Streaming Data: Das Modell

Berechnungen, bei fortlaufend einströmenden Daten

$$(\mathbf{x_t} \mid \mathbf{t} \geqslant \mathbf{0}),$$

sind in Echtzeit zu erbringen. Beispiele sind:

- Verkehrsmessungen im Internet,
- Datenanalyse in der Abwehr einer Denial-of-Service Attacke,
- Verarbeitung von durch Satelliten erfassten Daten, oder
- fortlaufende Protokollierung von Telefonverbindungen durch weltweit agierende Telefonunternehmen und die damit verbundenen Reaktionen auf überlastete Leitungen.

Algorithmische Fragestellungen (1/2)

- **Erstellung von Stichproben:**
  - Stichproben reduzieren die Größe der Datenmenge und sind deshalb ein wichtiges Hilfsmittel für viele einfache Probleme.
  - Beachte, dass sich Stichproben dynamisch mit dem Datenstrom ändern müssen.
- **Häufigkeitsanalyse für Datenströme:**
  - Wenn $\mathbf{a_u} = |\{\mathbf{i} \mid \mathbf{x_i} = \mathbf{u}\}|$ die Häufigkeit des Schlüssels u ist, dann berechne das $k$te Häufigkeitsmoment
$$\mathbf{H_k} = \sum_{\mathbf{u} \in \mathbf{U}} \mathbf{a_u^k}.$$
  - H_0 ist die Anzahl verschiedener Schlüssel, H_1 die Anzahl der bisher gesehenen Schlüssel.
  - H_2 misst wie gleichmäßig n Daten (auf m verschiedene Schlüssel) verteilt sind:
    - ★ Es ist $m \cdot \left(\frac{n}{m}\right)^2 = \frac{n^2}{m} \leqslant H_2 \leqslant n^2$.
    - ★ Kleine Werte von H_2 implizieren eine gleichmäßige Verteilung.

Algorithmische Fragestellungen (2/2)

Zeitfenster:

- Um das Verhalten eines Datenstroms in der jüngsten Vergangenheit bestimmen zu können, wertet man Datenströme in Zeitfenstern aus.
- Entwickle Methoden, um die quantitative Analyse nach dem Verschwinden veralteter und dem Erscheinen junger Schlüssel zu aktualisieren.

Reservoir Sampling: Stichproben für Datenströme

Reservoir Sampling

Berechne eine ohne Ersetzung gleichverteilt gezogenen Stichprobe $S \subseteq \{ (i, x_i) \mid 1 \leqslant i \leqslant n \}$ des Datenstroms $(x_i \mid i)$.

(1) T sei eine obere Schranke für die Größe der Stichprobe. Der Parameter t zählt die bisher gesehenen Schlüssel.
Setze $t = 0$ und STICHPROBE$= \emptyset$.

(2) Durchlaufe die Schlüssel nacheinander:

(2a) Setze $t = t + 1$.

(2b) $t \leqslant T$: Füge den Schlüssel in STICHPROBE ein.

(2c) $t > T$: Werfe eine Münze mit Erfolgswahrscheinlichkeit $\frac{T}{t}$.

- ★ Bei einem Erfolg entferne einen zufällig aus STICHPROBE gewählten Schlüssel; der aktuelle Schlüssel wird eingefügt.
- ★ Bei einem Misserfolg wird nichts unternommen.

Kommentar: Die Stichprobengröße T bleibt unverändert.

Analyse

Für alle $t \geqslant T$: Jede T-elementige Teilmenge $X \subseteq \{(i, x_i) \mid 1 \leqslant i \leqslant t\}$ tritt mit Wahrscheinlichkeit $1/\binom{t}{T}$ als Stichprobe auf.

- Beweis durch Induktion nach t: Die Aussage ist für $t = T$ richtig.
- Wir nehmen an, dass die Aussage für $t - 1$ richtig ist.
  - (t, x_t) wird mit Wahrscheinlichkeit $p = \frac{T}{t}$ aufgenommen.
  - p ist die Wahrscheinlichkeit, dass (t, x_t) in einer zufälligen T-elementigen Teilmenge $X \subseteq \{(i, x_i) \mid 1 \leqslant i \leqslant t\}$ enthalten ist, denn
$$\binom{t-1}{T-1} / \binom{t}{T} = \frac{T}{t}.$$
  - Wir erhalten eine T-elementige Stichprobe, wenn wir
    - mit Erfolgswahrscheinlichkeit $\frac{T}{t}$ entscheiden, ob (t, x_t) gewählt wird.
    - Wenn (t, x_t) nicht gewählt wird, dann wähle eine zufällige, T-elementige Stichprobe aus der Menge $\{(i, x_i) \mid 1 \leqslant i \leqslant t - 1\}$: Wende die Induktionshypothese an.

Und wenn (t, x_t) gewählt wird?

Wenn (t, x_t) gewählt wird:

- Dann ist eine zufällige Stichprobe mit $T-1$ Elementen aus der Menge $\{(i, x_i) \mid 1 \leqslant i \leqslant t-1\}$ zu wählen.
- Was macht Reservoir Sampling?
  - Die alte Stichprobe X ist eine zufällige T-elementige Teilmenge von $\{(i, x_i) \mid 1 \leqslant i \leqslant t-1\}$.
  - Reservoir Sampling entfernt ein zufälliges Element aus X und erhält damit eine zufällige Stichprobe aus $\{(i, x_i) \mid 1 \leqslant i \leqslant t-1\}$ mit $T-1$ Elementen.
  - Diese modifizierte Stichprobe wird um (t, x_t) vergrößert und wir erhalten eine zufällige T-elementige Stichprobe X.

Reservoir Sampling funktioniert.

Approximative Berechnung des Medians

Berechnung des approximativen Medians

1. Benutze Reservoir Sampling, um eine Stichprobe S der Größe s zu ziehen.
2. Bestimme den Median M von S und gib M als Approximation des tatsächlichen Medians aus.

- $\delta, \varepsilon \in [0, 1]$ seien vorgegeben.
- Arbeite mit einer Stichprobe der Größe

$$s = c \cdot \frac{1}{\varepsilon^2} \cdot \ln \frac{1}{\delta}$$

für ein hinreichend großes c.

Dann liegt der Rang des ausgegebenen Schlüssels mit Wahrscheinlichkeit $\geqslant 1 - \delta$ in dem Intervall $[\frac{n}{2} - \varepsilon \cdot n, \frac{n}{2} + \varepsilon \cdot n]$.

Analyse

- x_{unten} (bzw. x_{oben}) sei der Schlüssel vom Rang $(\frac{1}{2} - \varepsilon) \cdot n$ (bzw. Rang $(\frac{1}{2} + \varepsilon) \cdot n$).
- Wir haben nur Pech, wenn 50% aller Schlüssel der Stichprobe unterhalb von x_{unten} (bzw. oberhalb von x_{oben}) liegen.
  - Ein Schlüssel kleiner als x_{unten} wird mit Wahrscheinlichkeit $(\frac{1}{2} - \varepsilon)$ gezogen.
  - Die erwartete Anzahl dieser „kleinen" Schlüssel ist deshalb höchstens $(\frac{1}{2} - \varepsilon) \cdot s$.
  - Wir haben also nur Pech, wenn sogar $\frac{1}{2} \cdot s = (\frac{1}{2} - \varepsilon) \cdot s \cdot (1 + \frac{\varepsilon}{1/2 - \varepsilon})$ kleine Schlüssel gezogen werden.
  - Mit der **Chernoff-Schranke** passiert dies mit Wahrscheinlichkeit höchstens $e^{-\Omega(\varepsilon^2 \cdot s)}$.

