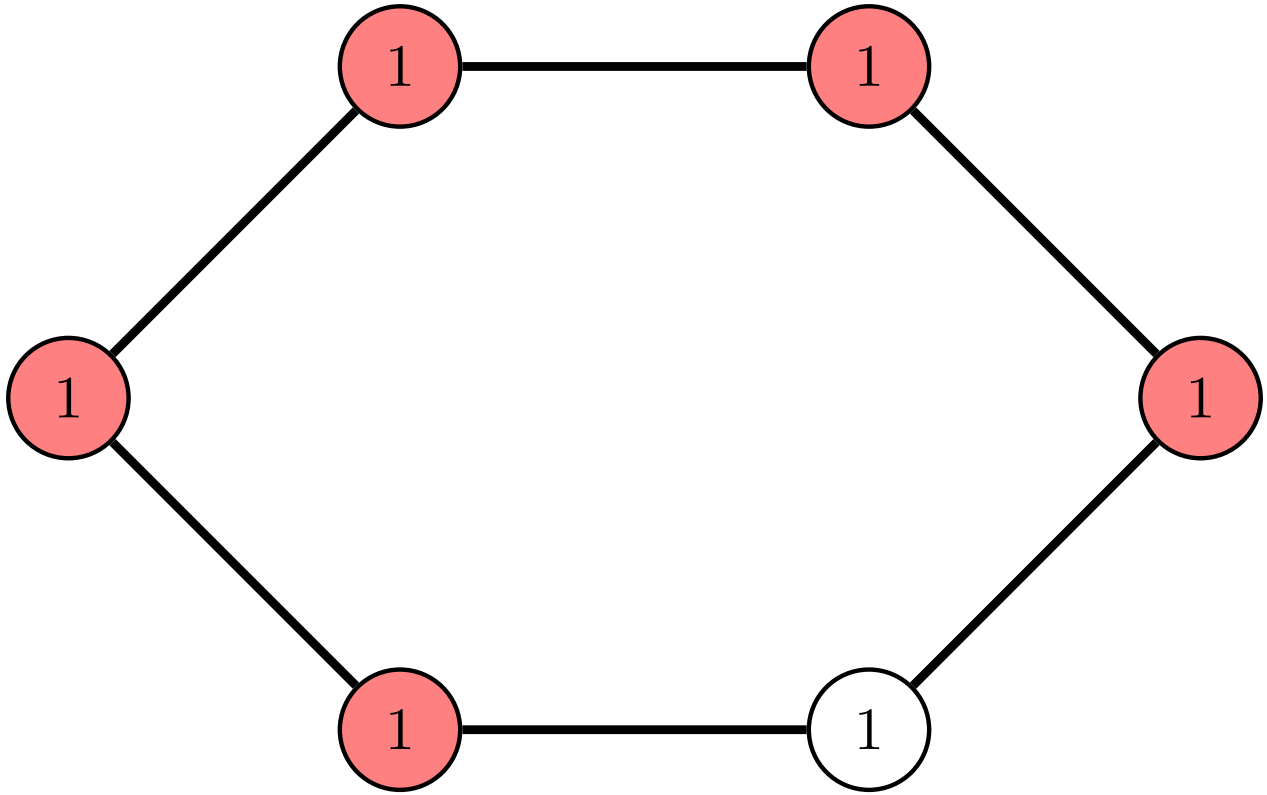
Beispiel



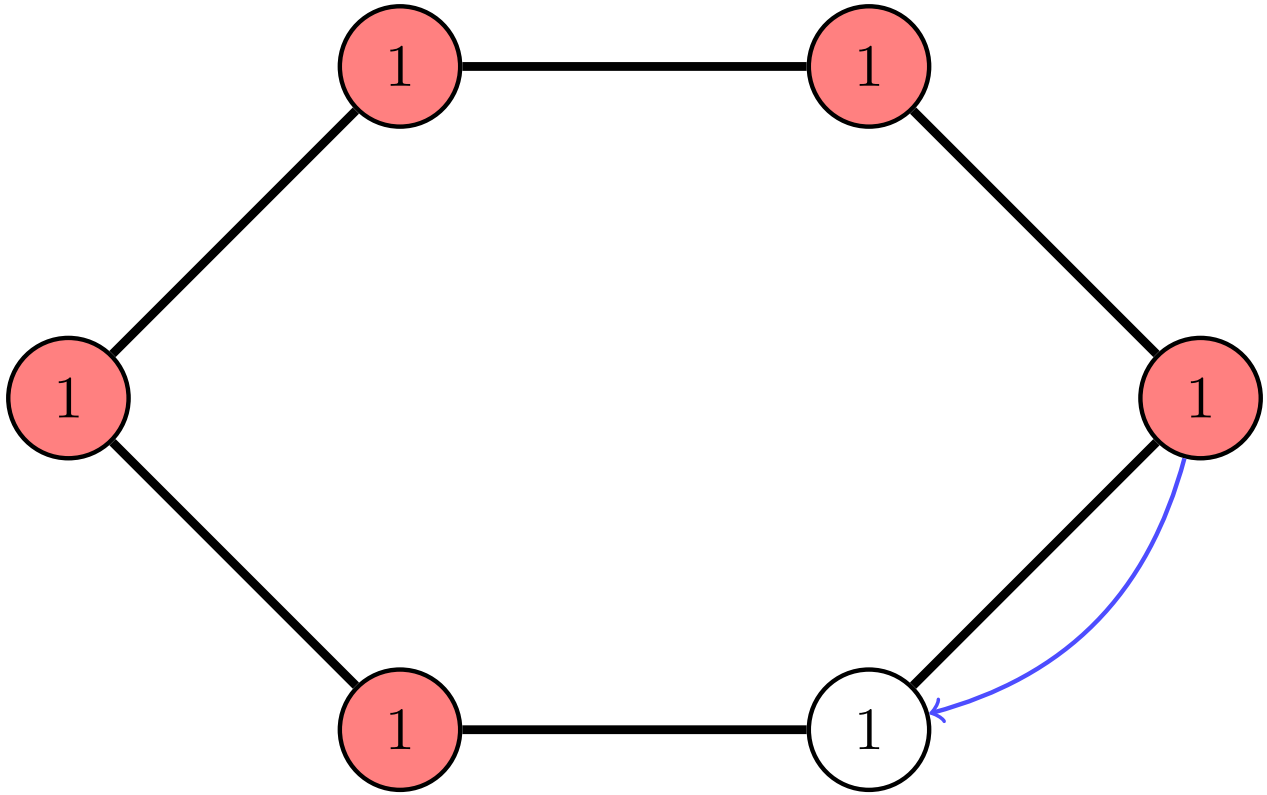
Beispiel



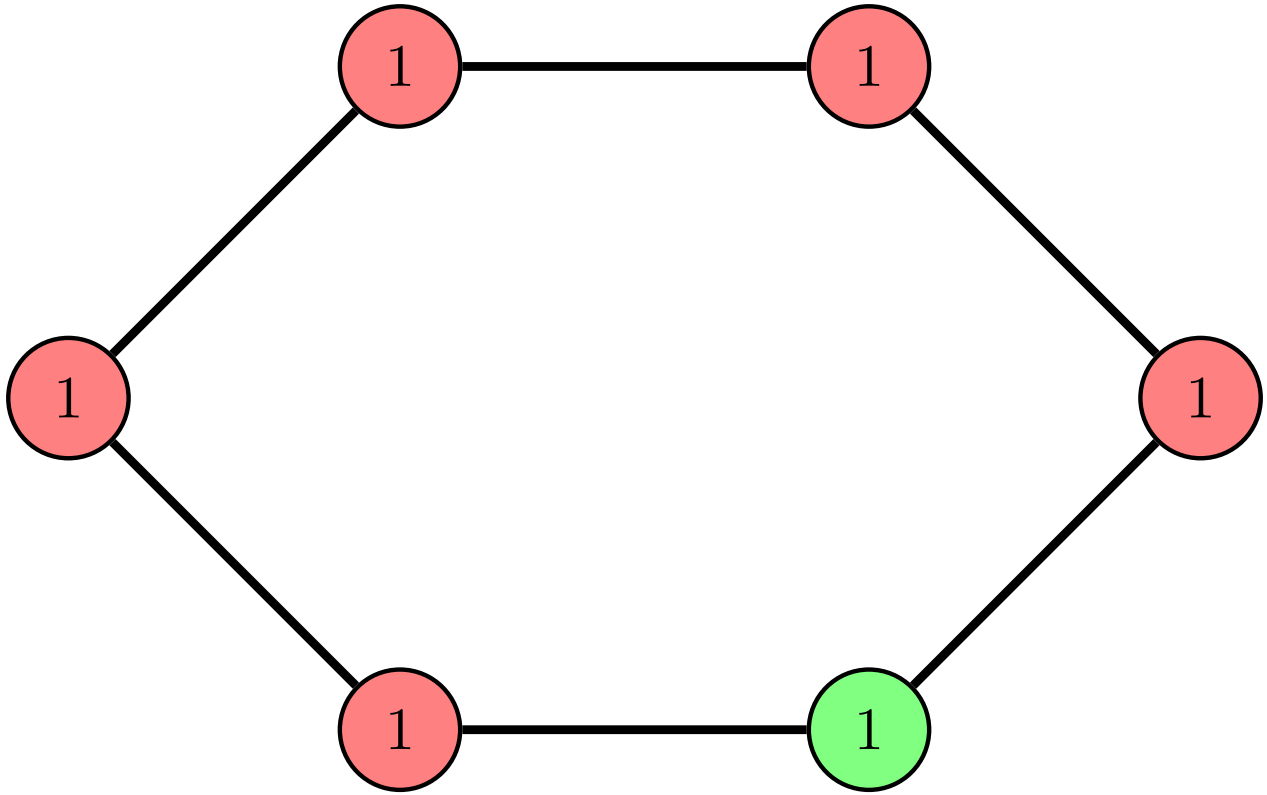
Beispiel



Beispiel



Beispiel



Analyse

Theorem

Nach höchstens $2n - 1$ Zeiteinheiten bestimmt der asynchrone Clockwise Algorithmus den Knoten mit der kleinsten ID zum Leader und alle anderen zu Followern (wenn die Übermittlung jeder Nachricht höchstens eine Zeiteinheit dauert).

Analyse

Theorem

Nach höchstens $2n - 1$ Zeiteinheiten bestimmt der asynchrone Clockwise Algorithmus den Knoten mit der kleinsten ID zum Leader und alle anderen zu Followern (wenn die Übermittlung jeder Nachricht höchstens eine Zeiteinheit dauert).

Beweisidee:

- Zeitmessung startet mit dem ersten Knoten, der initialisiert wird.

Analyse

Theorem

Nach höchstens $2n - 1$ Zeiteinheiten bestimmt der asynchrone Clockwise Algorithmus den Knoten mit der kleinsten ID zum Leader und alle anderen zu Followern (wenn die Übermittlung jeder Nachricht höchstens eine Zeiteinheit dauert).

Beweisidee:

- Zeitmessung startet mit dem ersten Knoten, der initialisiert wird.
- Dann vergehen höchstens $n - 1$ Zeiteinheit bis Knoten z mit kleinster ID eine Nachricht empfängt und das erste Mal aktiv wird.

Analyse

Theorem

Nach höchstens $2n - 1$ Zeiteinheiten bestimmt der asynchrone Clockwise Algorithmus den Knoten mit der kleinsten ID zum Leader und alle anderen zu Followern (wenn die Übermittlung jeder Nachricht höchstens eine Zeiteinheit dauert).

Beweisidee:

- Zeitmessung startet mit dem ersten Knoten, der initialisiert wird.
- Dann vergehen höchstens $n - 1$ Zeiteinheit bis Knoten z mit kleinster ID eine Nachricht empfängt und das erste Mal aktiv wird.
- Anschließend: Nach höchstens n weiteren Zeiteinheiten hat jeder Knoten $\mathrm{ID}(z)$ empfangen und sich entschieden.

