



Aufgaben zu Satz A

Aufgabe

Betrachten Sie die Funktion

$$f: \mathbb{R} \rightarrow \mathbb{R} \\ x \mapsto f(x) := \left\lceil \frac{1}{4} \cdot 2^{\lfloor x \rfloor \bmod 6} \right\rceil$$

Schätzen Sie möglichst gut den Mindestaufwand ab, der nötig ist, um $f(x)$ auf einer *random access machine* zu berechnen.

Lösung

Den mindestens nötigen Aufwand erhalten wir, indem wir das *lower-bound*-Theorem (Satz A) aus der Vorlesung auf die Funktion

$$f: x \mapsto \left\lceil \frac{1}{4} \cdot 2^{\lfloor x \rfloor \bmod 6} \right\rceil$$

anwenden.

Aus wie vielen rationalen Funktionen Q_i setzt sich f zusammen? Dazu müssen wir uns die Beschaffenheit der Funktion genauer anschauen:

Zunächst fällt auf, daß man $f(x)$ auch als

$$f(x) = \tilde{f}(\lfloor x \rfloor \bmod 6)$$

schreiben kann, daß der Funktionswert $f(x)$ also nur vom Rest der Ganzzahl-Division $\lfloor x \rfloor / 6$ abhängt. Demnach sind überhaupt nur 6 verschiedene Funktionswerte für f möglich.

Der Term

$$\frac{1}{4} \cdot 2^{\lfloor x \rfloor \bmod 6}$$

kann für $\lfloor x \rfloor \bmod 6 \in \{0, 1, 2, 3, 4, 5\}$ nur die sechs diskreten Werte

$$\frac{1}{4}, \frac{2}{4}, \frac{4}{4}, \frac{8}{4}, \frac{16}{4}, \frac{32}{4}$$

annehmen. Die *ceiling*-Funktion $\lceil \dots \rceil$ verringert diese möglichen Werte noch einmal auf vier: $|\{1, 1, 1, 2, 4, 8\}| = 4$.

Es gibt also genau vier verschiedene rationale Funktionen

$$Q_1: x \mapsto 1 \qquad Q_3: x \mapsto 4 \\ Q_2: x \mapsto 2 \qquad Q_4: x \mapsto 8,$$

aus denen f aufgebaut werden kann.

Wie in **Satz A** kann man also $q = 4$ verschiedene Punkte

$$x_1 = 2,5 \quad x_3 = 4,5$$

$$x_2 = 3,5 \quad x_4 = 5,5$$

und beispielsweise $\varepsilon = 1/3$ angeben, sodaß gilt

$$\forall x \in U(x_i, \varepsilon) : Q_i(x) = f(x)$$

Gemäß dem *lower-bound*-Theorem gilt nun, daß **im schlechtesten Fall die Zahl R der Vergleiche zur Berechnung von f**

$$R \geq \log_2 q = 2$$

beträgt.

Ein RAM-Algorithmus, der f berechnet, kann also im *worst case*¹ nicht mit weniger als zwei Vergleichen auskommen.

Anmerkung: Da ein reales (!) RAM-Programm sicherlich zunächst $\lfloor x \rfloor$ berechnen müßte, würde dies schon einen gewaltigen Aufwand bedeuten, sodaß die untere Schranke von 2 Vergleichsoperationen, die wir eben ermittelt haben, wohl kaum erreichbar sein dürfte.

Aufgabe

Gegeben seien $2n$ Intervalle $I_i = [a_i, b_i] \subset \mathbb{R}$ und $J_i = [c_i, d_i] \subset \mathbb{R}$ mit $i = 1, \dots, n$.

Zeigen Sie: Das Problem, die Frage

$$\left(\bigcup_{i=1}^n I_i \right) \cap \left(\bigcup_{i=1}^n J_i \right) \stackrel{?}{=} \{\}$$

zu beantworten, hat als untere Zeitschranke $T_{\max} \in \Omega(n \log n)$.

