

Leader Election II

Algorithmen für verteilte Systeme

Sebastian Forster

Universität Salzburg



Dieses Werk ist unter einer Creative Commons Namensnennung 4.0 International Lizenz lizenziert.

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
- Schwierigkeit beim Finden einer Lösung ist die konsistente Entscheidung für eine der Lösungen

Was bisher geschah...

Leader Election im Ring:

- ❶ Anonyme Ringe: unmöglich für deterministische Algorithmen

Was bisher geschah...

Leader Election im Ring:

- ❶ Anonyme Ringe: unmöglich für deterministische Algorithmen

Heute: Randomisierter Algorithmus

Was bisher geschah...

Leader Election im Ring:

- ❶ Anonyme Ringe: unmöglich für deterministische Algorithmen
Heute: Randomisierter Algorithmus
- ❷ Clockwise Algorithmus:
 - ▶ Deterministischer synchroner/asynchroner Algorithmus für uniforme/non-uniforme Ringe
 - ▶ $O(n)$ Runden, $O(n^2)$ Nachrichten

Was bisher geschah...

Leader Election im Ring:

- ❶ Anonyme Ringe: unmöglich für deterministische Algorithmen
Heute: Randomisierter Algorithmus
- ❷ Clockwise Algorithmus:
 - ▶ Deterministischer synchroner/asynchroner Algorithmus für uniforme/non-uniforme Ringe
 - ▶ $O(n)$ Runden, $O(n^2)$ Nachrichten
- ❸ Procrastination Algorithmus:
 - ▶ Deterministischer synchroner Algorithmus für non-uniforme Ringe
 - ▶ $O(n \cdot \min_v \text{ID}(v))$ Runden, $O(n)$ Nachrichten

Was bisher geschah...

Leader Election im Ring:

- ❶ Anonyme Ringe: unmöglich für deterministische Algorithmen

Heute: Randomisierter Algorithmus

- ❷ Clockwise Algorithmus:

- ▶ Deterministischer synchroner/asynchroner Algorithmus für uniforme/non-uniforme Ringe
- ▶ $O(n)$ Runden, $O(n^2)$ Nachrichten

- ❸ Procrastination Algorithmus:

- ▶ Deterministischer synchroner Algorithmus für non-uniforme Ringe
- ▶ $O(n \cdot \min_v \text{ID}(v))$ Runden, $O(n)$ Nachrichten
- ▶ Frage: Effizienz in beiden Metriken möglich?

Was bisher geschah...

Leader Election im Ring:

- ❶ Anonyme Ringe: unmöglich für deterministische Algorithmen

Heute: Randomisierter Algorithmus

- ❷ Clockwise Algorithmus:

- ▶ Deterministischer synchroner/asynchroner Algorithmus für uniforme/non-uniforme Ringe
- ▶ $O(n)$ Runden, $O(n^2)$ Nachrichten

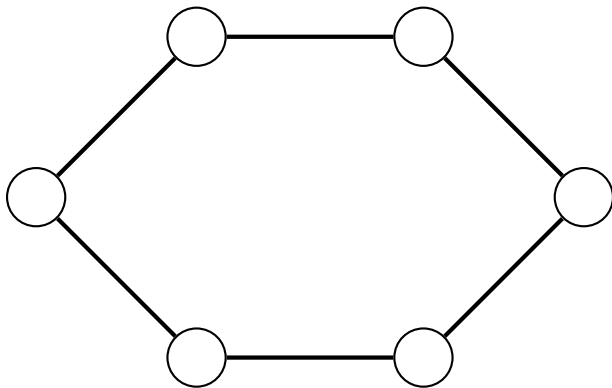
- ❸ Procrastination Algorithmus:

- ▶ Deterministischer synchroner Algorithmus für non-uniforme Ringe
- ▶ $O(n \cdot \min_v \text{ID}(v))$ Runden, $O(n)$ Nachrichten
- ▶ Frage: Effizienz in beiden Metriken möglich?

Heute: $O(n)$ Runden, $O(n \log n)$ Nachrichten

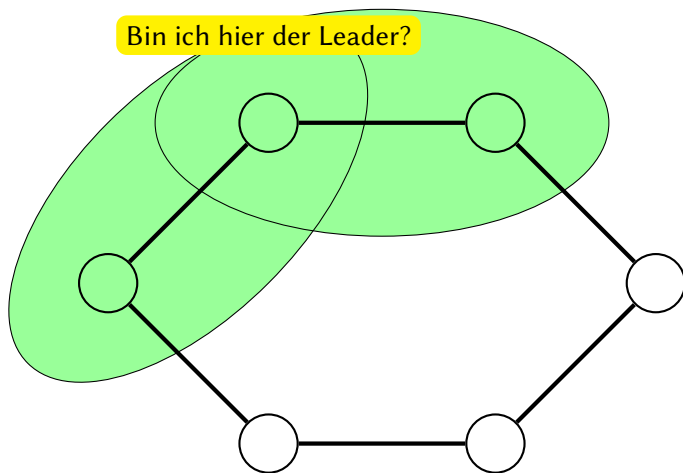
Idee: Radius Growth

Annahmen: synchron, identifizierbar, non-uniform



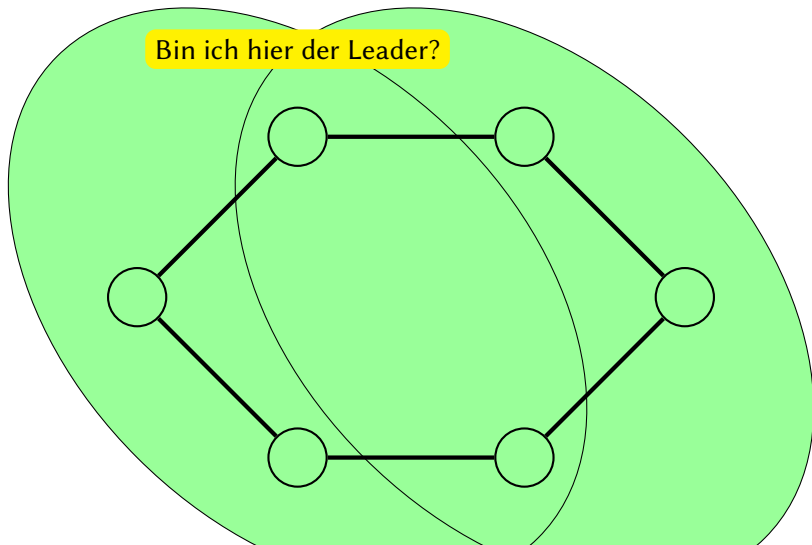
Idee: Radius Growth

Annahmen: synchron, identifizierbar, non-uniform



Idee: Radius Growth

Annahmen: synchron, identifizierbar, non-uniform



Radius Growth Algorithmus [Hirschberg/Sinclair '80]

- Unterteilung der Runden in $\lceil \log n \rceil$ aufeinanderfolgende Phasen
- Die i -te Phase dauert $2^{i-1} + 1$ Runden

Radius Growth Algorithmus [Hirschberg/Sinclair '80]

- Unterteilung der Runden in $\lceil \log n \rceil$ aufeinanderfolgende Phasen
- Die i -te Phase dauert $2^{i-1} + 1$ Runden

Algorithmus für jeden Knoten v :

Erste Runde jeder Phase:

Radius Growth Algorithmus [Hirschberg/Sinclair '80]

- Unterteilung der Runden in $\lceil \log n \rceil$ aufeinanderfolgende Phasen
- Die i -te Phase dauert $2^{i-1} + 1$ Runden

Algorithmus für jeden Knoten v :

Erste Runde jeder Phase:

- 1 **if** v noch kein Follower **then**
- 2 v sendet $ID(v)$ an Nachbarn im und gegen den Uhrzeigersinn

Radius Growth Algorithmus [Hirschberg/Sinclair '80]

- Unterteilung der Runden in $\lceil \log n \rceil$ aufeinanderfolgende Phasen
- Die i -te Phase dauert $2^{i-1} + 1$ Runden

Algorithmus für jeden Knoten v :

Erste Runde jeder Phase:

- 1 **if** v noch kein Follower **then**
- 2 v sendet $ID(v)$ an Nachbarn im und gegen den Uhrzeigersinn

Jede andere Runde:

- 1 **if** v empfängt Nachricht M von Nachbar gegen (im) den Uhrzeigersinn **then**

Radius Growth Algorithmus [Hirschberg/Sinclair '80]

- Unterteilung der Runden in $\lceil \log n \rceil$ aufeinanderfolgende Phasen
- Die i -te Phase dauert $2^{i-1} + 1$ Runden

Algorithmus für jeden Knoten v :

Erste Runde jeder Phase:

- 1 **if** v noch kein Follower **then**
- 2 v sendet $ID(v)$ an Nachbarn im und gegen den Uhrzeigersinn

Jede andere Runde:

- 1 **if** v empfängt Nachricht M von Nachbar gegen (im) den Uhrzeigersinn **then**
- 2 **if** $M < ID(v)$ **then**
- 3 v wird zum Follower (sofern nicht bereits vorher geschehen)
- 4 **if** nicht letzte Runde der Phase **then**
- 5 v sendet M an Nachbar im (gegen) Uhrzeigersinn

Radius Growth Algorithmus [Hirschberg/Sinclair '80]

- Unterteilung der Runden in $\lceil \log n \rceil$ aufeinanderfolgende Phasen
- Die i -te Phase dauert $2^{i-1} + 1$ Runden

Algorithmus für jeden Knoten v :

Erste Runde jeder Phase:

- 1 **if** v noch kein Follower **then**
- 2 v sendet $ID(v)$ an Nachbarn im und gegen den Uhrzeigersinn

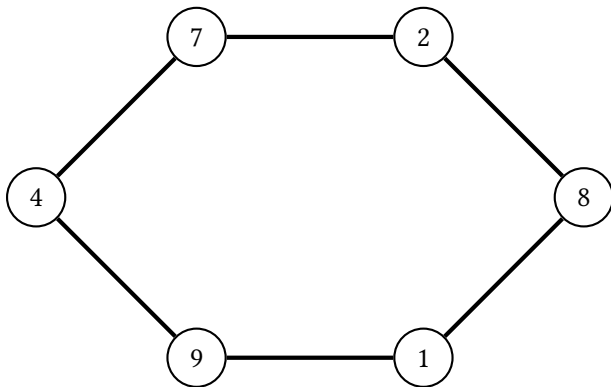
Jede andere Runde:

- 1 **if** v empfängt Nachricht M von Nachbar gegen (im) den Uhrzeigersinn **then**
- 2 **if** $M < ID(v)$ **then**
- 3 v wird zum Follower (sofern nicht bereits vorher geschehen)
- 4 **if** nicht letzte Runde der Phase **then**
- 5 v sendet M an Nachbar im (gegen) Uhrzeigersinn

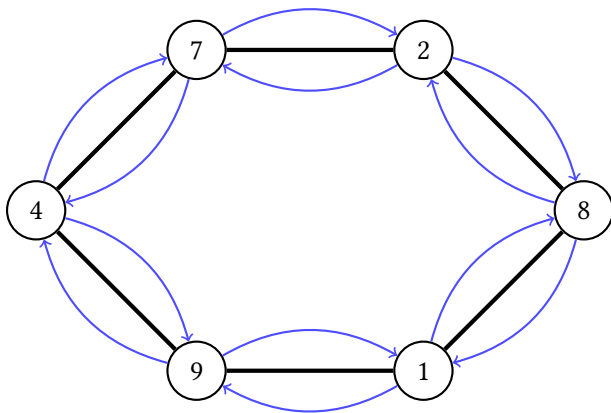
Zusätzlich in letzter Runde der letzten Phase:

- 1 **if** v ist noch kein Follower **then** v wird zum Leader

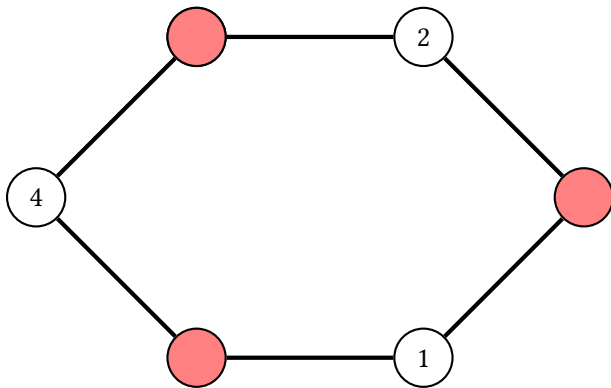
Beispiel



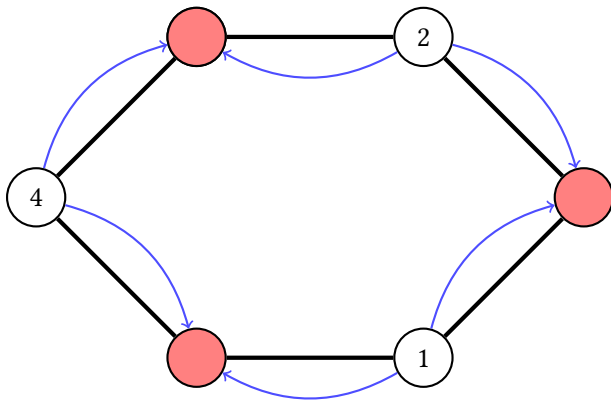
Beispiel



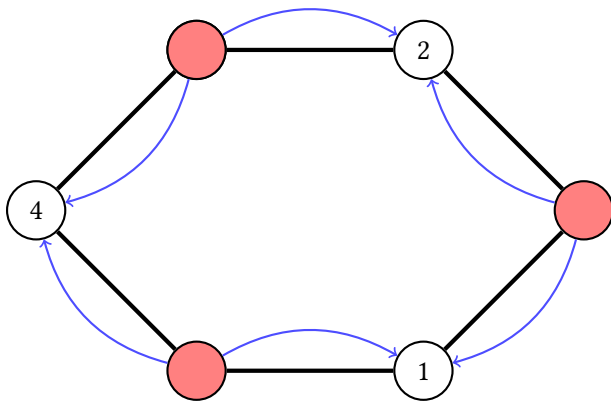
Beispiel



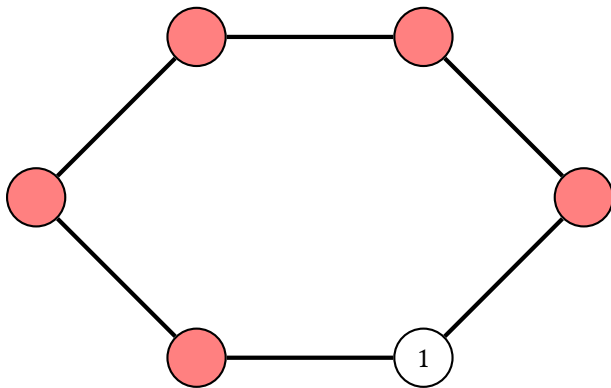
Beispiel



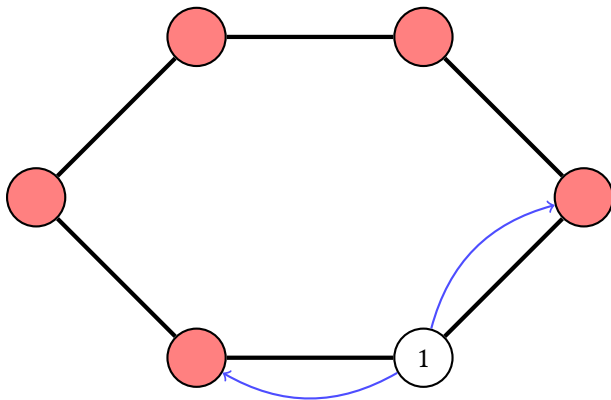
Beispiel



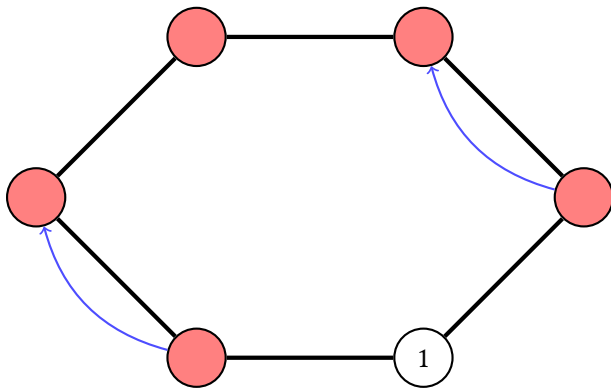
Beispiel



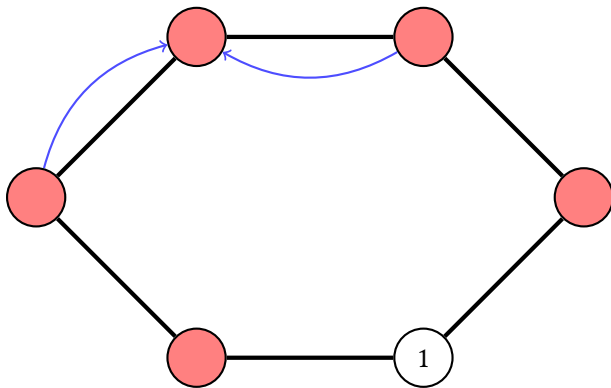
Beispiel



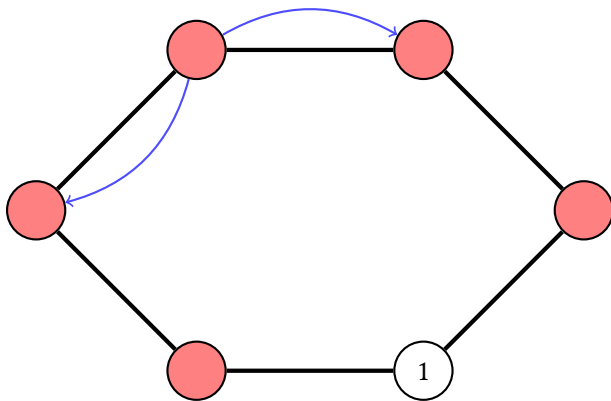
Beispiel



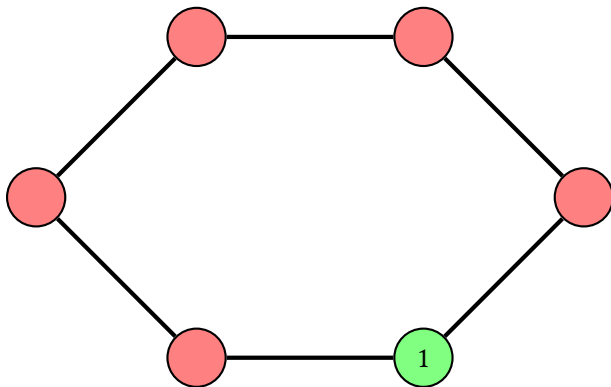
Beispiel



Beispiel



Beispiel



Korrektheit

Theorem

Der Radius Growth Algorithmus bestimmt einen eindeutigen Leader.

Theorem

Der Radius Growth Algorithmus bestimmt einen eindeutigen Leader.

Beweis:

- Knoten z mit kleinster ID kann nie Nachricht mit kleinerer ID erhalten, wird deshalb nie zum Follower und wird somit in letzter Runde von letzter Phase zum Leader

Theorem

Der Radius Growth Algorithmus bestimmt einen eindeutigen Leader.

Beweis:

- Knoten z mit kleinster ID kann nie Nachricht mit kleinerer ID erhalten, wird deshalb nie zum Follower und wird somit in letzter Runde von letzter Phase zum Leader
- Nachricht mit ID von z in letzter Phase ℓ (wobei $\ell = \lceil \log n \rceil$) erreicht 2^ℓ Knoten jeweils im und gegen den Uhrzeigersinn

Theorem

Der Radius Growth Algorithmus bestimmt einen eindeutigen Leader.

Beweis:

- Knoten z mit kleinster ID kann nie Nachricht mit kleinerer ID erhalten, wird deshalb nie zum Follower und wird somit in letzter Runde von letzter Phase zum Leader
- Nachricht mit ID von z in letzter Phase ℓ (wobei $\ell = \lceil \log n \rceil$) erreicht 2^ℓ Knoten jeweils im und gegen den Uhrzeigersinn
- Da $2^\ell = 2^{\lceil \log n \rceil - 1} \geq 2^{\log n - 1} = \frac{2^{\log n}}{2} = \frac{n}{2}$, erreicht Nachricht mit ID von z alle anderen Knoten, die spätestens dann zu Followern werden, da sie höhere ID als z haben

Theorem

Der Radius Growth Algorithmus bestimmt einen eindeutigen Leader.