Analyse

Theorem

Nach höchstens $2n - 1$ Zeiteinheiten bestimmt der asynchrone Clockwise Algorithmus den Knoten mit der kleinsten ID zum Leader und alle anderen zu Followern (wenn die Übermittlung jeder Nachricht höchstens eine Zeiteinheit dauert).

Beweisidee:

- Zeitmessung startet mit dem ersten Knoten, der initialisiert wird.
- Dann vergehen höchstens $n - 1$ Zeiteinheit bis Knoten z mit kleinster ID eine Nachricht empfängt und das erste Mal aktiv wird.
- Anschließend: Nach höchstens n weiteren Zeiteinheiten hat jeder Knoten $\mathrm{ID}(z)$ empfangen und sich entschieden.

Nachrichtenkomplexität: $O(n^2)$
Gleiches Argument wie für synchronen Algorithmus

Optimale Nachrichtenkomplexität

Frage

Kann ein Leader mit signifikant weniger als $\Theta(n^2)$ Nachrichten bestimmt werden?

Optimale Nachrichtenkomplexität

Frage

Kann ein Leader mit signifikant weniger als $\Theta(n^2)$ Nachrichten bestimmt werden?

„Verantwortlich ist man nicht nur, für das, was man tut, sondern auch für das, was man nicht tut.“

(Laozi)

Optimale Nachrichtenkomplexität

Frage

Kann ein Leader mit signifikant weniger als $\Theta(n^2)$ Nachrichten bestimmt werden?

„Verantwortlich ist man nicht nur, für das, was man tut, sondern auch für das, was man nicht tut.“

(Laozi)

Idee: *Keine* Nachricht zu senden darf als Zustimmung interpretiert werden

Wartezeit-Algorithmus [Attiya/Welch '98]

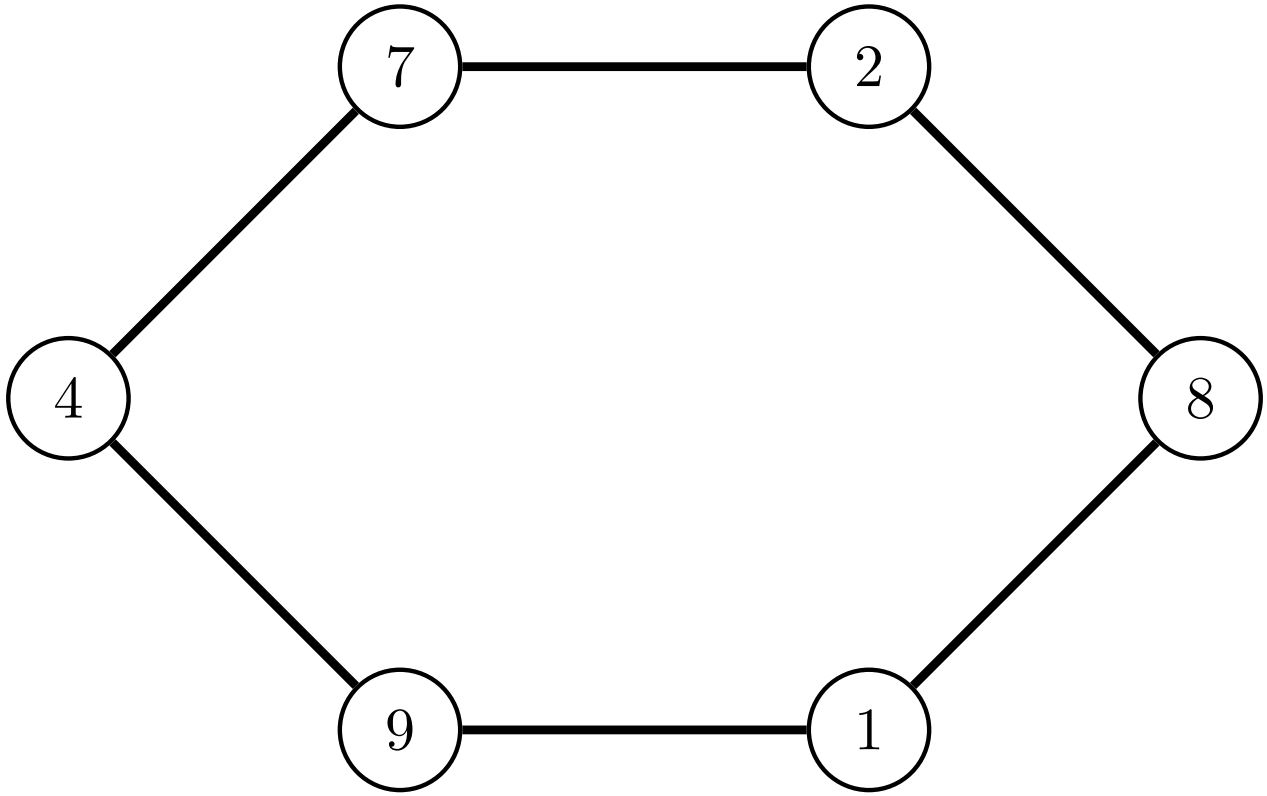
Annahmen: synchron, identifizierbar, non-uniform

Jeder Knoten v führt folgenden Algorithmus in jeder Runde aus:

```
1 if Leader-Nachricht empfangen und noch kein Leader then
2     Werde zum Follower
3     Leite Leader-Nachricht im Uhrzeigersinn weiter
4 if #Runden = ID(v) * n + 1 und v noch kein Follower then
5     Werde zum Leader
6     Sende Leader-Nachricht an Nachbar im Uhrzeigersinn
```

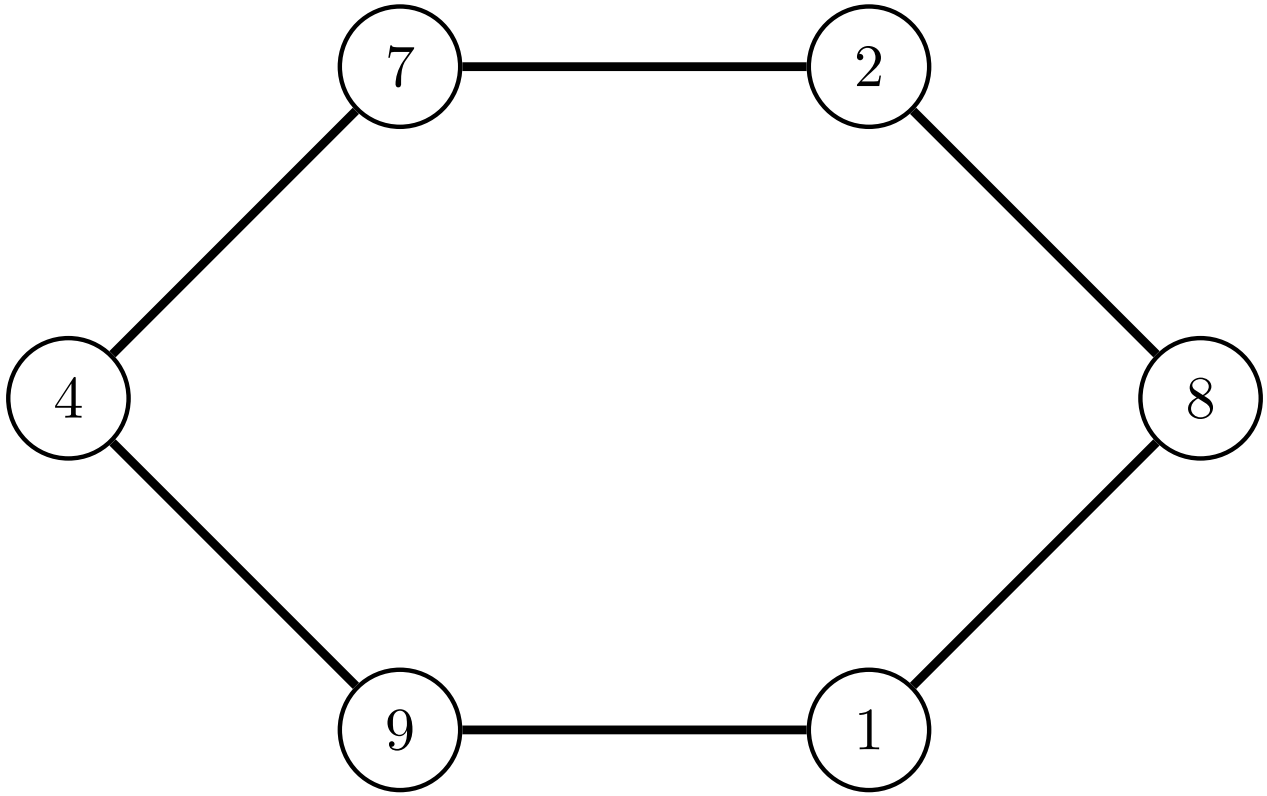
Beispiel

Runde 1:



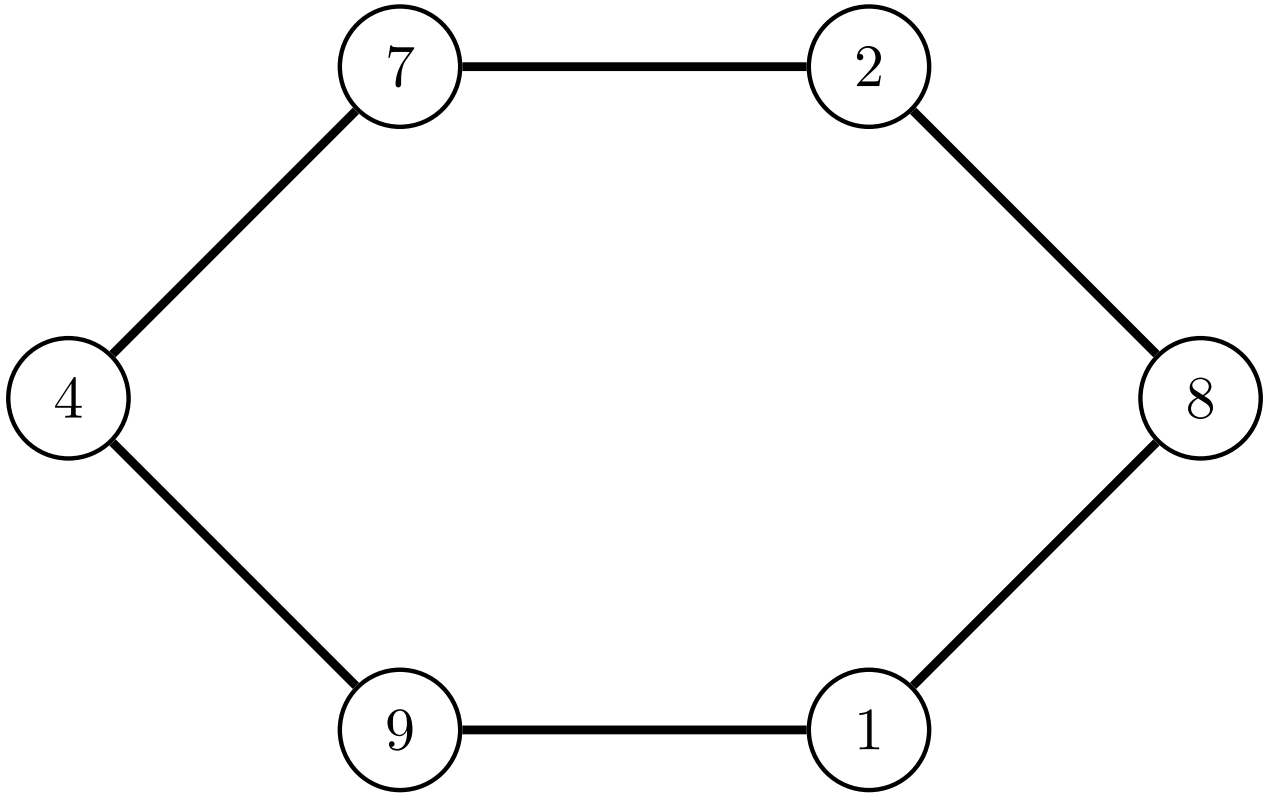
Beispiel

Runde 2:



