technische universität
dortmund

LS 2 /
Informatik

Datenstrukturen, Algorithmen und Programmierung 2 (DAP2)

Big Data

Datenströme

- Sehr viele Daten, die mit hoher Rate ankommen
- Beispiele: Internetdatenverkehr, Webcrawls, Börsentransaktionen, Sensordaten

Typisches Szenario

- Kann nicht alle Daten abspeichern
- Algorithmus soll schnell einzelne Daten verarbeiten
- Will Statistik der Daten aufrechterhalten

Big Data

Das Datenstrommodell

- Jedes Element ist Integer aus Universum $U = \{1, \dots, N\}$
- Ein Datenstrom ist eine Sequenz $a = (a_1, \dots, a_n)$ von n Elementen aus dem Universum; Elemente können mehrfach vorkommen
- n ist unbekannt
- Speicherplatz $\mathbf{O}\left((\log(nN))^c\right)$ für n groß genug und geeignete Konstante c
- Die Sequenz a wird Element für Element verarbeitet; freier Zugriff auf den Datenstrom ist nicht möglich

Der Häufigkeitsvektor

- Der Häufigkeitsvektor F ist ein $|U|$-dimensionaler Vektor, der für jedes Element $x \in U$ die Anzahl $F(x)$ seiner Vorkommen im Datenstrom enthält

Big Data

Aufgabe: Die fehlende Zahl

- Annahme: Im Datenstrom kommen alle Elemente aus U genau einmal vor bis auf ein Element, das nicht vorkommt
- Wie kann man die fehlende Zahl finden? (mit $\mathbf{O}(\log N)$ Speicher)

Big Data

Definition (Mehrheitselement)

Ein Element ist ein Mehrheitselement, wenn es häufiger im Datenstrom auftritt als alle andere Elemente zusammen, d.h. wenn es mehr als $n/2$ mal vorkommt.

Das Mehrheitsproblem

- Annahme: Ein Mehrheitselement existiert
- Aufgabe: Finde dieses Element

Big Data

Idee

- Finde solange zwei unterschiedliche Elemente und eliminiere sie aus dem Datenstrom, bis nur ein Element übrig bleibt

Lemma 39

- Das übriggebliebene Element ist Mehrheitselement.

Beweis

- Der Datenstrom hat n Elemente.

Big Data

Idee

- Finde solange zwei unterschiedliche Elemente und eliminiere sie aus dem Datenstrom, bis nur ein Element übrig bleibt

Lemma 39

- Das übriggebliebene Element ist Mehrheitselement.

Beweis

- Der Datenstrom hat n Elemente. Jedes mal, wenn eine Kopie des Mehrheitselements x aus dem Datenstrom entfernt wurde, wurde auch ein anderes Element entfernt.

Big Data

Idee

- Finde solange zwei unterschiedliche Elemente und eliminiere sie aus dem Datenstrom, bis nur ein Element übrig bleibt

Lemma 39

- Das übriggebliebene Element ist Mehrheitselement.

Beweis

- Der Datenstrom hat n Elemente. Jedes mal, wenn eine Kopie des Mehrheitselements x aus dem Datenstrom entfernt wurde, wurde auch ein anderes Element entfernt. Da x mehr als $n/2$ mal vorkommt, kann es nicht vollständig entfernt werden, da man sonst mehr als n Elemente entfernen müsste.

Big Data

Idee

- Finde solange zwei unterschiedliche Elemente und eliminiere sie aus dem Datenstrom, bis nur ein Element übrig bleibt

Lemma 39

- Das übriggebliebene Element ist Mehrheitselement.

Beweis

- Der Datenstrom hat n Elemente. Jedes mal, wenn eine Kopie des Mehrheitselements x aus dem Datenstrom entfernt wurde, wurde auch ein anderes Element entfernt. Da x mehr als $n/2$ mal vorkommt, kann es nicht vollständig entfernt werden, da man sonst mehr als n Elemente entfernen müsste.

Big Data

Heavy Hitter Problem

- Gegeben Parameter ε
- Finde Menge M, die alle Elemente $x \in U$ mit $F(x) > \varepsilon n$ enthält und kein Element mit $F(x) < \varepsilon n/2$

Bemerkungen

- Für $\varepsilon = 0.5$ ist dies (ungefähr) das Mehrheitsproblem
- Es ist nicht möglich mit wenig Speicherplatz genau die Elemente $x \in U$ mit $F(x) > \varepsilon n$ zu finden

Big Data

Anwendungen von Heavy Hitters:

DDoS Angriffe (Distributed Denial-of-Service)

- Internet Monitoring
- Ziel-IPs der Pakete im Internetdatenverkehr werden beobachtet
- Gibt es Adressen, die sehr viele Pakete erhalten?