Beweis:

- Knoten z mit kleinster ID kann nie Nachricht mit kleinerer ID erhalten, wird deshalb nie zum Follower und wird somit in letzter Runde von letzter Phase zum Leader
- Nachricht mit ID von z in letzter Phase ℓ (wobei $\ell = \lceil \log n \rceil$) erreicht 2^ℓ Knoten jeweils im und gegen den Uhrzeigersinn
- Da $2^\ell = 2^{\lceil \log n \rceil - 1} \geq 2^{\log n - 1} = \frac{2^{\log n}}{2} = \frac{n}{2}$, erreicht Nachricht mit ID von z alle anderen Knoten, die spätestens dann zu Followern werden, da sie höhere ID als z haben
- Somit: Gültige Einteilung in Leader und Follower

Laufzeit

Theorem

Der Radius Growth Algorithmus benötigt $O(n)$ Runden.

Laufzeit

Theorem

Der Radius Growth Algorithmus benötigt $O(n)$ Runden.

Beweis:

- Phase i benötigt $2^{i-1} + 1 \leq 2^i$ Runden

Theorem

Der Radius Growth Algorithmus benötigt $O(n)$ Runden.

Beweis:

- Phase i benötigt $2^{i-1} + 1 \leq 2^i$ Runden
- Gesamtzahl an Runden:

$$\sum_{i=1}^{\lceil \log n \rceil} 2^i$$

Theorem

Der Radius Growth Algorithmus benötigt $O(n)$ Runden.

Beweis:

- Phase i benötigt $2^{i-1} + 1 \leq 2^i$ Runden
- Gesamtzahl an Runden:

$$\sum_{i=1}^{\lceil \log n \rceil} 2^i \leq 2^{\lceil \log n \rceil + 1}$$

Geometrischen Reihe: $\sum_{k=0}^n r^k = \frac{r^{n+1}-1}{r-1} \leq r^{n+1}$ für $r \neq 1$ und $n \geq 0$

Theorem

Der Radius Growth Algorithmus benötigt $O(n)$ Runden.

Beweis:

- Phase i benötigt $2^{i-1} + 1 \leq 2^i$ Runden
- Gesamtzahl an Runden:

$$\sum_{i=1}^{\lceil \log n \rceil} 2^i \leq 2^{\lceil \log n \rceil + 1} \leq 2^{\log n + 2}$$

Geometrischen Reihe: $\sum_{k=0}^n r^k = \frac{r^{n+1}-1}{r-1} \leq r^{n+1}$ für $r \neq 1$ und $n \geq 0$

Theorem

Der Radius Growth Algorithmus benötigt $O(n)$ Runden.

Beweis:

- Phase i benötigt $2^{i-1} + 1 \leq 2^i$ Runden
- Gesamtzahl an Runden:

$$\sum_{i=1}^{\lceil \log n \rceil} 2^i \leq 2^{\lceil \log n \rceil + 1} \leq 2^{\log n + 2} \leq 4 \cdot 2^{\log n}$$

Geometrischen Reihe: $\sum_{k=0}^n r^k = \frac{r^{n+1}-1}{r-1} \leq r^{n+1}$ für $r \neq 1$ und $n \geq 0$

Theorem

Der Radius Growth Algorithmus benötigt $O(n)$ Runden.

Beweis:

- Phase i benötigt $2^{i-1} + 1 \leq 2^i$ Runden
- Gesamtzahl an Runden:

$$\sum_{i=1}^{\lceil \log n \rceil} 2^i \leq 2^{\lceil \log n \rceil + 1} \leq 2^{\log n + 2} \leq 4 \cdot 2^{\log n} = 4n$$

Geometrischen Reihe: $\sum_{k=0}^n r^k = \frac{r^{n+1}-1}{r-1} \leq r^{n+1}$ für $r \neq 1$ und $n \geq 0$

Nachrichtenkomplexität I

Wir nennen Knoten, die noch keine Follower sind, **aktiv**.

Nachrichtenkomplexität I

Wir nennen Knoten, die noch keine Follower sind, **aktiv**.

Lemma

Für jeden aktiven Knoten v gilt am Ende von Phase i : alle anderen Knoten in Distanz bis zu 2^{i-1} sind inaktiv.

Nachrichtenkomplexität I

Wir nennen Knoten, die noch keine Follower sind, **aktiv**.

Lemma

Für jeden aktiven Knoten v gilt am Ende von Phase i : alle anderen Knoten in Distanz bis zu 2^{i-1} sind inaktiv.

Beweis:

- Angenommen es gibt Knoten w in Distanz höchstens 2^{i-1} , der am Ende von Phase i aktiv ist

Nachrichtenkomplexität I

Wir nennen Knoten, die noch keine Follower sind, **aktiv**.

Lemma

Für jeden aktiven Knoten v gilt am Ende von Phase i : alle anderen Knoten in Distanz bis zu 2^{i-1} sind inaktiv.

Beweis:

- Angenommen es gibt Knoten w in Distanz höchstens 2^{i-1} , der am Ende von Phase i aktiv ist
- Dann war w auch am Anfang von Phase i aktiv

Nachrichtenkomplexität I

Wir nennen Knoten, die noch keine Follower sind, **aktiv**.

Lemma

Für jeden aktiven Knoten v gilt am Ende von Phase i : alle anderen Knoten in Distanz bis zu 2^{i-1} sind inaktiv.

Beweis:

- Angenommen es gibt Knoten w in Distanz höchstens 2^{i-1} , der am Ende von Phase i aktiv ist
- Dann war w auch am Anfang von Phase i aktiv
- Wenn $ID(v) > ID(w)$: v erhält in Phase i Nachricht mit $ID(w)$ (da w am Anfang der Phase aktiv war) und wird daher inaktiv: Widerspruch zur Annahme, dass v am Ende von Phase i aktiv ist

Nachrichtenkomplexität I

Wir nennen Knoten, die noch keine Follower sind, **aktiv**.

Lemma

Für jeden aktiven Knoten v gilt am Ende von Phase i : alle anderen Knoten in Distanz bis zu 2^{i-1} sind inaktiv.

Beweis:

- Angenommen es gibt Knoten w in Distanz höchstens 2^{i-1} , der am Ende von Phase i aktiv ist
- Dann war w auch am Anfang von Phase i aktiv
- Wenn $ID(v) > ID(w)$: v erhält in Phase i Nachricht mit $ID(w)$ (da w am Anfang der Phase aktiv war) und wird daher inaktiv: Widerspruch zur Annahme, dass v am Ende von Phase i aktiv ist
- Wenn $ID(v) < ID(w)$: w erhält in Phase i Nachricht mit $ID(v)$ und wird daher inaktiv: Widerspruch zur Annahme, dass w am Ende von Phase i aktiv ist

Nachrichtenkomplexität II

Lemma

Für $i \geq 2$ ist die Anzahl aktiver Knoten am Beginn von Phase i höchstens $n/2^{i-2}$.

Nachrichtenkomplexität II

Lemma

Für $i \geq 2$ ist die Anzahl aktiver Knoten am Beginn von Phase i höchstens $n/2^{i-2}$.

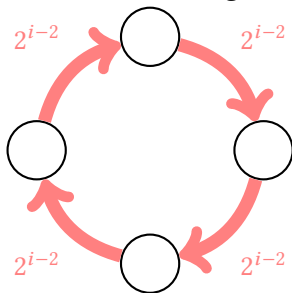
- Wegen vorigem Lemma: Jedem aktiven Knoten können eindeutig die 2^{i-2} nächsten inaktiven Knoten im Uhrzeigersinn zugeordnet werden

Nachrichtenkomplexität II

Lemma

Für $i \geq 2$ ist die Anzahl aktiver Knoten am Beginn von Phase i höchstens $n/2^{i-2}$.

- Wegen vorigem Lemma: Jedem aktiven Knoten können eindeutig die 2^{i-2} nächsten inaktiven Knoten im Uhrzeigersinn zugeordnet werden

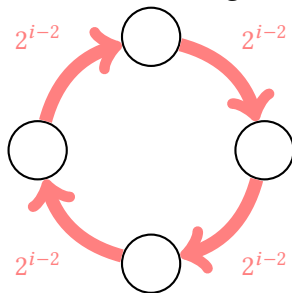


Nachrichtenkomplexität II

Lemma

Für $i \geq 2$ ist die Anzahl aktiver Knoten am Beginn von Phase i höchstens $n/2^{i-2}$.

- Wegen vorigem Lemma: Jedem aktiven Knoten können eindeutig die 2^{i-2} nächsten inaktiven Knoten im Uhrzeigersinn zugeordnet werden



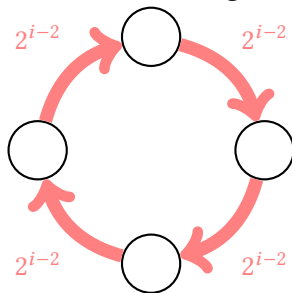
- Sei a die Anzahl aktiver Knoten

Nachrichtenkomplexität II

Lemma

Für $i \geq 2$ ist die Anzahl aktiver Knoten am Beginn von Phase i höchstens $n/2^{i-2}$.

- Wegen vorigem Lemma: Jedem aktiven Knoten können eindeutig die 2^{i-2} nächsten inaktiven Knoten im Uhrzeigersinn zugeordnet werden



- Sei a die Anzahl aktiver Knoten
- Dann gilt $a \cdot 2^{i-2} \leq n$ und somit $a \leq n/2^{i-2}$

Nachrichtenkomplexität III

Lemma

Für $i \geq 2$ ist die Anzahl aktiver Knoten am Beginn von Phase i höchstens $n/2^{i-2}$.

Nachrichtenkomplexität III

Lemma

Für $i \geq 2$ ist die Anzahl aktiver Knoten am Beginn von Phase i höchstens $n/2^{i-2}$.

Theorem

Der Radius Growth Algorithmus sendet höchstens $O(n \log n)$ Nachrichten.

Nachrichtenkomplexität III

Lemma

Für $i \geq 2$ ist die Anzahl aktiver Knoten am Beginn von Phase i höchstens $n/2^{i-2}$.

