Algorithmen für verteilte Systeme **Universität Salzburg**
Sommersemester 2018 **FB Computerwissenschaften**

Vorlesung 3: 20. März 2018

Vortrag: Sebastian Forster *Mitschrift: Sebastian Landl*

3.1 Leader Election II

3.1.1 Wiederholung

3.1.1.1 Motivation

Gesucht ist ein Algorithmus, der einen *Leader* bestimmt. Ein *Leader* ist ein Knoten in einem Netzwerk, auf den sich alle Knoten geeinigt haben. Alle anderen Knoten heißen *Follower*. Der Zweck eines solchen *Leaders* kann sein, dass dieser das Netzwerk koordinieren kann. Ein Problem bei der Suche nach einem Algorithmus, mit dem dieser *Leader* bestimmt werden kann, ist, dass sich jeder Knoten der n eines Netzwerks gleich gut als *Leader* eignet und es somit grundsätzlich n gleichwertige Möglichkeiten für den *Leader* gibt. Um dennoch genau einen Knoten zum *Leader* bestimmen zu können müssen sich die Knoten im Netzwerk auf eine der n gleichberechtigten Lösungen einigen.

3.1.1.2 Leader Election im Ring

In der letzten Vorlesung (Leader Election I) wurde festgestellt, dass es nicht möglich ist in einem anonymen Ring, also einem ringförmingen Netzwerk ohne eindeutige IDs, mithilfe eines deterministischen Algorithmus einen *Leader* zu bestimmen. In der heutigen Vorlesung werden wir einen randomisierten Algorithmus kennen lernen, um dieses Problem zu umgehen.

Weiters wurde der Clockwise Algorithmus vorgestellt, welcher $O(n)$ Runden und $O(n^2)$ Nachrichten benötigt, um einen *Leader* zu bestimmen. Ebenso haben wir einen nachrichteneffizienteren Algorithmus kennen gelernt, welcher mit gewissen Annahmen zwar $\Omega(n^2)$ Runden, aber dafür lediglich $O(n)$ Nachrichten benötigt. Es wird ein Algorithmus folgen, der die geringe Rundenanzahl und die geringe Nachrichtenkomplexität dieser beiden Algorithmen annäherend vereint und somit $O(n)$ Runden und $O(n \log n)$ Nachrichten benötigt.

3.2 Radius Growth

Dieser Algorithmus zur Leaderelection im Ring funktioniert nach folgendem Prinzip: jeder Knoten überprüft, ob er in einer gewissen Umgebung der *Leader* ist, was bedeutet, dass er schaut, ob er die größte ID hat. Wenn ja, so wird diese Umgebung vergrößert.

Die Ausgangssituation des Algorithmus ist, dass jeder der n Knoten als *Leader* infrage kommt. Im Laufe des Algorithmus werden sich dann $n - 1$ Knoten dazu entscheiden *Follower* zu sein und der

übrige Knoten wird der *Leader* sein. Knoten, die noch keine *Follower* sind nennen wir aktiv.

Die Runden werden in aufeinanderfolgende Phasen unterteilt, wobei die i-te Phase $2^{i-1} + 1$ Runden dauert. In der ersten Runde jeder Phase schickt jeder aktive Knoten seine ID an seine Nachbarn im Ring. In allen anderen Runden, wenn ein Knoten einen Nachricht erhält, sendet er diese weiter und entscheidet sich *Follower* zu sein, falls die empfangene ID größer als seine eigene ist. Falls die empfangene ID gleich seiner eigenen ist, so entscheidet sich der Knoten *Leader* zu werden, weil er eine Nachricht mit der eigenen ID erhalten hat.

Anmerkung: Selbst wenn es nur noch einen Knoten gibt, der noch kein *Follower* ist und es somit bereits klar ist, wer der *Leader* sein wird, muss der Knoten warten, bis er seine eigene ID erhalten hat, weil die Anzahl der Knoten im Netzwerk unbekannt ist.