Big Data

KSP-Algorithmus *(Karp, Shenker und Papadimitriou)*

1. Bezeichne $a[1] \ldots a[n]$ den Datenstrom
2. $K \leftarrow \emptyset$
3. **new** array count$[1..\lceil 2/\varepsilon \rceil]$ indiziert durch K
4. **for** $i \leftarrow 1$ **to** n **do**
5. **if** $a[i]$ ist in K **then** count$\left[a[i]\right] \leftarrow$ count$\left[a[i]\right] + 1$
6. **else** insert $a[i]$ in K; set count$\left[a[i]\right] \leftarrow 1$
7. **if** $|K| > \lceil 2/\varepsilon \rceil$ **then**
8. **for all** $x \in K$ **do**
9. count$[x] \leftarrow$ count$[x] - 1$;
10. **if** count$[x] = 0$ **then** delete x from K
11. **return** alle $x \in K$ mit count$[x] \geq \varepsilon n/2$

Big Data

Satz 40

Der KSP-Algorithmus findet alle Elemente $x \in U$ mit $F(x) > \varepsilon n$ und gibt kein Element mit $F(x) < \varepsilon n/2$ zurück.

Big Data

Satz 40

Der KSP-Algorithmus findet alle Elemente $x \in U$ mit $F(x) > \varepsilon n$ und gibt kein Element mit $F(x) < \varepsilon n/2$ zurück.

Beweis

- Da der count eines Elements höchstens so groß ist wie die Anzahl seiner Vorkommen im Datenstrom, wird kein Element mit $F(x) < \varepsilon n/2$ zurückgegeben.

Big Data

Satz 40

Der $\mathrm{KSP\text{-}Algorithmus}$ findet alle Elemente $x \in U$ mit $F(x) > \varepsilon n$ und gibt kein Element mit $F(x) < \varepsilon n/2$ zurück.

Beweis

- Da der count eines Elements höchstens so groß ist wie die Anzahl seiner Vorkommen im Datenstrom, wird kein Element mit $F(x) < \varepsilon n/2$ zurückgegeben.
- Wir müssen noch zeigen, dass jedes $x \in U$ mit $F(x) > \varepsilon n$ zurückgegeben wird.

Big Data

Satz 40

Der KSP-Algorithmus findet alle Elemente $x \in U$ mit $F(x) > \varepsilon n$ und gibt kein Element mit $F(x) < \varepsilon n/2$ zurück.

Beweis

- Da der count eines Elements höchstens so groß ist wie die Anzahl seiner Vorkommen im Datenstrom, wird kein Element mit $F(x) < \varepsilon n/2$ zurückgegeben.
- Wir müssen noch zeigen, dass jedes $x \in U$ mit $F(x) > \varepsilon n$ zurückgegeben wird.
- Bei jeder Eliminierung eines Vorkommens von x werden auch $\lceil 2/\varepsilon \rceil$ andere Symbole eliminiert.

Big Data

Satz 40

Der KSP-Algorithmus findet alle Elemente $x \in U$ mit $F(x) > \varepsilon n$ und gibt kein Element mit $F(x) < \varepsilon n/2$ zurück.

Beweis

- Da der count eines Elements höchstens so groß ist wie die Anzahl seiner Vorkommen im Datenstrom, wird kein Element mit $F(x) < \varepsilon n/2$ zurückgegeben.
- Wir müssen noch zeigen, dass jedes $x \in U$ mit $F(x) > \varepsilon n$ zurückgegeben wird.
- Bei jeder Eliminierung eines Vorkommens von x werden auch $\lceil 2/\varepsilon \rceil$ andere Symbole eliminiert.
- Es können daher maximal $n/\lceil 2/\varepsilon \rceil \leq \varepsilon n/2$ Kopien eines Elements entfernt werden.

Big Data

Satz 40

Der KSP-Algorithmus findet alle Elemente $x \in U$ mit $F(x) > \varepsilon n$ und gibt kein Element mit $F(x) < \varepsilon n/2$ zurück.

Beweis

- Da der count eines Elements höchstens so groß ist wie die Anzahl seiner Vorkommen im Datenstrom, wird kein Element mit $F(x) < \varepsilon n/2$ zurückgegeben.
- Wir müssen noch zeigen, dass jedes $x \in U$ mit $F(x) > \varepsilon n$ zurückgegeben wird.
- Bei jeder Eliminierung eines Vorkommens von x werden auch $\lceil 2/\varepsilon \rceil$ andere Symbole eliminiert.
- Es können daher maximal $n/\lceil 2/\varepsilon \rceil \leq \varepsilon n/2$ Kopien eines Elements entfernt werden.
- Gilt also $F(x) > \varepsilon n$, so sind am Ende des Datenstroms noch mehr als $\varepsilon n/2$ Kopien übrig und das Element wird zurückgegeben.