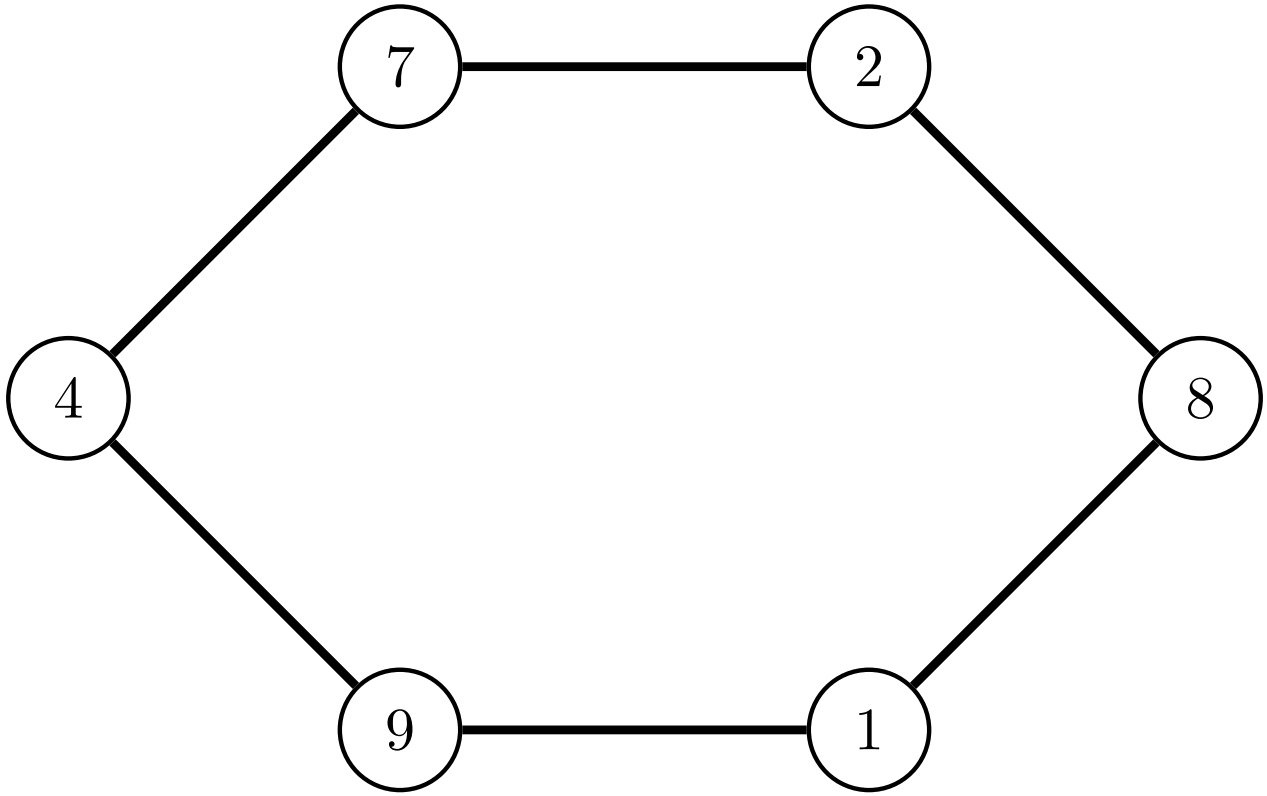
Beispiel

Runde 3:



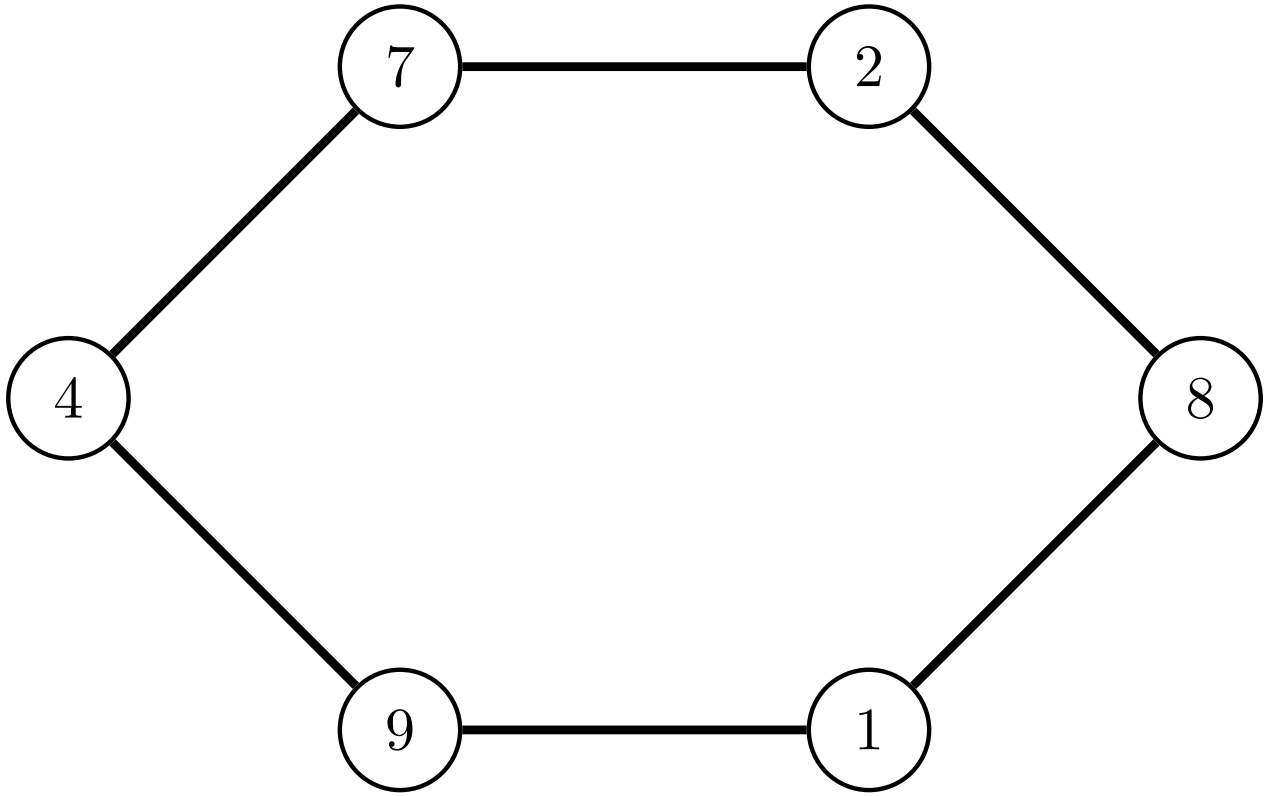
Beispiel

Runde 4:



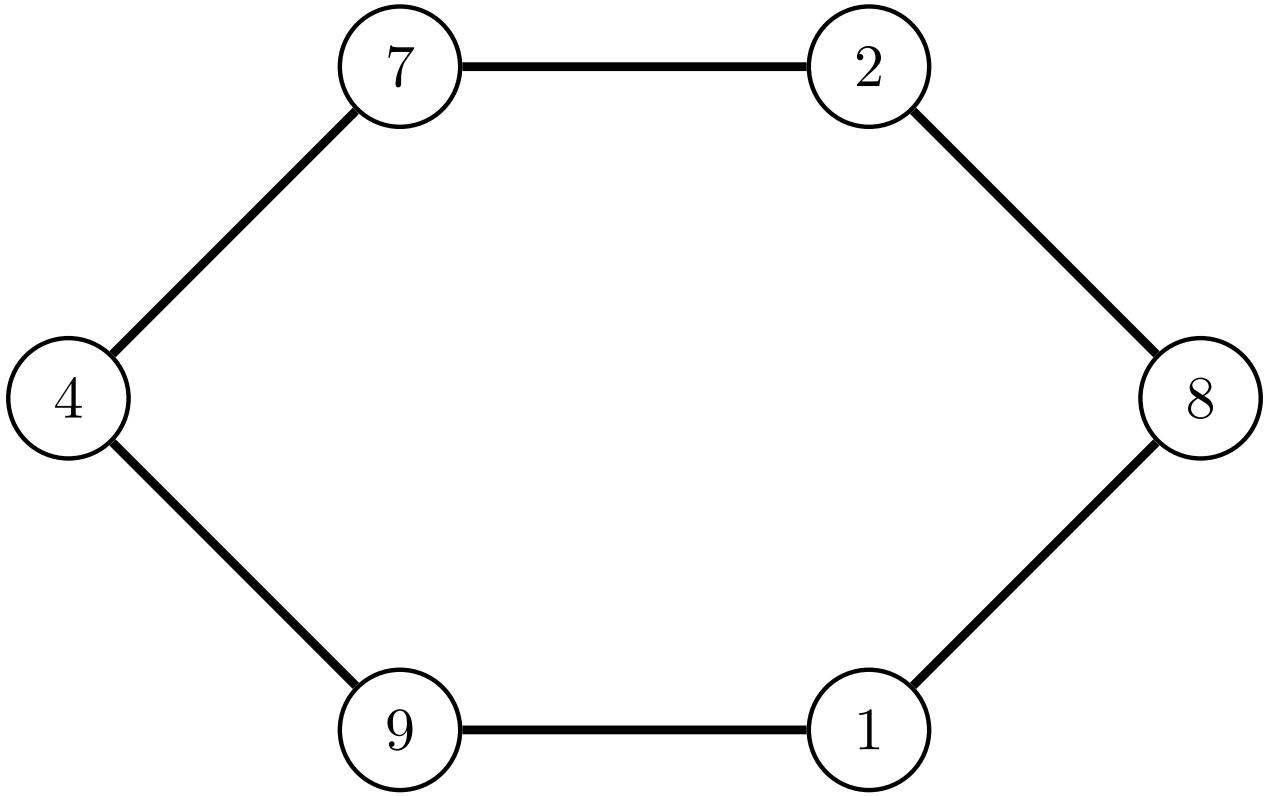
Beispiel

Runde 5:



