Karlsruher Institut für Technologie

Institut für Theoretische Informatik
T.T.-Prof. Dr. Thomas Bläsius
https://scale.iti.kit.edu

Zusatzaufgaben 03

Algorithmen I – Sommersemester 2023

Gesamtpunkte: 35

Aufgabe 1 - To negative infinity and beyond! (3 Punkte)

In der Vorlesung wurde der ∞-Trick für (a,b)-Bäume vorgestellt. Beim $-\infty$-Trick hat das Dummy-Element den Wert $-\infty$. Wie unterscheidet sich ein (a, b)-Baum, bei dem der $-\infty$-Trick angewendet wird von einem, bei dem der ∞-Trick angewendet wird? Beachte insbesondere Veränderungen in den Operationen, die wir in der Vorlesung kennengelernt haben. (3 Punkte)

Aufgabe 2 - Pseudocode (8 Punkte)

```
FIRST(A: [ℕ; n], x: ℕ, y: ℕ, z: ℕ)
    if x < y then
        u: ℕ = SECOND(A, x, y, z)
        z_1: ℕ = x + (z mod (u − x))
        z_2: ℕ = u + 1 + (z mod (y − u − 1))
        FIRST(A, x, u − 1, z_1)
        if u < y then
            FIRST(A, u + 1, y, z_2)
        end
    end
```

```
SECOND(A: [ℕ; n], x: ℕ, y: ℕ, z: ℕ): ℕ
    u: ℕ = A[z]
    SWAP(A[y], A[z])
    i: ℕ = x
    j: ℕ = x
    while j < y do
        if A[j] ≤ u then
            SWAP(A[i], A[j])
            i := i + 1
        end
        j := j + 1
    end
    SWAP(A[i], A[y])
    return i
```

1. Seien $A = \langle 4, 8, 12, 34, 19, 26 \rangle$ und $n = 6$. Gib die den Zustand von A nach einem Aufruf von FIRST mit Eingabe $(A, 0, n-1, 1)$ an. (1 Punkt)

2. Was berechnet FIRST für eine beliebiges Array A, was berechnet SECOND? Haben wir diesen Algorithmus schon in der Vorlesung kennengelernt? Wie war sein Name? (2 Punkte)

3. Ändere den Pseudocode von FIRST bzw. SECOND so ab, dass er stabil wird, sich die Laufzeit aber nicht ändert. Du darfst dabei die Funktion CONCAT$(A1, A2)$ benutzen um zwei Arrays in Linearzeit zu konkatenieren. (3 Punkte)

4. Der Algorithmus FIRST ist in-place, das heißt er benötigt nur konstant viel zusätzlichen Speicherplatz. Wie viel zusätzlichen Speicherplatz braucht dein Algorithmus? Gib im O-Kalkül an. (1 Punkt)

5. Nenne einen Algorithmus der das gleiche tut wie FIRST, aber stabil und in-place ist. Was ist seine Laufzeit in O-Kalkül? (1 Punkt)

Aufgabe 3 - Min-Heaps (11 Punkte)

1. Folgende Arrays stellen Max-Heaps dar. Falls die Heap-Eigenschaft verletzt wurde, gib an wo, wenn nicht, zeichne die Baumrepräsentation des

Heaps. (2 Punkte)

a) $\langle 56, 47, 56, 10, 20, 50, 51, 1, 2, 3, 18\rangle$

b) $\langle 56, 56, 47, 10, 20, 50, 51, 1, 2, 3, 18\rangle$

c) $\langle 56, 47, 56, 10, 51, 20, 50, 1, 2, 3, 18\rangle$

d) $\langle 56, 47, 56, 10, 5, 20, 50, 1, 2, 3, 18\rangle$

2. Erstelle aus Array $A = \langle 7, 8, 1, 2, 6, 7, 5\rangle$ einen binären Min-Heap mit BUILD aus der Vorlesung. Gib dabei das Array an, sobald sich Einträge darin ändern. Du musst nur die vertauschten Zahlen angeben, bei allen anderen wird angenommen, dass sie gleich bleiben. (2 Punkte)