    $e^{-\Omega(\varepsilon^2 \cdot s)} \leqslant \delta$ gilt, falls $\varepsilon^2 \cdot s = \Omega(\ln \frac{1}{\delta})$.

Die Behauptung folgt, da die Anzahl der „großen" Schlüssel ein analoges Verhalten zeigt.

Clustering

Clustering: Das k-Zentren Problem

- Ein vollständiger ungerichteter Graph $G = (V, E)$ und eine Metrik $d : V^2 \to \mathbb{R}_{\geqslant 0}$ ist gegeben.
- Für ein fixiertes k bestimme eine Menge $Z \subseteq V$ von k Knoten, so dass der maximale Abstand zu einem Zentrum, also

$$\max_{v \in V} \min_{w \in Z} \mathbf{d}(\mathbf{v}, \mathbf{w}),$$

kleinstmöglich ist.

Minimiere den **Radius**, also den größten Abstand eines Punktes vom nächstliegenden Cluster-Zentrum.

- Die Sprachenversion des *k-Zentren* Problems ist *NP*-vollständig.
- Bestimme eine approximative Lösung!

Clustering für Datenströme

1. Die Zahl k der erlaubten Cluster-Zentren ist gegeben ebenso wie die Metrik d.

   Die Folge $(x_j \mid j)$ bezeichne den Datenstrom.

2. Benutze Reservoir Sampling, um eine Stichprobe S der Größe s zu ziehen.

3. Setze $Z = \{x_j\}$ für einen beliebigen Schlüssel $x_j \in S$.
   Wiederhole $k - 1$ mal:
   - Bestimme einen Schlüssel $x_i \in S$, dessen minimaler Abstand zu einem Schlüssel in Z größtmöglich ist.
   - Füge x_i in die Menge Z ein.

4. Z wird als Menge der Cluster-Zentren ausgegeben.

Analyse

Das Clustering ist 2-approximativ auf der Stichprobe.

- Der optimale Radius sei **opt**.
- Angenommen, es gibt einen Punkt $p \in S$ mit einem Abstand von größer als **2**$\cdot$ **opt** zu allen Punkten in Z.
  - Nach Konstruktion von Z haben je zwei Punkte in $Z \cup \{p\}$ einen Abstand von größer als $2 \cdot \mathrm{opt}$.
  - Die Punkte aus $Z \cup \{p\}$ gehören zu verschiedenen Clustern der optimalen Lösung:
    - ⋆ Wenn ein Clusterpunkt y der optimalen Lösung nächstliegender Punkt für zwei Elemente $u, v \in Z \cup \{p\}$ ist, dann ist $d(u, v) \leqslant d(u, y) + d(y, v) \leqslant 2 \cdot \mathrm{opt}$.
  - Die optimale Lösung hat nur **k** Clusterpunkte: Widerspruch.

Der Algorithmus funktioniert für die Stichprobe,
aber wie gut ist die Stichprobe?

Das Resultat

Ein Datenstrom der Länge n sei gegeben.

- Wähle eine Stichprobe der Größe $s = \frac{k \cdot \ln n + \ln(\frac{1}{\delta})}{\varepsilon}$.
- Dann ist unser Clustering mit Wahrscheinlichkeit mindestens $1 - \delta$ auf einer Teilmenge der Größe $(1 - \varepsilon) \cdot n$ 2-approximativ.

Warum müssen wir eine kleine Menge von Punkten ausschließen?

- Einige wenige Ausreißer gehören hochwahrscheinlich nicht zur Stichprobe.

Die Wahrscheinlichkeit schlechter Zentren

- Sei **opt** der optimale Radius.
- Sei Z eine „**schlechte**“ Zentrenmenge:

  Die Menge $\text{Weitweg}(Z) = \{x_i \mid d(x_i, Z) > 2 \cdot \text{opt}\}$, also die Menge aller Punkte mit einem Abstand von mehr als **2**· **opt** von ihrem nächstliegenden Zentrum in Z, habe mehr als $\varepsilon \cdot n$ Elemente.

- Mit welcher Wahrscheinlichkeit verfehlt die Stichprobe S die Menge $\text{Weitweg}(Z)$?

$$\begin{aligned} \text{prob}[\, S \cap \text{Weitweg}(Z) = \emptyset \,] &\leqslant \left(\frac{(1-\varepsilon)n}{n}\right)^s = (1-\varepsilon)^s \\ &\leqslant e^{-\varepsilon \cdot s} = e^{-(k \cdot \ln n + \ln(\frac{1}{\delta}))} = \delta \cdot n^{-k} \\ &\leqslant \delta / \binom{n}{k}. \end{aligned}$$

- Und wenn $\text{Weitweg}(Z)$ getroffen wird?

Wenn die Stichprobe S die Menge Weitweg(Z) trifft:

- Dann gibt es $s \in S$ mit $d(s, z) > 2\mathrm{opt}$ für alle $z \in Z$.
- Aber wir berechnen ein Clustering, das auf Stichprobe S 2-approximativ oder besser ist: Widerspruch zur Definition von opt.

- Die Wahrscheinlichkeit irgendeine schlechte Zentrenmenge zu wählen, ist somit höchstens

$$\binom{n}{k} \cdot \delta / \binom{n}{k} = \delta.$$

- Unser Clustering wird mit Wahrscheinlichkeit mindestens $1 - \delta$ eine Zentrenmenge Z mit $|\text{Weitweg}(Z)| \leqslant \varepsilon \cdot n$ bestimmen und das war zu zeigen.

Was können Stichproben nicht?

Die Grenzen der Stichproben-Methode

Bestimme die Anzahl verschiedener Schlüssel approximativ.

- Betrachte alle (deterministischen oder probabilistischen) Algorithmen zur Stichproben-Erstellung, die nur $r << n$ Schlüssel inspizieren.

  Reservoir Sampling wird erfasst.
- Können diese Algorithmen die beiden folgenden Szenarien voneinander unterscheiden?
  - **Szenario 1** besteht aus der nur mit Einsen besetzten Folge.
  - **Szenario 2** besteht aus allen Folgen, für die jedes $i \in \{2, \ldots, k\}$ genau einmal auftritt. Alle restlichen Folgenelemente haben den Wert 1.

Für eine approximative Bestimmung der Anzahl verschiedener Schlüssel muss ein Algorithmus beide Szenarien voneinander unterscheiden.

Unterscheidung der beiden Szenarien

Sei A ein Algorithmus und sei X_i die Zufallsvariable, die den $i$ten von A ausgewählten Schlüssel als Wert besitzt. Dann gilt

$$\text{prob}[\, X_i = 1 \mid X_1 = X_2 = \cdots = X_{i-1} = 1 \,] = \frac{n - i - k + 1}{n - i + 1},$$

wenn nur Eingaben des Szenarios 2 auftreten.

- Wir betrachten Szenario 2.
- Angenommen, die ersten $i - 1$ inspizierten Schlüssel besitzen sämtlich den Wert 1:
  - Von den $n - k$ Schlüsseln mit Wert 1 verbleiben somit $n - k - (i - 1)$ noch nicht inspizierte Schlüssel mit Wert 1.
  - Aber insgesamt $n - (i - 1)$ Schlüssel wurden noch nicht inspiziert und die Behauptung folgt.

Wie wahrscheinlich sind Stichproben nur mit Einsen?

