Eine fehlende Zahl

untere Schranke: $\lceil \log n \rceil$ Bits

denn:

- Für jede Zahl $i \in \{1 \dots n\}$ betrachte die Eingabe $E_i := (i+1), (i+2), \dots, n, 1, 2, \dots, (i-1)$
  - d.h.: fehlende Zahl bei Eingabe E_i : i

- Sei A ein Datenstr.-Algo, der das Problem löst und dabei $\leq B$ Speicher-Bits benötig.

  Für jedes $i \in \{1 \dots n\}$ sei
  S_i der Speicherinhalt von A am Ende der Verarbeitung der Eingabe E_i

- Klar: Die Speicherinhalte $S_1, \dots, S_n$ müssen paarweise verschieden sein (denn: ist $S_i = S_j$, so liefert der Algo bei Eingabe E_i dieselbe Ausgabe wie bei Eingabe E_j — die korrekte Ausgabe ist aber die Zahl i bei Eingabe E_i und j bei Eingabe E_j).

- Somit: der Speicher muss groß genug sein, um n verschiedene Speicherinhalte zuzulassen.
  Bei B Speicher-Bits sind nur 2^B verschiedene Speicherinhalte möglich.
  $\Rightarrow 2^B \geq n$, also $B \geq \log n$.

MULTISET - EQUALITY

Satz: Zu entscheiden, ob zwei m-elementige Mengen

$a = \{x_1, \dots, x_m\} \subseteq \{1, \dots, n\}$ und

$b = \{y_1, \dots, y_m\} \subseteq \{1, \dots, n\}$

gleich sind, erfordert mindestens $\log \binom{n}{m}$ Kommunikations-Bits

Beweis: Hier für "1-Weg-Komm.protokolle", d.h.: nur Alice kann Bob Informationen schicken (Bob kann nichts an Alice schicken).

Beachte: Es gibt genau $\binom{n}{m}$ verschiedene m-elementige Teilmengen von $\{1, \dots, n\}$.

Sei $M_1, M_2, \dots, M_{\binom{n}{m}}$ eine Liste all dieser Teilmengen.

Klar: für jedes $i \in \{1, \dots, \binom{n}{m}\}$ muss für

$a = M_i$ und $b = M_i$

Bob zu dem Schluss kommen, dass die beiden Mengen gleich sind.

Angen., es gibt Indizes $i \neq j$ sd. bei $a = M_i$, $b = M_i$ dieselbe Kommunikation stattfindet, wie bei $a = M_j$, $b = M_j$. Dann muss diese Komm. auch bei $a = M_i$, $b = M_j$

stattfinden, und daher muss Bob auch dann zu dem Schluss kommen, dass

M_i und M_j gleich sind. —

Das ist aber falsch!

Daher: für jeden der Indices $i, j \in \{1, \dots, \binom{n}{m}\}$ muss eine andere Komm. stattfinden.

Daher $\geq \log \binom{n}{m}$ Bit nötig.

Nebenrechnung:

$n := m^2 \Rightarrow \log \binom{n}{m} \geq m \cdot \log m$,

denn: $\binom{m^2}{m} = \frac{m^2 \cdot (m^2-1) \cdots (m^2-m+1)}{m \cdot (m-1) \cdots 1} \geq \underbrace{m \cdot m \cdots m}_{m\text{-mal}} = m^m$

also $\log \binom{m^2}{m} \geq \log (m^m) = m \cdot \log m$.

Details zum Randomisierten Algorithmus zur Lösung des MULTISET-EQUALITY Problems:

Setze $m' := \log n$ (die zum Speichern einer Zahl $\in n$ benötigten Bits)

(1) Wähle eine Primzahl

$$p_1 \leq \underbrace{m^3 \cdot n^2 \cdot \log(m^3 \cdot n^2)}_{=: k}$$

zufällig, gleichverteilt aus der Menge aller Primzahlen $\leq k$.

(2) Sei p_2 eine beliebige Primzahl mit

$$3k < p_2 \leq 6k$$

Eine solche Primzahl p_2 gibt es gemäß dem Bertrandschen Postulat — das ist ein Satz der Zahlentheorie, der besagt, dass es für jede natürliche Zahl n eine Primzahl p gibt mit

$$n < p \leq 2n.$$

(3) Wähle eine Zahl

$$z \in \{1, 2, \ldots, p_2 - 1\}$$

zufällig, gleichverteilt.

(4) Für $i \in \{1, \dots, m\}$ setze

$$a_i' := a_i \bmod p_1$$

$$b_i' := b_i \bmod p_1$$

Falls

$$\sum_{i=1}^{m} z^{a_i'} \equiv \sum_{i=1}^{m} z^{b_i'} \bmod p_2,$$

so akzeptiere; ansonsten verwerfe

Implementierung als Datenstromalgorithmus:

Zu (1):
Wiederhole:
Wähle zufällig, gleichverteilt eine
natürliche Zahl $\leq k$
bis diese Zahl eine Primzahl ist.

