AlgoDat: 5.Hausaufgabe (17.05.23) - Cora Zeitler

$\Sigma: 13/32$

Mittwoch, 17. Mai 2023 03:23

Aufgabe 1:
Gegeben sind eine ungeordnete Folge von n Schlüsseln und eine natürliche Zahl k mit $1 \leq k \leq n$. Auszugeben sind diejenigen k Schlüssel, die am nächsten zum Median liegen und zwar sortiert nach dem Abstand zum Median.
Der naheliegende Algorithmus (sortieren und die gesuchten Elemente ausgeben) erfordert $O(n \log n)$ Zeit. Wir suchen nach einer schnelleren Lösung.

(a) Beschreiben Sie einen Algorithmus, der einen binären Heap als Datenstruktur nutzt. Analysieren Sie seine Laufzeit in Abhängigkeit von n und k.

(b) Entwerfen Sie einen Algorithmus, der eine Zeitschranke von $O(n + k \log k)$ erreicht.

Es kommt nicht darauf an, den Algorithmus in Pseudocode aufzuschreiben. Wichtiger ist, die Hauptschritte Ihres Algorithmus kurz und klar verbal zu beschreiben und jeweils eine Aussage über die Laufzeit zu treffen. (12 Punkte)

1a) Algorithmus:

1. leeren binären Heap (min-Heap) erstellen, der die Schlüssel enthält
2. ersten k Schlüssel der ungeordneten Folge in den Heap einfügen
3. für verbleibenden Schlüssel in der ungeordneten Folge gilt:
    -> Vergleichen Sie den aktuellen Schlüssel mit der Wurzel des Heaps (dem kleinsten Element).
    -> Wenn der aktuelle Schlüssel größer ist als die Wurzel des Heaps, entfernen Sie die Wurzel und fügen Sie den aktuellen Schlüssel in den Heap ein.
    -> Andernfalls fahren Sie mit dem nächsten Schlüssel fort, ohne den Heap zu ändern.
4. am Ende enthält der Heap die k Schlüssel, die dem Median am nächsten liegen f enthält die k größten Schlüssel

-> sind in aufsteigender Reihenfolge nach ihrem Abstand zum Median sortiert

Laufzeit: (hängt von n und k ab)

1. O(1) Zeit -> erstellen einen leeren Heap
2. O(k log k) Zeit -> Einfügen von k Schlüsseln hat in einen Heap logarithmische Zeitkomplexität
3. O((n-k) log k) Zeit -> überprüfen (n-k) Schlüssel und gegebenenfalls in Heap einfügen.
4. O(k log k) Zeit -> entfernen k Schlüssel in aufsteigender Reihenfolge aus Heap

Insgesamt beträgt die Laufzeit des Algorithmus O(k log k + (n-k) log k) = O(n log k). ✓

Oder anderer Algorithmus:

1. leeren Max-Heap mit k Elementenerstellen
2. ersten k Schlüssel der ungeordneten Folge in den Max-Heap einfügen
3. fürverbleibenden Schlüssel in der ungeordneten Folge gilt:
    -> aktuellen Schlüssel mit dem Wurzelelement des Max-Heaps vergleichen
    -> wenn aktuelle Schlüssel kleiner ist als Wurzelelement, dann Wurzelelement entfernen und aktuellen Schlüssel in den Max-Heap einfügen
    -> Andernfalls fahren Sie mit dem nächsten Schlüssel fort, ohne Max-Heap zu ändern
4. am Ende enthält der Max-Heap die k Schlüssel, die Median am nächsten liegen
    -> sind in absteigender Reihenfolge nach ihrem Abstand zum Median sortiert

4/12

Laufzeit:

1. O(1) Zeit -> erstellen von leeren Max-Heap
2. O(k) Zeit -> einfügen der ersten k Schlüssel in den Max-Heap
3. O((n-k) log k) Zeit -> überprüfen (n-k) Schlüssel und fügen gegebenenfalls in den Max-Heap ein
4. O(k log k) Zeit -> k Schlüssel in absteigender Reihenfolge aus dem Max-Heap zu entfernen

Insgesamt beträgt die Laufzeit des Algorithmus O((n-k) log k), da k ≤ n.