Sei I das Ereignis, dass alle r inspizierten Schlüssel den Wert 1 besitzen.

$$\begin{aligned}
\text{prob}[\, I \,] &= \Pi_{i=1}^{r} \text{prob}[\, X_i = 1 \mid X_1 = X_2 = \cdots = X_{i-1} = 1 \,] \\
&= \Pi_{i=1}^{r} \frac{n-i-k+1}{n-i+1} \\
&\geqslant \left(\frac{n-r-k+1}{n-r+1} \right)^r \geqslant \left(\frac{n-r-k}{n-r} \right)^r \\
&= \left(1 - \frac{k}{n-r} \right)^r \geqslant e^{-\frac{2 \cdot k \cdot r}{n-r}}, \qquad \text{falls } \frac{k}{n-r} \leqslant \frac{1}{2},
\end{aligned}$$

denn $1 - z \geqslant e^{-2 \cdot z}$ für $0 \leqslant z \leqslant \frac{1}{2}$.

Wie wahrscheinlich sind Stichproben nur aus Einsen?

Es ist prob[I] $\geqslant e^{-\frac{2 \cdot k \cdot r}{n-r}}$, falls $\frac{k}{n-r} \leqslant \frac{1}{2}$.

- Wir setzen $k = \frac{n-r}{2 \cdot r} \cdot \ln(2)$ und prob[I] $\geqslant \frac{1}{2}$ folgt.
- Es ist $\frac{k}{n-r} \leqslant \frac{1}{2}$ für $r \geqslant 2$.

- Es gelte $k = \frac{n-r}{2 \cdot r} \cdot \ln(2)$ sowie $r \geqslant 2$.
- Wenn Algorithmus A nur r Schlüssel inspiziert, dann bestimmt A mit Wahrscheinlichkeit $\geqslant 1/2$ eine Stichprobe nur aus Einsen, obwohl der Datenstrom k verschiedene Schlüssel besitzt.

Kommunikation

Das 2-Parteien Kommunikationsmodell:

- Zwei Parteien, Alice und Bob, besitzen Eingaben x bzw. y, wobei weder Alice noch Bob die Eingabe des Partners kennt.
- Alice schickt eine Nachricht **message(x)** an Bob.

  Bob muss das Ergebnis nur in Abhängigkeit von seiner Eingabe y und der von Alice geschickten Nachricht berechnen.
- Alice und Bob besitzen, im Rahmen der ihnen zur Verfügung stehenden Informationen, eine **unbeschränkte Rechenkraft**.

- Deterministische oder probabilistische Protokolle bestimmen die Nachricht von Alice und die von Bob berechnete Antwort.
- Das Ziel:
  - Berechne einen Funktionswert **f(x, y)** zumindest approximativ.
  - Minimiere die Länge der längsten von Alice geschickten Nachricht.

Kommunikation: Ein Beispiel

- Alice erhält $x \in \{0,1\}^n$, Bob erhält $y \in \{0,1\}^n$.
- Entscheide, ob $x = y$ gilt.

- **Deterministische Kommunikation**:
  - Angenommen, Alice schickt für die beiden Zeichenketten $x_1, x_2 \in \{0,1\}^n$ dieselbe Nachricht m.
  - Bob weiss nicht, ob Alice die Eingabe x_1 oder x_2 besitzt und kann nicht fehlerfrei arbeiten.
  - Alice wird Nachrichten mit mindestens n Bits verschicken müssen.
- **Probabilistische Kommunikation**:
  - Alice und Bob interpretieren ihre Eingaben als Zahlen $0 \leqslant x, y \leqslant 2^n - 1$.
  - Alice würfelt eine Primzahl $p \leqslant n^2$ aus und verschickt das Paar $(x \mod p, p)$ mit $O(\log_2 n)$ Bits.
    - ★ Bob entscheidet auf $x \neq y$, wenn $x \mod p \neq y \mod p$: Kein Fehler.
    - ★ Bob entscheidet auf $x = y$, wenn $x \mod p = y \mod p$: Kleiner Fehler.

Kommunikation und Streaming Data

- Warum interessiert uns das Kommunikationsmodell?
- Sei A ein Streaming Data Algorithmus, der auf Eingaben $x_1 x_2$ mit $x_1, x_2 \in \{0,1\}^{n/2}$ höchstens Speicherplatz $s(n)$ benutzt.
  - Simuliere die Berechnung von A durch ein Kommunikationsprotokoll: Alice erhält x_1, Bob erhält x_2.
  - Wenn A die Eingabe x_1 abgearbeitet hat, sei $w \in \{0,1\}^{s(n)}$ der Inhalt des Speichers.
    - ★ Alice kennt w, verschickt w und den gegenwärtigen Zustand an Bob.
    - ★ Bob kann die Berechnung von A erfolgreich zu Ende führen.
    - ★ Nachrichten mit $s(n) + O(1)$ Bits reichen aus.

Kommunikation: Ein erstes Fazit

Die Funktion $f_n : \{0,1\}^n \to X$ sei zu berechnen.

- Jeder Algorithmus, der f

  deterministisch oder probabilistisch

  mit Speichergröße $\mathbf{s(n)}$ berechnet, kann durch ein

  deterministisches oder probabilistisches

  Kommunikationsprotokoll simuliert werden, das Nachrichten mit höchstens $\mathbf{s(n) + O(1)}$ Bits verschickt.
- Wenn Kommunikationsprotokolle mindestens $\omega(s)$ Bits benötigen, dann kann es keine Streaming Data Algorithmen mit Speichergröße $O(s)$ geben!

Probabilistische Kommunikation kann sehr viel effizienter als deterministische Kommunikation sein.

ε-approximative Berechnungen

Für Mengen A und B ist $f : A \times B \to \mathbb{R}$ zu berechnen.

(a) Ein deterministisches Protokoll heißt genau dann **ε-approximativ**, wenn Bob für jedes Eingabepaar (x, y) ein Ergebnis $a(x, y)$ mit $(1 - \varepsilon) \cdot f(x, y) \leqslant a(x, y) \leqslant (1 + \varepsilon) \cdot f(x, y)$ berechnet.

(b) $C^{\varepsilon}(f)$ ist die Länge der längsten Nachricht eines besten ε-approximativen deterministischen Protokolls für f.

(c) Ein probabilistisches Protokoll ist **ε-approximativ mit Fehler δ**, wenn Bob für alle Eingaben (x, y) mit Wahrscheinlichkeit mindestens $1 - \delta$ ein Ergebnis $a(x, y)$ mit $(1 - \varepsilon) \cdot f(x, y) \leqslant a(x, y) \leqslant (1 + \varepsilon) \cdot f(x, y)$ berechnet.

(d) $C_{\delta}^{\varepsilon}(f)$ ist die Länge der längsten Nachricht eines besten ε-approximativen probabilistischen Protokolls, das f mit Fehler δ berechnet.

Das Disjunktheitsproblem

- Im Disjunktheitsproblem der Größe n erhalten Alice und Bob Inzidenzvektoren der Teilmengen $x, y \subseteq \{1, \ldots, n\}$.
- Es ist zu entscheiden, ob $x \cap y \neq \emptyset$ gilt.

- Definiere die Funktion D_n durch

$$D_n(x, y) = \begin{cases} 1 & x \cap y = \emptyset \\ 0 & \text{sonst.} \end{cases}$$

- Man kann zeigen: $C^{\varepsilon}_{\delta}(D_n) = \Omega(n)$ für alle $\varepsilon, \delta < \frac{1}{2}$:

  Eine randomisierte Lösung des Disjunktheitsproblems ist selbst dann schwierig, wenn die Fehlerwahrscheinlichkeit δ eine beliebige Konstante kleiner als $\frac{1}{2}$ ist.

Hat diese negative Aussage zum Beispiel Konsequenzen für die Berechnung der größten Häufigkeit?

