



Übungen zu Woche 5: Randomisierte Algorithmen I

Dienstag

👤👍 **5.1 Contention Resolution.** Führe das *contention resolution* Protokoll mit 4 Prozessen von Hand aus. Benutze hierzu zwei Münzen und werfe sie, um die zufällige Auswahl zu simulieren. Wie viele Runden hast du gebraucht, bis alle Prozesse erfolgreich auf die Datenbank zugegriffen haben? Inwiefern passt dein Ergebnis zu der theoretisch vorhergesagten Anzahl an benötigten Runden zusammen?

moodle 🧠 **5.2 Mehrheit.** Gegeben ist eine Sequenz x_1, x_2, \dots, x_n von n ganzen Zahlen. Die Sequenz hat das *Mehrheitselement* t , wenn die Zahl t öfter als $\frac{n}{2}$ -mal in der Sequenz vorkommt. Zum Beispiel hat die Sequenz 1, 2, 3, 1, 2, 2, 2 das Mehrheitselement 2, während die Sequenz 2, 2, 1, 2, 3, 3 kein Mehrheitselement hat. Im Folgenden ist der randomisierte Algorithmus FINDEMEHRHEITSELEMENT beschrieben, der das Mehrheitselement finden soll.

FINDEMEHRHEITSELEMENT(x_1, \dots, x_n):

Ziehe einen Index i uniform zufällig aus $\{1, \dots, n\}$. Prüfe dann, ob x_i öfter als $\frac{n}{2}$ -mal in der Sequenz vorkommt. Wenn ja, dann gib x_i aus. Ansonsten gib aus, dass es kein Mehrheitselement gibt.

- 👍 Was ist die Laufzeit von FINDEMEHRHEITSELEMENT?
- 👍 Kann FINDEMEHRHEITSELEMENT ein Mehrheitselement zurückgeben, wenn die Sequenz keines hat? Begründe deine Antwort.
- 👍 Kann FINDEMEHRHEITSELEMENT ausgeben, dass kein Mehrheitselement existiert, obwohl die Sequenz ein solches besitzt? Begründe deine Antwort.
- 🔑 Bestimme die Wahrscheinlichkeit, dass FINDEMEHRHEITSELEMENT eine falsche Antwort liefert.

🧠 **5.3 Wahrscheinlichkeitsrechnung.**

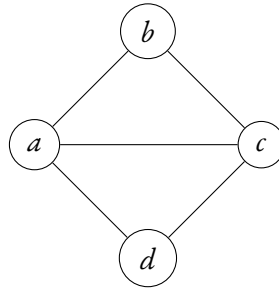
- 👍 Seien E und F zwei Ereignisse mit $\Pr(E|F) = \Pr(E)$ und $\Pr(F|E) = P(F)$. Zeige, dass dann auch $\Pr(E \cap F) = \Pr(E) \cdot \Pr(F)$ gilt. *Tipp: ~~überprüfe die Definitionen~~ beachte die Definitionen*
- 🔑 Betrachte die folgende Anwendung der *union bound* auf den Folien zu *contention resolution* mit n Prozessen:

$$\Pr\left(\bigcup_{i=1}^n F_{i,t}\right) \leq \sum_{i=1}^n \Pr(F_{i,t}).$$

Hierbei ist $F_{i,t}$ das Ereignis, dass Prozess i es in keiner der Runden $1, \dots, t$ geschafft hat, auf die Datenbank zuzugreifen. Sind die einzelnen Ereignisse $F_{i,t}$ hierbei disjunkt oder überschneiden sie sich? Warum?

Donnerstag

🧑 5.4 Minimaler Schnitt. Betrachte den Beispielgraphen zum Kontraktionsalgorithmus aus der Vorlesung:



- 👉 Zeige, dass es eine Sequenz von Kontraktionen gibt, die zu einem nicht-minimalen Schnitt führt.
- 🔑 Berechne die genaue Wahrscheinlichkeit dafür, dass der Kontraktionsalgorithmus den minimalen Schnitt ausgibt.

moodle 🧑 5.5 Schneller Kontraktionsalgorithmus. Wie kann man den Kontraktionsalgorithmus für minimale Schnitte effizient implementieren? Beschreibe deine Lösung als Pseudocode, unter Verwendung aller nötigen Datenstrukturen und Algorithmen. Analysiere die Laufzeit sowie den Platzverbrauch.