1b) Algorithmus:

1. Median der ungeordneten Folge von Schlüsseln finden (mit Median of Median) - in O(n) Zeit ✓
2. zwei leere Arrays, "kleiner" und "größer" erstellen, um Schlüssel zu speichern, die kleiner bzw. größer als der Median sind.
3. ungeordnete Folge der Schlüssel durchlaufen und die Arrays in "kleiner" und "größer" aufteilen (nach Verhältnis des Medians) - in O(n) Zeit
4. die Arrays "kleiner" und "größer" separat sortieren - in O(k log k) Zeit
5. ausgeben der ersten k/2 Elemente des sortierten "kleiner"-Arrays und der ersten k/2 Elemente des sortierten "größer"-Arrays

Es könnten theoretisch auch alle k Elemente rechts vom Median liegen

-> Elemente, die dem Median am nächsten liegen und sortiert nach ihrem Abstand zum Median - in O(k) Zeit

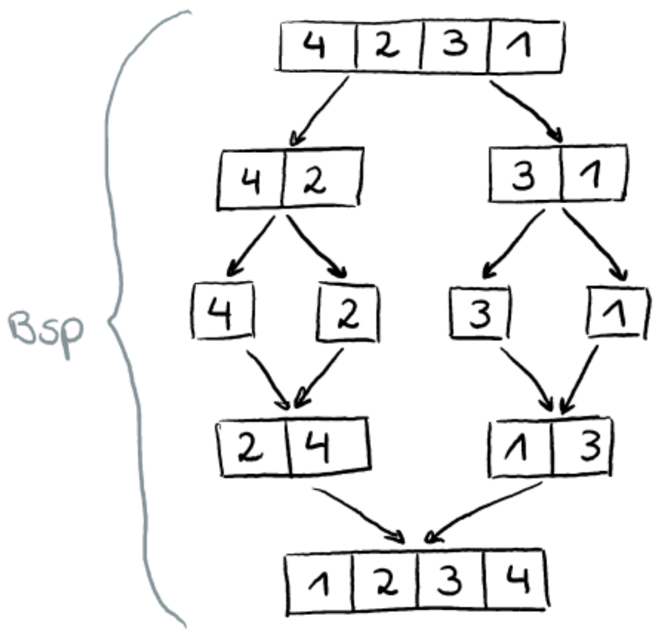
Laufzeit:

Insgesamt beträgt die Laufzeit des Algorithmus O(n + k log k)

-> Median wird effizient gefunden
-> werden nur k Schlüssel sortiert, die Median am nächsten liegen
-> aufteilen der Schlüssel in zwei separate Arrays wird vermieden

Aufgabe 2:
Erstellen Sie einen Entscheidungsbaum für MERGE-SORT auf einem Feld von 4 Elementen.
(5 Punkte)

2) Eingabe: Feld A = [4, 2, 3, 1]



→ Zeigt rekursive Aufteilung und das Zsmfügen der Elemente für ein Feld von 4 Elementen

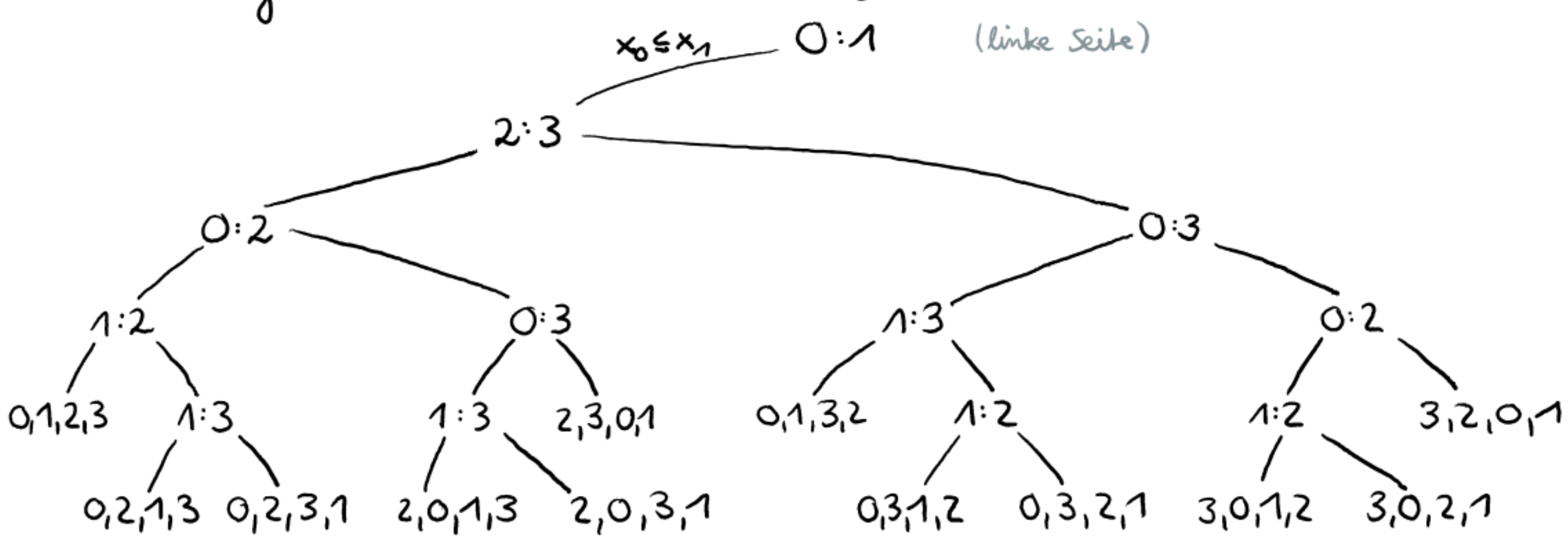
→ in jedem Schritt wird Feld in zwei Hälften aufgeteilt bis nur noch einzelne Elemente vorhanden sind