Big Data

Satz 40

Der KSP-Algorithmus findet alle Elemente $x \in U$ mit $F(x) > \varepsilon n$ und gibt kein Element mit $F(x) < \varepsilon n/2$ zurück.

Beweis

- Da der count eines Elements höchstens so groß ist wie die Anzahl seiner Vorkommen im Datenstrom, wird kein Element mit $F(x) < \varepsilon n/2$ zurückgegeben.
- Wir müssen noch zeigen, dass jedes $x \in U$ mit $F(x) > \varepsilon n$ zurückgegeben wird.
- Bei jeder Eliminierung eines Vorkommens von x werden auch $\lceil 2/\varepsilon \rceil$ andere Symbole eliminiert.
- Es können daher maximal $n/\lceil 2/\varepsilon \rceil \leq \varepsilon n/2$ Kopien eines Elements entfernt werden.
- Gilt also $F(x) > \varepsilon n$, so sind am Ende des Datenstroms noch mehr als $\varepsilon n/2$ Kopien übrig und das Element wird zurückgegeben.

Big Data

KSP-Algorithmus *(Karp, Shenker und Papadimitriou)*

1. Bezeichne $a[1] \ldots a[n]$ den Datenstrom
2. $K \leftarrow \emptyset$
3. **new** array count$[1..\lceil 2/\varepsilon \rceil]$ indiziert durch K
4. **for** $i \leftarrow 1$ **to** n **do**
5. **if** $a[i]$ ist in K **then** count$\left[a[i]\right] \leftarrow$ count$\left[a[i]\right] + 1$
6. **else** insert $a[i]$ in K; set count$\left[a[i]\right] \leftarrow 1$
7. **if** $|K| > \lceil 2/\varepsilon \rceil$ **then**
8. **for all** $x \in K$ **do**
9. count$[x] \leftarrow$ count$[x] - 1$;
10. **if** count$[x] = 0$ **then** delete x from K
11. **return** alle $x \in K$ mit count$[x] \geq \varepsilon n/2$

Frage: Wie wird K umgesetzt?

Big Data

KSP-Algorithm *(Papadimitriou)*

K wird als AVL-Baum realisiert.

1. Bezeichne
2. $K \leftarrow \emptyset$
3. **new** array count$[1..\lceil 2/\varepsilon \rceil]$ indiziert durch K
4. **for** $i \leftarrow 1$ **to** n **do**
5. **if** $a[i]$ ist in K **then** count$\left[a[i]\right] \leftarrow$ count$\left[a[i]\right] + 1$
6. **else** insert $a[i]$ in K; set count$\left[a[i]\right] \leftarrow 1$
7. **if** $|K| > \lceil 2/\varepsilon \rceil$ **then**
8. **for all** $x \in K$ **do**
9. count$[x] \leftarrow$ count$[x] - 1$;
10. **if** count$[x] = 0$ **then** delete x from K
11. **return** alle $x \in K$ mit count$[x] \geq \varepsilon n/2$

Big Data

KSP-Algorithm *(Papadimitriou)*

1. Bezeichne
2. $K \leftarrow \emptyset$
3. **new** array count$[1..\lceil 2/\varepsilon \rceil]$ indiziert durch K
4. **for** $i \leftarrow 1$ **to** n **do**
5. **if** $a[i]$ ist in K **then** count$[a[i]] \leftarrow$ count$[a[i]$
6. **else** insert $a[i]$ in K; set count$[a[i]] \leftarrow 1$
7. **if** $|K| > \lceil 2/\varepsilon \rceil$ **then**
8. **for all** $x \in K$ **do**
9. count$[x] \leftarrow$ count$[x] - 1$;
10. **if** count$[x] = 0$ **then** delete x from K
11. **return** alle $x \in K$ mit count$[x] \geq \varepsilon n/2$

K wird als AVL-Baum realisiert.

count ist unsortiertes Feld. Jeder Knoten des AVL-Baums speichert den zugehörigen Index des Arrays.

Big Data

KSP-Algorithm *(Papadimitriou)*

1. Bezeichne
2. $K \leftarrow \emptyset$
3. **new** array count$[1..\lceil 2/\varepsilon \rceil]$ indiziert durch K
4. **for** $i \leftarrow 1$ **to** n **do**
5. **if** $a[i]$ ist in K **then** count$\big[a[i]\big] \leftarrow$ count$\big[a[i]$
6. **else** insert $a[i]$ in K; set count$\big[a[i]\big] \leftarrow 1$
7. **if** $|K| > \lceil 2/\varepsilon \rceil$ **then**
8. **for all** $x \in K$ **do**
9. count$[x] \leftarrow$ count$[x] - 1$;
10. **if** count$[x] = 0$ **then** delete x from K
11. **return** alle $x \in K$ mit count$[x] \geq \varepsilon n/2$

K wird als AVL-Baum realisiert.

count ist unsortiertes Feld. Jeder Knoten des AVL-Baums speichert den zugehörigen Index des Arrays.