Bestimmung der größten Häufigkeit

- $\varepsilon, \delta < \frac{1}{2}$ seinen beliebig. A sei ein probabilistischer Algorithmus, der die größte Häufigkeit ε-**approximativ** mit **Fehlerwahrscheinlichkeit höchstens** δ im Streaming-Data Modell berechnet.
- Dann benötigt A die Speichergröße $\Omega(m)$, wenn m die Anzahl verschiedener Schlüssel ist.

- Der probabilistische Algorithmus A berechne die Häufigkeit des häufigsten Schlüssels approximativ.
- Wir lösen das Disjunkheitsproblem mit Hilfe von A:
  - Alice und Bob erhalten die Teilmengen $x, y \subseteq \{1, \ldots, m\}$.
  - Für die Funktion D_m des Disjunkheitsproblems gilt:

$$D_m(x, y) = \begin{cases} 1 & 1 = \text{ größte Häufigkeit für } (x, y), \\ 0 & 2 \leqslant \text{ größte Häufigkeit für } (x, y) \end{cases}$$

  - Aber es ist $C_\delta^\varepsilon(D_m) = \Omega(m)$ und die Behauptung folgt.

Häufigkeitsmomente

- Sei $\delta < \frac{1}{2}$ beliebig und sei A ein probabilistischer Algorithmus, der H_k (für $k \neq 1$) **exakt** mit **Fehlerwahrscheinlichkeit höchstens** δ im Streaming-Data Modell berechnet.
- Dann benötigt A Speicherkomplexität mindestens $\Omega(m)$, wenn m die Anzahl verschiedener Schlüssel ist.

- Wir lösen das Disjunktionsproblem mit Hilfe von A.
- Setze $m^* = |x| + |y|$ und beachte für $k = 0$

$$D_m(x, y) = \begin{cases} 1 & H_0 = m^* \\ 0 & H_0 < m^*, \end{cases}$$

beziehungsweise für $k > 1$

$$D_m(x, y) = \begin{cases} 1 & H_k = m^* \\ 0 & H_k > m^*. \end{cases}$$

Das zweite Häufigkeitsmoment

$$H_2 = \sum_{u \in U} a_u^2$$

Berechnung von H_2

Das Ziel: Bestimme

$$H_2 = \sum_{u \in U} a_u^2,$$

wobei $a_u = |\{i \mid x_i = u\}|$ die Häufigkeit des Schlüssels u ist.

- Warum ist die Bestimmung des zweiten Häufigkeitsmoments H_2 interessant?
  - Je größer H_2, umso unregelmäßiger sind die Schlüssel verteilt.
  - H_2 misst die Regelmäßigkeit der Schlüsselverteilung.
- Eine exakte Berechnung von H_2 gelingt nur, wenn die Menge aller ankommenden Schlüssel klein ist:
  - Vermerke die individuellen Häufigkeiten in einer Hashtabelle.

Was tun, wenn zu viele Schlüssel „im Spiel sind“?

Sketching H_2

1. Bestimme eine zufällige Hashfunktion

$$h : U \to \{-1, 1\}$$

   und setze SUMME $= 0$.
2. Wiederhole für alle Daten x_i

   Setze SUMME $=$ SUMME $+h(x_i)$.

SUMME^2 ist eine Approximation von H_2. Warum?

Der Erwartungswert von SUMME^2

$\mathbb{E}[\,\mathrm{SUMME}^2] = H_2.$

$$\begin{aligned}
\mathbb{E}[\,\mathrm{SUMME}^2\,] &= \mathbb{E}[\,(\sum_{u=1}^{m} a_u \cdot h(u))^2\,] \\
&= \mathbb{E}[\,\sum_{u=1}^{m} a_u^2 \cdot h(u)^2\,] + \mathbb{E}[\,\sum_{u \neq v} a_u \cdot a_v \cdot h(u) \cdot h(v)\,] \\
&= \mathbb{E}[\,\sum_{u=1}^{m} a_u^2\,] = \sum_{u=1}^{m} a_u^2 = H_2,
\end{aligned}$$

denn

- $h(u)^2 = 1$ für jedes u und
- $\mathbb{E}[\,h(u) \cdot h(v)\,] = 0$ für $u \neq v$.

Wie genau ist die Schätzung?

- Die Ungleichung von Tschebyscheff: Große Abweichungen vom Erwartungswert sind unwahrscheinlich, wenn die Varianz klein ist.
- Wir müssen die Varianz von SUMME^2 bestimmen!

- $V[\ \text{SUMME}^2\ ] = \mathbb{E}[\ \text{SUMME}^4\ ] - \mathbb{E}[\ \text{SUMME}^2\ ]^2$.
- Es ist $\mathbb{E}[\ \text{SUMME}^4\ ] = \mathbb{E}[\ (\sum_{u=1}^{m} a_u \cdot h(u))^4\ ]$:
  - Beim Ausmultiplizieren treten Terme der Form $\mathbb{E}[\ h(u_1) \cdot h(u_2) \cdot h(u_3) \cdot h(u_4)\ ]$ auf.
  - Wir nehmen vierfache Unabhängigkeit an!

$$\begin{aligned}\mathbb{E}[\ \text{SUMME}^4\ ] &= \mathbb{E}[\ \sum_{u=1}^{m} a_u^4 \cdot h(u)^4\ ] + \binom{4}{2} \cdot \mathbb{E}[\ \sum_{u \neq v} a_u^2 a_v^2 \cdot h(u)^2 h(v)^2\ ] \\ &= \sum_{u=1}^{m} a_u^4 + 6 \cdot \sum_{u \neq v} a_u^2 \cdot a_v^2\end{aligned}$$

Wie genau ist die Schätzung?

$\mathbb{E}[\text{SUMME}^4] = \sum_{u=1}^{m} a_u^4 + 6 \cdot \sum_{u \neq v} a_u^2 \cdot a_v^2$.

und wir erhalten die Varianz

$$\begin{aligned} V[\,\text{SUMME}^2\,] &= \sum_{u=1}^{m} a_u^4 + 6 \cdot \sum_{u \neq v} a_u^2 \cdot a_v^2 - \left(\sum_{u=1}^{m} a_u^2 \right)^2 \\ &= 4 \cdot \sum_{u \neq v} a_u^2 \cdot a_v^2 \leqslant 2 \cdot H_2^2. \end{aligned}$$

Als Konsequenz der Tschebyscheff Ungleichung folgt

$$\text{prob}[\,|\text{SUMME}^2 - H_2| > \lambda \cdot H_2\,] \leqslant \frac{2 \cdot H_2^2}{\lambda^2 \cdot H_2^2} = \frac{2}{\lambda^2}.$$

Große Abweichungen haben eine zu hohe Wahrscheinlichkeit!

Boosting

- Wähle k Hashfunktionen $h_1, \ldots, h_k$.
- Gib $\text{SCHÄTZUNG}_k = \frac{\sum_{i=1}^{k} \text{SUMME}_i^2}{k}$ aus.
  - Der Erwartungswert bleibt stabil, aber die Varianz fällt um Faktor k.
  - Es ist deshalb $\text{prob}[\ |\text{SCHÄTZUNG}_k - H_2| > \lambda \cdot H_2\ ] \leqslant \frac{2}{k \cdot \lambda^2}$.
  - Für $k = \frac{8}{\lambda^2}$ ist die Fehlerwahrscheinlichkeit höchstens $\frac{1}{4}$.
- Wiederhole den „Sketching Algorithmus“ $k^* = k \cdot s$ mal.
  - Teile die Summenergebnisse in s Klassen von je k Wiederholungen auf und berechne das Mittel für jede Klasse.
  - Bestimme den Median M der s Schätzungen.