Theorem

Der Radius Growth Algorithmus sendet höchstens $O(n \log n)$ Nachrichten.

Beweis:

- In jeder Phase initiiert jeder aktive Knoten zwei Nachrichten, diese werden in jeder Runde der Phase einmal weitergeleitet

Nachrichtenkomplexität III

Lemma

Für $i \geq 2$ ist die Anzahl aktiver Knoten am Beginn von Phase i höchstens $n/2^{i-2}$.

Theorem

Der Radius Growth Algorithmus sendet höchstens $O(n \log n)$ Nachrichten.

Beweis:

- In jeder Phase initiiert jeder aktive Knoten zwei Nachrichten, diese werden in jeder Runde der Phase einmal weitergeleitet
- Anzahl an Nachrichten in Phase $i \geq 2$: $\leq \frac{n}{2^{i-2}} \cdot 2 \cdot 2^{i-1} = 4n$

Nachrichtenkomplexität III

Lemma

Für $i \geq 2$ ist die Anzahl aktiver Knoten am Beginn von Phase i höchstens $n/2^{i-2}$.

Theorem

Der Radius Growth Algorithmus sendet höchstens $O(n \log n)$ Nachrichten.

Beweis:

- In jeder Phase initiiert jeder aktive Knoten zwei Nachrichten, diese werden in jeder Runde der Phase einmal weitergeleitet
- Anzahl an Nachrichten in Phase $i \geq 2$: $\leq \frac{n}{2^{i-2}} \cdot 2 \cdot 2^{i-1} = 4n$
- Anzahl an Nachrichten in Phase 1: $2n$

Nachrichtenkomplexität III

Lemma

Für $i \geq 2$ ist die Anzahl aktiver Knoten am Beginn von Phase i höchstens $n/2^{i-2}$.

Theorem

Der Radius Growth Algorithmus sendet höchstens $O(n \log n)$ Nachrichten.

Beweis:

- In jeder Phase initiiert jeder aktive Knoten zwei Nachrichten, diese werden in jeder Runde der Phase einmal weitergeleitet
- Anzahl an Nachrichten in Phase $i \geq 2$: $\leq \frac{n}{2^{i-2}} \cdot 2 \cdot 2^{i-1} = 4n$
- Anzahl an Nachrichten in Phase 1: $2n$
- $\lceil \log n \rceil$ Phasen, daher insgesamt $O(n \log n)$ Nachrichten

Radius Growth:

- Bestimmt Knoten mit kleinster ID zum Leader
- Laufzeit: $O(n)$
- Nachrichtenkomplexität: $O(n \log n)$
- Kann auch als asynchroner Algorithmus in uniformen Ringen formuliert werden

Randomisierter Algorithmus [Attiya/Welch '98]

Annahmen: synchron, anonym, non-uniform

Randomisierter Algorithmus [Attiya/Welch '98]

Annahmen: synchron, anonym, non-uniform

Algorithmus:

- 1 Jeder Knoten wählt uniform zufällige ID von 0 bis $2n - 1$
- 2 Knoten führen Clockwise Algorithmus mit gewählten IDs aus

Randomisierter Algorithmus [Attiya/Welch '98]

Annahmen: synchron, anonym, non-uniform

Algorithmus:

- 1 Jeder Knoten wählt uniform zufällige ID von 0 bis $2n - 1$
- 2 Knoten führen Clockwise Algorithmus mit gewählten IDs aus

Problem:

- ▶ IDs können mehrfach vergeben sein
- ▶ Algorithmus könnte falsches Ergebnis liefern
- ▶ Clockwise macht alle Knoten mit kleinster vergebener ID zu Leadern

Randomisierter Algorithmus [Attiya/Welch '98]

Annahmen: synchron, anonym, non-uniform

Algorithmus:

- ① Jeder Knoten wählt uniform zufällige ID von 0 bis $2n - 1$
- ② Knoten führen Clockwise Algorithmus mit gewählten IDs aus

Problem:

- ▶ IDs können mehrfach vergeben sein
 - ▶ Algorithmus könnte falsches Ergebnis liefern
 - ▶ Clockwise macht alle Knoten mit kleinster vergebener ID zu Leadern
- ③ Knoten verifizieren, ob Ergebnis korrekt ist

Randomisierter Algorithmus [Attiya/Welch '98]

Annahmen: synchron, anonym, non-uniform

Algorithmus:

- 1 Jeder Knoten wählt uniform zufällige ID von 0 bis $2n - 1$
- 2 Knoten führen Clockwise Algorithmus mit gewählten IDs aus

Problem:

- ▶ IDs können mehrfach vergeben sein
 - ▶ Algorithmus könnte falsches Ergebnis liefern
 - ▶ Clockwise macht alle Knoten mit kleinster vergebener ID zu Leadern
- 3 Knoten verifizieren, ob Ergebnis korrekt ist
- Gültigkeit einer Einteilung in Leader und Follower kann in $O(n)$ Runden mit $O(n)$ Nachrichten überprüft werden

Randomisierter Algorithmus [Attiya/Welch '98]

Annahmen: synchron, anonym, non-uniform

Algorithmus:

- 1 Jeder Knoten wählt uniform zufällige ID von 0 bis $2n - 1$
- 2 Knoten führen Clockwise Algorithmus mit gewählten IDs aus

Problem:

- ▶ IDs können mehrfach vergeben sein
 - ▶ Algorithmus könnte falsches Ergebnis liefern
 - ▶ Clockwise macht alle Knoten mit kleinster vergebener ID zu Leadern
- 3 Knoten verifizieren, ob Ergebnis korrekt ist
Gültigkeit einer Einteilung in Leader und Follower kann in $O(n)$ Runden mit $O(n)$ Nachrichten überprüft werden
 - 4 Falls Ergebnis nicht korrekt, wiederhole ab Schritt 1

Binomialverteilung

Definition

Ein Zufallsexperiment, das nur zwei mögliche Ergebnisse – Erfolg mit Wahrscheinlichkeit p und Misserfolg mit Wahrscheinlichkeit $1 - p$ – hat, heißt **Bernoulli-Experiment**.

Binomialverteilung

Definition

Ein Zufallsexperiment, das nur zwei mögliche Ergebnisse – Erfolg mit Wahrscheinlichkeit p und Misserfolg mit Wahrscheinlichkeit $1 - p$ – hat, heißt **Bernoulli-Experiment**.

Definition

Eine **Bernoulli-Kette** der Länge n ist eine n -fache Wiederholung eines Bernoulli-Experimente mit der jeweils gleichen Erfolgswahrscheinlichkeit p .

Binomialverteilung

Definition

Ein Zufallsexperiment, das nur zwei mögliche Ergebnisse – Erfolg mit Wahrscheinlichkeit p und Misserfolg mit Wahrscheinlichkeit $1 - p$ – hat, heißt **Bernoulli-Experiment**.

Definition

Eine **Bernoulli-Kette** der Länge n ist eine n -fache Wiederholung eines Bernoulli-Experimente mit der jeweils gleichen Erfolgswahrscheinlichkeit p .

Definition

Sei X die Zufallsvariable, die angibt wie viele Erfolge in einer Bernoulli-Kette der Länge n erzielt werden. Dann wird die Wahrscheinlichkeitsverteilung von X **Binomialverteilung** mit Parametern n und p genannt.

Binomialverteilung

Definition

Ein Zufallsexperiment, das nur zwei mögliche Ergebnisse – Erfolg mit Wahrscheinlichkeit p und Misserfolg mit Wahrscheinlichkeit $1 - p$ – hat, heißt **Bernoulli-Experiment**.

Definition

Eine **Bernoulli-Kette** der Länge n ist eine n -fache Wiederholung eines Bernoulli-Experimente mit der jeweils gleichen Erfolgswahrscheinlichkeit p .

Definition

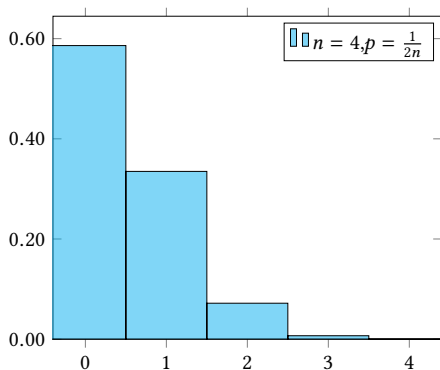
Sei X die Zufallsvariable, die angibt wie viele Erfolge in einer Bernoulli-Kette der Länge n erzielt werden. Dann wird die Wahrscheinlichkeitsverteilung von X **Binomialverteilung** mit Parametern n und p genannt.

Theorem

$$\Pr[X = k] = \binom{n}{k} p^k (1 - p)^{n-k} \text{ für alle } k \geq 0$$

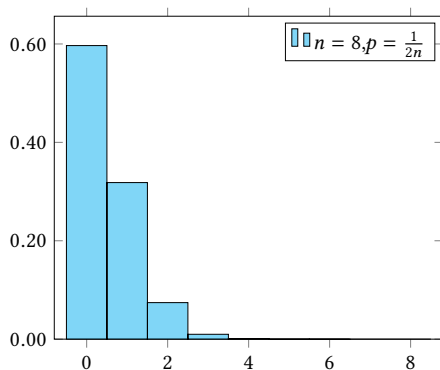
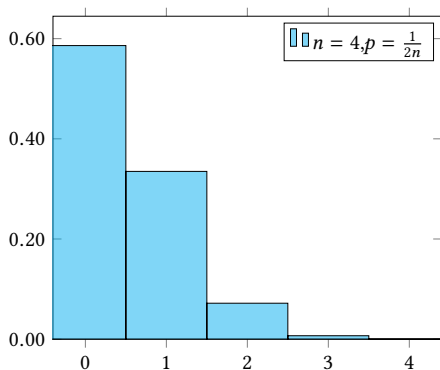
Beispiel Binomialverteilung

Wahrscheinlichkeitsverteilung für k Erfolge in Bernoulli-Kette der Länge n mit Einzel-Erfolgswahrscheinlichkeit p :



Beispiel Binomialverteilung

Wahrscheinlichkeitsverteilung für k Erfolge in Bernoulli-Kette der Länge n mit Einzel-Erfolgswahrscheinlichkeit p :



Erfolgswahrscheinlichkeit einer Iteration

Beobachtung

Eine Iteration des Algorithmus ist genau dann erfolgreich, wenn die kleinste vergebene ID genau einmal vergeben wurde.