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 im (gegen) den Uhrzeigersinn then
2     if M > ID(v) then
3         v entscheidet sich Follower zu sein (sofern nicht bereits vorher geschehen)
4         v sendet M an Nachbar gegen (im) Uhrzeigersinn
5     if M = ID(v) then
6         v entscheidet sich Leader zu sein
```

3.2.1 Korrektheit und Laufzeit

Theorem 3.1. *Der Radius Growth Algorithmus bestimmt in Phase* $\lceil \log n \rceil + 1$ *einen eindeutigen Leader.*

Beweis. Sei z der Knoten mit der höchsten ID.

- z muss in Phase $i = \lceil \log n \rceil + 1$ noch aktiv sein, weil er keine höhere ID gesehen haben kann, da er selbst die höchste ID besitzt und jede ID eindeutig, also nur einmal vergeben, ist.
- In dieser Phase schickt z eine Nachricht im Uhrzeigersinn los, welche $2^{i-1} = 2^{\lceil \log n \rceil} \geq 2^{\log n} = n$ viele Knoten erreicht, weil die Phase eine Länge von $2^{i-1} + 1$ Runden hat.
- Somit erhalten alle Knoten des Netzwerks diese Nachricht von z, was zur Folge hat, dass z *Leader* wird und alle anderen Knoten *Follower* werden.

□

Theorem 3.2. *Der Radius Growth Algorithmus benötigt $O(n)$ Runden.*

Beweis. Die Phase i benötigt $2^{i-1} + 1 \leq 2^i$ Runden. Somit beträgt die Gesamtanzahl an Runden: $\sum_{i=1}^{\lceil \log n \rceil + 1} 2^i \leq 2^{\lceil \log n \rceil + 2} \leq 2^{\log n + 3} \leq 8 \cdot 2^{\log n} = 8n$ □

Lemma 3.3. *Für jeden aktiven Knoten gilt am Ende von Phase i: alle anderen Knoten in Distanz bis zu 2^{i-1} sind inaktiv.*

Die Aussage dieses Lemmas ist, dass es am Ende von Phase i im Bereich 2^{i-1} einen eindeutigen *Leader* gibt.

Beweis. Wir führen einen Widerspruchsbeweis. Sei v ein Knoten, der am Ende von Phase i noch aktiv ist. Angenommen es gibt einen Knoten w mit einer Distanz höchstens 2^{i-1} vom Knoten v, der am Ende von Phase i noch aktiv ist. Dann war w war auch am Anfang von Phase i aktiv.

Fallunterscheidung:

- Fall $\text{ID}(w) < \text{ID}(v)$: w erhält in Phase i eine Nachricht mit $\text{ID}(v)$ und wird daher inaktiv $\Rightarrow$ Widerspruch zur Annahme, dass w am Ende von Phase i noch aktiv ist
- Fall $\text{ID}(w) > \text{ID}(v)$: v erhält in Phase i eine Nachricht mit $\text{ID}(w)$ und wird daher inaktiv $\Rightarrow$ Widerspruch zur Annahme, dass v am Ende von Phase i noch aktiv ist
- Der Fall $\text{ID}(w) = \text{ID}(v)$ kann nicht eintreten, da die IDs eindeutig sind.

□

Lemma 3.4. *Die Anzahl aktiver Knoten in Phase i ist höchstens $n/2^{i-2}$.*

Beweis. Von Lemma 3.3 wissen wir, dass jedem aktiven Knoten eindeutig 2^{i-2} inaktive Knoten im Uhrzeigersinn zugeordnet werden können.
Sei a die Anzahl an aktiven Knoten