Es ist $|\text{M} - H_2| > \lambda \cdot H_2$ mit Wahrscheinlichkeit höchstens $2^{-\Omega(s)}$.

- Wenn $|\text{M} - H_2| > \lambda \cdot H_2$, dann liegt mindestens die Hälfte aller Einzelschätzungen außerhalb.
- Mindestens doppelt so viele Einzelschätzungen wie erwartet liegen außerhalb: Wende die Chernoff-Schranke an.

Das Ergebnis

- Eine bis auf den Faktor $1 + \lambda$ exakte Schätzung von H_2 wird mit Wahrscheinlichkeit mindestens $1 - \delta$ erreicht.
- Die Speicherplatzkomplexität und die Laufzeit pro Schlüssel ist durch $O(\frac{1}{\lambda^2} \cdot \log_2(\frac{1}{\delta}))$ beschränkt.

- Es ist $|\mathrm{M} - H_2| > \lambda \cdot H_2$ mit Wahrscheinlichkeit höchstens $2^{-\Omega(s)}$.
- Setze $s = \log_2(\frac{1}{\delta})$ und die Fehlerwahrscheinlichkeit ist durch $O(\delta)$ beschränkt.
- Es ist $k = \frac{8}{\lambda^2}$: Pro Schlüssel müssen $O(\frac{\log_2(\frac{1}{\delta})}{\lambda^2})$ Hashfunktionen ausgewertet werden.

H_2 lässt sich schnell und speichereffizient approximieren, solange keine zu scharfe Approximation verlangt wird.

Das nullte Häufigkeitsmoment

$$|\{ u \in U \mid a_u \geqslant 1 \}|$$

Die Bestimmung von H_0

- Wenn die **Hauptspeicher**größe mindestens $\Omega(n)$ beträgt:
  - Benutze Hashing.
- Wenn die Daten in den **Sekundärspeicher** passen:
  - Lege die Daten im Sekundärspeicher ab und sortiere zum Beispiel mit Mergesort.
- Wenn selbst der Sekundärspeicher überfordert ist:
  - Wir müssen eine approximative Berechnung von H_0 erlauben
  - und benutzen randomisierte Algorithmen.

Hashing für die Berechnung von H_0

1. Die Vorbereitungsphase:
   - Wähle eine natürliche Zahl $T > 1$ und
   - eine zufällige Hashfunktion $h : U \to \{1, \ldots, T\}$.
   - Setze die boolesche Variable GROSS auf falsch.
2. Durchlaufe die Eingabe $(x_1, \ldots, x_n)$:
   - Wenn $h(x_i) = T$ für ein i, dann setze GROSS auf wahr.
3. Gib die Antwort GROSS aus.

Eine Diskussion

- Die Intuition:
  - Die Wahrscheinlichkeit, dass $h(x) = T$ gilt, ist $\frac{1}{T}$.
  - Wenn der Hashwert T in t Versuchen nicht erreicht wird, dann geschieht dies mit Wahrscheinlichkeit
$$(\frac{T-1}{T})^t = (1 - \frac{1}{T})^t \approx \exp^{-t/T},$$
vorausgesetzt, alle t Zahlen sind paarweise verschieden.
  - Für $t \leqslant T$ verschiedene Schlüssel wird die Wahrscheinlichkeit eines „Treffers“ beobachtbar kleiner als für $t \geqslant (1 + \varepsilon) \cdot T$ Schlüssel sein.
- Schätzungen mit Hilfe dieses Verfahrens sind ungenau und besitzen substantielle Fehlerwahrscheinlichkeiten.
  - Arbeite mit mehreren, unabhängig von einander ausgewürfelten Hashfunktionen
  - und variiere den Schwellenwert T in Potenzen von $1 + \varepsilon$.

Stichproben verschiedener Schlüssel

H_0 und Stichproben

- Eine Stichprobe gleichverteilt gezogener Schlüssel ist ungeeignet.
- Berechne stattdessen eine **Stichprobe verschiedener Schlüssel**.
- Und wenn die Stichprobe überläuft?
  - Weise jedem Schlüssel eine **Priorität** zu.
  - Wenn die Stichprobe S alle Schlüssel der Mindestpriorität P enthält und wenn S überläuft, dann setze die Mindestpriorität auf $P + 1$:

    Entferne alle Schlüssel der Priorität P aus der Stichprobe.

- Wie sollten wir Prioritäten berechnen?
- Von der Anzahl verschiedener Schlüssel der Priorität mindestens P müssen wir auf die Anzahl der verschiedenen Schlüssel **rückschließen** können.
  - Prioritäten müssen randomisiert berechnet werden.

Berechnung der Prioritäten

Die Zweierpotenz m sei mindestens so groß wie die Anzahl aller möglichen verschiedenen Schlüssel.

- Wähle Parameter $A \in \{1, \ldots, m-1\}$ und $B \in \{0, \ldots, m-1\}$ zufällig.
- Arbeite mit der Hashfunktion $h_{A,B}(u) = (A \cdot u + B) \bmod m$.
- Die **Prioritätszuweisung**:

$$p(u) \quad = \quad \text{die Anzahl der führenden Nullen in der Binärdarstellung von } h_{A,B}(u).$$

Die Binärdarstellung von $h_{A,B}(u)$ werde durch führende Nullen auf die exakte Länge $\log_2 m$ aufgefüllt.

Der relative Anteil der Schlüssel u mit $p(u) \geqslant k$ beträgt $\mathbf{2^{-k}}$.

Der Algorithmus

1. Würfle eine Hashfunktion $h_{A,B}$ aus.
2. Setze **PRIORITÄT**$= 0$ und **STICHPROBE**$= \emptyset$.
   S sei die Maximalgröße einer Stichprobe.
3. Wiederhole für $i = 1, \ldots,$:
   - Füge x_i zur Menge STICHPROBE hinzu, falls $p(x_i) \geqslant$ PRIORITÄT.
   - Solange STICHPROBE mehr als S Schlüssel besitzt, entferne alle Schlüssel x mit $p(x) =$ PRIORITÄT und erhöhe PRIORITÄT um 1.
4. Setze $p =$ PRIORITÄT und gib die Schätzung

$$2^p \cdot |\textbf{STICHPROBE}|$$

aus.

Analyse

Es ist $\text{prob}[\, p(x) \geqslant p \,] = 2^{-p}$.

- Wenn p die Priorität am Ende der Berechnung ist, gilt

$$\begin{aligned}
\mathbb{E}[\, |\textbf{STICHPROBE}| \,] &= \mathbb{E}[\, |\{x \mid p(x) \geqslant p, \text{ es gibt } i \text{ mit } x = x_i\}| \,] \\
&= \sum_{x \text{ ein Schluessel}} \text{prob}[\, p(x) \geqslant p \,] \\
&= 2^{-p} \cdot \text{Anzahl verschiedener Schlüssel.}
\end{aligned}$$

- Also folgt

$$\mathbb{E}[\, 2^p \cdot |\textbf{STICHPROBE}| \,] = \text{Anzahl verschiedener Schlüssel.}$$

Die Schätzung ist gut, wenn große Abweichungen vom Erwartungswert $\mathbb{E}[\, |\textbf{STICHPROBE}| \,]$ unwahrscheinlich sind.

