4. Tutorium - Algorithmen I

Nina Zimbel

13.05.2015

- Anmerkungen zu den Übungsblättern
- Wahrscheinlichkeitstheorie
- Datenstrukturen

Zu den Übungsblättern

- Nr. 3: Auch bei "geben Sie an" ist eine Begründung gefordert.
- geschlossene Form, aber auch möglichst einfach $\rightarrow$ Summen ausrechnen
- Nr. 2: Pointersalat...

Wahscheinlichkeitstheorie

- Zufallsvariable:
- Erwartungswert einer Zufallsvariablen X, die die Werte $(x_i)_{i \in I}$ mit den jeweiligen Wahrscheinlichkeiten $(p_i)_{i \in I}$ annimmt:
- relative Häufigkeit eines Merkmals A:

Wahscheinlichkeitstheorie

- Zufallsvariable:
- Zuordnungsvorschrift, die jedem möglichen Ergebnis eines Zufallsexperiments eine Größe zuordnet. Z.B. gewürfelte Zahl eines Würfels, Anzahl der gewürfelten 6en, ...
- Erwartungswert einer Zufallsvariablen X, die die Werte $(x_i)_{i \in I}$ mit den jeweiligen Wahrscheinlichkeiten $(p_i)_{i \in I}$ annimmt:

- relative Häufigkeit eines Merkmals A:

Wahscheinlichkeitstheorie

- Zufallsvariable:
- Zuordnungsvorschrift, die jedem möglichen Ergebnis eines Zufallsexperiments eine Größe zuordnet. Z.B. gewürfelte Zahl eines Würfels, Anzahl der gewürfelten 6en, ...
- Erwartungswert einer Zufallsvariablen X, die die Werte $(x_i)_{i \in I}$ mit den jeweiligen Wahrscheinlichkeiten $(p_i)_{i \in I}$ annimmt:
- $\mathrm{E}(X) = \sum_{i \in I} x_i p_i = \sum_{i \in I} x_i P(X = x_i)$
- relative Häufigkeit eines Merkmals A:

Wahscheinlichkeitstheorie

- Zufallsvariable:
- Zuordnungsvorschrift, die jedem möglichen Ergebnis eines Zufallsexperiments eine Größe zuordnet. Z.B. gewürfelte Zahl eines Würfels, Anzahl der gewürfelten 6en, ...
- Erwartungswert einer Zufallsvariablen X, die die Werte $(x_i)_{i \in I}$ mit den jeweiligen Wahrscheinlichkeiten $(p_i)_{i \in I}$ annimmt:
- $\mathrm{E}(X) = \sum_{i \in I} x_i p_i = \sum_{i \in I} x_i P(X = x_i)$
- relative Häufigkeit eines Merkmals A:
- $h_n(A) = \frac{H_n(A)}{n}$ wobei: $H_n(A)$ die absolute Häufigkeit des Merkmals A ist

Beispiel: Variante des Geburtstagsparadoxon

Wieviele Gäste muss eine Geburtstagsparty “im Mittel” haben, damit mindestens zwei Gäste den gleichen Geburtstag haben?

Gäste (Keys) $1..n$.

Elementarereignisse: $h \in \Omega = \{0..364\}^{\{1..n\}}$.

Definiere Indikator-ZV $I_{ij} = 1$ gdw $h(i) = h(j)$.

Anzahl Paare mit gleichem Geburtstag: $X = \sum_{i=1}^{n} \sum_{j=i+1}^{n} I_{ij}$.

$$\begin{aligned} \mathrm{E}[X] &= \mathrm{E}[\sum_{i=1}^{n} \sum_{j=i+1}^{n} I_{ij}] = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \mathrm{E}[I_{ij}] \\ &= \sum_{i=1}^{n} \sum_{j=i+1}^{n} \mathbb{P}\left[I_{ij} = 1\right] = \frac{n(n-1)}{2} \cdot \frac{1}{365} \\ &\stackrel{!}{=} 1 \Leftrightarrow n = -\frac{1}{2} + \sqrt{\frac{1}{2^2} + 730} \approx 26.52 \end{aligned}$$

Aufgabe

Es werden n identische Bälle auf b verschiedene Urnen geworfen. Die Würfe sind unabhängig, und bei jedem Wurf landet der Ball mit gleicher Wahrscheinlichkeit in einer der Urnen.

- Wieviele Bälle fallen in eine gegebene Urne?
- Wie viele Bälle müssen im Mittel geworfen werden, bis eine gegebene Urne einen Ball enthält?
- Wie viele Bälle müssen geworfen werden, bis jede der Urnen wenigstens einen Ball enthält?

Aufgabe - Sparse Array

Entwerfen Sie ein *SparseArray* (soviel wie „spärlich besetztes Array“).

Eigenschaften: wie beschränktes Array, zusätzlich schnelle Erzeugung und schneller Reset.

Nehmen Sie dabei an, dass **allocate** beliebig viel *uninitialisierten* Speicher in konstanter Zeit liefert.

Im Detail habe das *SparseArray* folgende Eigenschaften:

- Ein *SparseArray* mit n Slots braucht O(n) Speicher.
- Erzeugen eines leeren *SparseArray* mit n Slots: O(1)
- *reset* (in leeren Zustand versetzen): O(1)
- $get(i)$ liefert den Wert, der sich im i-ten Slot befindet wenn dort ein bestimmter Wert gesetzt wurde, sonst $\perp$.
- $set(i, x)$ setzt das Element im i-ten Slot auf den Wert x.
- $get(i)$ und $set(i, x)$: O(1) (*wahlfreien Zugriff*, engl. *random access*)

b) Nehmen Sie an, dass die Datenelemente, die Sie im *SparseArray* ablegen wollen, recht groß sind (also z.B. nicht nur einzelne Zahlen, sondern Records mit 10, 20 oder mehr Einträgen).

Geht Ihre Realisierung unter dieser Annahme sparsam oder verschwenderisch mit dem Speicherplatz um?
Wie geht es besser?

c) Vergleichen Sie Ihr *SparseArray* mit Bounded-Arrays.
Welche Vorteile und Nachteile sehen Sie im Hinblick auf den Speicherverbrauch und auf das Iterieren über alle Elemente mit *set* eingefügten Elemente?