Zu (2):

p_2 kann man z.B. finden, indem man nacheinander für alle ungeraden Zahlen zwischen $3k$ und $6k$ testet, ob sie eine Primzahl sind und stoppt, sobald man eine Primzahl gefunden hat.

Die Schritte (1) - (3) werden durchgeführt, bevor die Verarbeitung des eigentlichen Datenstroms (der aus den Zahlen $a_1, \dots, a_m, b_1, \dots, b_n$ besteht) beginnt.

Zu (4): Hier findet die eigentliche Verarbeitung des Datenstroms statt!

Zu jedem Zeitpunkt während des Lesens des Datenstroms merkt sich der Algorithmus zwei Zahlen $A, B \in \{0, 1, \dots, p_2 - 1\}$.

Initialisierung:

$A := 0$

$B := 0$

Beim Lesen einer Zahl a_i:

$a_i' := a_i \bmod p_1$

$H := z^{a_i'} \bmod p_2$

$A := A + H$

Beim Lesen einer Zahl b_i:

$b_i' := b_i \bmod p_1$

$H := z^{b_i'} \bmod p_2$

$B := B + H$

Am Ende der Verarbeitung des Datenstroms gilt dann

$$A = \sum_{i=1}^{m} z^{a_i} \mod p_2$$

$$B = \sum_{i=1}^{m} z^{b_i} \mod p_2$$

Falls $A = B$, so akzeptiere; ansonsten verwerfe.

Benutzter Speicherplatz:

Es werden konstant viele natürliche Zahlen $\leq p_2$ gespeichert, für $p_2 \leq 6k$ mit

$k = m^3 \cdot n' \cdot \log(m^3 \cdot n')$.

Zum Speichern einer solchen Zahl reichen $\lceil \log(6k) \rceil$ Bits.

Beachte: $\log(6k) = \log(6) + \log(k)$ und

$$\begin{aligned} \log(k) &= \log(m^3 \cdot n' \cdot \log(m^3 \cdot n')) \\ &= \log(m^3 \cdot n') + \log\log(m^3 \cdot n') \\ &\leq 3 \cdot \log(m \cdot n') + \log(3 \cdot \log(m \cdot n')) \\ &= O(\log N) \end{aligned}$$

wobei N die Bit-Länge des Eingabestroms ist.

Korrektheitsanalyse:

Ziel: Nachweis, dass der Algorithmus mit hoher Wahrscheinlichkeit tatsächlich das korrekte Ergebnis liefert.

Fall 1: Die beiden Multimengen sind gleich, d.h. $\{a_1, \dots, a_m\} = \{b_1, \dots, b_m\}$, und somit auch $\{a_1', \dots, a_m'\} = \{b_1', \dots, b_m'\}$.

Am Ende der Berechnung gilt:

$$A = \sum_{i=1}^{m} z^{a_i'} \mod p_2 \quad \text{und}$$

$$B = \sum_{i=1}^{m} z^{b_i'} \mod p_2 .$$

D.h.: $A = f(z) \mod p_2$ und

$B = g(z) \mod p_2$

für die Polynome

$$f(x) := \sum_{i=1}^{m} x^{a_i'} \quad \text{und} \quad g(x) := \sum_{i=1}^{m} x^{b_i'}$$

Wegen $\{a_1', \dots, a_m'\} = \{b_1', \dots, b_m'\}$, sind diese beiden

Polynome identisch.
Insbes gilt also für die Zahl z, dass
$f(z) = g(z)$.
Daher gilt $A = B$, und unser Algo akzeptiert (korrekterweise), und zwar für jede Wahl der Werte p_1, p_2, z.

Fall 2: Die beiden Multimengen sind verschieden, d.h. $\{a_1, \dots, a_n\} \neq \{b_1, \dots, b_m\}$.

Wir wollen zeigen, dass die Wahrscheinlichkeit, dass unser Algorithmus dennoch akzeptiert (d.h. die falsche Antwort liefert) sehr klein ist.

Behauptung 1: Die Wahrscheinlichkeit, dass $\{a_1', \dots, a_n'\} = \{b_1', \dots, b_m'\}$ ist, ist $\leq O(\frac{1}{m})$

(Die Wahrscheinlichkeit läuft hier darüber, dass p_1 zufällig gewählt wird).

Beweis:

Wir nutzen hier folgenden Satz, den wir hier nicht beweisen:

Satz (siehe z.B. Theorem 7.5 im Buch "Randomized Algorithms" von Motwani und Raghavan, 1995)

Seien $\ell, n' \in \mathbb{N}$, sei $k := \ell \cdot n' \cdot \log(\ell \cdot n')$ und sei $0 < x < 2^{n'}$.