$\Rightarrow a \cdot 2^{i-2} \leq n$, weil a multipliziert mit der Anzahl an a zugeordneten inaktiven Knoten nicht mehr als die Gesamtanzahl an Knoten im Ring sein kann
$\Rightarrow a \leq n/2^{i-2}$

□

Theorem 3.5. *Der Radius Growth Algorithmus sendet höchstens $O(n \log n)$ Nachrichten.*

Beweis. Zu Beginn jeder Phase schickt jeder aktive Knoten zwei Nachrichten los, eine im und eine gegen den Uhrzeigersinn. Diese Nachrichten werden in jeder Runde der Phase einmal weitergeleitet. Somit ist die Anzahl der Nachrichten in Phase $i \geq 2$: $\leq \frac{n}{2^{i-2}} \cdot 2 \cdot (2^{i-1} + 1) \leq 6n$ ($\frac{n}{2^{i-2}}$ ist die maximale Anzahl an aktiven Knoten (Lemma 3.4); jeder aktive Knoten schickt 2 Nachrichten; die Anzahl der Runden der Phase i beträgt $2^{i-1} + 1$ und in jeder dieser Runden werden die zu Beginn erzeugten Nachrichten einmal weitergeleitet).
Die Anzahl der Nachrichten in Phase 1 beträgt $2n$.
Nach $\lceil \log n \rceil + 1$ Phasen werden keine Nachrichten mehr verschickt. Da in jeder Phase höchstens $6n$

Nachrichten versendet werden und es $\lceil \log n \rceil + 1$ viele Phasen gibt, liegt die Nachrichtenkomplexität des Radius Growth Algorithmus in $O(n \log n)$.

□

3.2.2 Zusammenfassung

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

3.3 Randomisierter Algorithmus für anonyme Ringe

Anonymer Ring bedeutet, dass keine IDs vergeben sind. Jedoch treffen wir die Annahme, dass die Gesamtanzahl n an Knoten im Ring jedem Knoten bekannt ist.
Die Idee: Können wir IDs zufällig vergeben? Und einen Algorithmus folgender Form verwenden:

1. Jeder Knoten wählt uniform zufällige ID von 0 bis $n - 1$
2. Knoten führen Leader Election Algorithmus mit gewählten IDs aus
3. Knoten verifizieren das Ergebnis

Das Problem hierbei: wir haben keine Garantie, dass keine ID mehrfach vergeben wird und es somit mehrere Knoten mit der höchsten ID gibt, was zu einem falschen Ergebnis führen würde, weil dadurch mehr als ein einziger, eindeutiger *Leader* bestimmt werden würde. Um dieses Problem zu lösen, müssen die Knoten in Schritt 3 verifizieren, ob das Ergebnis, also die Einteilung in *Leader* und *Follower* korrekt ist (es also genau einen *Leader* und sonst nur *Follower* gibt), was in $O(n)$ Runden und mit $O(n)$ Nachrichten möglich ist (siehe Übungsaufgabe). Wenn das Ergebnis nicht korrekt ist, so muss wieder bei Schritt 1 angefangen werden. Diese Schritte werden so lange wiederholt, bis ein eindeutiger *Leader* bestimmt wurde.

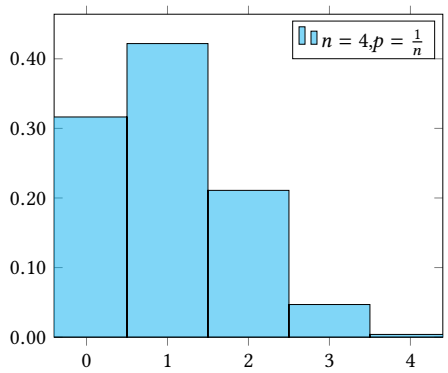
3.3.1 Bernoulli-Experimente

Definition 3.6 (Bernoulli-Experiment)**.** *Ein Zufallsexperiment, das nur zwei mögliche Ergebnisse (Erfolg mit Wahrscheinlichkeit p und Misserfolg mit Wahrscheinlichkeit $1 - p$) hat, heißt Bernoulli-Experiment.*