3. Füge in den Min-Heap$A = \langle 2, 7, 3, 9, 7, 5, 5\rangle$ mit Hilfe von PUSH zuerst ein Element mit Priorität 4 und anschließend ein Element mit Priorität 1 ein. Gib deinen Heap nach jeder Einfügeoperation an. (2 Punkte)

4. Lösche aus dem Min-Heap $A = \langle 2, 7, 3, 9, 7, 5, 5\rangle$ mit Hilfe von POPMIN zwei Mal das kleinste Element. Gib das Array nach jeder Operation an. (2 Punkte)

5. Man kann ein Array A sortieren, indem man mit BUILD einen Heap aus A ausbaut und mit POPMIN wiederholt das Minimum entfernt. Dieses Sortierverfahren heißt HEAPSORT und ist im Allgemeinen nicht stabil.

   a) Gib ein Beispiel an, bei dem die Stabilität eines Arrays verloren geht. (1 Punkt)

   b) Beschreibe, wie man ohne die Laufzeit oder Funktionsweise von HEAPSORT zu verändern Stabilität erzwingen kann. Du hast dazu $\mathcal{O}(n)$ zusätzlichen Speicher gegeben. (2 Punkte)

Aufgabe 4 - Stamm-Haltung (9 Punkte)

Auf dem ersten Zusatzblatt wurde beschrieben, dass Dr. Meta jeden Biberfreund und -feind sehr genau im Auge behält. Hierfür möchte er eine Datenbank D erstellt bekommen, welche es ihm ermöglicht alle gesammelten Informationen zu den ihm bekannten Bibern zu sammeln. Ein Eintrag in D soll hierbei ein k-Tupel sein, zum Beispiel (Name, Alter, Beruf, Sektion, verdächtig, ...). Eine Zeile in dieser Datenbank ist also ein k-Tupel und eine Spalte i stellt

den i-ten Eintrag von allen Tupeln dar. Die Anzahl der k-Tupel in D wird mit n bezeichnet.
Da Dr. Meta immer genauestens informiert sein möchte, greift er sehr häufig auf diese Datenbank zu, um sich Einträge nach einer bestimmten Spalte gefiltert anzusehen.

1. Beschreibe eine Methode FIND(`filter`, `zeile`), welche in $\mathcal{O}(\log(m) + j)$ alle k-Tupel ausgibt, die in der Zeile `zeile` den Eintrag `filter` haben. Dabei bezeichnet m die Anzahl verschiedener Einträge in dieser Zeile und j die Anzahl der k-Tupel mit dem Eintrag `filter`. Beschreibe zusätzlich hierfür eine Datenstruktur, auf welcher die Laufzeit von FIND gewährleistet werden kann. (5 Punkte)

2. Beschreibe in Abhängigkeit von n, m und j eine möglichst laufzeiteffiziente Methode INSERT(`tupel`), welche ein k-Tupel der Datenstruktur hinzufügt. Gib dessen Laufzeit so genau wie möglich an und begründe diese. (2 Punkte)

3. Begründe warum deine INSERT Methode keine bessere Laufzeit haben kann. Gehe hierbei auf die von dir gewählte Datenstruktur sowie Funktionsweise von FIND ein. (2 Punkte)

Aufgabe 5 - Kleinaufgaben (4 Punkte)

1. Ordne die folgenden Algorithmen absteigend nach ihrer Worst-Case Laufzeitkomplexität: Quicksort, Bucketsort, Mergesort (1 Punkt)

2. Wie lautet die untere Schranke für die Worst-Case Laufzeitkomplexität bei vergleichsbasiertem Sortieren? Gilt diese untere Schranke auch für Sortieralgorithmen für ganze Zahlen? Begründe. (1 Punkt)

3. Was macht ein stabiles Sortierverfahren aus? Nenne ein Beispiel für ein stabiles Sortierverfahren. (1 Punkt)

4. Nenne die Heap-Eigenschaft bezüglich eines Arrays $A[0, \ldots, n-1]$, welches implizit einen binären Heap darstellt. (1 Punkt)