Die Tschebyscheff Ungleichung

V sei die Varianz der Zufallsvariablen X. Dann gilt

$$\text{prob}[\ |X - \mathbb{E}[X]| > t\ ] \leqslant \frac{V}{t^2}.$$

- **Übungsaufgabe**: var|STICHPROBE| $\leqslant \mathbb{E}[\ |\text{STICHPROBE}|\ ]$.
- Für $E = \mathbb{E}[\ |\text{STICHPROBE}|\ ]$ ist

$$\text{prob}[\ |\ E - |\text{STICHPROBE}|\ |\ > \varepsilon \cdot E\ ] = O(\frac{E}{\varepsilon^2 \cdot E^2}) = O(\frac{1}{\varepsilon^2 \cdot E}).$$

- $A = 2^p \cdot E$ gilt für die Anzahl A verschiedener Schlüssel:

$$\begin{aligned} & \text{prob}[\ |\ A - 2^p \cdot |\text{STICHPROBE}|\ |\ > \varepsilon \cdot A\ ] \\ = \ & \text{prob}[\ 2^p \cdot |\ E - |\text{STICHPROBE}|\ |\ > \varepsilon \cdot 2^p \cdot E\ ] \\ = \ & \text{prob}[\ |\ E - |\text{STICHPROBE}|\ |\ > \varepsilon \cdot E\ ] = O(\frac{1}{\varepsilon^2 \cdot E}). \end{aligned}$$

Zusammenfassung

Mit Wahrscheinlichkeit höchstens $O(\frac{1}{\varepsilon^2 . E})$ weicht die Schätzung um mehr als $\varepsilon \cdot A$ von der Anzahl A verschiedener Schlüssel ab.

Wie lässt sich die Schätzung verbessern?

- Arbeite mit mehreren, unabhängig voneinander ausgewürfelten Hashfunktionen.
- Gib den Median der Schätzungen als endgültige Schätzung aus.
  - Warum den Median?
  - Der Einfluss von Ausreissern auf den Median ist nur schwach.

Wir benutzen dasselbe Verfahren wie für die Bestimmung von H_2.

Heavy Hitter

Heavy Hitter: Was geht und was nicht?

- Wir wissen: Die Bestimmung eines Schlüssels mit großer Häufigkeit verlangt einen zu großen Speicher.
- Übungsaufgabe: Deterministische Algorithmen benötigen Speichergröße $\Omega(m)$ bei m verschiedenen Schlüsseln, wenn der Schlüssel mit Häufigkeit größer als 50% zu bestimmen ist.

- Verlange nur, dass eine Menge K von höchstens $\frac{1}{p}$ Schlüsseln berechnet wird, unter denen sich alle Schlüssel mit Häufigkeit größer als p% befinden.
- Wie bestimmt man einen Schlüssel mit Häufigkeit $> 50\%$?
  - Entferne **Paare verschiedener Schlüssel**.
  - Der Mehrheitschlüssel verliert nie seine Mehrheitseigenschaft und wird bis zuletzt überleben.

Der Zähleralgorithmus

1. Der Datenstrom bestehe aus dem Vektor $x = (x_1, x_2, \ldots, x_n)$ mit $x_i \in \{1, \ldots, m\}$.
2. Die Menge K ist anfänglich leer und das Array "Zähler" ist auf Null initialisiert.
3. For $i = 1$ to n do
   (a) Wenn $x_i \in K$, dann erhöhe den Zähler von x_i um Eins.
   (b) Wenn $x_i \notin K$, dann füge x_i zu K hinzu und setze den Zähler von x_i auf Eins.
   - ★ Wenn K jetzt mehr als $\frac{1}{p}$ Elemente besitzt, dann erniedrige alle Zähler um Eins und entferne alle Schlüssel mit Zählerstand Null.

Übungsaufgabe:
Alle Schlüssel mit Häufigkeit größer als p% liegen in K.

Der Algorithmus rechnet in Linearzeit und die Menge K besteht aus höchstens $1/p$ Schlüsseln.

Eine verallgemeinerte Häufigkeitsanfrage

- Der Datenstrom habe die Form $((x_i, c_i) \mid i)$ mit $c_i \geqslant 0$.
- $H(x) = \sum_{x_k = x} c_k$ ist die verallgemeinerte Häufigkeit von x.
  Wir erhalten die konventionelle Häufigkeit für $c_i = 1$.
- Wir möchten $H(x)$ mit einer „Ungenauigkeit“ von höchstens ε und einer Wahrscheinlichkeit von mindestens $1 - \delta$ berechnen.

- Wir arbeiten mit $k = \lceil \ln(\frac{1}{\delta}) \rceil$ Hashfunktionen $h_1, \ldots, h_k : U \to \{1, \ldots, w\}$ für

$$w = \frac{e}{\varepsilon}$$

  Zellen.
- Wir verwenden **zählende Bloom-Filter**, die den einzelnen Hashfunktionen disjunkte Teilarrays $Z_1, \ldots, Z_k$ des Arrays $Z = (Z_1, \ldots, Z_k)$ zuweisen.
  - Jedes Z_i besteht aus w Zellen.

Der Bloom Filter Sketch

1. Die Arrays $Z_1, \ldots, Z_k$ sind alle zu Null initialisiert.
2. Wiederhole: Wenn das Paar (x_i, c_i) empfangen wird, dann setze für jedes $j \leqslant k$

$$Z_j[h_j(x_i)] = Z_j[h_j(x_i)] + c_i.$$

3. Eine verallgemeinerte Häufigkeitsfrage für Schlüssel x wird mit

$$H'(x) = \min_{1 \leqslant j \leqslant k} Z_j[h_j(x)]$$

beantwortet.

Analyse (1/3)

- Es ist stets $H(x) \leqslant Z_j[h_j(x)]$ und $H(x) \leqslant H'(x)$ folgt.
- Mit welcher Wahrscheinlichkeit wird $H'(x)$ zu groß?

Definiere die Zufallsvariable $K_j^x = \sum_{y \in U \text{ mit } x \neq y \text{ und } h_j(x)=h_j(y)} H(y)$.

Es ist $\text{prob}[\ H'(x) > H(x) + \varepsilon \cdot \sum_{y \in U} H(y)\ ]$

$$
\begin{aligned}
&= \text{prob}[\ \forall j \leqslant k :\ Z_j[h_j(x)] > H(x) + \varepsilon \cdot \sum_{y \in U} H(y)\ ] \\
&= \text{prob}[\ \forall j \leqslant k :\ H(x) + K_j^x > H(x) + \varepsilon \cdot \sum_{y \in U} H(y)\ ] \\
&= \text{prob}[\ \forall j \leqslant k :\ K_j^x > \varepsilon \cdot \sum_{y \in U} H(y)\ ] \\
&= (\text{prob}[\ K_j^x > \varepsilon \cdot \sum_{y \in U} H(y)\ ])^k.
\end{aligned}
$$

$\text{prob}[\, H'(x) > H(x) + \varepsilon \cdot \sum_{y \in U} H(y) \,] = (\text{prob}[\, K_j^x > \varepsilon \cdot \sum_{y \in U} H(y) \,])^k$.

Wie groß kann K_j^x werden?