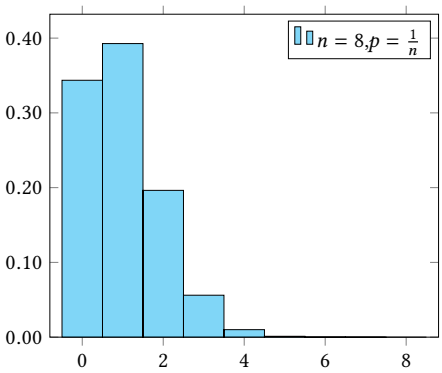
Definition 3.7 (Bernoulli-Kette)**.** *Eine Bernoulli-Kette der Länge n ist eine n-fache Wiederholung eines Bernoulli-Experimente mit der jeweils gleichen Erfolgswahrscheinlichkeit p.*

Theorem 3.8 (Binomialverteilung)**.** *Die Wahrscheinlichkeit, dass in einer Bernoulli-Kette der Länge n genau k Erfolge erzielt werden, ist $\binom{n}{k} p^k (1-p)^{n-k}$.*

Zur Veranschaulichung der Binominalverteilung folgen zwei Beispiele:
Die x-Achse entspricht den k erwünschten Erfolgen und die y-Achse entspricht der Wahrscheinlichkeit, dass genau diese k Erfolge eintreten.



Beim linken Diagramm werden $n = 4$ Bernoulli-Experimente durchgeführt. Jedes davon hat eine Erfolgswahrscheinlichkeit von $p = \frac{1}{n} = \frac{1}{4}$. Wenn die Erfolgswahrscheinlichkeit $p = \frac{1}{2}$ wäre, so wäre dieses Diagramm der Binominalverteilung symmetrisch. Da jedoch $p < \frac{1}{2}$, ist die Wahrscheilichkeit nach links verschoben, was bedeutet, dass es einfacher ist, Misserfolge zu haben.



Beim rechten Diagramm werden $n = 8$ Bernoulli-Experimente durchgeführt. Jedes davon hat eine Erfolgswahrscheinlichkeit von $p = \frac{1}{n} = \frac{1}{8}$. Diese geringe Erfolgswahrscheinlichkeit gemeinsam mit der höheren Anzahl an hintereinander ausgeführten Bernoulli-Experimenten sorgt dafür, dass die Wahrscheilichkeit noch mehr nach links verschoben ist.

3.3.2 Erfolgswahrscheinlichkeit des randomisierten Algorithmus

Eine Iteration des Algorithmus ist genau dann erfolgreich, wenn die höchste vergebene ID genau einmal vergeben wurde. In diesem Fall können Algorithmen, die wir bereits kennen gelernt haben (Clockwise, Radius Growth), einen eindeutigen *Leader* bestimmen und somit ein korrektes Ergebnis liefern.

Davon ausgehend die Erfolgswahrscheinlichkeit auszurechnen, ist jedoch etwas schwierig. Daher wenden wir folgenden Trick an: Wir wissen, dass wenn die ID $n-1$ genau einmal vergeben wird, daraus folgt, dass die Iteration erfolgreich ist. Nun lassen wir die Möglichkeit außer Acht, dass eine ID echt kleiner als $n-1$ nur einmal vergeben wird und die größte ID ist, was jedoch kein Problem ist, weil wir damit unsere Erfolgswahrscheinlichkeit nicht überschätzen. Darüber hinaus funktioniert diese Schlussfolgerung nur, wenn die zufällig gewählten IDs im Bereich 0 bis $n-1$ sind, was wir ganz am Anfang angenommen haben.

Wir betrachten nun das Wählen der IDs in einer Iteration als wiederholte Ausführung eines Bernoulli-Experiments, wobei n = Anzahl der Knoten viele Bernoulli-Experimente ausgeführt werden. Wir definieren die Wahl der ID $n-1$ als Erfolg und haben somit eine Erfolgswahrscheinlichkeit $p = \frac{1}{n}$.