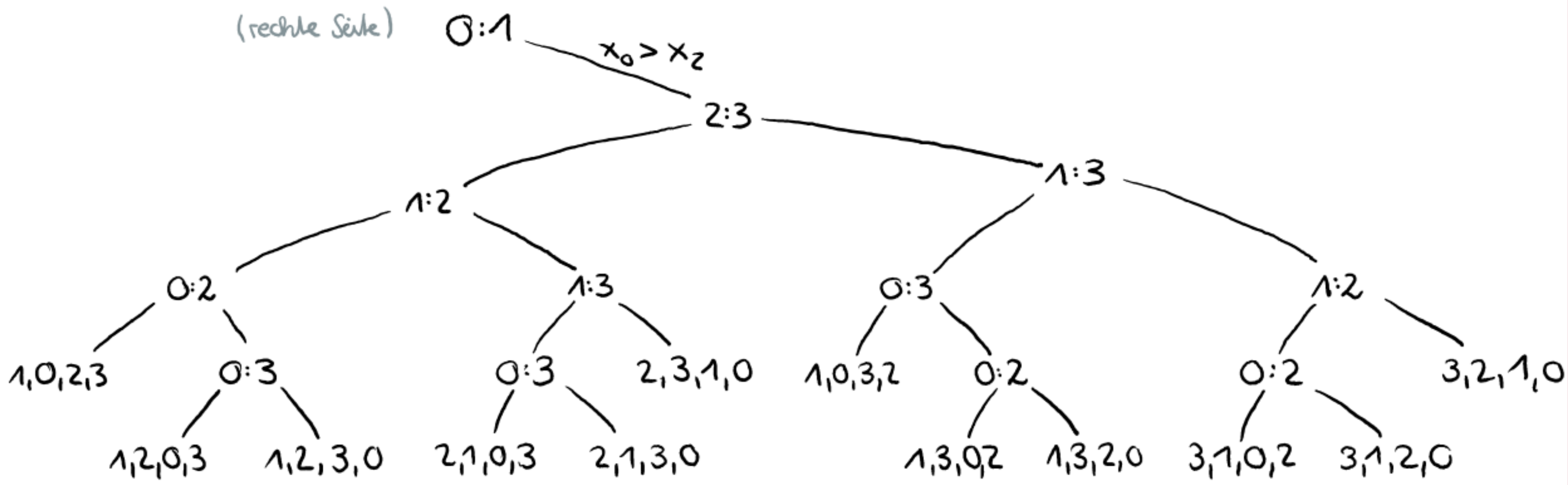
→ Einzelelemente werden zu sortierten Teilfelder schrittweise zsmgeführt bis zum sortierten ganzen Feld

→ Ausgabe: A = [1, 2, 3, 4]

5/5

Entscheidungsbaum: $A = [x_0, x_1, x_2, x_3]$





Aufgabe 3:
Ein Sortieralgorithmus heißt stabil, wenn Elemente mit gleichem Schlüssel in der sortierten Folge in der gleichen Reihenfolge auftauchen, wie in der Input-Folge.

(a) Begründen Sie, dass COUNTING-SORT stabil ist.

(b) Zeigen Sie durch vollständige Induktion, dass RADIX-SORT eine Folge von n d-stelligen Dezimalzahlen korrekt sortiert.