Für eine zufällig gleichverteilt gewählte Primzahl $p \leq k$ gilt:

$$\Pr(x \equiv 0 \bmod p) \leq O\left(\frac{1}{\ell}\right) \xrightarrow{\ell \to \infty} 0$$

Wir wenden diesen Satz für $\ell := m^3$ und alle $i,j \in \{1, \dots, m\}$ mit $a_i \neq b_j$ auf die Zahl $x := |a_i - b_j|$ an und erhalten dadurch:

$$\Pr(\exists i,j \leq m \text{ mit } a_i \neq b_j \text{ und } a_i \equiv b_j \bmod p)$$

$$\leq \sum_{\substack{i,j \leq m \\ \text{mit } a_i \neq b_j}} \Pr(a_i \equiv b_j \bmod p)$$

$$= \sum_{\substack{i,j \leq m \\ \text{mit } a_i \neq b_j}} \underbrace{\Pr(|a_i - b_j| \equiv 0 \bmod p)}_{\text{Satz} \leq O\left(\frac{1}{\ell}\right)}$$

$$\leq O\left(m^2 \cdot \frac{1}{\ell}\right) \underset{\ell = m^3}{=} O\left(\frac{1}{m}\right) \xrightarrow{m \to \infty} 0$$

Somit gilt auch:

$$\Pr\left(\{a_1,\dots,a_m\} = \{b_1,\dots,b_m\}\right) \leq O\left(\frac{1}{m}\right).$$

□ Beh 1

Gemäß Beh 1 ist

$$\Pr\left(\{a_1,\dots,a_m\} \neq \{b_1,\dots,b_m\}\right) \geq 1 - O\left(\frac{1}{m}\right) \xrightarrow{m\to\infty} 1$$

○ Wir betrachten daher im Folgenden nur noch den (sehr wahrscheinlichen) Fall, dass

$$\{a_1,\dots,a_m\} \neq \{b_1,\dots,b_m\}.$$

Die beiden Polynome

$$f(x) := \sum_{i=1}^{m} x^{a_i} \quad \text{und} \quad g(x) := \sum_{i=1}^{m} x^{b_i}$$

○ sind dann verschieden, und das Polynom

$$h(x) := f(x) - g(x)$$

ist $\neq 0$ und hat einen Grad ≥ 1 und $< p_1$, und hat daher $< p_1$ viele Nullstellen.

Jeder Koeffizient des Polynoms h ist eine ganze Zahl zwischen $-m$ und m.

Indem wir das Polynom "modulo p_2" auswerten, betrachten wir es als

Polynom über dem Körper $GF(p_2)$ (bei dem "+" und "·" modulo p_2 gerechnet werden). Auch dort hat das Polynom $h(x)$ weniger als p_1 viele Nullstellen.

Für unser zufällig gewähltes

$$z \in \{1, 2, \ldots, p_2 - 1\}$$

gilt also

$$Pr\left(h(z) \equiv 0 \mod p_2\right) < \frac{p_1}{p_2 - 1} \leq \frac{1}{3}$$

$p_1 \leq k$, $p_2 > 3k$

d.h. unser Algorithmus akzeptiert fälschlicherweise, obwohl sowohl $\{a_1, \ldots, a_m\} \neq \{b_1, \ldots, b_m\}$ und $\{a'_1, \ldots, a'_m\} \neq \{b'_1, \ldots, b'_m\}$ gilt.

Insgesamt erhalten wir im Fall, dass $\{a_1, \ldots, a_m\} \neq \{b_1, \ldots, b_m\}$:

Die Wahrscheinlichkeit, dass unser Algorithmus (fälschlicherweise) akzeptiert, ist

$$\leq \underbrace{O\left(\frac{1}{m}\right)}_{\text{Wk, dass } \{a'_1, \ldots, a'_m\} = \{b'_1, \ldots, b'_m\}} + \underbrace{\frac{1}{3}}_{\text{Wk, dass } h(z) \equiv 0 \mod p_2} \leq \frac{1}{2}$$

für hinreichend großes m

Indem wir gleichzeitig und unabhängig voneinander $s \geq 1$ Instanzen des Algorithmus laufen lassen und nur dann akzeptieren, wenn jede der s Instanzen akzeptiert, erhalten wir einen Algorithmus, der

- Im Fall, dass $\{a_1, \ldots, a_n\} = \{b_1, \ldots, b_m\}$ ist, auf jeden Fall akzeptiert
- Im Fall, dass $\{a_1, \ldots, a_n\} \neq \{b_1, \ldots, b_m\}$ ist, mit Wahrscheinlichkeit

$$< \left(\frac{1}{2}\right)^s = \frac{1}{2^s}$$

fälschlicherweise akzeptiert, also mit Wahrscheinlichkeit

$$\geq 1 - \frac{1}{2^s}$$

korrekterweise verwirft.

Wenn wir $s = 4$ wählen, ist $2^s = 16 > 10$ und $\frac{1}{2^s} < \frac{1}{10}$ und $1 - \frac{1}{2^s} > 0{,}9$.

Dies beendet den Beweis des Satzes über einen randomisierten Datenstrom-Algorithmus für das MULTISET-EQUALITY Problem. □