Algorithmik Blatt 3 Teil 1

Mtr.-Nr. 6329857

Universität Hamburg — 6. November 2019

Aufgabe 6

Was macht der Algorithmus?

Der Wert des Zählers wird durch die Differenz zweier Teilzähler P und N gebildet. Wir haben bereits gesehen, dass ein einzelner Zähler in armotisierter worst-case Zeit $\mathcal{O}(1)$ inkrementiert werden kann. Das Hinzunehmen der Decrement Operation verschlechtert die armotisierte Laufzeit jedoch auf $\mathcal{O}(n)$, weil im worst-case eine Sequenz aus abwechselnen Increment und Decrement dazu führen, dass in jeder Operation alle n Bits gewechselt werden müssen.

Der neue Algorithmus umgeht das Problem, indem die Decrement Operation einfach als Increment auf einem zweiten Zähler agiert. Nun sind abwechselnde Increment und Decrement einfach nur Increments auf unterschiedlichen Zählern und die Argumentation für die amortisierte $\mathcal{O}(1)$ Laufzeit von Increment lässt sich auf {Increment, Decrement} übertragen.

Allerdings muss der Algorithmus einen Sonderheit beachten: Jedes Bit darf nur in einem der beiden Zähler auf 1 stehen. Das ist wichtig, um ungewollte mehrfache Repräsentation von Werten zu verhindern. Ansonsten stellten z.B. $(P = 00, N = 00)$, $(P = 01, N = 01)$, $(P = 10, N = 10)$ und $(P = 11, N = 11)$ alle den Gesamtwert 0 dar. Dadurch wäre bei gleicher Anzahl Bits der Wertebereich eingeschränkt. Immer wenn der Algorithmus ein Bit auf 1 setzen will, muss er zunächst prüfen, ob das Partnerbit im anderen Zähler schon auf 1 steht und falls ja, dieses dort stattdessen auf 0 setzen.

Accounting-Methode

Die relevanten Operationen der Increment und Decrement Funktion sind die Schreibzugriffe auf die jeweils $log(n)$ breiten Zähler P und N. Die Operation die ein Bit i auf 0 setzt nennen wir $\text{RESET}_P(i)$ bzw. $\text{RESET}_N(i)$. Die Operation die ein Bit i auf 1 setzt nennen wir $\text{SET}_P(i)$ bzw. $\text{SET}_N(i)$.

Die Laufzeit von Increment ist proportional zur Anzahl der ausgeführten $\text{RESET}_P(i)$ Operationen, denn diese kommt unbedingt genau einmal in der einzigen Schleife vor. Entsprechend ist die Laufzeit von Decrement ist proportional zur Anzahl der ausgeführten $\text{RESET}_N(i)$ Operationen.

Zu Beginn seien alle Bits auf 0 gesetzt. $\text{RESET}_P(i)$ wird nur durchgeführt, wenn Bit i auf 1 gesetzt ist. Dies kann nur der Fall sein, wenn vorher ein $\text{SET}_P(i)$ durchgeführt wurde. Analog gilt dies für $\text{RESET}_N(i)$.

Also gilt (wobei #(*Op*) die Anzahl der Ausführungen von *Op* ist):

$$\#(\text{RESET}_N) \leq \#(\text{SET}_N) \tag{1}$$

Wir können im Accounting die RESET Operationen also die Kosten für die SET Operationen übernehmen lassen. Somit veranschlagen wir:

$$\begin{aligned} T_{SET_P} &= 2 \\ T_{SET_N} &= 2 \\ T_{RESET_P} &= 0 \\ T_{RESET_N} &= 0 \end{aligned} \tag{2}$$

Für n ausführungen von INCREMENT ergibt sich eine gesamte Laufzeit von:

$$\begin{aligned} T_{INC}(p) &= p \cdot (T_{SET_P} + log(p) \cdot T_{RESET_P} + T_{RESET_N}) \\ &= p \cdot T_{SET_P} \end{aligned} \tag{3}$$

Für DECREMENT entsprechend

$$\begin{aligned} T_{DEC}(q) &= q \cdot (T_{SET_N} + log(q) \cdot T_{RESET_N} + T_{RESET_P}) \\ &\text{weil } T_{RESET_P} = T_{RESET_N} = 0 \\ T_{DEC}(q) &= q \cdot T_{SET_N} \end{aligned} \tag{4}$$

Für n ausgeführte INCREMENT und m ausgeführte INCREMENT ergibt sich:

$$\begin{aligned} T_{INC}(p) + T_{DEC}(q) &= p \cdot T_{SET_P} + q \cdot T_{SET_N} \\ T_{SET_P} &= T_{SET_N} = 2 \\ T_{INC}(p) + T_{DEC}(q) &= (p+q) \cdot 2 \end{aligned} \tag{5}$$

Da INCREMENT und DECREMENT die einzig erlaubten Operationen sind ergibt sich für jede beliebe Sequenz aus $n = (p+q)$ Operationen eine Laufzeit von $\mathcal{O}(n \cdot 2) = \mathcal{O}(n)$ und damit einer amortisierte worst-case Laufzeit von $\mathcal{O}(1)$.

