Matthias Krause
Alexander Moch

Mannheim, den 05.10.2020

CS 307 Algorithmen und Datenstrukturen, Herbstsemester 2020

Lösungsskizze zum Übungsblatt 2

AUFGABE 2.1: Der folgende Algorithmus benutzt die Methode der binären Suche, um festzustellen, ob ein $x \in U$ in einem (aufsteigend) sortierten (Teil-)Array $A[l \cdots r]$ von Elementen aus U vorkommt.

algorithm *BINARY-SEARCH* (A, l, r, x)
1 **if** $(l > r)$ **return** *FALSE;*
2 $m := \lfloor (l + r)/2 \rfloor$;
3 **if** $(x = A[m])$ **return** *TRUE;*
4 **if** $(x > A[m])$ **return** *BINARY-SEARCH* $(A, m + 1, r, x)$;
5 **else** **return** *BINARY-SEARCH* $(A, l, m - 1, x)$;

a) Beweisen Sie die Korrektheit von *BINARY-SEARCH*.

Behauptung: *BINARY-SEARCH liefert TRUE, falls x in $A[l \cdots r]$ vorkommt; sonst liefert er FALSE.*
Beweis (*Induktion über* $(r - l)$) :
Induktionsanfang $(r - l \leq 0)$: *Offensichtlich liefert BINARY-SEARCH TRUE, wenn $l = r$ und $x = A[l]$; sonst liefert er FALSE.*
Induktionsschritt $(r - l > 0)$: *Sei* $m := \lfloor (l + r)/2 \rfloor$. *Wir unterscheiden drei Fälle:*

I) $(x = A[m])$: *BINARY-SEARCH liefert korrekterweise TRUE.*
II) $(x > A[m])$: *Da A sortiert ist, kommt x genau dann in $A[l \cdots r]$ vor, wenn x in $A[m + 1 \cdots r]$ vorkommt. Nach Induktionsvoraussetzung gilt die Behauptung.*
III) $(x < A[m])$: *Wie* (II).

b) Geben Sie die maximale Anzahl der Vergleiche bei der Ausführung von *BINARY-SEARCH* $(A, 1, n, x)$ in der O-Notation an. Bei welchen Eingaben tritt der worst case ein?

Wir bezeichnen die gesuchte Zahl mit $V(n)$. Offensichtlich gilt: $V(n) \leq V(\lfloor n/2 \rfloor) + c$ (für eine Konstante c). Die Anwendung des Master-Theorems (Fall (2)) liefert $V(n) \in O(\log n)$.
Der worst case tritt ein, wenn x nicht in $A[1 \cdots n]$ vorkommt.

c) Geben Sie eine nicht-rekursive Realisierung der binären Suche in Pseudocode an.

algorithm *ITERATIVE-BINARY-SEARCH* (A, l, r, x)
1 **while** $(l <= r)$
2 $m := \lfloor (l + r)/2 \rfloor$;
3 **if** $(x = A[m])$ **return** *TRUE;*
4 **if** $(x > A[m])$ $l := m + 1$;
5 **else** $r := m - 1$;
6 **return** *FALSE;*

AUFGABE 2.2: Gegeben sind ein Array $A[1 \cdot \cdot n]$ von Integers und ein Integer y. Zu entscheiden ist, ob es Indizes i und j, $1 \leq i \leq j \leq n$, gibt mit $y = A[i] + A[j]$. Beschreiben Sie, wie man dieses Problem in Zeit $O(n \log n)$ und Platz $\Theta(n)$ lösen kann.

Sortiere A (z.B.) mit Mergesort. Für $i = 1, \ldots, n$ entscheide mit Hilfe der binären Suche, ob $x := y - A[i]$ in A vorkommt.

AUFGABE 2.3: Schreiben Sie ein Pseudocodeprogramm für die Operation Merge(A, p, s, q). Hierbei ist A ein Feld mit Einträgen aus einer geordneten Menge $(M, \leq)$. Es gilt $1 \leq p \leq s \leq q \leq \text{length}(A)$. Merge$(A, p, s, q)$ sortiert, unter Nutzung eines Hilfsfeldes B, in Linearzeit $O(q - p)$ das Teilfeld $A[p], \ldots, A[q]$ unter der Bedingung, dass die Teilfelder $A[p], \ldots, A[s]$ und $A[s+1], \ldots, A[q]$ bereits sortiert sind.

MERGE(A, p, s, q)

1: $n_1 \leftarrow s - p + 1$
2: $n_2 \leftarrow q - s$
3: erzeuge die Felder $B_L[1 \ldots (n_1 + 1)]$ *und* $B_R[1 \ldots (n_2 + 1)]$
4: **for** $i \leftarrow 1$ **to** n_1
5: **do** $B_L[i] \leftarrow A[p + i - 1]$
6: **for** $j \leftarrow 1$ **to** n_2
7: **do** $B_R[j] \leftarrow A[s + j]$
8: $B_L[n_1 + 1] \leftarrow \infty$
9: $B_R[n_2 + 1] \leftarrow \infty$
10: $i \leftarrow 1$
11: $j \leftarrow 1$
12: **for** $k \leftarrow p$ **to** q
13: **do if** $B_L[i] \leq B_R[j]$
14: **then** $A[k] \leftarrow B_L[i]$
15: $i \leftarrow i + 1$
16: **else** $A[k] \leftarrow B_R[j]$
17: $j \leftarrow j + 1$

(vgl. Cormen, Leiserson, Rivest, Stein: Algorithmen - Eine Einführung)

AUFGABE 2.4: Man bestimme mit der Summations-Integrationsmethode eine möglichst genaue untere und obere Schranke für $\sum_{k=1}^{n} k^2 \ln(k)$.

Stammfunktion von $k^2 \ln(k)$ *ist*

$$\frac{k^3 \ln(k)}{3} - \frac{k^3}{9},$$

Damit können wir in die Formel einsetzen und integrieren:

$$\begin{aligned}
& \int_a^{b-1} f(x)dx && \leq \sum_{k=a}^{b-1} f(k) && \leq \int_a^{b} f(x)dx \\
\Rightarrow \quad & \int_1^{n} x^2 \ln(x)dx && \leq \sum_{k=1}^{n} k^2 \ln(k) && \leq \int_1^{n+1} x^2 \ln(x)dx \\
\Rightarrow \quad & \left[\frac{x^3 \ln(x)}{3} - \frac{x^3}{9}\right]_1^{n} && \leq \sum_{k=1}^{n} k^2 \ln(k) && \leq \left[\frac{x^3 \ln(x)}{3} - \frac{x^3}{9}\right]_1^{n+1} \\
\Rightarrow \quad & \frac{n^3 \ln(n)}{3} - \frac{n^3}{9} - 0 + \frac{1}{9} && \leq \sum_{k=1}^{n} k^2 \ln(k) && \leq \frac{(n+1)^3 \ln(n+1)}{3} - \frac{(n+1)^3}{9} - 0 + \frac{1}{9}
\end{aligned}$$

womit man die untere Schranke

$$\frac{n^3 \ln(n)}{3} - \frac{n^3}{9} + \frac{1}{9}$$

und die obere Schranke

$$\frac{(n+1)^3 \ln(n+1)}{3} - \frac{(n+1)^3}{9} + \frac{1}{9}$$

erhält.