Damit beträgt die Wahrscheinlichkeit q, dass genau ein Knoten die ID $n-1$ erhält:

$$q = n \cdot p(1-p)^{n-1} = \left(1 - \frac{1}{n}\right)^{n-1} \geq \frac{1}{e} \approx 0.36$$

weil gilt: $\forall x > 1 : \left(1 - \frac{1}{x}\right)^{x-1} \geq \frac{1}{e}$

Somit beträgt die Erfolgswahrscheinlichkeit unseres Algorithmus in jeder Iteration mindestens $q \geq \frac{1}{e}$.

3.3.3 Korrektheit

Theorem 3.9. *Der randomisierte Leader Election Algorithmus terminiert mit Wahrscheinlichkeit* 1.

Beweis. In jeder Iteration gibt es eine konstante Wahrscheinlichkeit $q > 0$, dass der Algorithmus erfolgreich ist und terminiert. Somit beträgt die Wahrscheinlichkeit, dass der Algorithmus nach k Iterationen noch nicht terminiert ist: $(1-q)^k$.

Wenn nun k gegen Unendlich geht, dann gilt: $\lim_{k\to\infty}(1-q)^k = 0$, da $q > 0$

Da die Gegenwahrscheinlichkeit zu 0 eine Wahrscheinlichkeit von 1 ist, gilt, dass der randomisierte Leader Election Algorithmus mit Wahrscheinlichkeit 1 terminiert.

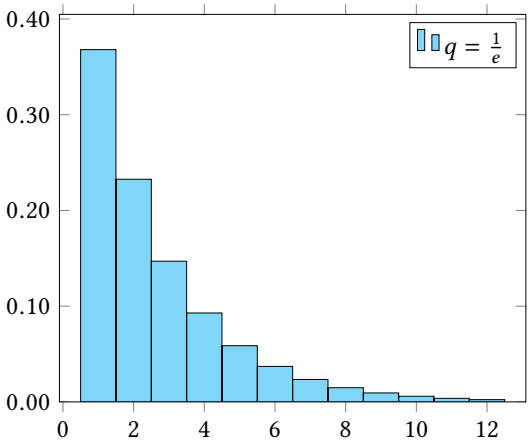
□

Jedoch ist dabei zu beachten, dass ”Wahrscheinlichkeit 1” nicht mit ”immer” gleichzusetzen ist; das Ereignis, nicht zu terminieren, ist prinzipiell möglich, auch wenn es Wahrscheinlichkeit 0 hat.

Unsere bisherige Analyse hat ergeben, dass dieser Algorithmus irgendwann terminiert. Kommen wir nun zu einer genaueren Analyse der Anzahl an Iterationen, die der Algorithmus benötigen wird.

Definition 3.10 (Geometrische Verteilung)**.** *Betrachte ein ein Bernoulli-Experiment mit Erfolgswahrscheinlichkeit q und sei X die Zufallsvariable, die angibt nach wie vielen Wiederholungen der erste Treffer erzielt wird. Dann wird die Wahrscheinlichkeitsverteilung von X die geometrische Verteilung mit Einzel-Erfolgswahrscheinlichkeit q genannt.*

Kommen wir erneut zu einem kurzen Beispiel:



Hier beträgt die Erfolgswahrscheinlichkeit $q = \frac{1}{e}$. Die Wahrscheinlichkeit bei der ersten Wiederholung gleich einen Treffer zu erzielen ist genau die Erfolgswahrscheinlichkeit. Die Wahrscheinlichkeit bei der Zweiten Wiederholung einen Treffer zu erzielen beträgt $(1-\frac{1}{e}) \cdot \frac{1}{e}$, bei der dritten Wiederholung $(1-\frac{1}{e})^2 \cdot \frac{1}{e}$ und so weiter. Dieses Diagramm ist nur bis 12 Wiederholungen gezeichnet, würde jedoch bis ins Unendliche weiter gehen, mit immer geringeren Wahrscheinlichkeiten.

Wir wollen einen Aussage treffen können, die etwa so aussieht: Der Erwartungswert der Anzahl an benötigten Iterationen ist x.

Theorem 3.11. *Der Erwartungswert der geometrischen Verteilung ist* $E[X] = \frac{1}{q}$.

