Karlsruher Institut für Technologie

Stefan Walzer
ITI Prof. Sanders
Wintersemester 2023/2024

Übungsblatt 8 – Bounded Differences and Bloom Filters

Randomisierte Algorithmik

Aufgabe 1 – Kollisionszahl

Eine Schlüsselmenge $S = \{x_1, \dots, x_n\} \subseteq D$ werde rein zufällig auf Hashwerte $h(x_1), \dots, h(x_n) \sim \mathcal{U}([m])$ abgebildet wobei $\alpha := \frac{n}{m} = O(1)$ gelte. Die Kollisionszahl ist definiert als $C = |\{(i,j) \mid 1 \leq i < j \leq n \mid h(x_i) = h(x_j)\}|$. Ziel dieser Aufgabe ist es, eine Konzentrationsschranke an C zu bestimmen.

(i) Zeige $\mathbb{E}[C] = \Theta(n)$.

(ii) Sei $j \in [m]$ und $f_j = |\{x \in S \mid h(x) = j\}|$. Zeige $\Pr[f_j \geq \log(n)] = O(1/n^2)$.
Hinweis: Union Bound. Eventuell nützlich ist, dass $\frac{\alpha^{\log n}}{(\log n)!}$ asymptotisch kleiner als jedes Polynom ist (z.B. $\frac{\alpha^{\log n}}{(\log n)!} = O(n^{-42})$).

(iii) Definiere $C_{\text{cap}} := \sum_{j \in [m]} \binom{\min(\log n, f_j)}{2}$. Zeige $\Pr[C_{\text{cap}} = C] = 1 - O(1/n)$.

(iv) Zeige $\Pr[C_{\text{cap}} - \mathbb{E}[C_{\text{cap}}] \geq t] \leq \exp(-2t^2/(n \cdot \log^2 n))$.

(v) Zeige $\Pr[C - \mathbb{E}[C] \geq n^{2/3}] = O(1/n)$.

Folgende Aufgabe hat nicht wirklich mit randomisierten Algorithmen zu tun, außer dass wir die Abschätzung in der Vorlesung gebraucht haben. Daher "Bonus".

Aufgabe 2 – Bonus: Approximationen von e

Du weißt sicher, dass $e = \lim_{n \to \infty} (1 + \frac{1}{n})^n$ gilt (das ist sogar eine gängige *Definition* von e). Leite daraus ab, dass die folgenden beiden Gleichungen für alle $n \in \mathbb{N}$ gelten:

$$(1 + \tfrac{1}{n})^n \leq e \leq (1 + \tfrac{1}{n})^{n+1}$$
$$(1 - \tfrac{1}{n})^n \leq e^{-1} \leq (1 - \tfrac{1}{n})^{n-1}.$$

Folgende Schritte bieten sich an.

(i) Zeige die linken beiden Ungleichungen.
Hinweis: Benutze mal wieder $1 + x \leq e^x$.

(ii) Zeige, dass die rechten Seiten monoton in n fallen.

(iii) Folgere aus (ii) und einer Grenzwertbetrachtung die rechten beiden Ungleichungen.

Aufgabe 3 – Counting Bloomfilter (a.k.a.: Count-Min-Sketch)

Ein Counting Bloomfilter ist zunächst wie ein Bloomfilter: Er verwaltet n Schlüssel in einem Array der Größe m, der *load factor* ist $\alpha := \frac{n}{m}$ und es gibt k Hashfunktionen. Die Parameter seien wie in der Vorlesung gewählt, sodass ein (gewöhnlicher) Bloomfilter eine Falsch-Positiv-Wahrscheinlichkeit von ε hätte. Das Array enthält nun keine Bits mehr sondern natürliche Zahlen. Neben insert und query gibt es nun als weitere Operation delete.

Algorithm insert(x):
  for $i \in [k]$ **do**
    $A[h_i(x)]++$

Algorithm delete(x):
  for $i \in [k]$ **do**
    $A[h_i(x)]--$

Algorithm query(x):
  for $i \in [k]$ **do**
    if $A[h_i(x)] = 0$ **then**
      return false
  return true

Wir lassen zu, dass ein Schlüssel *mehrfach* in den Counting Bloomfilter eingefügt wird. Insofern ist das verwaltete Objekt S nun eine *Multimenge*, deren n Elemente $\{x_1, \ldots, x_n\}$ jeweils mit einer Vielfachheit $a_1, \ldots, a_n$ vorliegen. Die Operation insert(x) soll der Multimenge S eine Kopie von x hinzufügen, das heißt die Vielfachheit von x erhöhen, falls x bereits in S enthalten war oder ein weiteres Element mit Vielfachheit 1 in S aufnehmen, falls x noch nicht in S enthalten war. Die Bedeutung von delete ist analog.

Wie zuvor auch darf query falsch-positive, aber keine falsch-negativen Antworten liefern.