Erfolgswahrscheinlichkeit einer Iteration

Beobachtung

Eine Iteration des Algorithmus ist genau dann erfolgreich, wenn die kleinste vergebene ID genau einmal vergeben wurde.

Beobachtung

Wenn genau ein Knoten die ID 0 erhält, dann ist die Iteration erfolgreich.

Erfolgswahrscheinlichkeit einer Iteration

Beobachtung

Eine Iteration des Algorithmus ist genau dann erfolgreich, wenn die kleinste vergebene ID genau einmal vergeben wurde.

Beobachtung

Wenn genau ein Knoten die ID 0 erhält, dann ist die Iteration erfolgreich.

- Jeder Knoten erhält ID 0 mit Wahrscheinlichkeit $p := \frac{1}{2n}$ (uniform)

Erfolgswahrscheinlichkeit einer Iteration

Beobachtung

Eine Iteration des Algorithmus ist genau dann erfolgreich, wenn die kleinste vergebene ID genau einmal vergeben wurde.

Beobachtung

Wenn genau ein Knoten die ID 0 erhält, dann ist die Iteration erfolgreich.

- Jeder Knoten erhält ID 0 mit Wahrscheinlichkeit $p := \frac{1}{2n}$ (uniform)
- Sei X Anzahl der Knoten mit ID 0 (Binomialverteilung):

$$q := \Pr[X = 1] = n \cdot p(1 - p)^{n-1}$$

Erfolgswahrscheinlichkeit einer Iteration

Beobachtung

Eine Iteration des Algorithmus ist genau dann erfolgreich, wenn die kleinste vergebene ID genau einmal vergeben wurde.

Beobachtung

Wenn genau ein Knoten die ID 0 erhält, dann ist die Iteration erfolgreich.

- Jeder Knoten erhält ID 0 mit Wahrscheinlichkeit $p := \frac{1}{2n}$ (uniform)
- Sei X Anzahl der Knoten mit ID 0 (Binomialverteilung):

$$q := \Pr[X = 1] = n \cdot p(1 - p)^{n-1} = \frac{1}{2} \cdot \left(1 - \frac{1}{2n}\right)^{n-1}$$

Erfolgswahrscheinlichkeit einer Iteration

Beobachtung

Eine Iteration des Algorithmus ist genau dann erfolgreich, wenn die kleinste vergebene ID genau einmal vergeben wurde.

Beobachtung

Wenn genau ein Knoten die ID 0 erhält, dann ist die Iteration erfolgreich.

- Jeder Knoten erhält ID 0 mit Wahrscheinlichkeit $p := \frac{1}{2n}$ (uniform)
- Sei X Anzahl der Knoten mit ID 0 (Binomialverteilung):

$$q := \Pr[X = 1] = n \cdot p(1-p)^{n-1} = \frac{1}{2} \cdot \left(1 - \frac{1}{2n}\right)^{n-1} \geq \frac{1}{2} \cdot \left(1 - \frac{n-1}{2n}\right)$$

Bernoulli-Ungleichung: $(1+x)^n \geq 1+xn$ für $x \geq -1$ und $n \geq 0$

Erfolgswahrscheinlichkeit einer Iteration

Beobachtung

Eine Iteration des Algorithmus ist genau dann erfolgreich, wenn die kleinste vergebene ID genau einmal vergeben wurde.

Beobachtung

Wenn genau ein Knoten die ID 0 erhält, dann ist die Iteration erfolgreich.

- Jeder Knoten erhält ID 0 mit Wahrscheinlichkeit $p := \frac{1}{2n}$ (uniform)
- Sei X Anzahl der Knoten mit ID 0 (Binomialverteilung):

$$\begin{aligned} q := \Pr[X = 1] &= n \cdot p(1-p)^{n-1} = \frac{1}{2} \cdot \left(1 - \frac{1}{2n}\right)^{n-1} \geq \frac{1}{2} \cdot \left(1 - \frac{n-1}{2n}\right) \\ &\geq \frac{1}{2} \cdot \left(1 - \frac{n}{2n}\right) \end{aligned}$$

Bernoulli-Ungleichung: $(1+x)^n \geq 1+xn$ für $x \geq -1$ und $n \geq 0$

Erfolgswahrscheinlichkeit einer Iteration

Beobachtung

Eine Iteration des Algorithmus ist genau dann erfolgreich, wenn die kleinste vergebene ID genau einmal vergeben wurde.

Beobachtung

Wenn genau ein Knoten die ID 0 erhält, dann ist die Iteration erfolgreich.

- Jeder Knoten erhält ID 0 mit Wahrscheinlichkeit $p := \frac{1}{2n}$ (uniform)
- Sei X Anzahl der Knoten mit ID 0 (Binomialverteilung):

$$\begin{aligned} q := \Pr[X = 1] &= n \cdot p(1-p)^{n-1} = \frac{1}{2} \cdot \left(1 - \frac{1}{2n}\right)^{n-1} \geq \frac{1}{2} \cdot \left(1 - \frac{n-1}{2n}\right) \\ &\geq \frac{1}{2} \cdot \left(1 - \frac{n}{2n}\right) = \frac{1}{4} \end{aligned}$$

Bernoulli-Ungleichung: $(1+x)^n \geq 1+xn$ für $x \geq -1$ und $n \geq 0$

Erfolgswahrscheinlichkeit einer Iteration

Beobachtung

Eine Iteration des Algorithmus ist genau dann erfolgreich, wenn die kleinste vergebene ID genau einmal vergeben wurde.

Beobachtung

Wenn genau ein Knoten die ID 0 erhält, dann ist die Iteration erfolgreich.

- Jeder Knoten erhält ID 0 mit Wahrscheinlichkeit $p := \frac{1}{2n}$ (uniform)
- Sei X Anzahl der Knoten mit ID 0 (Binomialverteilung):

$$\begin{aligned} q := \Pr[X = 1] &= n \cdot p(1-p)^{n-1} = \frac{1}{2} \cdot \left(1 - \frac{1}{2n}\right)^{n-1} \geq \frac{1}{2} \cdot \left(1 - \frac{n-1}{2n}\right) \\ &\geq \frac{1}{2} \cdot \left(1 - \frac{n}{2n}\right) = \frac{1}{4} \end{aligned}$$

Bernoulli-Ungleichung: $(1+x)^n \geq 1+xn$ für $x \geq -1$ und $n \geq 0$

- **Somit:** Jede Iteration mit Wahrscheinlichkeit $q \geq \frac{1}{4}$ erfolgreich

Geometrische Verteilung

Definition

Sei Y die Zufallsvariable, die angibt nach wie vielen Wiederholungen eines Bernoulli-Experiments mit Erfolgswahrscheinlichkeit q der erste Erfolg erzielt wird. Dann ist die Wahrscheinlichkeitsverteilung von Y **geometrische Verteilung** mit Parameter q genannt.

Geometrische Verteilung

Definition

Sei Y die Zufallsvariable, die angibt nach wie vielen Wiederholungen eines Bernoulli-Experiments mit Erfolgswahrscheinlichkeit q der erste Erfolg erzielt wird. Dann ist die Wahrscheinlichkeitsverteilung von Y **geometrische Verteilung** mit Parameter q genannt.

Theorem

$$\Pr[Y = k] = (1 - q)^{k-1}q \text{ für alle } k \geq 1$$

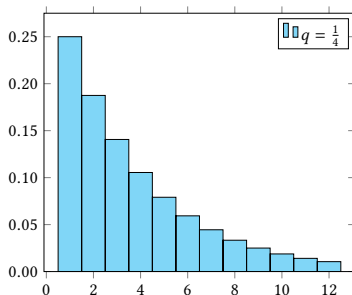
Geometrische Verteilung

Definition

Sei Y die Zufallsvariable, die angibt nach wie vielen Wiederholungen eines Bernoulli-Experiments mit Erfolgswahrscheinlichkeit q der erste Erfolg erzielt wird. Dann ist die Wahrscheinlichkeitsverteilung von Y **geometrische Verteilung** mit Parameter q genannt.

Theorem

$$\Pr[Y = k] = (1 - q)^{k-1}q \text{ für alle } k \geq 1$$



Korrektheit

Theorem

Der randomisierte Leader Election Algorithmus terminiert mit Wahrscheinlichkeit 1.

Korrektheit

Theorem

Der randomisierte Leader Election Algorithmus terminiert mit Wahrscheinlichkeit 1.

Beweis:

- Der Algorithmus terminiert, wenn die Anzahl an Iterationen endlich ist

Korrektheit

Theorem

Der randomisierte Leader Election Algorithmus terminiert mit Wahrscheinlichkeit 1.

Beweis:

- Der Algorithmus terminiert, wenn die Anzahl an Iterationen endlich ist
- Analyse mit geometrischer Verteilung: Sei Y Anzahl an Iterationen bis genau ein Knoten ID 0 erhält (Wiederholungen bis zum ersten Erfolg)

Korrektheit

Theorem

Der randomisierte Leader Election Algorithmus terminiert mit Wahrscheinlichkeit 1.

Beweis:

- Der Algorithmus terminiert, wenn die Anzahl an Iterationen endlich ist
- Analyse mit geometrischer Verteilung: Sei Y Anzahl an Iterationen bis genau ein Knoten ID 0 erhält (Wiederholungen bis zum ersten Erfolg)

$$\Pr \left[\bigvee_{k \geq 1} Y = k \right]$$

Theorem

Der randomisierte Leader Election Algorithmus terminiert mit Wahrscheinlichkeit 1.

Beweis:

- Der Algorithmus terminiert, wenn die Anzahl an Iterationen endlich ist
- Analyse mit geometrischer Verteilung: Sei Y Anzahl an Iterationen bis genau ein Knoten ID 0 erhält (Wiederholungen bis zum ersten Erfolg)

$$\Pr \left[\bigvee_{k \geq 1} Y = k \right] = \sum_{k \geq 1} \Pr[Y = k]$$

Korrektheit

Theorem

Der randomisierte Leader Election Algorithmus terminiert mit Wahrscheinlichkeit 1.