Wenn wir als Beispiel einen Münzwurf betrachten, dann ist die Erfolgswahrscheinlichkeit $q = \frac{1}{2}$ für das gewünschte Ereignis "Kopf". Der Erwartungswert $E[X]$ wäre $\frac{1}{\frac{1}{2}} = 2$, was bedeuten würde, dass in Erwartung in zwei Münzwürfen das Ereignis "Kopf" eintritt, was der Intuition entspricht.

Beweis. Was ist die Wahrscheinlichkeit genau bei der k-ten Wiederholung erfolgreich zu sein?
$\Pr[X = k] = (1-q)^{k-1}q$, also die Wahrscheinlichkeit für $k-1$ Misserfolge und einen Erfolg.
Laut Definition des Erwartungswertes gilt $E[X] = \sum_{k=1}^{\infty} k \cdot \Pr[X = k]$.
(Summe über alle Werte der Zufallsvariable mal der dazugehörigen Wahrscheinlichkeit)
Wir setzen nun ein:

$$\begin{aligned}
E[X] = \sum_{k=1}^{\infty} k \cdot \Pr[X = k] &\overset{1.}{=} \sum_{k=1}^{\infty} k(1-q)^{k-1}q \overset{2.}{=} \sum_{k=0}^{\infty}(k+1)(1-q)^k q \\
&\overset{3.}{=} \sum_{k=0}^{\infty} k(1-q)^k q + \sum_{k=0}^{\infty}(1-q)^k q \\
&\overset{4.}{=} \sum_{k=1}^{\infty} k(1-q)^k q + \sum_{k=1}^{\infty}(1-q)^{k-1} q \\
&\overset{5.}{=} (1-q)\sum_{k=1}^{\infty} k(1-q)^{k-1}q + \sum_{k=1}^{\infty} \Pr[X = k]
\end{aligned}$$

1. Einsetzen
2. Index-Shift
3. Ausmultiplizieren von $(k+1)\cdot\ldots$ und Aufteilung in zwei Summen
4. Bei der ersten Summe: für $k = 0$ ist der Summand gleich 0, somit kann man diese Summe auch bei 1 beginnen lassen
   Bei der zweiten Summe: Index-Shift in die anderen Richtung
5. Bei der ersten Summe: Ausklammern von $1-q$
   Bei der zweiten Summe: die Definition von $\Pr[X = k]$ einsetzen

Nun entspricht die linke Summe der Definition des Erwartungswertes und die rechte Summe ist gleich 1 sein. *Warum?* Wir betrachten die Wahrscheinlichkeiten aller möglichen Ausprägungen von k, also die Wahrscheinlichkeiten, dass in der Runde k der erste Treffer erzielt wird, für alle Runden k. Diese Ereignisse sind disjunkt (der erste Treffer kann nicht in zwei unterschiedlichen Runden auftreten) und spannen der gesamten Ereignisraum auf (eines der Ereignisse *muss* auftreten, weil die Wartezeit eine natürliche Zahl größer gleich 1 sein muss) entspricht diese Summe 1.

Somit gilt: $E[X] = (1-q)E[X] + 1$, also $E[X] = \frac{1}{q}$

□

Theorem 3.12. *Die erwartete Anzahl an Iterationen des randomisierten Leader Election Algorithmus ist $O(1)$.*

Beweis. Wir wissen bereits, dass wir in jeder Iteration eine konstante Erfolgswahrscheinlichkeit von $q \geq \frac{1}{e}$ haben.

Aus Definition 3.10 und Theorem 3.11 folgt nun, dass die erwartete Anzahl an Iterationen $\frac{1}{q} \leq e$ beträgt, was in $O(1)$ liegt, weil e eine Konstante ist.

□

