Humboldt-Universität zu Berlin
Institut für Informatik
Prof. Dr. Ulf Leser
E. Angriman, F. Nelles, M. Sänger, Dr. P. Schäfer

Berlin, den 22.05.2019

Übungen zur Vorlesung
Algorithmen und Datenstrukturen

Übungsblatt 4

Abgabe: Mittwoch, den 05.06.2019, bis 11:10 Uhr vor der Vorlesung im Hörsaal. Die Übungsblätter sind in Gruppen von 2/3 Personen zu bearbeiten. Die Lösungen sind auf nach Aufgaben getrennten Blättern abzugeben. Heften Sie bitte die zu einer Aufgabe gehörenden Blätter vor der Abgabe zusammen. Vermerken Sie auf allen Abgaben Ihre Namen, Ihre **CMS-Benutzernamen**, Ihre **Abgabegruppe** (z.B. AG123) aus Moodle, und den **Übungstermin** (z.B. Gruppe 2 oder Mo 13 Uhr bei M. Sänger), an dem Sie Ihre korrigierten Blätter zurückerhalten möchten.
Beachten Sie die Informationen auf der Übungswebseite (https://hu.berlin/algodat19).

Konventionen:

- Mit log wird der Logarithmus $\log_2$ zur Basis 2 bezeichnet.
- Für ein Array A ist $|A|$ die Länge von A, also die Anzahl der Elemente in A. Die Indizierung aller Arrays auf diesem Blatt beginnt bei 1 (und endet also bei $|A|$).
- Mit der Aufforderung "Analysieren Sie die Laufzeit" ist gemeint, dass Sie eine möglichst gute obere Schranke der Zeitkomplexität angeben und diese begründen sollen.

Aufgabe 1 (Fibonacci-Suche) — 6 + 4 + 4 = 14 Punkte

1. Führen Sie einen Schreibtischtest für den Algorithmus *Fibonacci-Search* aus der VL durch, bei dem das folgende Array A nach dem Wert $c = 17$ durchsucht wird. Geben Sie die aktuellen Belegungen der Variablen $fib2$, $fib3$, und m vor jedem Aufruf von Zeile 8 im Pseudocode von Folie 13 an.

$A =$

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

2. Die Fibonacci-Zahlen sind wie folgt definiert: $F_1 = 1$, $F_2 = 1$ und $F_k = F_{k-2} + F_{k-1}$ für alle $k > 2$. Zeigen Sie unter Verwendung dieser Definition für Fibonacci-Zahlen, dass für $k \geq 1$ die Beziehung $F_k \geq \frac{1}{3} F_{k+2}$ gilt.

3. Benutzen Sie Teilaufgabe 2, um zu zeigen, dass die Fibonacci-Suche in einem sortierten Array mit n Elementen die Worst-Case Laufzeit $\mathcal{O}(\log n)$ hat.

Aufgabe 2 (Suche in sortierten Arrays) **6 + 8 = 14 Punkte**

Implementieren Sie die nachfolgenden zwei Suchverfahren für die Suche eines Werts k in einem aufsteigend sortierten Array A mit Werten vom Typ `Long`:

(a) *binäre Suche* und

(b) *Interpolationssuche*.

Gehen Sie für die Implementierung der Interpolationssuche analog zur Vorlesung von gleichverteilten Daten aus. Ergänzen Sie den fehlenden Code in der Vorlage `SortedSearch.java`, welche Sie auf der Übungswebseite[1] und Moodle vorfinden. Sie können beliebige neue Variablen und Hilfsmethoden zur Klasse hinzufügen, dürfen jedoch keine außer den von Java bereitgestellten Standard-Bibliotheken verwenden. Verwenden Sie für alle Vergleiche zweier Werte im Array A die vorgegebene Klasse `CountingComparator`. Mittels des Comperators wird die Anzahl der Vergleiche gezählt. Ein Beispiel für die Implementierung einer Suche in dem vorgegebenen Quellcode-Gerüst finden Sie in der Klasse `LinearSearch`. Stellen Sie sicher, dass alle Testfälle in der main-Methode der Datei SortedSearch.java erfolgreich durchlaufen. Achten Sie außerdem auf Randbedingungen und Spezialfälle, die von den Testfällen vielleicht nicht vollständig abgedeckt werden.