- Die Null-Eins Zufallsvariable $K_j^x(y)$ nehme genau dann den Wert Eins an, wenn $h_j(x) = h_j(y)$.
  - Es ist $\mathbb{E}[K_j^x(y)] \leqslant 1/w$ für $x \neq y$.
- Wir schätzen den Erwartungswert von K_j^x ab: $\mathbb{E}[\, K_j^x \,]$

$$= \mathbb{E}\Big[\sum_{y \in U:\ x \neq y, h_j(x) = h_j(y)} H(y) \,\Big] = \sum_{y \in U \text{ mit } x \neq y} H(y) \cdot \mathbb{E}[\, K_j^x(y) \,]$$

$$\leqslant \sum_{y \in U \text{ mit } x \neq y} \frac{H(y)}{w} \leqslant \frac{\varepsilon}{e} \cdot \sum_{y \in U} H(y), \quad \text{denn } w = \frac{e}{\varepsilon}.$$

Analyse (3/3)

Wende die Markoff-Ungleichung $\mathrm{pr}[\,|X| \geqslant a] \leqslant \mathbb{E}[|X|]/a$ an:

$$\begin{aligned} & \mathrm{prob}[\; H'(x) > H(x) + \varepsilon \cdot \sum_{y \in U} H(y) \;] \\ = \;& (\mathrm{prob}[\; K_j^x > \varepsilon \cdot \sum_{y \in U} H(y) \;])^k \\ \leqslant \;& (\frac{\varepsilon}{e} \cdot \sum_{y \in U} H(y) \;/\; \varepsilon \cdot \sum_{y \in U} H(y))^k = e^{-k} \leqslant \delta. \end{aligned}$$

Für $k = \lceil \ln(\frac{1}{\delta}) \rceil$ und $w = \frac{e}{\varepsilon}$ gilt

$$H(x) \leqslant H'(x) \leqslant H(x) + \varepsilon \cdot \sum_{y \in U} H(y)$$

mit Wahrscheinlichkeit mindestens $1 - \delta$.

Heavy Hitter: Zusammenfassung

Der **Zähleralgorithmus**: Alle Schlüssel mit Häufigkeit größer als $p\%$ werden bestimmt; allerdings werden noch weitere, höchstens aber $\frac{1}{p}$ Schlüssel ausgegeben.

Der Algorithmus rechnet in Linearzeit.

Der **Bloom Filter Sketch**: Für $k = \lceil \ln(\frac{1}{\delta}) \rceil$ und $w = \frac{e}{\varepsilon}$ gilt

$$H'(x) \leqslant H(x) + \varepsilon \cdot \sum_{y \in U} H(y)$$

mit Wahrscheinlichkeit mindestens $1 - \delta$. Höchstens $O(\frac{1}{\varepsilon} \cdot \ln(\frac{1}{\delta}))$ Zellen werden benutzt und die Laufzeit pro Anfrage ist höchstens $O(\ln(\frac{1}{\delta}))$.

In Anwendungen muss ε SEHR klein gewählt werden!

Häufigkeitseigenschaften

Eigenschaften von Häufigkeiten

- Wir interpretieren Teilmengen $E_n \subseteq \{1, \ldots, n\}$ als Eigenschaften von Häufigkeiten:
  - Schlüssel x hat Eigenschaft E_n, falls die Häufigkeit von x –unter den ersten n Schlüsseln– in E_n liegt.

  Beispiel: $E_n = \{1, \ldots, K\}$ formalisiert die Eigenschaft „höchstens K-mal aufzutreten“.
- $E_n^* = \{x_j \mid j \leqslant n \text{ und } x_j \text{ hat Eigenschaft } E_n\}$ ist die Menge der Schlüssel mit Eigenschaft E_n. Wenn V_n die Menge der verschiedenen Schlüssel (unter den ersten n) ist, dann ist

$$p[E_n] = \frac{|E_n^*|}{|V_n|}$$

  die Wahrscheinlichkeit von E_n.

Wie scharf kann $p[E_n]$ approximiert werden?

Der Jaccard Koeffizient

Wir erinnern an den Jaccard Koeffizienten

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}.$$

Es ist $\min_\pi(A) = \min_\pi(B)$ mit Wahrscheinlichkeit $J(A,B)$, wobei

$$\min_\pi(X) = \min\{\pi(x) \mid x \in X\}.$$

- Es ist $J(E_n^*, V_n) = \frac{|E_n^* \cap V_n|}{|E_n^* \cup V_n|} = \frac{|E_n^*|}{|V_n|} = p[E_n]$.
- Als Konsequenz:

$$\text{prob}[\min_\pi(E_n^*) = \min_\pi(V_n)] = p[E_n].$$

Berechnung von $p[E_n]$

1. Wähle zufällig k Permutationen $\pi_1, \ldots, \pi_k$.
   Setze $\min_j(0) = \infty$ sowie $\text{Zähler}_j = 0$ für alle $j \leqslant k$.
2. Wenn x_{i+1} zu verarbeiten ist, dann wiederhole für jedes $j \leqslant k$
   (a) Berechne $\pi_j(x_{i+1})$.
   (b) Wenn $\pi_j(x_{i+1}) < \min_j(i)$, dann setze
   $\min_j(i+1) = \pi_j(x_{i+1})$ und $\text{Zähler}_j = 1$.

   /* Der Schlüssel x_{i+1} tritt zum ersten Mal auf und sein Zähler wird deshalb auf Eins gesetzt. */
   (c) Wenn $\pi_j(x_{i+1}) = \min_j(i)$, dann setze $\text{Zähler}_j = \text{Zahler}_j + 1$.

   /* Ein weiteres Vorkommen des minimalen Schlüssels x_{i+1} für π_j: Sein Zähler wird inkrementiert. */
3. k^* sei die Anzahl der Schlüssel $\min_j(n)$ mit Eigenschaft E_n:
   - Gib k^*/k als Schätzung für $p[E_n]$ aus.

Definiere die Zufallsvariablen

$$X_j = \begin{cases} 1 & \min_j(n) \in E_n^*, \\ 0 & \text{sonst.} \end{cases}$$

- $\mathbb{E}[\sum_{j=1}^{k} X_j] = \sum_{j=1}^{k} \mathbb{E}[X_j] = k \cdot p[E_n]$.
- $X_1, \ldots, X_k$ sind unabhängig: Die Chernoff-Schranke liefert

$$\text{prob}[\ |\sum_{j=1}^{k} X_j - k \cdot p[E_n]| \geqslant \varepsilon \cdot k \cdot p[E_n]\ ] \leqslant 2 \cdot e^{-k \cdot p[E_n] \cdot \varepsilon^2/3}.$$

- Also folgt $\text{prob}[\ |k^*/k - p[E_n]| \geqslant \varepsilon \cdot p[E_n]\ ]$

$$\begin{aligned} &= \text{prob}[\ |k^* - k \cdot p[E_n]| \geqslant \varepsilon \cdot k \cdot p[E_n]\ ] \\ &\leqslant 2 \cdot e^{-k \cdot p[E_n] \cdot \varepsilon^2/3}. \end{aligned}$$

Analyse (2/2)

$\text{prob}[\ |k^*/k - p[E_n]| \geqslant \varepsilon \cdot p[E_n]\ ] \leqslant 2 \cdot e^{-k \cdot p[E_n] \cdot \varepsilon^2/3}$.

- Für eine ε-Approximation mit Fehlerwahrscheinlichkeit höchstens δ fordere

$$2 \cdot e^{-k \cdot p[E_n] \cdot \varepsilon^2/3} \leqslant \delta.$$

- Äquivalent dazu $-k \cdot p[E_n] \cdot \varepsilon^2/3 \leqslant \ln(\frac{\delta}{2})$,
  bzw. $k \geqslant \frac{3}{\varepsilon^2 \cdot p[E_n]} \cdot \ln(\frac{2}{\delta})$.

- Für $k = O(\frac{1}{\varepsilon^2 \cdot p[E_n]} \cdot \ln(\frac{1}{\delta}))$ erreichen wir eine $(1+\varepsilon)$-Approximation von $p[E_n]$ mit Wahrscheinlichkeit mindestens $1 - \delta$.
- Die Laufzeit pro Schlüssel ist durch $O(k)$ und die Speicherplatzkomplexität durch $O(k \cdot \log_2 n)$ beschränkt.