Manchmal reicht uns jedoch der Erwartungswert nicht, da eine große Varianz um diesen Erwartungswert herrschen könnte und es somit viele Ausreißer nach oben geben könnte. Und obwohl man in Erwartung eine konstante Anzahl an Iterationen hat, kann die Anzahl an Iterationen oft dennoch deutlich größer sein.

Daher untersucht man nicht nur wie viele Iterationen bei konstanter Wahrscheinlichkeit notwendig sind, sondern auch wie viele Iterationen "mit hoher Wahrscheinlichkeit" benötigt werden. Somit ist unser Maß an Erfolg keine konstante Wahrscheinlichkeit, sondern eine invers polynomielle Wahrscheinlichkeit.

Theorem 3.13. *Für jede Konstante c gilt: Mit Wahrscheinlichkeit mindestens $1 - \frac{1}{n^c}$ („mit hoher Wahrscheinlichkeit") benötigt der randomisierte Leader Election Algorithmus $O(\log n)$ Iterationen.*

Beweis. Die Wahrscheinlichkeit, dass der Algorithmus nach $k = \lceil ce \ln n \rceil$ (c ist eine beliebige Konstante und e wurde so gewählt, weil diese Zahl der Kehrwert der Erfolgswahrscheinlichkeit q ist) Iterationen terminiert ist beträgt: $1 - (1-q)^k$, also 1 minus der Wahrscheinlichkeit, dass der Algorithmus nach k Iterationen noch nicht terminiert ist.

Wir erinnern uns: $q \geq \frac{1}{e}$

$1-(1-q)^k \geq 1-(1-\frac{1}{e})^k \geq 1-(1-\frac{1}{e})^{ce\ln n} = (1-(1-\frac{1}{e})^e)^{c\ln n} \geq 1-(\frac{1}{e})^{c\ln n} = 1-\frac{1}{(e^{\ln n})^c} = 1-\frac{1}{n^c}$

weil $\forall x \geq 1 : \left(1-\frac{1}{x}\right)^x \leq \frac{1}{e}$ (wegen $\frac{1}{e}$ wählen wir bei k den ln)

□

3.4 Allgemeines Prinzip

Theorem 3.14 (Monte Carlo → Las Vegas)**.**

- *Sei $\mathcal{A}$ ein randomisierter Algorithmus für ein Problem $\mathcal{P}$, der für jede Eingabe mit Wahrscheinlichkeit q 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 terminiert und dabei eine korrekte Lösung berechnet und dafür $\mathcal{A}$ und $\mathcal{V}$ in Erwartung $O(1/q)$ Mal aufruft.*

Definition 3.15. *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.*

Bei einem Monte Carlo Algorithmus wirkt sich die Randomisierung darauf aus, ob der Algorithmus am Ende ein korrektes Ergebnis liefert oder nicht.

Bei einem Las Vegas Algorithmus wirkt sich die Randomisierung auf die verbrauchten Ressourcen (Anzahl an benötigten Runden, Nachrichtenkomplexität, Laufzeit, Speicher) aus und ein solcher Algorithmus liefert stets ein korrektes Ergebnis.

Man kann nicht jeden Monte Carlo Algorithmus in einen Las Vegas Algorithmus überführen, da es nicht immer so leicht ist einen Verifizierer zu finden.

3.5 Fazit

Leader Election in anonymen Ringen ist mit Randomisierung möglich.

Prinzipien und Techniken verteilter Algorithmen bei Leader Election im Ring:

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

Literatur

[1] Nancy A. Lynch (1996) *Distributed Algorithms*, Kapitel 3, Morgan Kaufmann.