Beweis:

- Der Algorithmus terminiert, wenn die Anzahl an Iterationen endlich ist
- Analyse mit geometrischer Verteilung: Sei Y Anzahl an Iterationen bis genau ein Knoten ID 0 erhält (Wiederholungen bis zum ersten Erfolg)

$$\Pr \left[\bigvee_{k \geq 1} Y = k \right] = \sum_{k \geq 1} \Pr[Y = k] = \sum_{k \geq 1} (1 - q)^{k-1} q$$

Theorem

Der randomisierte Leader Election Algorithmus terminiert mit Wahrscheinlichkeit 1.

Beweis:

- Der Algorithmus terminiert, wenn die Anzahl an Iterationen endlich ist
- Analyse mit geometrischer Verteilung: Sei Y Anzahl an Iterationen bis genau ein Knoten ID 0 erhält (Wiederholungen bis zum ersten Erfolg)

$$\Pr \left[\bigvee_{k \geq 1} Y = k \right] = \sum_{k \geq 1} \Pr[Y = k] = \sum_{k \geq 1} (1 - q)^{k-1} q = q \cdot \sum_{k \geq 1} (1 - q)^{k-1}$$

Theorem

Der randomisierte Leader Election Algorithmus terminiert mit Wahrscheinlichkeit 1.

Beweis:

- Der Algorithmus terminiert, wenn die Anzahl an Iterationen endlich ist
- Analyse mit geometrischer Verteilung: Sei Y Anzahl an Iterationen bis genau ein Knoten ID 0 erhält (Wiederholungen bis zum ersten Erfolg)

$$\begin{aligned}\Pr\left[\bigvee_{k \geq 1} Y = k\right] &= \sum_{k \geq 1} \Pr[Y = k] = \sum_{k \geq 1} (1 - q)^{k-1} q = q \cdot \sum_{k \geq 1} (1 - q)^{k-1} \\ &= q \cdot \sum_{k \geq 0} (1 - q)^k\end{aligned}$$

Korrektheit

Theorem

Der randomisierte Leader Election Algorithmus terminiert mit Wahrscheinlichkeit 1.

Beweis:

- Der Algorithmus terminiert, wenn die Anzahl an Iterationen endlich ist
- Analyse mit geometrischer Verteilung: Sei Y Anzahl an Iterationen bis genau ein Knoten ID 0 erhält (Wiederholungen bis zum ersten Erfolg)

$$\begin{aligned}\Pr\left[\bigvee_{k \geq 1} Y = k\right] &= \sum_{k \geq 1} \Pr[Y = k] = \sum_{k \geq 1} (1 - q)^{k-1} q = q \cdot \sum_{k \geq 1} (1 - q)^{k-1} \\ &= q \cdot \sum_{k \geq 0} (1 - q)^k = q \cdot \frac{1}{1 - (1 - q)}\end{aligned}$$

Geometrische Reihe: $\sum_{k \geq 0} r^k = \frac{1}{1-r}$ für $-1 < r < 1$

Theorem

Der randomisierte Leader Election Algorithmus terminiert mit Wahrscheinlichkeit 1.

Beweis:

- Der Algorithmus terminiert, wenn die Anzahl an Iterationen endlich ist
- Analyse mit geometrischer Verteilung: Sei Y Anzahl an Iterationen bis genau ein Knoten ID 0 erhält (Wiederholungen bis zum ersten Erfolg)

$$\begin{aligned}\Pr\left[\bigvee_{k \geq 1} Y = k\right] &= \sum_{k \geq 1} \Pr[Y = k] = \sum_{k \geq 1} (1 - q)^{k-1} q = q \cdot \sum_{k \geq 1} (1 - q)^{k-1} \\ &= q \cdot \sum_{k \geq 0} (1 - q)^k = q \cdot \frac{1}{1 - (1 - q)} = q \cdot \frac{1}{q}\end{aligned}$$

Geometrische Reihe: $\sum_{k \geq 0} r^k = \frac{1}{1-r}$ für $-1 < r < 1$

Korrektheit

Theorem

Der randomisierte Leader Election Algorithmus terminiert mit Wahrscheinlichkeit 1.

Beweis:

- Der Algorithmus terminiert, wenn die Anzahl an Iterationen endlich ist
- Analyse mit geometrischer Verteilung: Sei Y Anzahl an Iterationen bis genau ein Knoten ID 0 erhält (Wiederholungen bis zum ersten Erfolg)

$$\begin{aligned}\Pr\left[\bigvee_{k \geq 1} Y = k\right] &= \sum_{k \geq 1} \Pr[Y = k] = \sum_{k \geq 1} (1 - q)^{k-1} q = q \cdot \sum_{k \geq 1} (1 - q)^{k-1} \\ &= q \cdot \sum_{k \geq 0} (1 - q)^k = q \cdot \frac{1}{1 - (1 - q)} = q \cdot \frac{1}{q} = 1\end{aligned}$$

Geometrische Reihe: $\sum_{k \geq 0} r^k = \frac{1}{1-r}$ für $-1 < r < 1$

Korrektheit

Theorem

Der randomisierte Leader Election Algorithmus terminiert mit Wahrscheinlichkeit 1.

Beweis:

- Der Algorithmus terminiert, wenn die Anzahl an Iterationen endlich ist
- Analyse mit geometrischer Verteilung: Sei Y Anzahl an Iterationen bis genau ein Knoten ID 0 erhält (Wiederholungen bis zum ersten Erfolg)

$$\begin{aligned}\Pr\left[\bigvee_{k \geq 1} Y = k\right] &= \sum_{k \geq 1} \Pr[Y = k] = \sum_{k \geq 1} (1 - q)^{k-1} q = q \cdot \sum_{k \geq 1} (1 - q)^{k-1} \\ &= q \cdot \sum_{k \geq 0} (1 - q)^k = q \cdot \frac{1}{1 - (1 - q)} = q \cdot \frac{1}{q} = 1\end{aligned}$$

Geometrische Reihe: $\sum_{k \geq 0} r^k = \frac{1}{1-r}$ für $-1 < r < 1$

Achtung: „Wahrscheinlichkeit 1“ \neq „immer“ – Gegenereignis ist möglich!

Waiting Time Bound

Theorem

Der Erwartungswert der geometrischen Verteilung ist $\text{Ex}[Y] = \frac{1}{q}$.

Waiting Time Bound

Theorem

Der Erwartungswert der geometrischen Verteilung ist $\text{Ex}[Y] = \frac{1}{q}$.

Beweis:

- Beobachtung: $\Pr[Y = k] = (1 - q)^{k-1}q$

Waiting Time Bound

Theorem

Der Erwartungswert der geometrischen Verteilung ist $\text{Ex}[Y] = \frac{1}{q}$.

Beweis:

- Beobachtung: $\Pr[Y = k] = (1 - q)^{k-1}q$
- Definition des Erwartungswerts:

$$\text{Ex}[Y] = \sum_{k \geq 1} k \cdot \Pr[Y = k]$$

Waiting Time Bound

Theorem

Der Erwartungswert der geometrischen Verteilung ist $\text{Ex}[Y] = \frac{1}{q}$.

Beweis:

- Beobachtung: $\Pr[Y = k] = (1 - q)^{k-1}q$
- Definition des Erwartungswerts:

$$\text{Ex}[Y] = \sum_{k \geq 1} k \cdot \Pr[Y = k] = \sum_{k \geq 1} k(1 - q)^{k-1}q$$

Waiting Time Bound

Theorem

Der Erwartungswert der geometrischen Verteilung ist $\text{Ex}[Y] = \frac{1}{q}$.

Beweis:

- Beobachtung: $\Pr[Y = k] = (1 - q)^{k-1}q$
- Definition des Erwartungswerts:

$$\text{Ex}[Y] = \sum_{k \geq 1} k \cdot \Pr[Y = k] = \sum_{k \geq 1} k(1 - q)^{k-1}q = \sum_{k \geq 0} (k + 1)(1 - q)^k q$$

Waiting Time Bound

Theorem

Der Erwartungswert der geometrischen Verteilung ist $\text{Ex}[Y] = \frac{1}{q}$.

Beweis:

- Beobachtung: $\Pr[Y = k] = (1 - q)^{k-1}q$
- Definition des Erwartungswerts:

$$\begin{aligned}\text{Ex}[Y] &= \sum_{k \geq 1} k \cdot \Pr[Y = k] = \sum_{k \geq 1} k(1 - q)^{k-1}q = \sum_{k \geq 0} (k + 1)(1 - q)^k q \\ &= \sum_{k \geq 0} k(1 - q)^k q + \sum_{k \geq 0} (1 - q)^k q\end{aligned}$$

Waiting Time Bound

Theorem

Der Erwartungswert der geometrischen Verteilung ist $\text{Ex}[Y] = \frac{1}{q}$.

Beweis:

- Beobachtung: $\Pr[Y = k] = (1 - q)^{k-1}q$
- Definition des Erwartungswerts:

$$\begin{aligned}\text{Ex}[Y] &= \sum_{k \geq 1} k \cdot \Pr[Y = k] = \sum_{k \geq 1} k(1 - q)^{k-1}q = \sum_{k \geq 0} (k + 1)(1 - q)^k q \\ &= \sum_{k \geq 1} k(1 - q)^k q + \sum_{k \geq 1} (1 - q)^k q \\ &= \sum_{k \geq 1} k(1 - q)^k q + \sum_{k \geq 0} (1 - q)^{k-1} q\end{aligned}$$

Waiting Time Bound

Theorem

Der Erwartungswert der geometrischen Verteilung ist $\text{Ex}[Y] = \frac{1}{q}$.

Beweis:

- Beobachtung: $\Pr[Y = k] = (1 - q)^{k-1}q$
- Definition des Erwartungswerts:

$$\begin{aligned}\text{Ex}[Y] &= \sum_{k \geq 1} k \cdot \Pr[Y = k] = \sum_{k \geq 1} k(1 - q)^{k-1}q = \sum_{k \geq 0} (k + 1)(1 - q)^k q \\ &= \sum_{k \geq 1} k(1 - q)^k q + \sum_{k \geq 0} (1 - q)^k q \\ &= \sum_{k \geq 1} k(1 - q)^k q + \sum_{k \geq 1} (1 - q)^{k-1} q \\ &= (1 - q) \sum_{k \geq 1} k(1 - q)^{k-1} q + \sum_{k \geq 1} \Pr[Y = k]\end{aligned}$$

Waiting Time Bound

Theorem

Der Erwartungswert der geometrischen Verteilung ist $\text{Ex}[Y] = \frac{1}{q}$.

