Algorithmen und Datenstrukturen 2

ALGO2 · WiSe-2021/22 · tcs.uni-frankfurt.de/algo2 · 2021-11-13 · 737950d

Übungen zu Woche 5: Randomisierte Algorithmen I

Das Übungsblatt enthält alle empfohlenen Lernaktivitäten für die aktuelle Woche.

- **Heimarbeit bis Montag 17:00.**
  - Schau die Videos an und lies die Buchkapitel.
  - Bearbeite die 🌱-Aufgabe in Moodle. (Feste Abgabefrist!)
  - Lese den Aufgabentext aller Übungsaufgaben.
- **Heimarbeit.** Bearbeite die Übungsaufgaben soweit möglich. Probier zumindest alle mal!
- **Dienstag/Donnerstag.**
  - **8:00–8:15.** Besprechung im Hörsaal.
  - **8:15–9:15.** Bearbeite jetzt die Übungen, die du noch nicht lösen konntest. Sprich mit anderen Studis! Frag das Vorlesungsteam um Hilfe!
  - **9:15–9:45.** Lösungsspaziergang zu den Aufgaben für heute.
- **Heimarbeit bis Freitag, den 19.11., 17:00.** Gib deine Lösungen zu der ⭐-Aufgabe von diesem Übungsblatt in Moodle ab. (Feste Abgabefrist!)

Dienstag

Aufgabe 5.1 (Wahrscheinlichkeitsrechnung).

a) Seien E und F zwei Ereignisse, für die $\Pr(E|F) = \Pr(E)$ und $\Pr(F|E) = P(F)$ gelten. Zeige, dass dann auch $\Pr(E \cap F) = \Pr(E) \cdot \Pr(F)$ gilt.

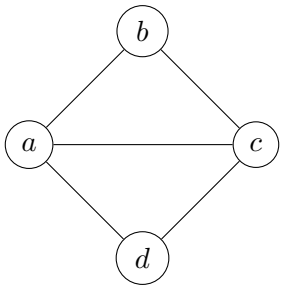
b) Betrachte *contention resolution* mit n Prozessen. Betrachte die folgende Anwendung der *union bound* aus der Vorlesung:

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

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

Aufgabe 5.2 (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?

Aufgabe 5.3 (Minimaler Schnitt). Betrachte den Beispielgraphen zum Kontraktionsalgorithmus aus der Vorlesung:



a) Zeige, dass es eine Sequenz von Kontraktionen gibt, die zu einem nicht minimalen Schnitt führt.
b) Berechne die genaue Wahrscheinlichkeit, dass der Kontraktionsalgorithmus den minimalen Schnitt ausgibt.

Donnerstag

Aufgabe 5.4 (Schneller Kontraktionsalgorithmus). Wie kann man den Kontraktionsalgorithmus für minimale Schnitte effizient implementieren? Beschreibe alle nötigen Datenstrukturen und Algorithmen und analysiere die Laufzeit sowie den Platzverbrauch.

Aufgabe 5.5 (Kontraktionsalgorithmus analysieren). Betrachte die folgende Analyse der Erfolgswahrscheinlichkeit für den Kontraktionsalgorithmus. Die folgende Herleitung führt zum gewünschten Ergebnis:

$$
\begin{aligned}
&\Pr(E_{n-2} \cap \cdots \cap E_1) \\
&= \Pr(E_{n-3} \cap \cdots \cap E_1) \cdot \Pr(E_{n-2} | E_{n-3} \cap \cdots \cap E_1) && (1)\\
&= \Pr(E_1) \cdot \Pr(E_2|E_1) \cdots \Pr(E_{j+1}|E_j \cap \cdots \cap E_1) \cdots \Pr(E_{n-2}|E_{n-3} \cap \cdots \cap E_1) && (2)\\
&\geq \left(1 - \frac{2}{n}\right) \cdot \left(1 - \frac{2}{n-1}\right) \cdot \left(1 - \frac{2}{n-2}\right) \cdots \left(1 - \frac{2}{3}\right) && (3)\\
&= \left(\frac{n-2}{n}\right) \cdot \left(\frac{n-3}{n-1}\right) \cdot \left(\frac{n-4}{n-2}\right) \cdots \left(\frac{2}{4}\right) \cdot \left(\frac{1}{3}\right) && (4)\\
&= \left(\frac{2}{n(n-1)}\right) \geq \frac{2}{n^2} && (5)
\end{aligned}
$$

a) Benutze zwei verschiedene Wege, um (1) zu zeigen. Erstens, indem du die Formel für die bedingte Wahrscheinlichkeit benutzt. Zweitens, mache dir bewusst, was die linke und rechte Seite bedeuten.
b) Zeige (2).
   Hinweis: Benutze das Ergebnis aus Teilaufgabe a)
c) Schätze jeden Faktor einzeln ab, um (3) zu zeigen.
d) Zeige (4).
e) Zeige (5).

Aufgabe 5.6 (Mehrheit). Gegeben ist eine Sequenz $x_1, x_2, \ldots, x_n$ 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 randomisierter Algorithmus FindeMehrheitsElement beschrieben, der das Mehrheitselement finden soll.

> FindeMehrheitsElement:
> Ziehe eine uniform zufällige Zahl x_i aus der Sequenz. Prüfe dann, ob x_i öfter als $\frac{n}{2}$-mal in der Sequenz vorkommt. Wenn ja, dann ist x_i das Mehrheitselement. Wenn das nicht der Fall ist, gibt der Algorithmus aus, dass es kein Mehrheitselement gibt.

a) Was ist die Laufzeit von FindeMehrheitsElement?

b) Kann FindeMehrheitsElement ein Mehrheitselement zurückgeben, wenn die Sequenz keines hat? Begründe deine Antwort.

c) Kann FindeMehrheitsElement ausgeben, dass kein Mehrheitselement existiert, obwohl die Sequenz ein solches besitzt? Begründe deine Antwort.

d) Bestimme die Wahrscheinlichkeit, dass FindeMehrheitsElement eine falsche Antwort liefert.

Sternaufgabe

Aufgabe 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 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. Da man das allgemeine Independent Set Problem auf dieses Problem reduzieren kann, ist es 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 einer kleinen Anzahl anderer Aufgaben und entscheidet dann, ob sie in die Menge S gehört oder nicht. Wir nehmen an, dass jeder Knoten genau d Nachbarn in G hat. Das heißt, jede Aufgabe hat einen Konflikt zu genau d anderen Aufgaben. Schaue 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 die erwartete Größe von S in Abhängigkeit der Anzahl n an Aufgaben und der Anzahl d an Konflikten pro Aufgabe an. Begründe, warum deine Formel korrekt ist.