Hinweis: Ihr Java-Programm muss auf dem Rechner *gruenau2* laufen. Kommentieren Sie Ihren Code, sodass die einzelnen Schritte nachvollziehbar sind. Die Abgabe des von Ihnen modifizierten Quellcodes `SortedSearch.java` erfolgt über Moodle.

Aufgabe 3 (Suchen und Auswahl) **8 + 4 = 12 Punkte**

Angenommen, Sie haben ein *unendlich* langes Array A gegeben, d.h., die Länge des Arrays n kann nicht bestimmt werden. In A kommen nur die Buchstaben **a**, **b** und **c** vor. Die Elemente sind so geordnet, dass zuerst alle **a**s kommen, dann an Position p ein einzelnes **b**, und danach ab Position $p + 1$ nur noch **c**s. Sie können annehmen, dass Sie auf alle Positionen in konstanter Zeit zugreifen können.

1. Geben Sie einen Algorithmus in Pseudocode an, der in $\mathcal{O}(\log p)$ vielen Vergleichen die Position p bestimmt, an der das eine **b** steht.

2. Begründen Sie, warum die Anzahl der Vergleiche Ihres Algorithmus in $\mathcal{O}(\log p)$ liegt.

[1] https://hu.berlin/algodat19

Aufgabe 4 (Laufzeitanalyse) **2 + 2 + 3 + 3 = 10 Punkte**

Anton will seine Sammlung alter Liebesbriefe in einem (feuerfesten) Papierkorb verbrennen, der anfangs leer ist. Er tut dies mit Hilfe von zwei Operationen:

`rein`: Anton benötigt eine Sekunde, um einen Liebesbrief in den Papierkorb zu werfen. Es sei angenommen, dass der Papierkorb für eine beliebige Anzahl Liebesbriefe Platz hat.

`brand`: Anton setzt die Liebesbriefe im Papierkorb in Brand. In diesem Fall wartet Anton auf jeden Fall, bis alle Liebesbriefe im Papierkorb rückstandslos verbrannt sind. Liegen k Liebesbriefe im Papierkorb, so dauert dies genau k Sekunden.

a) Zeigen Sie zuerst, dass Anton im Worst Case $\mathcal{O}(n^2)$ Sekunden benötigt, um eine beliebige Folge von $n > 0$ Operationen `rein` und `brand` auszuführen.

Wir wollen nun nun mit Hilfe der Potentialmethode zeigen, dass Anton im Worst Case tatsächlich nur $\mathcal{O}(n)$ Sekunden benötigt, um eine beliebige Folge von $n > 0$ Operationen `rein` und `brand` auszuführen. Im Folgenden bezeichnen wir mit D_0 den leeren Anfangszustand des Papierkorbs und mit D_i den Zustand des Papierkorbs nach Ausführung der ersten i Operationen einer Folge Q von Operationen.

b) Beschreiben Sie zuerst eine geeignete Potentialfunktion, die jedem Zustand D des Papierkorbs ein Potential $\Phi(D)$ zuordnet.

c) Gegeben sei die Folge Q der Operationen {`rein`, `rein`, `rein`, `brand`, `rein`, `brand`} der Länge $n = 6$. Notieren Sie tabellarisch für jede Operation der Folge Q die folgenden drei Werte:

- die realen Kosten c_i (in Sekunden) der i-ten Operation von Q,
- das Potential $\Phi(D_i)$ des Papierkorbs nach Ausführung der i-ten Operation und
- die amortisierten Kosten $d_i \;=\; c_i + \Phi(D_i) - \Phi(D_{i-1})$ der i-ten Operation von Q.

d) Zeigen Sie mit Hilfe der Potentialmethode, dass die tatsächlichen Kosten zur Ausführung aller n Operationen der Folge Q in $\mathcal{O}(n)$ liegen, d.h., dass

$$\sum_{i=1}^{n} c_i \;\in\; \mathcal{O}(n)\,.$$

Hinweis: Zeigen Sie dafür zunächst, dass für $1 \leq i \leq n$ die amortisierten Kosten d_i der i-ten Operation aus Q höchstens 2 betragen.