Beweis:

- Beobachtung: $\Pr[Y = k] = (1 - q)^{k-1}q$
- Definition des Erwartungswerts:

$$\begin{aligned}\text{Ex}[Y] &= \sum_{k \geq 1} k \cdot \Pr[Y = k] = \sum_{k \geq 1} k(1 - q)^{k-1}q = \sum_{k \geq 0} (k + 1)(1 - q)^k q \\ &= \sum_{k \geq 1} k(1 - q)^k q + \sum_{k \geq 1} (1 - q)^k q \\ &= \sum_{k \geq 1} k(1 - q)^k q + \sum_{k \geq 1} (1 - q)^{k-1} q \\ &= (1 - q) \sum_{k \geq 1} k(1 - q)^{k-1} q + \sum_{k \geq 1} \Pr[Y = k]\end{aligned}$$

- **Somit:** $\text{Ex}[Y] = (1 - q) \text{Ex}[Y] + 1$

Waiting Time Bound

Theorem

Der Erwartungswert der geometrischen Verteilung ist $\text{Ex}[Y] = \frac{1}{q}$.

Beweis:

- Beobachtung: $\Pr[Y = k] = (1 - q)^{k-1}q$
- Definition des Erwartungswerts:

$$\begin{aligned}\text{Ex}[Y] &= \sum_{k \geq 1} k \cdot \Pr[Y = k] = \sum_{k \geq 1} k(1 - q)^{k-1}q = \sum_{k \geq 0} (k + 1)(1 - q)^k q \\ &= \sum_{k \geq 1} k(1 - q)^k q + \sum_{k \geq 1} (1 - q)^k q \\ &= \sum_{k \geq 1} k(1 - q)^k q + \sum_{k \geq 1} (1 - q)^{k-1} q \\ &= (1 - q) \sum_{k \geq 1} k(1 - q)^{k-1} q + \sum_{k \geq 1} \Pr[Y = k]\end{aligned}$$

- **Somit:** $\text{Ex}[Y] = (1 - q) \text{Ex}[Y] + 1$, also $\text{Ex}[Y] = \frac{1}{q}$

Laufzeitanalyse I

Theorem

Die erwartete Anzahl an Iterationen des randomisierten Leader Election Algorithmus ist $O(1)$.

Laufzeitanalyse I

Theorem

Die erwartete Anzahl an Iterationen des randomisierten Leader Election Algorithmus ist $O(1)$.

Beweis:

- Erfolgswahrscheinlichkeit $q \geq \frac{1}{4}$ in jeder Iteration

Laufzeitanalyse I

Theorem

Die erwartete Anzahl an Iterationen des randomisierten Leader Election Algorithmus ist $O(1)$.

Beweis:

- Erfolgswahrscheinlichkeit $q \geq \frac{1}{4}$ in jeder Iteration
- Mit Waiting Time Bound: erwartete Anzahl an Iterationen ist $\frac{1}{q} \leq 4 = O(1)$

Laufzeitanalyse I

Theorem

Die erwartete Anzahl an Iterationen des randomisierten Leader Election Algorithmus ist $O(1)$.

Beweis:

- Erfolgswahrscheinlichkeit $q \geq \frac{1}{4}$ in jeder Iteration
- Mit Waiting Time Bound: erwartete Anzahl an Iterationen ist $\frac{1}{q} \leq 4 = O(1)$

Frage: Erwartungswert erlaubt Ausreißer nach oben; ab wievielen Iterationen kann man sich „ziemlich sicher“ sein, dass der Algorithmus terminiert?

Laufzeitanalyse I

Theorem

Die erwartete Anzahl an Iterationen des randomisierten Leader Election Algorithmus ist $O(1)$.

Beweis:

- Erfolgswahrscheinlichkeit $q \geq \frac{1}{4}$ in jeder Iteration
- Mit Waiting Time Bound: erwartete Anzahl an Iterationen ist $\frac{1}{q} \leq 4 = O(1)$

Frage: Erwartungswert erlaubt Ausreißer nach oben; ab wievielen Iterationen kann man sich „ziemlich sicher“ sein, dass der Algorithmus terminiert?

Definition

Ein Ereignis findet **mit hoher Wahrscheinlichkeit** statt, wenn es mit Wahrscheinlichkeit mindestens $1 - \frac{1}{n^c}$, für eine beliebige gewählte Konstante $c \geq 1$, stattfindet.

Laufzeitanalyse I

Theorem

Die erwartete Anzahl an Iterationen des randomisierten Leader Election Algorithmus ist $O(1)$.

Beweis:

- Erfolgswahrscheinlichkeit $q \geq \frac{1}{4}$ in jeder Iteration
- Mit Waiting Time Bound: erwartete Anzahl an Iterationen ist $\frac{1}{q} \leq 4 = O(1)$

Frage: Erwartungswert erlaubt Ausreißer nach oben; ab wievielen Iterationen kann man sich „ziemlich sicher“ sein, dass der Algorithmus terminiert?

Definition

Ein Ereignis findet **mit hoher Wahrscheinlichkeit** statt, wenn es mit Wahrscheinlichkeit mindestens $1 - \frac{1}{n^c}$, für eine beliebige gewählte Konstante $c \geq 1$, stattfindet. (Insbesondere gilt: $\lim_{n \rightarrow \infty} 1 - \frac{1}{n^c} = 1$)

Laufzeitanalyse II

Theorem

Für jedes $c \geq 1$ gilt: Mit Wahrscheinlichkeit mindestens $1 - \frac{1}{n^c}$ benötigt der randomisierte Leader Election Algorithmus $O(c \log n)$ Iterationen.

Laufzeitanalyse II

Theorem

Für jedes $c \geq 1$ gilt: Mit Wahrscheinlichkeit mindestens $1 - \frac{1}{n^c}$ benötigt der randomisierte Leader Election Algorithmus $O(c \log n)$ Iterationen.

Beweis:

- Wir zeigen: $\Pr[Y \leq k] \leq 1 - \frac{1}{n^c}$ für $k = \lceil c \log_{4/3} n \rceil$

Theorem

Für jedes $c \geq 1$ gilt: Mit Wahrscheinlichkeit mindestens $1 - \frac{1}{n^c}$ benötigt der randomisierte Leader Election Algorithmus $O(c \log n)$ Iterationen.

Beweis:

- Wir zeigen: $\Pr[Y \leq k] \leq 1 - \frac{1}{n^c}$ für $k = \lceil c \log_{4/3} n \rceil$
- Wahrscheinlichkeit für Gegenereignis (erste k Versuche erfolglos):

$$\Pr[Y > k] = (1 - q)^k$$

Theorem

Für jedes $c \geq 1$ gilt: Mit Wahrscheinlichkeit mindestens $1 - \frac{1}{n^c}$ benötigt der randomisierte Leader Election Algorithmus $O(c \log n)$ Iterationen.

Beweis:

- Wir zeigen: $\Pr[Y \leq k] \leq 1 - \frac{1}{n^c}$ für $k = \lceil c \log_{4/3} n \rceil = \lceil \log_{4/3} n^c \rceil$
- Wahrscheinlichkeit für Gegenereignis (erste k Versuche erfolglos):

$$\Pr[Y > k] = (1 - q)^k$$

Laufzeitanalyse II

Theorem

Für jedes $c \geq 1$ gilt: Mit Wahrscheinlichkeit mindestens $1 - \frac{1}{n^c}$ benötigt der randomisierte Leader Election Algorithmus $O(c \log n)$ Iterationen.

Beweis:

- Wir zeigen: $\Pr[Y \leq k] \leq 1 - \frac{1}{n^c}$ für $k = \lceil c \log_{4/3} n \rceil = \lceil \log_{4/3} n^c \rceil$
- Wahrscheinlichkeit für Gegenereignis (erste k Versuche erfolglos):

$$\Pr[Y > k] = (1 - q)^k \leq \left(1 - \frac{1}{4}\right)^k$$

Laufzeitanalyse II

Theorem

Für jedes $c \geq 1$ gilt: Mit Wahrscheinlichkeit mindestens $1 - \frac{1}{n^c}$ benötigt der randomisierte Leader Election Algorithmus $O(c \log n)$ Iterationen.

Beweis:

- Wir zeigen: $\Pr[Y \leq k] \leq 1 - \frac{1}{n^c}$ für $k = \lceil c \log_{4/3} n \rceil = \lceil \log_{4/3} n^c \rceil$
- Wahrscheinlichkeit für Gegenereignis (erste k Versuche erfolglos):

$$\Pr[Y > k] = (1 - q)^k \leq \left(1 - \frac{1}{4}\right)^k \leq \left(\frac{3}{4}\right)^k$$

Theorem

Für jedes $c \geq 1$ gilt: Mit Wahrscheinlichkeit mindestens $1 - \frac{1}{n^c}$ benötigt der randomisierte Leader Election Algorithmus $O(c \log n)$ Iterationen.

Beweis:

- Wir zeigen: $\Pr[Y \leq k] \leq 1 - \frac{1}{n^c}$ für $k = \lceil c \log_{4/3} n \rceil = \lceil \log_{4/3} n^c \rceil$
- Wahrscheinlichkeit für Gegenereignis (erste k Versuche erfolglos):

$$\Pr[Y > k] = (1 - q)^k \leq \left(1 - \frac{1}{4}\right)^k \leq \left(\frac{3}{4}\right)^k \leq \left(\frac{3}{4}\right)^{\log_{4/3} n^c}$$

Theorem

Für jedes $c \geq 1$ gilt: Mit Wahrscheinlichkeit mindestens $1 - \frac{1}{n^c}$ benötigt der randomisierte Leader Election Algorithmus $O(c \log n)$ Iterationen.

Beweis:

- Wir zeigen: $\Pr[Y \leq k] \leq 1 - \frac{1}{n^c}$ für $k = \lceil c \log_{4/3} n \rceil = \lceil \log_{4/3} n^c \rceil$
- Wahrscheinlichkeit für Gegenereignis (erste k Versuche erfolglos):

$$\begin{aligned}\Pr[Y > k] &= (1 - q)^k \leq \left(1 - \frac{1}{4}\right)^k \leq \left(\frac{3}{4}\right)^k \leq \left(\frac{3}{4}\right)^{\log_{4/3} n^c} \\ &= \left(\frac{1}{4/3}\right)^{\log_{4/3} n^c}\end{aligned}$$

Theorem

Für jedes $c \geq 1$ gilt: Mit Wahrscheinlichkeit mindestens $1 - \frac{1}{n^c}$ benötigt der randomisierte Leader Election Algorithmus $O(c \log n)$ Iterationen.