🧑 5.6 Kontraktionsalgorithmus analysieren. Wir vervollständigen jetzt die Analyse der Erfolgswahrscheinlichkeit des Kontraktionsalgorithmus. Die folgende Herleitung führt zum gewünschten Ergebnis:

$$\Pr(E_{n-2} \cap \dots \cap E_1) = \Pr(E_{n-3} \cap \dots \cap E_1) \cdot \Pr(E_{n-2} | E_{n-3} \cap \dots \cap E_1) \quad (1)$$

$$= \Pr(E_1) \cdot \Pr(E_2 | E_1) \cdot \dots \cdot \Pr(E_{j+1} | E_j \cap \dots \cap E_1) \cdot \dots \cdot \Pr(E_{n-2} | E_{n-3} \cap \dots \cap E_1) \quad (2)$$


$$\geq \left(1 - \frac{2}{n}\right) \cdot \left(1 - \frac{2}{n-1}\right) \cdot \left(1 - \frac{2}{n-2}\right) \cdot \dots \cdot \left(1 - \frac{2}{3}\right) \quad (3)$$

$$= \left(\frac{n-2}{n}\right) \cdot \left(\frac{n-3}{n-1}\right) \cdot \left(\frac{n-4}{n-2}\right) \cdot \dots \cdot \left(\frac{2}{4}\right) \cdot \left(\frac{1}{3}\right) \quad (4)$$

$$= \left(\frac{2}{n(n-1)}\right) \geq \frac{2}{n^2} \quad (5)$$

- 👉 Zeige (1) formal, indem du die Definition der bedingten Wahrscheinlichkeit benutzt. Erkläre außerdem die Intuition, indem du dir bewusst machst, was die linke und rechte Seite bedeuten.
- 🔑 Zeige (2). *Tipp: was ist die Wahrscheinlichkeit, dass ein bestimmtes Element zum ersten Mal in der Liste ist?*
- 👉 Zeige (3), indem du jeden Faktor einzeln abschätzt.
- 🔑 Zeige (4). *Tipp: was ist die Wahrscheinlichkeit, dass ein bestimmtes Element zum ersten Mal in der Liste ist?*
- 👉 Zeige (5) durch elementare Algebra.

Weitere Aufgaben und Projekte

 **5.7 Zufällige Konfliktfreiheit.** In Abschnitt 13.1 (KT) haben wir ein einfaches verteiltes Protokoll gesehen, um ein *contention resolution* Problem zu lösen. Eine zweite Möglichkeit, um *contention resolution* mittels Randomisierung zu lösen, ist eine verteilte Konstruktion eines Independent Set.



In einem System mit n Aufgaben P_1, \dots, P_n stehen einige Paare von Aufgaben miteinander im Konflikt, beispielsweise wenn beide Aufgaben Zugang zu derselben Ressource brauchen. Das Ziel ist es, in einem gegebenen Zeitintervall eine möglichst große Menge S von ausführbaren Aufgaben zu finden, sodass keine zwei Aufgaben aus S in einem Konflikt stehen. Die Menge S nennen wir *konfliktfrei*.

Diesen Vorgang kann man sich als einen Graphen $G = (V, E)$ vorstellen. Dabei repräsentiert jeder Knoten $v \in V$ eine Aufgabe und eine Kante $e = (u, v)$ stellt einen Konflikt zwischen zwei Aufgaben u und v dar. Ist eine Menge S von Aufgaben konfliktfrei, bilden sie eine *unabhängige Menge* (*independent set*) in G . Es ist im Allgemeinen schwierig, eine maximale und konfliktfreie Menge S für einen beliebigen Konfliktgraphen G zu finden.

Deshalb benutzen wir eine Heuristik mit einer einfachen dezentralisierten Methode, die eine vernünftig große und konfliktfreie Menge findet: Jede Aufgabe „kommuniziert“ nur mit den Aufgaben, mit denen sie in Konflikt steht, und entscheidet dann, ob sie in die Menge S gehört oder nicht. Wir nehmen an, dass der Konfliktgraph G d -regulär ist, das heißt, jeder Knoten hat genau d Nachbarn. Schau dir nun das folgende Verfahren an:

FINDEKONFLIKTFREIEMENGE:

Jede Aufgabe P_i wählt unabhängig einen zufälligen Wert x_i . Dabei nimmt x_i mit einer Wahrscheinlichkeit von $\frac{1}{2}$ den Wert 0 und mit einer Wahrscheinlichkeit von $\frac{1}{2}$ den Wert 1 an. Die Aufgabe kommt genau dann in die Menge S , wenn sie $x_i = 1$ gewählt hat und jede Aufgabe, mit der sie im Konflikt steht, den Wert 0 gewählt hat.

- a)  Beweise, dass die durch FINDEKONFLIKTFREIEMENGE entstehende Menge S immer konfliktfrei ist.
- b)  Gib außerdem eine Formel für $\mathbf{E}[|S|]$ an, also die erwartete Größe von S , in Abhängigkeit von der Anzahl n an Aufgaben und der Anzahl d an Konflikten pro Aufgabe. Begründe, warum deine Formel korrekt ist.