Wird ein Knoten aus dem Baum gelöscht, so wird der zugehörige Eintrag in count gelöscht.

Big Data

Satz 41

Die Laufzeit des KSP-Algorithmus ist $\mathbf{O}(n \log(1/\varepsilon))$.

Beweis:

- Zeile 5 und 6 benötigen $\mathbf{O}(\log(1/\varepsilon))$ Zeit
  (Einfügen in AVL-Baum und Suche in AVL-Baum)

Big Data

Satz 41

Die Laufzeit des KSP-Algorithmus ist $\mathbf{O}(n \log(1/\varepsilon))$.

Beweis:

- Zeile 5 und 6 benötigen $\mathbf{O}(\log(1/\varepsilon))$ Zeit
  (Einfügen in AVL-Baum und Suche in AVL-Baum)
- Zeile 8 wird $\mathbf{O}(1/\varepsilon)$ mal wiederholt

Big Data

Satz 41

Die Laufzeit des KSP-Algorithmus ist $\mathbf{O}(n \log(1/\varepsilon))$.

Beweis:

- Zeile 5 und 6 benötigen $\mathbf{O}(\log(1/\varepsilon))$ Zeit (Einfügen in AVL-Baum und Suche in AVL-Baum)
- Zeile 8 wird $\mathbf{O}(1/\varepsilon)$ mal wiederholt
- Zeile 10 benötigt $\mathbf{O}(\log(1/\varepsilon))$ Laufzeit

Big Data

Satz 41

Die Laufzeit des KSP-Algorithmus ist $\mathbf{O}(n \log(1/\varepsilon))$.

Beweis:

- Zeile 5 und 6 benötigen $\mathbf{O}(\log(1/\varepsilon))$ Zeit (Einfügen in AVL-Baum und Suche in AVL-Baum)
- Zeile 8 wird $\mathbf{O}(1/\varepsilon)$ mal wiederholt
- Zeile 10 benötigt $\mathbf{O}(\log(1/\varepsilon))$ Laufzeit
- Gesamtlaufzeit: $\mathbf{O}(n/\varepsilon \log(1/\varepsilon))$

Big Data

Satz 41

Die Laufzeit des KSP-Algorithmus ist $\mathbf{O}(n \log(1/\varepsilon))$.

Beweis:

- Zeile 5 und 6 benötigen $\mathbf{O}(\log(1/\varepsilon))$ Zeit (Einfügen in AVL-Baum und Suche in AVL-Baum)
- Zeile 8 wird $\mathbf{O}(1/\varepsilon)$ mal wiederholt
- Zeile 10 benötigt $\mathbf{O}(\log(1/\varepsilon))$ Laufzeit
- Gesamtlaufzeit: $\mathbf{O}(n/\varepsilon \log(1/\varepsilon))$

- Verbesserung: Die Laufzeit von Zeile 8 und 9 ist insgesamt $\mathbf{O}(n \log(1/\varepsilon))$, da jedes Element, das gelöscht wird, vorher eingefügt wurde und es insgesamt n Elemente gibt

Big Data

Satz 41

Die Laufzeit des KSP-Algorithmus ist $\mathbf{O}(n \log(1/\varepsilon))$.

Beweis:

- Zeile 5 und 6 benötigen $\mathbf{O}(\log(1/\varepsilon))$ Zeit (Einfügen in AVL-Baum und Suche in AVL-Baum)
- Zeile 8 wird $\mathbf{O}(1/\varepsilon)$ mal wiederholt
- Zeile 10 benötigt $\mathbf{O}(\log(1/\varepsilon))$ Laufzeit
- Gesamtlaufzeit: $\mathbf{O}(n/\varepsilon \log(1/\varepsilon))$

- Verbesserung: Die Laufzeit von Zeile 8 und 9 ist insgesamt $\mathbf{O}(n \log(1/\varepsilon))$, da jedes Element, das gelöscht wird, vorher eingefügt wurde und es insgesamt n Elemente gibt

Zusammenfassung

Datenströme

- Sehr große Datenmengen, die als Sequenz auftreten

Datenstrommodell

- Daten treten als Sequenz auf
- Wenig Speicher
- Die Sequenz kann nur einmal und nur in der vorgegebenen Reihenfolge gelesen werden

KSP-Algorithmus

- Findet „Heavy Hitter“