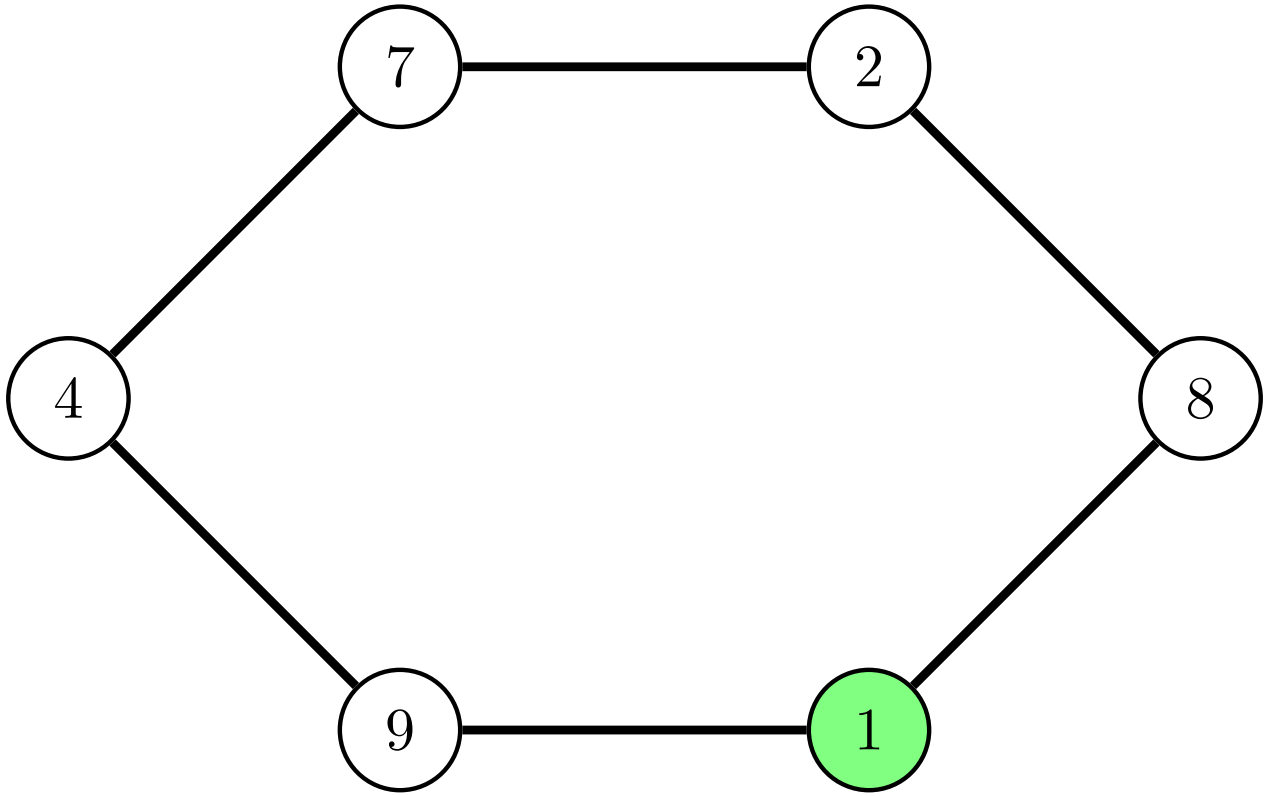
Beispiel

Runde 6:



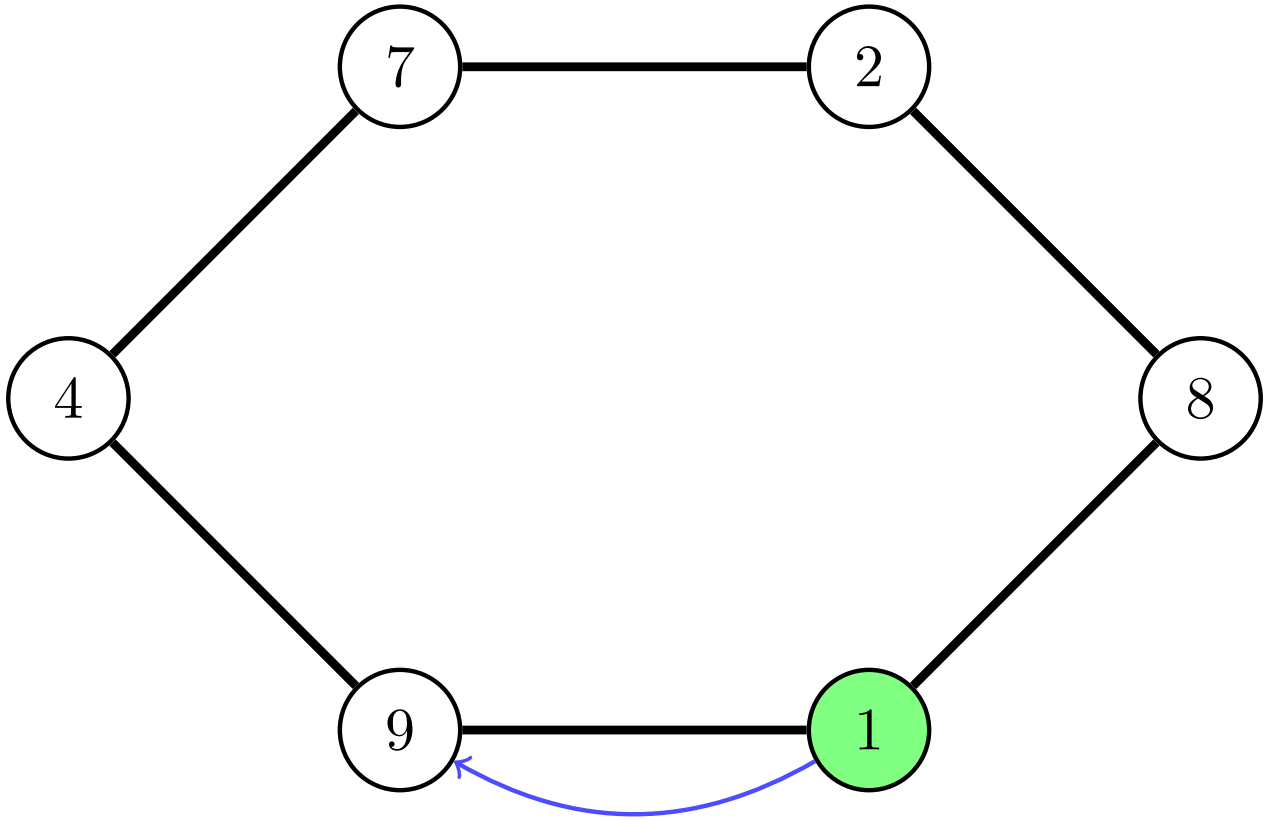
Beispiel

Runde 7:



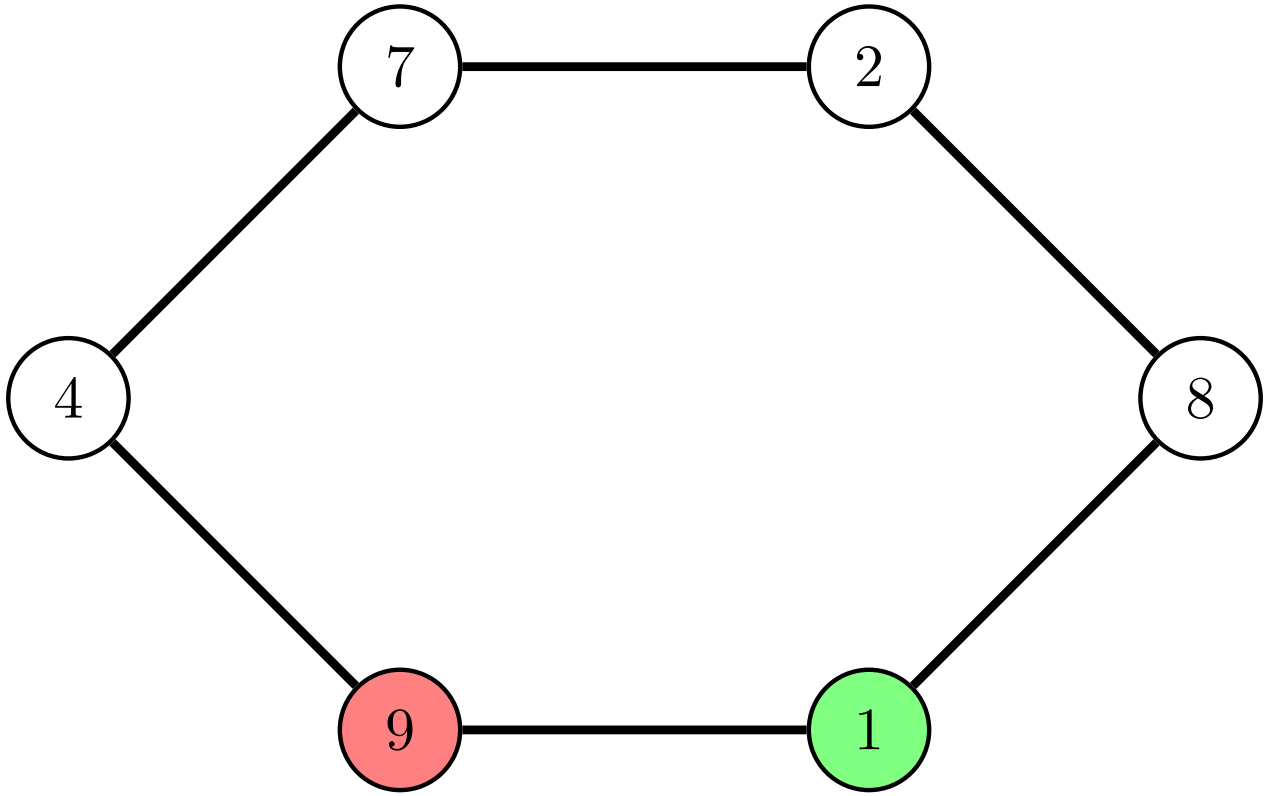
Beispiel

Runde 7:



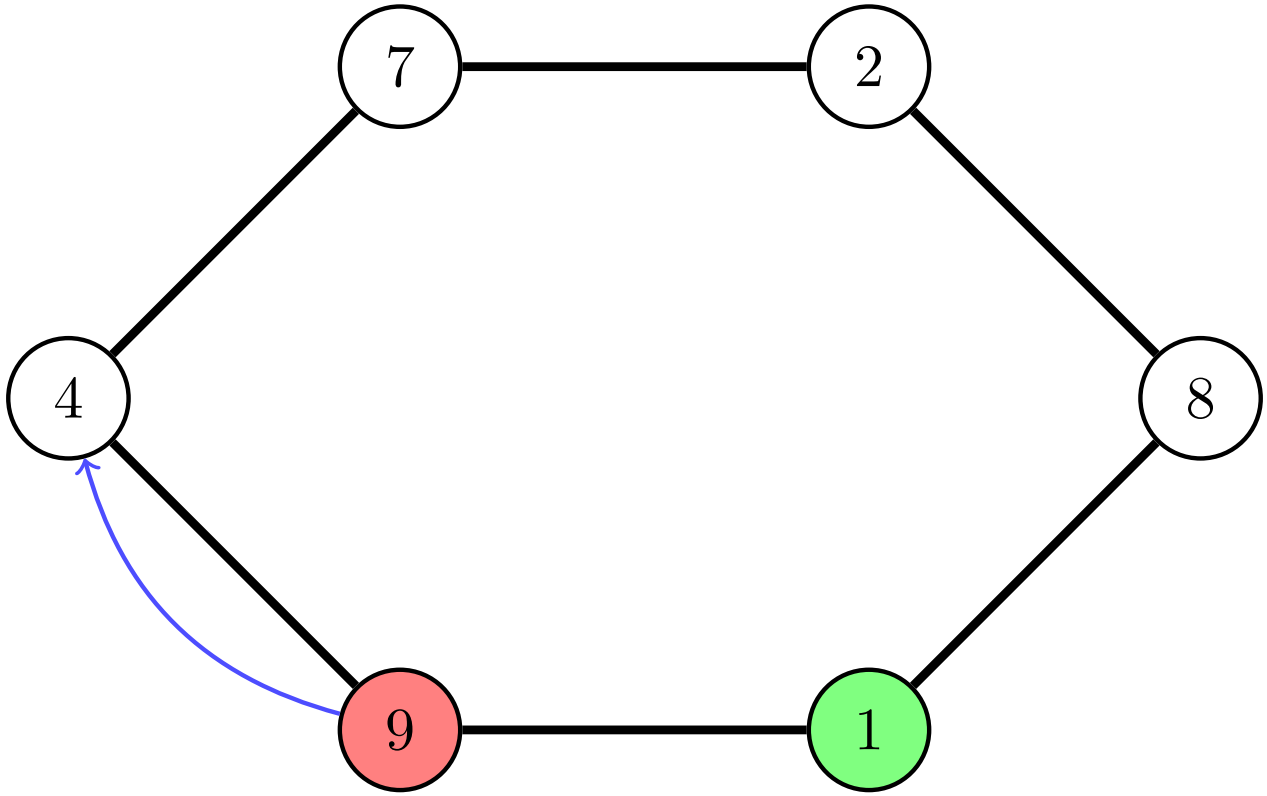
Beispiel

Runde 8:



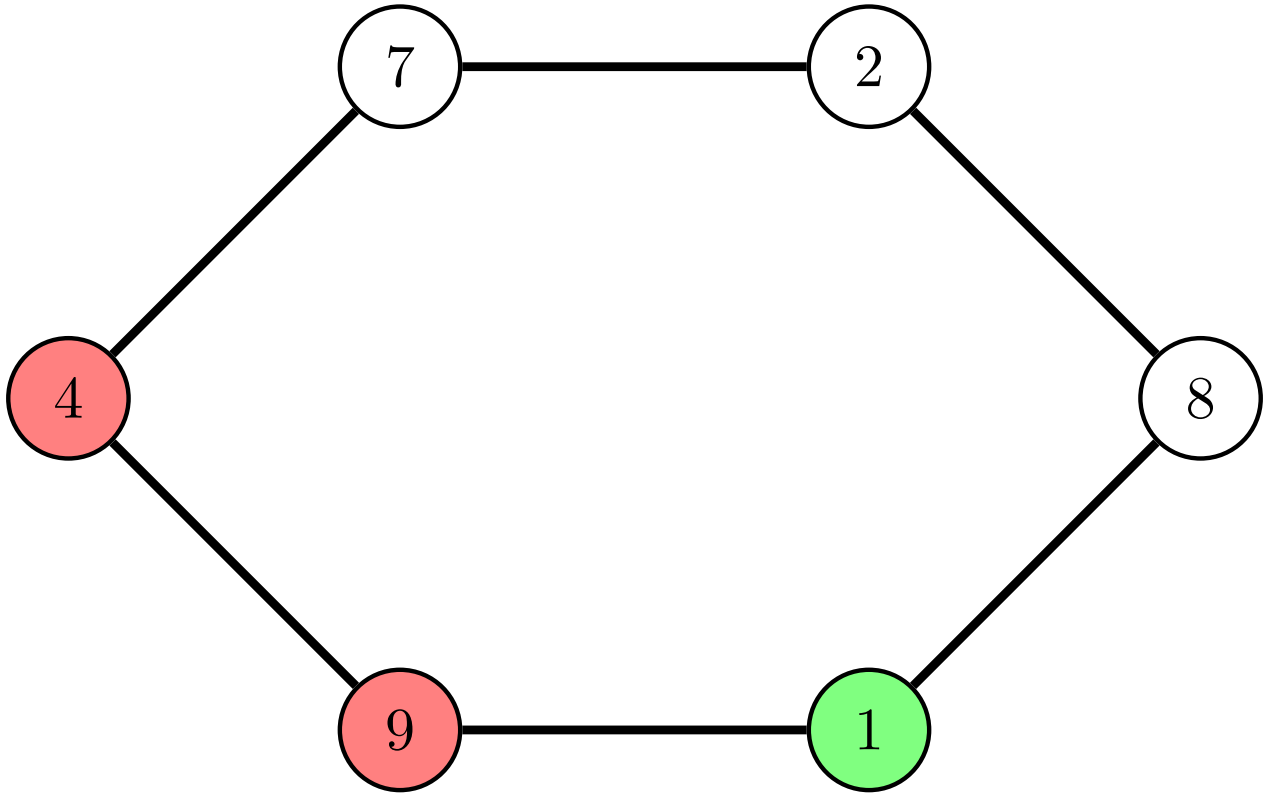
Beispiel

Runde 8:



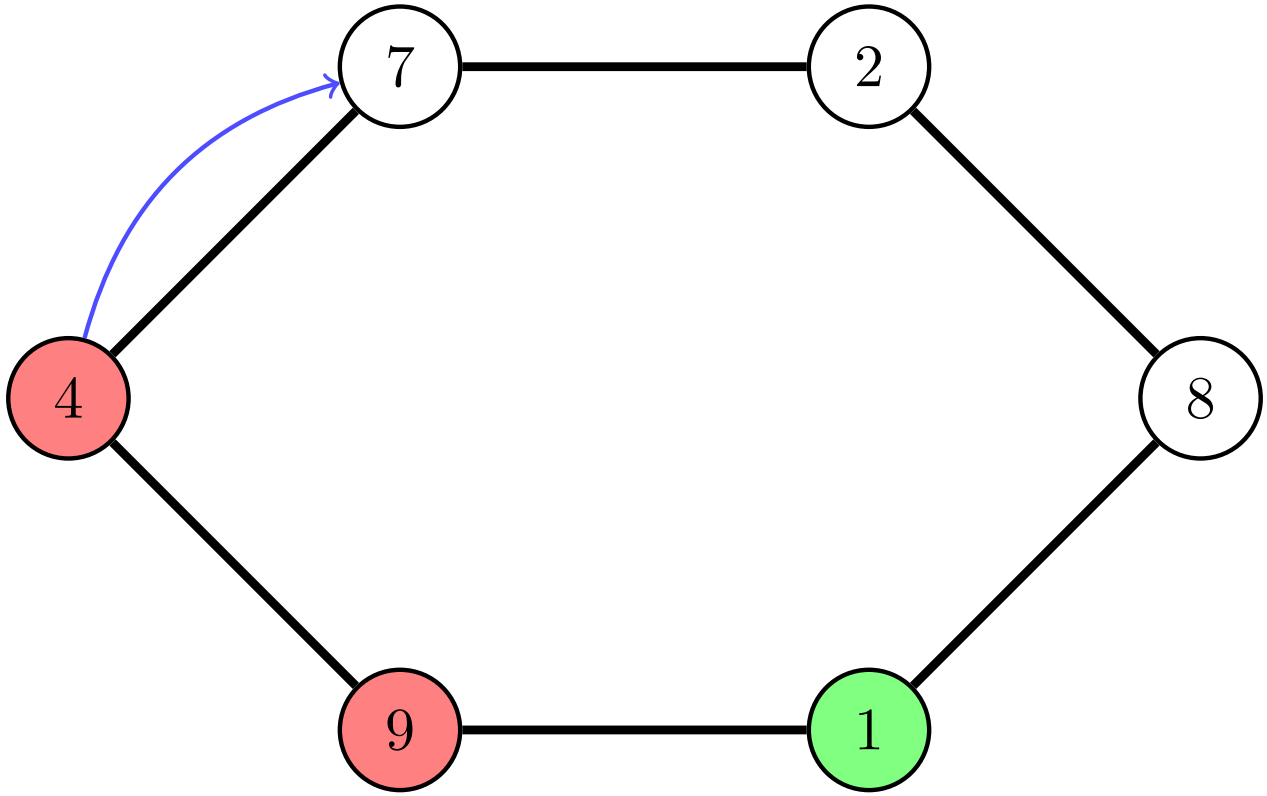
Beispiel

Runde 9:



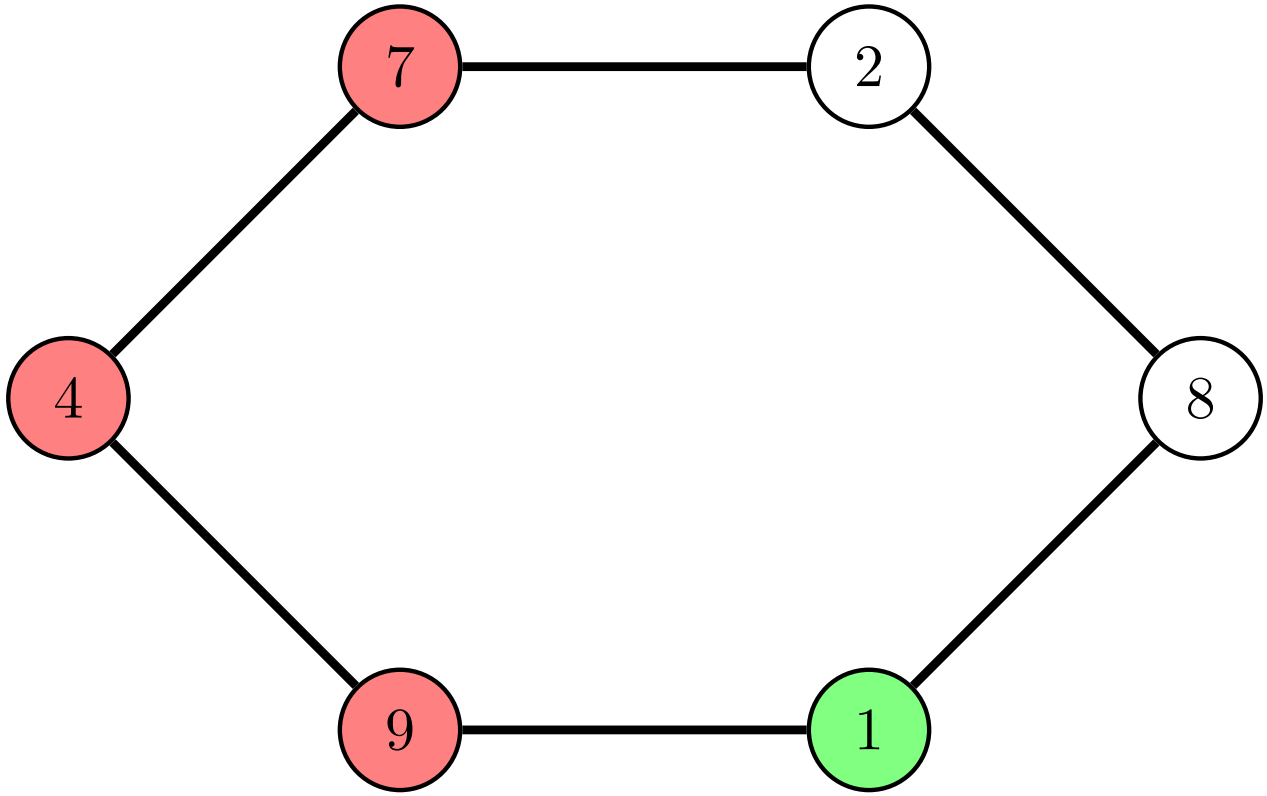
Beispiel

Runde 9:



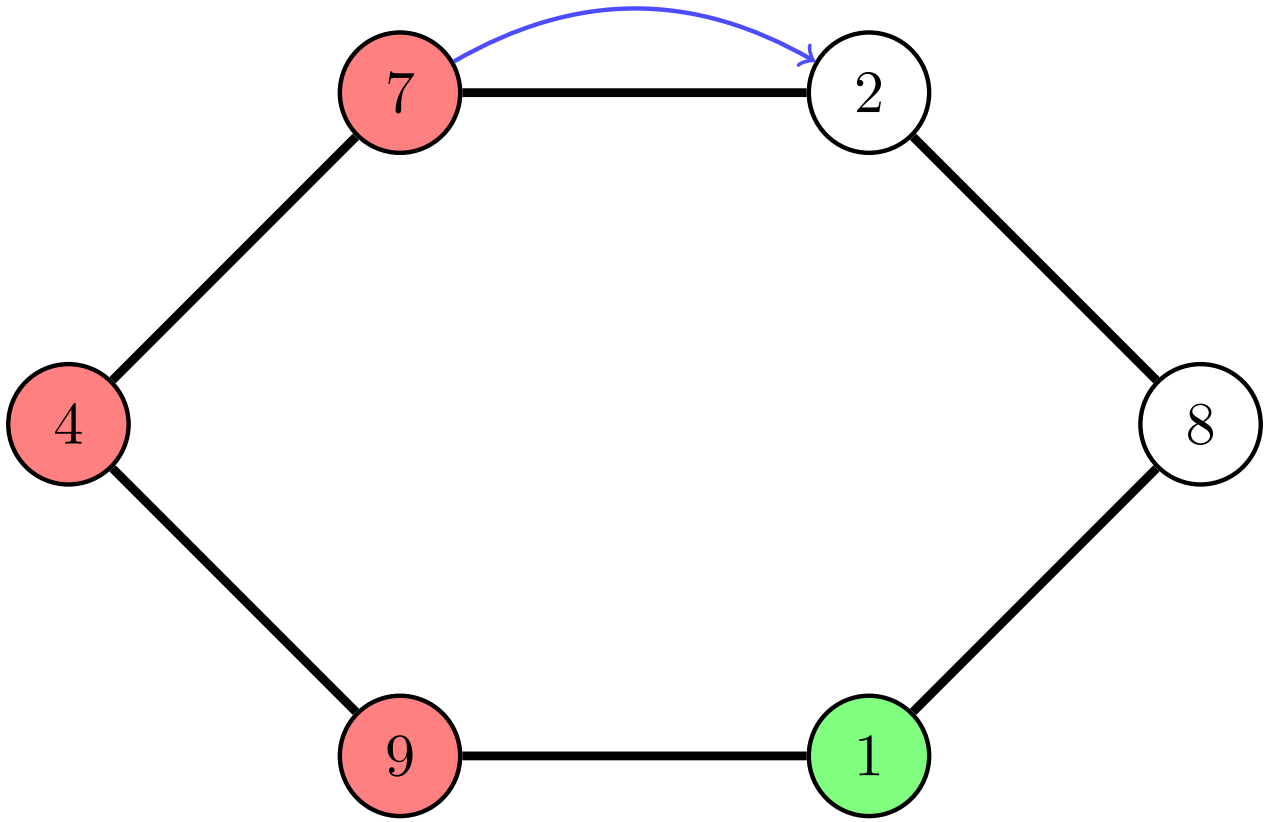
Beispiel

Runde 10:



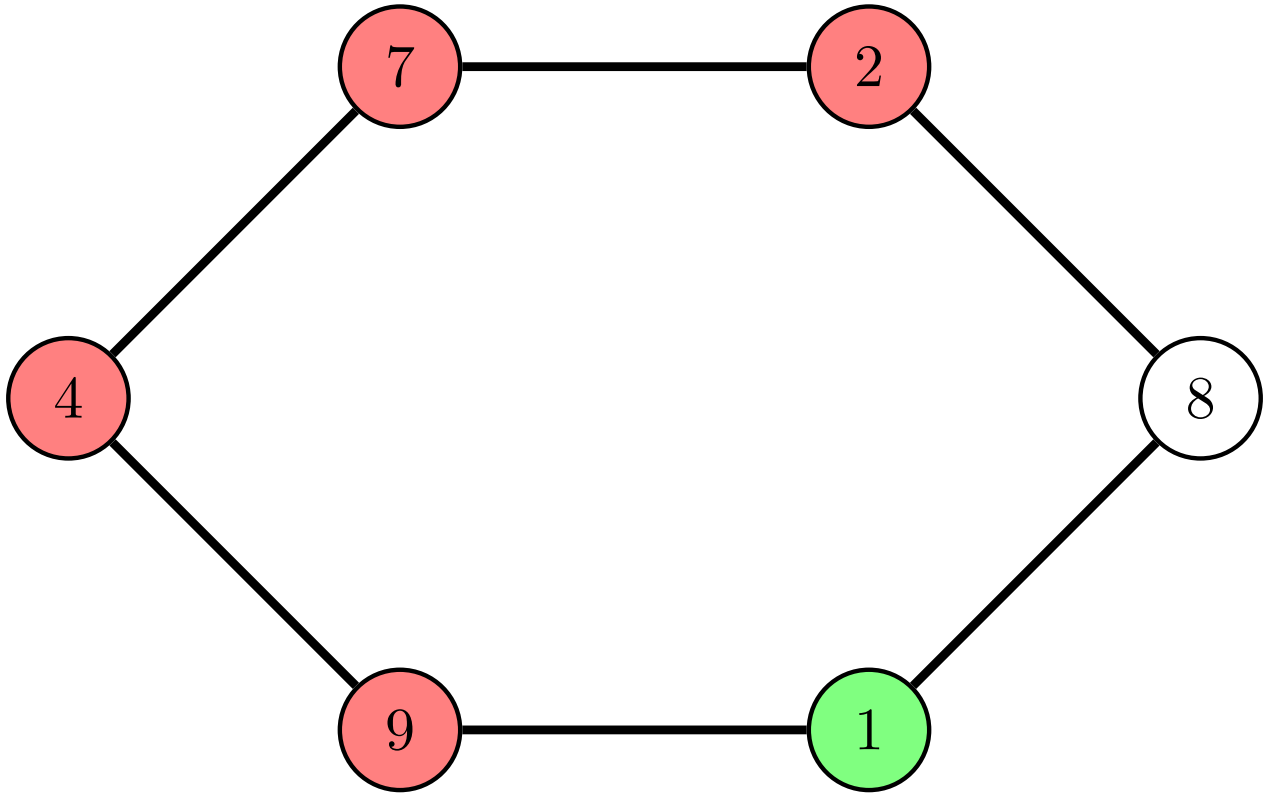
Beispiel

Runde 10:



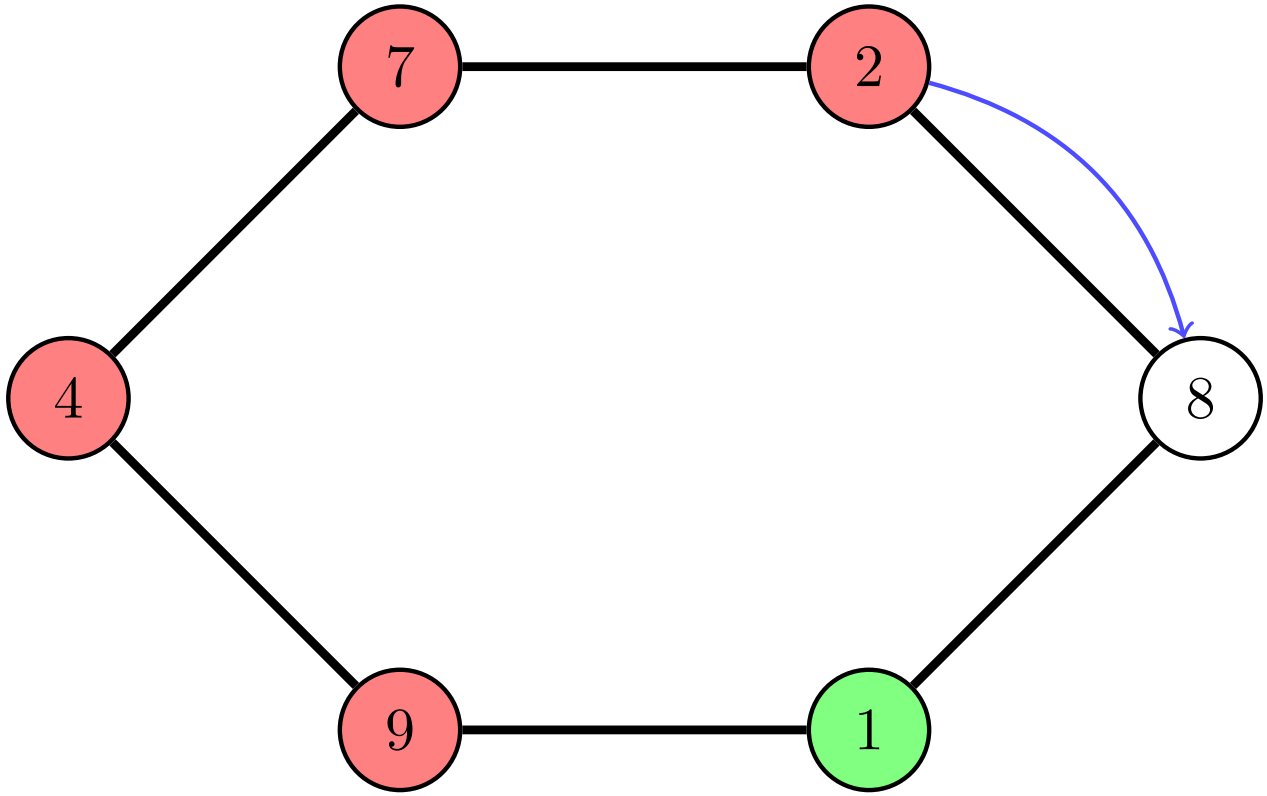
Beispiel

Runde 11:



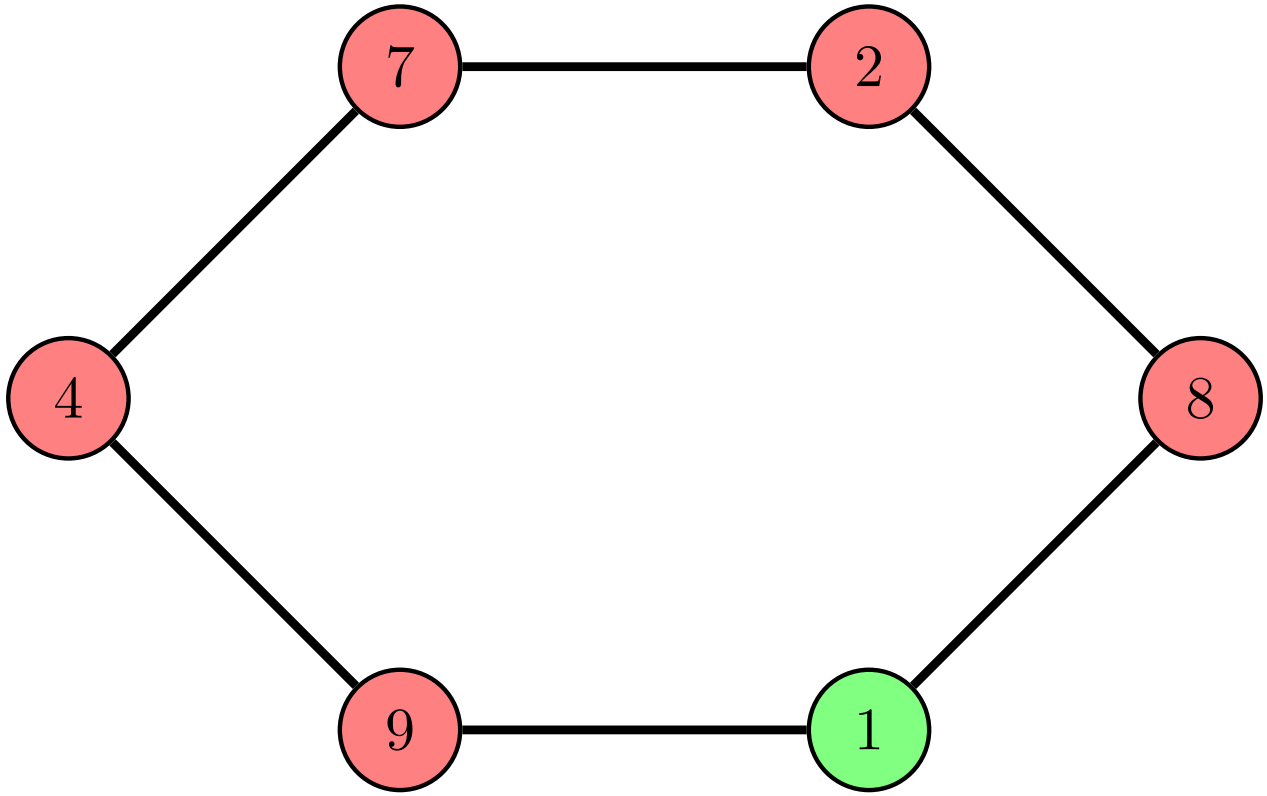
Beispiel

Runde 11:



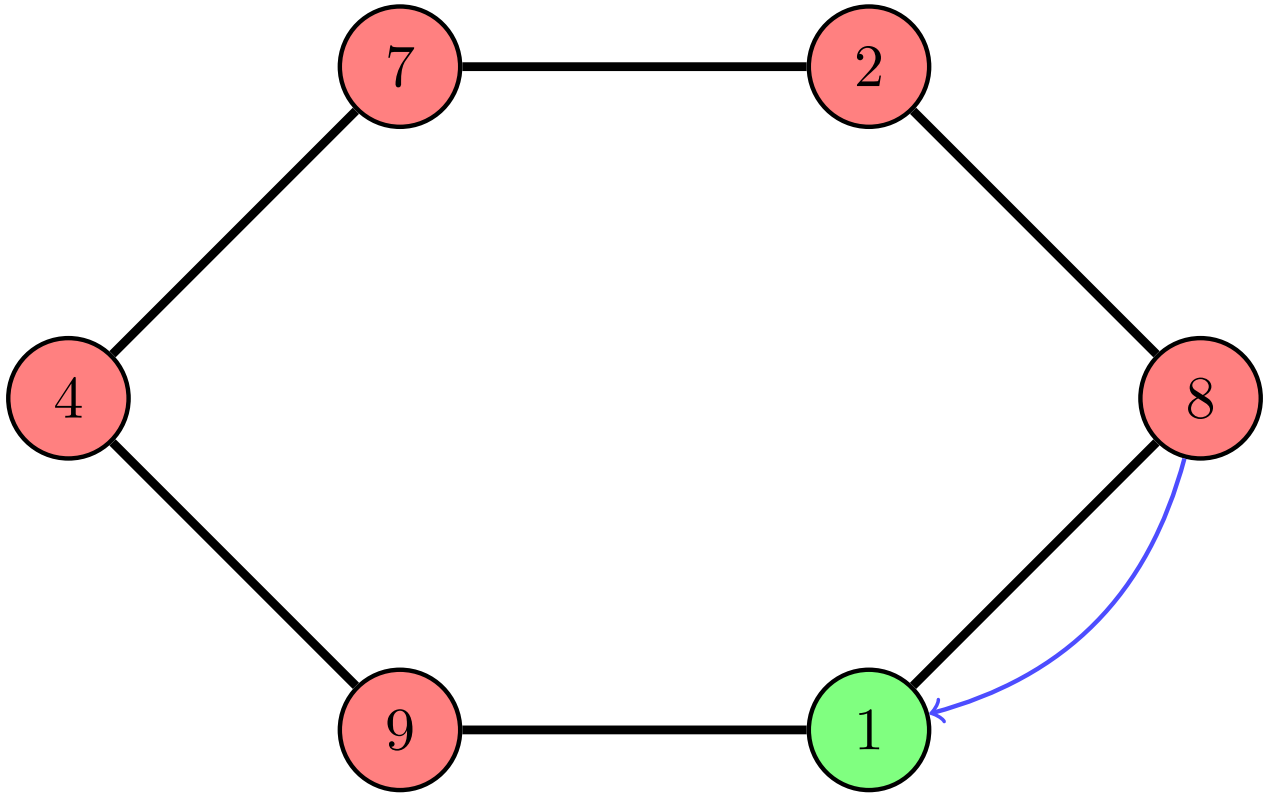
Beispiel

Runde 12:



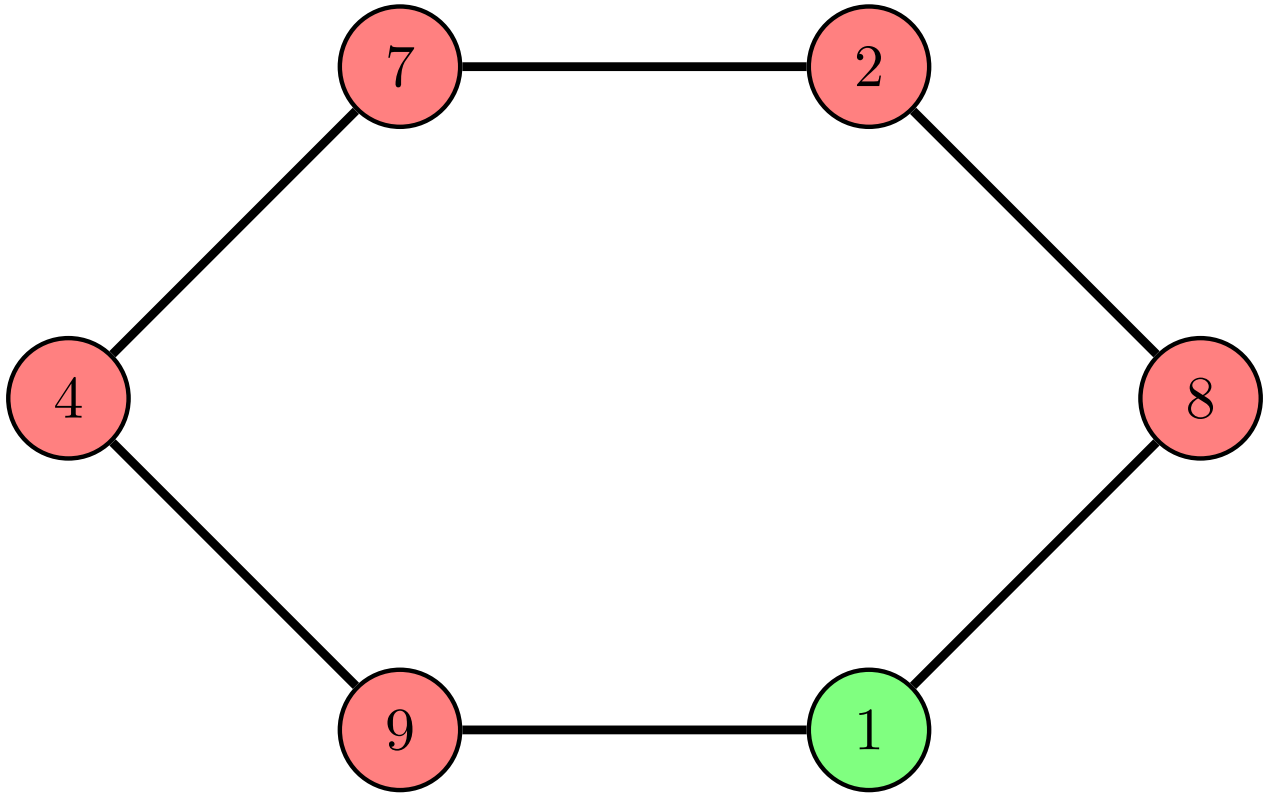
Beispiel

Runde 12:



Beispiel

Runde 13:



Analyse

- Alle n Runden entscheidet sich höchstens ein Knoten Leader zu werden.

Analyse

- Alle n Runden entscheidet sich höchstens ein Knoten Leader zu werden.
- Leader muss also innerhalb von n Runden alle anderen Knoten informieren, damit Algorithmus korrekt ist.

Analyse

- Alle n Runden entscheidet sich höchstens ein Knoten Leader zu werden.
- Leader muss also innerhalb von n Runden alle anderen Knoten informieren, damit Algorithmus korrekt ist.
- Dies wird durch Weiterleiten der Nachricht im Ring garantiert.

Analyse

- Alle n Runden entscheidet sich höchstens ein Knoten Leader zu werden.
- Leader muss also innerhalb von n Runden alle anderen Knoten informieren, damit Algorithmus korrekt ist.
- Dies wird durch Weiterleiten der Nachricht im Ring garantiert.
- Anzahl Nachrichten: $O(n)$

Analyse

- Alle n Runden entscheidet sich höchstens ein Knoten Leader zu werden.
- Leader muss also innerhalb von n Runden alle anderen Knoten informieren, damit Algorithmus korrekt ist.
- Dies wird durch Weiterleiten der Nachricht im Ring garantiert.
- Anzahl Nachrichten: $O(n)$
- Rundenzahl: $\Theta(n \cdot \min_v \mathrm{ID}(v))$

Analyse

- Alle n Runden entscheidet sich höchstens ein Knoten Leader zu werden.
- Leader muss also innerhalb von n Runden alle anderen Knoten informieren, damit Algorithmus korrekt ist.
- Dies wird durch Weiterleiten der Nachricht im Ring garantiert.
- Anzahl Nachrichten: $O(n)$
- Rundenzahl: $\Theta(n \cdot \min_v \mathrm{ID}(v))$
  Beispiel: Für IDs im Bereich von 0 bis $2n$ könnte kleinste ID den Wert n haben. Dann werden bereits $\Theta(n^2)$ Runden benötigt.

Anonyme Netzwerke

Anonym: Knoten haben keine IDs

Anonyme Netzwerke

Anonym: Knoten haben keine IDs

Anmerkung: Nach Leader Election ist Vergabe von IDs relativ einfach

Anonyme Netzwerke

Anonym: Knoten haben keine IDs

Anmerkung: Nach Leader Election ist Vergabe von IDs relativ einfach

Theorem ([Angluin '80])

Leader Election in anonymen Netzwerken ist mit deterministischen Algorithmen unmöglich, sogar im synchronen Ring.

Anonyme Netzwerke

Anonym: Knoten haben keine IDs

Anmerkung: Nach Leader Election ist Vergabe von IDs relativ einfach

Theorem ([Angluin '80])

Leader Election in anonymen Netzwerken ist mit deterministischen Algorithmen unmöglich, sogar im synchronen Ring.

Knackpunkt: Wie sind deterministische Algorithmen formal definiert?

Anonyme Netzwerke

Anonym: Knoten haben keine IDs

Anmerkung: Nach Leader Election ist Vergabe von IDs relativ einfach

Theorem ([Angluin '80])

Leader Election in anonymen Netzwerken ist mit deterministischen Algorithmen unmöglich, sogar im synchronen Ring.

Knackpunkt: Wie sind deterministische Algorithmen formal definiert?

- Funktion f angewendet auf jeden Knoten v:

Anonyme Netzwerke

Anonym: Knoten haben keine IDs

Anmerkung: Nach Leader Election ist Vergabe von IDs relativ einfach

Theorem ([Angluin '80])

Leader Election in anonymen Netzwerken ist mit deterministischen Algorithmen unmöglich, sogar im synchronen Ring.

Knackpunkt: Wie sind deterministische Algorithmen formal definiert?

- Funktion f angewendet auf jeden Knoten v:
  - **Eingabe:** Anzahl an Knoten n, Anzahl an Ports von v, gesamte Historie von v (Protokoll aller bisher empfangen Nachrichten eines Knotens mit Zeitpunkt und Eingangsport)

Anonyme Netzwerke

Anonym: Knoten haben keine IDs

Anmerkung: Nach Leader Election ist Vergabe von IDs relativ einfach

Theorem ([Angluin '80])

Leader Election in anonymen Netzwerken ist mit deterministischen Algorithmen unmöglich, sogar im synchronen Ring.

Knackpunkt: Wie sind deterministische Algorithmen formal definiert?

- Funktion f angewendet auf jeden Knoten v:
  - **Eingabe:** Anzahl an Knoten n, Anzahl an Ports von v, gesamte Historie von v (Protokoll aller bisher empfangen Nachrichten eines Knotens mit Zeitpunkt und Eingangsport)
  - **Ausgabe:** Zu sendende Nachrichten des Knotens an Ausgangsports und gegebenenfalls Entscheidung, ob v Leader oder Follower wird

Anonyme Netzwerke

Anonym: Knoten haben keine IDs

Anmerkung: Nach Leader Election ist Vergabe von IDs relativ einfach

Theorem ([Angluin '80])

Leader Election in anonymen Netzwerken ist mit deterministischen Algorithmen unmöglich, sogar im synchronen Ring.