Wahrscheinlichkeit von Eigenschaften

- Wir erreichen eine $(1+\varepsilon)$-Approximation von $p[E_n]$ mit Wahrscheinlichkeit mindestens $1-\delta$ für $k = O(\frac{1}{\varepsilon^2 \cdot p[E_n]} \cdot \ln(\frac{1}{\delta}))$.
- Die Laufzeit pro Schlüssel ist durch $O(k)$ und die Speicherplatzkomplexität durch $O(k \cdot \log_2 n)$ beschränkt.

+ Eine gute Approximation gelingt mit Min-Hashing relativ schnell und mit kleinem Speicher.
- Allerdings darf die Eigenschaft nicht zu unwahrscheinlich sein: Die Laufzeit und die Speichergröße wächst mit $\frac{1}{p[E_n]}$.
- Ein weiterer Wermutstropfen: Wir können nicht alle Schlüssel mit Eigenschaft E_n liefern, sondern nur eine kleine Auswahl.

Zeitfenster

Zeitfenster

Die Fenstergröße N sei gegeben.

- Wir betrachten einen Datenstrom und möchten die Anzahl „auffälliger“ Daten in einem Zeitfenster der Länge N zählen.
- Wir modellieren dieses Problem als das Zählen der Einsen unter den letzten N Bits in einem Datenstrom von Nullen und Einsen.

- Jeder deterministische Algorithmus, der die Anzahl der Einsen im Zeitfenster der Größe N mit relativem Fehler ε bestimmt, benötigt mindestens Speicherplatz $\Omega(\frac{1}{\varepsilon}\log_2^2 \varepsilon N)$.
- Wir zeigen, dass Speicherplatz $O(\frac{1}{\varepsilon}\log_2^2 N)$ ausreicht.

Die Datenstruktur der Körbe

Wir arbeiten mit Körben $K_1, \ldots K_m$. Der Korb K_i speichert die Anzahl der Einsen in einem sich dynamisch ändernden Zeitintervall Z_i.

- Die Zeitintervalle $Z_1, \ldots, Z_m$ bilden eine disjunkte Zerlegung eines $\{1, \ldots, N\}$ umfassenden Zeitraums.
- Die Zeitintervalle sind zeitlich geordnet $Z_1 < \cdots < Z_m$, wobei
  - Z_1 den gegenwärtigen Zeitpunkt und
  - Z_m den letzten interessanten Zeitpunkt N der Vergangenheit enthält.
- Korb K_i speichert
  - den Zeitstempel des Korb, nämlich das Alter der jüngsten erfassten Eins im Zeitintervall Z_i und
  - und die Korbgröße G_i, also die Anzahl der Einsen im Zeitintervall Z_i.
- Die Korbgrößen G_i sind stets Zweierpotenzen und die Körbe $K_1 \ldots, K_m$ besitzen aufsteigende Korbgrößen.

Zeitstempel

- Der Zeitstempel verrät, ob ein Korb noch aktuell ist:

  Wenn der Zeitstempel des letzten Korbs größer als N ist, dann kann der Korb entleert und wiederverwandt werden.
- Für alle Körbe kennen wir die exakte Anzahl der Einsen ihres Zeitintervalls, nicht aber das Ende des Zeitintervalls.
  - Wenn der letzte Korb K_m noch aktuell ist, dann folgt nur

$$\sum_{i=1}^{m-1} G_i + 1 \leqslant \text{Exakt} \leqslant \sum_{i=1}^{m-1} G_i + G_m.$$

  - Wir geben die Schätzung $\sum_{i=1}^{m-1} G_i + \frac{G_m}{2}$ ab.
  - Um den relativen Fehler durch 2ε zu beschränken, genügt die Forderung

$$\frac{G_m/2}{\sum_{i=1}^{m-1} G_i + 1} \leqslant 2 \cdot \varepsilon, \text{ bzw. stärker } \frac{(G_m - 1)/2}{\sum_{i=1}^{m-1} G_i + 1} \leqslant \varepsilon.$$

Die Größe des letzten Korbs

(1) Die Forderung $\frac{(G_m-1)/2}{\sum_{i=1}^{m-1} G_i+1} \leqslant \varepsilon$ soll stets erfüllt sein!

(2) Fordere für $k = \lceil \frac{1}{\varepsilon} \rceil$: Wenn 2^h die Korbgröße des letzten Korbs K_m ist, dann gibt es zu jedem $h' < h$ mindestens $\frac{k}{2}$ und höchstens $\frac{k}{2} + 1$ Körbe der Größe $2^{h'}$.

(1) folgt aus (2):

- Wenn Korb K_m die Korbgröße 2^r besitzt, dann gibt es mindestens $\frac{k}{2}$ Körbe der Größen $1, 2, 4, \ldots, 2^{r-1}$.
- Also gilt

$$\sum_{i=1}^{m-1} G_i + 1 \geqslant \frac{k}{2} \cdot \sum_{i=0}^{r-1} 2^i = \frac{k}{2} \cdot (2^r - 1) \geqslant \frac{1}{\varepsilon} \cdot \frac{G_m - 1}{2}.$$

Der Algorithmus

1. Zwei Variablen „TOTAL” und „LETZTER” speichern die Gesamtgröße aller Körbe, bzw. den Index des letzten Korbs.
2. Wenn der letzte Korb nicht aktuell ist, dann entferne ihn und aktualisiere TOTAL und LETZTER.
3. Wenn das neue Bit eine Null ist, dann wird es ignoriert. Sonst:
   - erzeuge einen neuen Korb mit Korbgröße 1,
   - halte den Zeitpunkt fest und erhöhe TOTAL um eins.
4. Gibt es jetzt $\frac{k}{2} + 2$ Körbe der Korbgröße 1, dann vereinige die beiden ältesten Körbe zu einem Korb der Größe 2 und bestimme den Zeitstempel des neuen Korbs.
   - Nach Verschmelzung gibt es möglicherweise zu viele Körbe der Größe 2.
   - Also setze die Überprüfung für Korbgröße 2 fort....
   - und aktualisiere gegebenenfalls die Variable LETZTER.

Ein Beispiel

- Es sei $k = 2$.
- Die Korbgrößen seien $(1, 1, 2, 4, 4, 8, 8, 16, 32)$.
- Wenn eine neue Eins eintrifft, dann
  - fasse Körbe zusammen, um $(1, 2, 2, 4, 4, 8, 8, 16, 32)$ zu erhalten.
- Nach zwei weiteren Einsen und der entsprechenden Korbzusammenfassung ergibt sich dann der Vektor $(1, 2, 4, 8, 16, 16, 32)$.

Jedes neue Bit verursacht im worst-case $O(\log_2 N)$ Operationen, während die durchschnittliche Laufzeit sogar nur konstant ist.

Das Ergebnis

- Für jedes N wird die Anzahl der Einsen unter den letzten N Bits mit relativem Fehler höchstens 2ε bestimmt.
- Die worst-case Laufzeit pro Bit ist $O(\log_2 N)$ und die amortisierte Laufzeit ist konstant.
- Die Speicherplatzkomplexität ist durch $O(\frac{1}{\varepsilon} \cdot \log_2^2 N)$ beschränkt.

Die Speichergröße:

- Es gibt bis zu $O(k \cdot \log_2 N)$ Körbe mit $k = \lceil \frac{1}{\varepsilon} \rceil$.
- Jeder Korb speichert sein Alter mit bis zu $O(\log_2 N)$ Bits.