(a) Wir fordern, dass delete nur für solche Elemente aufgerufen wird, die auch wirklich in S enthalten sind (mit Vielfachheit mindestens 1). Was geht kaputt wenn wir das nicht fordern?

Mit Counting Bloomfiltern lassen sich noch zusätzliche nützliche Operationen implementieren.

(b) Implementiere eine Operation count, die für $x \in D$ einen Schätzwert count(x) für die Vielfachheit a von x in S angibt. Zeige, dass $\Pr[a \neq \text{count}(x)] \leq \varepsilon$ gilt.

(c) Das wichtigste Argument dafür (Counting-) Bloomfilter zu verwenden, ist der geringe Speicherplatz im Vergleich zu einer exakten Datenstruktur. Wir sollten daher besser nicht große Integer Datentypen (etwa 64 Bit) für die Zähler verwenden. Stellen wir uns einen Anwendungsfall vor, in dem die „meisten“ Zähler niemals 8 Bit überschreiten. Wir nutzen daher ein Array A von 8-Bit Zählern. Diskutiere (kurz) die folgenden Vorschläge zum Umgang mit Zählerüberläufen. Was sind die Vorteile und welche Kompromisse werden eingegangen?

- Alice verhindert lediglich Zählerüberläufe, passt also die $A[h_i(x)]++$ Operation in insert und die $A[h_i(x)]--$ Operation in delete so an, dass der Zähler nur inkrementiert bzw. dekrementiert wird, wenn der maximale darstellbare Wert bzw. der minimale darstellbare Wert noch nicht erreicht ist.
- Bob schlägt vor, einen Zähler, der den Wert $(11111111)_2 = 255$ erreicht hat „einzufrieren“, das heißt zukünftige insert oder delete Operationen werden den Zähler nie mehr anpassen.

- Carol schlägt vor, die Bitfolge $(11111111)_2 = 255$ als Markierung dafür zu verwenden, dass der wahre Zählerwert Größer als 254 ist. Der wahre Zählerwert wird in diesem Sonderfall in einer Hashtabelle gespeichert.

Aufgabe 4 – Schnitte Schätzen mit Bloomfiltern

Seien $n, m, k \in \mathbb{N}$. Alice und Bob wollen abschätzen, wie ähnlich ihr Musikgeschmack ist. Seien X die n Lieblingslieder von Alice und Y die n Lieblingslieder von Bob. Zu schätzen ist $\gamma := \frac{|X \operatorname{cap} Y|}{n} \in [0, 1]$. Beide gehen folgendermaßen vor.

- Alice konstruiert einen Bloomfilter $A[1..m] \in \{0, 1\}^m$ für X unter Verwendung von k Hashfunktionen $h_1, \ldots, h_k$.
- Bob konstruiert einen Bloomfilter $B[1..m] \in \{0, 1\}^m$ für Y unter Verwendung *derselben* k Hashfunktionen.
- Alice und Bob tauschen ihre Filter aus und berechnen $\delta := \frac{|\{i \in [m] | A[i] \neq B[i]\}|}{m}$.
- Alice und Bob berechnen basierend auf δ eine Schätzung $\bar{\gamma}$ für γ.

Löse folgende Teilaufgaben:

(a) Diskutiere: Welche Vor- und Nachteile könnte das Verfahren im Vergleich zum direkten Austausch von X und Y haben?

(b) Gewinne Intuition: Welche Werte von δ erwartest du (in etwa) für die Extremfälle, in denen $\gamma = 1$ bzw. $\gamma = 0$ gilt?
Hinweis: Du darfst hier und im Folgenden davon ausgehen, dass den Bloomfiltern eine „optimale“ Konfiguration mit $\alpha k = \ln(2)$ zugrundegelegt wurde.

(c) Berechne $\mathbb{E}[\delta]$ als Funktion von γ. Du darfst hierbei Terme niederer Ordnung unter den Tisch fallen lassen, also z.B. $(1 - \frac{1}{m})^m \approx e^{-1}$ schreiben, ohne ein $o(1)$ mitzuführen.
Hinweis: Zunächst scheint es, als könnten andere Parameter (z.B. $n, m, k, \alpha, \varepsilon$) auch eine Rolle spielen. Deren Einfluss verschwindet aber in Termen niederer Ordnung.

(d) Diskutiere: Welche Konzentrationsschranke eignet sich, um zu beweisen, dass δ mit hoher Wahrscheinlichkeit nahe an $\mathbb{E}[\delta]$ liegt?

(e) Stelle die Gleichung aus (c) um, sodass ersichtlich wird, wie eine Schätzung $\bar{\gamma}$ für γ aus δ berechnet werden kann.

(f) Spekuliere: Welche Rolle spielt die Wahl von k (bzw. von ε) im vorliegenden Kontext?