Knackpunkt: Wie sind deterministische Algorithmen formal definiert?

- Funktion f angewendet auf jeden Knoten v:
  - **Eingabe:** Anzahl an Knoten n, Anzahl an Ports von v, gesamte Historie von v (Protokoll aller bisher empfangen Nachrichten eines Knotens mit Zeitpunkt und Eingangsport)
  - **Ausgabe:** Zu sendende Nachrichten des Knotens an Ausgangsports und gegebenenfalls Entscheidung, ob v Leader oder Follower wird
- Funktion löst das Problem, wenn für jedes Netzwerk nach endlich vielen Anwendungen von f ein Knoten als Leader und alle anderen als Follower festgelegt wurden

Unmöglichkeitsbeweis

Induktionsaussage

Für jeden deterministischen Algorithmus sind in jeder Runde jeweils die im und gegen den Uhrzeigersinn empfangenen Nachrichten für alle Knoten gleich.

Unmöglichkeitsbeweis

Induktionsaussage

Für jeden deterministischen Algorithmus sind in jeder Runde jeweils die im und gegen den Uhrzeigersinn empfangenen Nachrichten für alle Knoten gleich.

Wenn Induktionsaussage gilt:

- Wegen Induktionsaussage haben in jeder Runde alle Knoten die gleiche Historie

Unmöglichkeitsbeweis

Induktionsaussage

Für jeden deterministischen Algorithmus sind in jeder Runde jeweils die im und gegen den Uhrzeigersinn empfangenen Nachrichten für alle Knoten gleich.

Wenn Induktionsaussage gilt:

- Wegen Induktionsaussage haben in jeder Runde alle Knoten die gleiche Historie
- Da außerdem die Anzahl der Ports für jeden Knoten gleich ist, ist in jeder Runde die Eingabe der Funktion f für alle Knoten gleich

Unmöglichkeitsbeweis

Induktionsaussage

Für jeden deterministischen Algorithmus sind in jeder Runde jeweils die im und gegen den Uhrzeigersinn empfangenen Nachrichten für alle Knoten gleich.

Wenn Induktionsaussage gilt:

- Wegen Induktionsaussage haben in jeder Runde alle Knoten die gleiche Historie
- Da außerdem die Anzahl der Ports für jeden Knoten gleich ist, ist in jeder Runde die Eingabe der Funktion f für alle Knoten gleich
- Deshalb ist in jeder Runde die Ausgabe der Funktion f für alle Knoten gleich

Unmöglichkeitsbeweis

Induktionsaussage

Für jeden deterministischen Algorithmus sind in jeder Runde jeweils die im und gegen den Uhrzeigersinn empfangenen Nachrichten für alle Knoten gleich.

Wenn Induktionsaussage gilt:

- Wegen Induktionsaussage haben in jeder Runde alle Knoten die gleiche Historie
- Da außerdem die Anzahl der Ports für jeden Knoten gleich ist, ist in jeder Runde die Eingabe der Funktion f für alle Knoten gleich
- Deshalb ist in jeder Runde die Ausgabe der Funktion f für alle Knoten gleich
- Somit: In jeder Runde macht f entweder alle Knoten zu Leadern, alle Knoten zu Followern oder trifft für keinen Knoten eine Entscheidung

Unmöglichkeitsbeweis

Induktionsaussage

Für jeden deterministischen Algorithmus sind in jeder Runde jeweils die im und gegen den Uhrzeigersinn empfangenen Nachrichten für alle Knoten gleich.

Wenn Induktionsaussage gilt:

- Wegen Induktionsaussage haben in jeder Runde alle Knoten die gleiche Historie
- Da außerdem die Anzahl der Ports für jeden Knoten gleich ist, ist in jeder Runde die Eingabe der Funktion f für alle Knoten gleich
- Deshalb ist in jeder Runde die Ausgabe der Funktion f für alle Knoten gleich
- Somit: In jeder Runde macht f entweder alle Knoten zu Leadern, alle Knoten zu Followern oder trifft für keinen Knoten eine Entscheidung
- f liefert also entweder ein falsches Ergebnis oder terminiert nicht

Induktionsbeweis

Induktionsaussage

Für jeden deterministischen Algorithmus sind in jeder Runde jeweils die im und gegen den Uhrzeigersinn empfangenen Nachrichten für alle Knoten gleich.

Induktionsbeweis

Induktionsaussage

Für jeden deterministischen Algorithmus sind in jeder Runde jeweils die im und gegen den Uhrzeigersinn empfangenen Nachrichten für alle Knoten gleich.

Induktionsbasis: Gilt trivialerweise da in der ersten Runde noch keine Nachrichten empfangen werden.

Induktionsbeweis

Induktionsaussage

Für jeden deterministischen Algorithmus sind in jeder Runde jeweils die im und gegen den Uhrzeigersinn empfangenen Nachrichten für alle Knoten gleich.

Induktionsbasis: Gilt trivialerweise da in der ersten Runde noch keine Nachrichten empfangen werden.

Induktionsschritt:

- Da Induktionsaussage für alle vorherigen Runden gilt, hat jeder Knoten die gleiche Historie.

Induktionsbeweis

Induktionsaussage

Für jeden deterministischen Algorithmus sind in jeder Runde jeweils die im und gegen den Uhrzeigersinn empfangenen Nachrichten für alle Knoten gleich.

Induktionsbasis: Gilt trivialerweise da in der ersten Runde noch keine Nachrichten empfangen werden.

Induktionsschritt:

- Da Induktionsaussage für alle vorherigen Runden gilt, hat jeder Knoten die gleiche Historie.
- Da außerdem die Anzahl der Ports für jeden Knoten gleich ist, ist die Eingabe der Funktion f für alle Knoten gleich

Induktionsbeweis

Induktionsaussage

Für jeden deterministischen Algorithmus sind in jeder Runde jeweils die im und gegen den Uhrzeigersinn empfangenen Nachrichten für alle Knoten gleich.

Induktionsbasis: Gilt trivialerweise da in der ersten Runde noch keine Nachrichten empfangen werden.

Induktionsschritt:

- Da Induktionsaussage für alle vorherigen Runden gilt, hat jeder Knoten die gleiche Historie.
- Da außerdem die Anzahl der Ports für jeden Knoten gleich ist, ist die Eingabe der Funktion f für alle Knoten gleich
- Deshalb ist auch die Ausgabe der Funktion f für alle Knoten gleich

Induktionsbeweis

Induktionsaussage

Für jeden deterministischen Algorithmus sind in jeder Runde jeweils die im und gegen den Uhrzeigersinn empfangenen Nachrichten für alle Knoten gleich.

Induktionsbasis: Gilt trivialerweise da in der ersten Runde noch keine Nachrichten empfangen werden.

Induktionsschritt:

- Da Induktionsaussage für alle vorherigen Runden gilt, hat jeder Knoten die gleiche Historie.
- Da außerdem die Anzahl der Ports für jeden Knoten gleich ist, ist die Eingabe der Funktion f für alle Knoten gleich
- Deshalb ist auch die Ausgabe der Funktion f für alle Knoten gleich
- Somit: Alle Knoten senden die jeweils gleiche Nachricht im und gegen den Uhrzeigersinn

Induktionsbeweis

Induktionsaussage

Für jeden deterministischen Algorithmus sind in jeder Runde jeweils die im und gegen den Uhrzeigersinn empfangenen Nachrichten für alle Knoten gleich.

Induktionsbasis: Gilt trivialerweise da in der ersten Runde noch keine Nachrichten empfangen werden.

Induktionsschritt:

- Da Induktionsaussage für alle vorherigen Runden gilt, hat jeder Knoten die gleiche Historie.
- Da außerdem die Anzahl der Ports für jeden Knoten gleich ist, ist die Eingabe der Funktion f für alle Knoten gleich
- Deshalb ist auch die Ausgabe der Funktion f für alle Knoten gleich
- Somit: Alle Knoten senden die jeweils gleiche Nachricht im und gegen den Uhrzeigersinn
- Wegen Ring-Topologie: Alle empfangen jeweils gleiche Nachricht im und gegen den Uhrzeigersinn

Zusammenfassung

Leader Election im Ring:

1. Anonyme Ringe: unmöglich für deterministische Algorithmen

Zusammenfassung

Leader Election im Ring:

1. Anonyme Ringe: unmöglich für deterministische Algorithmen
   Nächste Vorlesungseinheit: Randomisierter Algorithmus

Zusammenfassung

Leader Election im Ring:

1. Anonyme Ringe: unmöglich für deterministische Algorithmen
   Nächste Vorlesungseinheit: Randomisierter Algorithmus
2. Clockwise Algorithmus:
   - Deterministischer synchroner/asynchroner Algorithmus für uniforme/non-uniforme Ringe
   - $O(n)$ Runden, $O(n^2)$ Nachrichten

Zusammenfassung

Leader Election im Ring:

1. Anonyme Ringe: unmöglich für deterministische Algorithmen
   Nächste Vorlesungseinheit: Randomisierter Algorithmus
2. Clockwise Algorithmus:
   - Deterministischer synchroner/asynchroner Algorithmus für uniforme/non-uniforme Ringe
   - $O(n)$ Runden, $O(n^2)$ Nachrichten
3. Wartezeit Algorithmus:
   - Deterministischer synchroner Algorithmus für non-uniforme Ringe
   - $O(n \cdot \min_v \mathrm{ID}(v))$ Runden, $O(n)$ Nachrichten

Zusammenfassung

Leader Election im Ring:

1. Anonyme Ringe: unmöglich für deterministische Algorithmen
   Nächste Vorlesungseinheit: Randomisierter Algorithmus
2. Clockwise Algorithmus:
   - Deterministischer synchroner/asynchroner Algorithmus für uniforme/non-uniforme Ringe
   - $O(n)$ Runden, $O(n^2)$ Nachrichten
3. Wartezeit Algorithmus:
   - Deterministischer synchroner Algorithmus für non-uniforme Ringe
   - $O(n \cdot \min_v \mathrm{ID}(v))$ Runden, $O(n)$ Nachrichten
   - Frage: Effizienz in beiden Metriken möglich?

Zusammenfassung

Leader Election im Ring:

1. Anonyme Ringe: unmöglich für deterministische Algorithmen
   Nächste Vorlesungseinheit: Randomisierter Algorithmus
2. Clockwise Algorithmus:
   - Deterministischer synchroner/asynchroner Algorithmus für uniforme/non-uniforme Ringe
   - $O(n)$ Runden, $O(n^2)$ Nachrichten
3. Wartezeit Algorithmus:
   - Deterministischer synchroner Algorithmus für non-uniforme Ringe
   - $O(n \cdot \min_v \mathrm{ID}(v))$ Runden, $O(n)$ Nachrichten
   - Frage: Effizienz in beiden Metriken möglich?

   Nächste Vorlesungseinheit: $O(n)$ Runden, $O(n \log n)$ Nachrichten

Quellen

Der Inhalt dieser Vorlesungseinheit basiert zum Teil auf Vorlesungseinheiten von Robert Elsässer und Stefan Schmid.

Literatur:

- Dana Angluin. „Local and Global Properties in Networks of Processors (Extended Abstract)“. In: *Proc. of the Symposium on Theory of Computing (STOC)*. 1980, S. 82–93
- Hagit Attiya, Jennifer Welch (2004) *Distributed Computing*, Kapitel 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, S. 155–160
- Nancy A. Lynch (1996) *Distributed Algorithms*, Kapitel 3, Morgan Kaufmann.