Lösung

Was nicht zu zeigen ist: Es gibt eine Vielzahl von Algorithmen, um das Problem mit $O(n \log n)$ Schritten zu lösen.

- Man könnte die I_i nach Endpunkten b_i und die J_i nach Anfangspunkten c_i sortieren – Aufwand $O(n \log n)$ –, um dann die I_i in aufsteigender Reihenfolge zu durchlaufen und jeweils mit den bisher nicht betrachteten J_j , die bei Koordinaten $c_j \leq b_i$ beginnen, auf gemeinsame Punkte hin zu überprüfen: ist $d_j \geq a_i$?² Bei diesem letzten Verfahren wird jedes Intervall I_i und J_j genau einmal betrachtet – Aufwand $O(n)$.
- Man sortiert die $\{J_i \mid i = 1, \dots, n\}$ – Aufwand $O(n \log n)$ – und kann dann die I_i der Reihe nach durchgehen und mit einer Variante des binären Suchens – Aufwand n mal $\log n$ – irgendwie ein Intervall J_j finden, indem es evtl. gemeinsame Punkte gibt.
- ...

¹“gute” und “schlechte” Fälle unterscheiden sich hier in der Wahl des Parameter-Wertes für x

²Das Verfahren kann in Fällen versagen, wenn für zwei Intervalle I_i, I_k gilt $a_i > a_k$ und $b_i < b_k$
... (Skizze!)

Zu zeigen ist, daß $T_{\max} \in \Omega(n \log n)$ tatsächlich eine untere Schranke *des Problems* ist.

Die Aufgabenstellung lädt dazu ein, einen Reduktionsbeweis zu führen. Ähnlich wie in der Vorlesung (Kapitel 1.7), wo gezeigt wurde, wie die Berechnung von Voronoi-Gebieten dazu mißbraucht werden kann, Skalare zu sortieren, wollen wir einen Beweis durch Widerspruch führen: Wir wollen wie bei einem Widerspruchsbeweis zunächst annehmen, daß die gestellte Aufgabe mit weniger als $\Omega(n \log n)$ Schritten zu lösen sei. Wenn wir dann **ein bereits bekanntes Problem** der Größe n , dessen Lösung als untere Schranke bewiesenermaßen $\Omega(n \log n)$ Schritte benötigt, **auf die Frage** $(\bigcup_{i=1}^n I_i) \cap (\bigcup_{i=1}^n J_i) \stackrel{?}{=} \{\}$ **reduzieren** können – nicht umgekehrt! –, ist die Annahme offensichtlich ein Irrtum.

Als dieses “bereits bekannte Problem” wählen wir das Suchen von n Zahlen $\{z_1, z_2, \dots, z_n\}$ in einem Suchbaum von n Schlüsseln $\{s_1, s_2, \dots, s_n\}$. Der asymptotische Aufwand für dieses Problem kann in schlechten Fällen nicht weniger als

$$\underbrace{n}_{\text{die } z_i} \text{ mal je } \Omega(\log \underbrace{n}_{\text{die } s_i})$$

Operationen betragen, da der Baum die Höhe $\Omega(\log n)$ hat (vgl. Lower-Bound-Theorem). Die Reduktion gestaltet sich wie folgt:

$$I_i := [s_i, s_i]$$

$$J_i := [z_i, z_i]$$

Die Intervalle haben also alle die Länge Null. Die Frage, ob mindestens ein z_j unter den $\{s_i\}$ vorkommt, läßt sich nun reduzieren auf

$$\left(\bigcup_{i=1}^n I_i \right) \cap \left(\bigcup_{i=1}^n J_i \right) \stackrel{?}{=} \{\}$$

Die Lösung dieses “neuen” Problems kann also nicht mit weniger als $\Omega(n \log n)$ Schritten gelöst werden.

q. e. d.