Beweis:

- Wir zeigen: $\Pr[Y \leq k] \leq 1 - \frac{1}{n^c}$ für $k = \lceil c \log_{4/3} n \rceil = \lceil \log_{4/3} n^c \rceil$
- Wahrscheinlichkeit für Gegenereignis (erste k Versuche erfolglos):

$$\begin{aligned}\Pr[Y > k] &= (1 - q)^k \leq \left(1 - \frac{1}{4}\right)^k \leq \left(\frac{3}{4}\right)^k \leq \left(\frac{3}{4}\right)^{\log_{4/3} n^c} \\ &= \left(\frac{1}{4/3}\right)^{\log_{4/3} n^c} = \frac{1}{(4/3)^{\log_{4/3} n^c}}\end{aligned}$$

Theorem

Für jedes $c \geq 1$ gilt: Mit Wahrscheinlichkeit mindestens $1 - \frac{1}{n^c}$ benötigt der randomisierte Leader Election Algorithmus $O(c \log n)$ Iterationen.

Beweis:

- Wir zeigen: $\Pr[Y \leq k] \leq 1 - \frac{1}{n^c}$ für $k = \lceil c \log_{4/3} n \rceil = \lceil \log_{4/3} n^c \rceil$
- Wahrscheinlichkeit für Gegenereignis (erste k Versuche erfolglos):

$$\begin{aligned}\Pr[Y > k] &= (1 - q)^k \leq \left(1 - \frac{1}{4}\right)^k \leq \left(\frac{3}{4}\right)^k \leq \left(\frac{3}{4}\right)^{\log_{4/3} n^c} \\ &= \left(\frac{1}{4/3}\right)^{\log_{4/3} n^c} = \frac{1}{(4/3)^{\log_{4/3} n^c}} = \frac{1}{n^c}\end{aligned}$$

Theorem

Für jedes $c \geq 1$ gilt: Mit Wahrscheinlichkeit mindestens $1 - \frac{1}{n^c}$ benötigt der randomisierte Leader Election Algorithmus $O(c \log n)$ Iterationen.

Beweis:

- Wir zeigen: $\Pr[Y \leq k] \leq 1 - \frac{1}{n^c}$ für $k = \lceil c \log_{4/3} n \rceil = \lceil \log_{4/3} n^c \rceil$
- Wahrscheinlichkeit für Gegenereignis (erste k Versuche erfolglos):

$$\begin{aligned}\Pr[Y > k] &= (1 - q)^k \leq \left(1 - \frac{1}{4}\right)^k \leq \left(\frac{3}{4}\right)^k \leq \left(\frac{3}{4}\right)^{\log_{4/3} n^c} \\ &= \left(\frac{1}{4/3}\right)^{\log_{4/3} n^c} = \frac{1}{(4/3)^{\log_{4/3} n^c}} = \frac{1}{n^c}\end{aligned}$$

- **Somit:** $\Pr[Y \leq k] = 1 - \Pr[Y > k] \geq 1 - \frac{1}{n^c}$

Randomisierter Algorithmus :

- Wählt IDs zufällig
- Macht Knoten mit kleinster ID zum Leader
- Wiederholung bis Leader Election erfolgreich
- Laufzeit: $O(n)$ Runden in Erwartung
- Nachrichtenkomplexität: $O(n^2)$ in Erwartung Bei naiver Analyse mit Clockwise Algorithmus
- Kann auch als asynchroner Algorithmus formuliert werden

Allgemeines Prinzip

Wir haben implizit gezeigt:

Allgemeines Prinzip

Wir haben implizit gezeigt:

Theorem (Monte Carlo \rightarrow Las Vegas)

- *Sei \mathcal{A} ein randomisierter Algorithmus für ein Problem \mathcal{P} , der für jede Eingabe mit Wahrscheinlichkeit $q > 0$ eine korrekte Lösung berechnet.*

Allgemeines Prinzip

Wir haben implizit gezeigt:

Theorem (Monte Carlo \rightarrow Las Vegas)

- Sei \mathcal{A} ein randomisierter Algorithmus für ein Problem \mathcal{P} , der für jede Eingabe mit Wahrscheinlichkeit $q > 0$ eine korrekte Lösung berechnet.
- Sei \mathcal{V} ein Verifizierer, der für jede Ausgabe von \mathcal{A} ermitteln kann, ob die Lösung korrekt ist.

Allgemeines Prinzip

Wir haben implizit gezeigt:

Theorem (Monte Carlo \rightarrow Las Vegas)

- Sei \mathcal{A} ein randomisierter Algorithmus für ein Problem \mathcal{P} , der für jede Eingabe mit Wahrscheinlichkeit $q > 0$ eine korrekte Lösung berechnet.
- Sei \mathcal{V} ein Verifizierer, der für jede Ausgabe von \mathcal{A} ermitteln kann, ob die Lösung korrekt ist.
- Dann gibt es einen Algorithmus \mathcal{B} , der mit Wahrscheinlichkeit 1 eine korrekte Lösung berechnet und dafür \mathcal{A} und \mathcal{V} in Erwartung $O(1/q)$ Mal aufruft (oder: $O(c \log_{1-1/q} n)$ Mal mit Wahrscheinlichkeit $1 - \frac{1}{n^c}$).

Allgemeines Prinzip

Wir haben implizit gezeigt:

Theorem (Monte Carlo \rightarrow Las Vegas)

- Sei \mathcal{A} ein randomisierter Algorithmus für ein Problem \mathcal{P} , der für jede Eingabe mit Wahrscheinlichkeit $q > 0$ eine korrekte Lösung berechnet.
- Sei \mathcal{V} ein Verifizierer, der für jede Ausgabe von \mathcal{A} ermitteln kann, ob die Lösung korrekt ist.
- Dann gibt es einen Algorithmus \mathcal{B} , der mit Wahrscheinlichkeit 1 eine korrekte Lösung berechnet und dafür \mathcal{A} und \mathcal{V} in Erwartung $O(1/q)$ Mal aufruft (oder: $O(c \log_{1-1/q} n)$ Mal mit Wahrscheinlichkeit $1 - \frac{1}{n^c}$).

Definition

Ein randomisierter Algorithmus \mathcal{B} , dessen Ergebnis immer korrekt ist, heißt **Las Vegas** Algorithmus.

Allgemeines Prinzip

Wir haben implizit gezeigt:

Theorem (Monte Carlo \rightarrow Las Vegas)

- Sei \mathcal{A} ein randomisierter Algorithmus für ein Problem \mathcal{P} , der für jede Eingabe mit Wahrscheinlichkeit $q > 0$ eine korrekte Lösung berechnet.
- Sei \mathcal{V} ein Verifizierer, der für jede Ausgabe von \mathcal{A} ermitteln kann, ob die Lösung korrekt ist.
- Dann gibt es einen Algorithmus \mathcal{B} , der mit Wahrscheinlichkeit 1 eine korrekte Lösung berechnet und dafür \mathcal{A} und \mathcal{V} in Erwartung $O(1/q)$ Mal aufruft (oder: $O(c \log_{1-1/q} n)$ Mal mit Wahrscheinlichkeit $1 - \frac{1}{n^c}$).

Definition

Ein randomisierter Algorithmus \mathcal{B} , dessen Ergebnis immer korrekt ist, heißt **Las Vegas** Algorithmus. Ein randomisierter Algorithmus \mathcal{A} , dessen Ergebnis wahrscheinlich korrekt ist, heißt **Monte Carlo** Algorithmus.

Leader Election im Ring verdeutlicht grundlegende Prinzipien und Techniken verteilter Algorithmen:

Leader Election im Ring verdeutlicht grundlegende Prinzipien und Techniken verteilter Algorithmen:

- Vielfalt an Modellen: synchron/asynchron, anonym/identifizierbar, uniform/non-uniform

Leader Election im Ring verdeutlicht grundlegende Prinzipien und Techniken verteilter Algorithmen:

- Vielfalt an Modellen: synchron/asynchron, anonym/identifizierbar, uniform/non-uniform
- Deterministische vs. randomisierte Algorithmen

Leader Election im Ring verdeutlicht grundlegende Prinzipien und Techniken verteilter Algorithmen:

- Vielfalt an Modellen: synchron/asynchron, anonym/identifizierbar, uniform/non-uniform
- Deterministische vs. randomisierte Algorithmen
- Komplexitätsmaße: Zeit, #Nachrichten

Leader Election im Ring verdeutlicht grundlegende Prinzipien und Techniken verteilter Algorithmen:

- Vielfalt an Modellen: synchron/asynchron, anonym/identifizierbar, uniform/non-uniform
- Deterministische vs. randomisierte Algorithmen
- Komplexitätsmaße: Zeit, #Nachrichten
- Obere und untere Schranken (bzw. Unmöglichkeit)

Leader Election im Ring verdeutlicht grundlegende Prinzipien und Techniken verteilter Algorithmen:

- Vielfalt an Modellen: synchron/asynchron, anonym/identifizierbar, uniform/non-uniform
- Deterministische vs. randomisierte Algorithmen
- Komplexitätsmaße: Zeit, #Nachrichten
- Obere und untere Schranken (bzw. Unmöglichkeit)
- Zwei Möglichkeiten Symmetrien zu brechen:
 - ▶ Eindeutige IDs
 - ▶ Randomisierung

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

Literatur:

- Hagit Attiya, Jennifer Welch (2004) *Distributed Computing*, Kapitel 3 u. 14, Wiley.
- Daniel S. Hirschberg, James B. Sinclair. „Decentralized Extrema-Finding in Circular Configurations of Processors“. *Communications of the ACM* 23(11): 627–628 (1980)
- Alon Itai, Michael Rodeh. „Symmetry breaking in distributed networks“. *Information and Computation* 88(1): 60–87 (1990)
- Nancy A. Lynch (1996) *Distributed Algorithms*, Kapitel 3, Morgan Kaufmann.