(5 Punkte)

3b) IA: $d = 1$

↳ Radix-Sort ist korrekt und stabil sortiert
(Weil Countingsort das tut)

4/5

IV: für d Ziffern ist Radix-Sort korrekt:

IB: dann ist Radix-Sort auch für d+1 korrekt

IS: Angenommen Radix-Sort ist korrekt nach d Ziffern der Zahl und dann werden die d+1 Ziffern mit den stabilen Sortieralgorithmus sortiert, dass Radix-Sort nutzt

→ Angenommen: [a bis zur d+1 Ziffer] < [b bis zur d+1 Ziffer]

→ dann gibt es 2 Fälle:

① [d+1 Ziffer von a] < [d+1 Ziffer von b]

→ a ist vor b, also noch korrekt?

Begründungen sehr schwammig

② [d+1 Ziffern von a] = [d+1 Ziffern von b]

→ Nach IV sind [d Ziffern von a] < [d Ziffern von b], also kommt a vor b, weil da d+1 Ziffer gleich sind ändert sich die Reihenfolge auch nicht

→ Angenommen: [a bis zur d+1 Ziffer] = [b bis zur d+1 Ziffer]

→ da alle Zahlen gleich sind, weil der Sortieralgorithmus stabil ist bleiben a und b in der gleichen Reihenfolge sind damit d+1 auch stabil

Aufgabe 4:
Gegeben seien n paarweise verschiedene Elemente $x_1, x_2, \ldots, x_n$ mit positiven Gewichten $w_1, w_2, \ldots, w_n$, $\sum_{i=1}^{n} w_i = 1$. Der gewichtete Median ist definiert als das Element x_k mit

$$\sum_{x_i < x_k} w_i < \frac{1}{2} \quad \text{und} \quad \sum_{x_i > x_k} w_i \leq \frac{1}{2}$$

(a) Bestimmen Sie für $(x_1, x_2, \ldots, x_{11}) = (52, 93, 10, 56, 100, 88, 7, 47, 19, 42, 33)$ mit den Gewichten $(w_1, w_2, \ldots, w_{11}) = (\frac{1}{20}, \frac{1}{20}, \frac{1}{5}, \frac{3}{20}, \frac{1}{20}, \frac{1}{10}, \frac{1}{10}, \frac{1}{20}, \frac{1}{20}, \frac{1}{10}, \frac{1}{10})$ den gewichteten Median.

(b) Zeigen Sie, dass der Median von $x_1, x_2, \ldots, x_n$ der gewichtete Median von $x_1, x_2, \ldots, x_n$ mit den Gewichten $w_i = 1/n$, $i = 1, \ldots, n$, ist.

(c) Wie kann man den gewichteten Median in $O(n \log n)$ Zeit unter Benutzung von Sortieren berechnen?

(10 Punkte)

4a) geg: 52 93 10 56 100 88 7 47 19 42 33

→ $\frac{1}{20}$ $\frac{1}{20}$ $\frac{1}{5}$ $\frac{3}{20}$ $\frac{1}{20}$ $\frac{1}{10}$ $\frac{1}{10}$ $\frac{1}{20}$ $\frac{1}{20}$ $\frac{1}{10}$ $\frac{1}{10}$

↳ $\frac{1}{20} + \frac{1}{20} + \frac{4}{20} + \frac{3}{20} = \frac{9}{20} < \frac{1}{2}$

$\frac{2}{20} + \frac{2}{20} + \frac{1}{20} + \frac{1}{20} + \frac{2}{20} + \frac{2}{20} = \frac{1}{2} \leq \frac{1}{2}$

$\Longrightarrow x_k = x_5 = 100$ f

4b) Dadurch das n die Anzahl der Elemente ist:

$x_k = \lceil \frac{n}{2} \rceil$, da $\sum \frac{1}{n} < \frac{1}{2} \iff \frac{1}{n} \cdot x_i < \frac{1}{2} \rightsquigarrow \frac{1}{n} \cdot x_i < \frac{1}{2} \quad | \cdot n$

4b) Dadurch das n die Anzahl der Elemente ist:

$x_k = \lceil \frac{n}{2} \rceil$, da $\sum_{x_i < x_k} \frac{1}{n} < \frac{1}{2} \overset{f}{\Longleftrightarrow} \underset{x_i = ?}{\frac{1}{n} \cdot x_i} < \frac{1}{2} \rightsquigarrow \frac{1}{n} \cdot x_i < \frac{1}{2} \quad | \cdot n$

$\underline{x_i < \frac{1}{2} n}$

und: $\sum_{x_i > x_k} \frac{1}{n} \leq \frac{1}{2} \Longleftrightarrow \frac{1}{n} \cdot x_i \leq \frac{1}{2} \Longleftrightarrow x_i \leq \frac{1}{2} \cdot n$

$\Longrightarrow x_i < \frac{n}{2} \leq x_k \leq x_i \leq \frac{1}{2} \cdot n$

4c) ?

0/10