Potential-Funktion

Die tatsächlichen Kosten c_i der INCREMENT Operation von Schritt i einer Sequenz von Operationen ergeben sich aus der Anzahl der Teiloperationen RESET_P, RESET_N und SET_P in Schritt i:

$$c_i = \#_i(\text{RESET}_P) + \#_i(\text{RESET}_N) + \#_i(\text{SET}_P)$$

Wobei $\#_i(Op)$ die Anzahl der ausgeführten Teiloperationen Op in Schritt i der Sequenz ist. Die amortisierten Kosten der i-ten Operation sind:

$$c_i' = c_i + \phi_i(P,N) - \phi_{i-1}(P,N) \tag{6}$$

Wählen wir als Potentialfunktion ϕ die Gesamtzahl der in P und N auf 1 gesetzten Bits, ergibt sich:

$$\phi_i(P,N) - \phi_{i-1}(P,N) = \#_i(\text{SET}_P) - \#_i(\text{RESET}_P) - \#_i(\text{RESET}_N) \tag{7}$$

Das Potential kann nie negativ sein, da die Anzahl der auf 1 gesetzten Bits nicht negativ sein kann.

Fügen wir die drei Gleichungne zusammen ergibt sich:

$$\begin{aligned} c_i' &= \#_i(\text{RESET}_P) + \#_i(\text{RESET}_N) + \#_i(\text{SET}_P) \\ &+ \#_i(\text{SET}_P) - \#_i(\text{RESET}_P) - \#_i(\text{RESET}_N) \\ &= 2 \cdot \#_i(\text{SET}_P) \end{aligned} \tag{8}$$

Da SET_P nur höchstens einmal pro INCREMENT vorkommt, ergibt sich $c_i' = 2 \le 2$ als amortisierte Kosten der INCREMENT Operation.

Analog funktioniert die Argumentation für Decrement:

$$\begin{aligned} c_i' &= c_i + \phi_i(P,N) - \phi_{i-1}(P,N) \\ c_i &= \#_i(\text{RESET}_N) + \#_i(\text{RESET}_P) + \#_i(\text{SET}_N) \\ \phi_i(P,N) - \phi_{i-1}(P,N) &= \#_i(\text{SET}_N) - \#_i(\text{RESET}_N) - \#_i(\text{RESET}_P) \\ c_i' &= \#_i(\text{RESET}_N) + \#_i(\text{RESET}_P) + \#_i(\text{SET}_N) \\ &+ \#_i(\text{SET}_N) - \#_i(\text{RESET}_N) - \#_i(\text{RESET}_P) \\ &= 2 \cdot \#_i(\text{SET}_N) \\ c_i' &= 2 \le 2 \end{aligned} \tag{9}$$

Für eine Sequenz aus n beliebigen Operationen aus der Menge $\{\text{INCREMENT}, \text{DECREMENT}\}$ ergibt sich $T(n) = 2 \cdot n$, also eine amortisierte worst-case Laufzeit von $\mathcal{O}(1)$.

Aufgabe 7

Wir wählen als Potenzialfunktion:

$$\phi(T) = \left| k \cdot T_n - T_g \right| \text{ für ein festes } k \ge 1 \tag{10}$$

Sie erfüllt die Forderung immer positiv zu sein auf Grund des Betrags-Operators.

Als nächstes bestimmen wir die laut ϕ amortisierte Laufzeit für INSERT. Dafür müssen wir 4 getrennt Fälle betrachten. In allen Fällen sei:

- n_i die Anzahl der Belegten Felder in Schritt i
- g_i die GröSSe des Array in Schritt i
- α_i der Belegungsfaktor $\frac{n_i}{g_i}$
- c_i die realen Kosten für Schritt i
- $n_i = n_{i-1} + 1$ gilt, weil INSERT ein Element einfügt.
- $c_i' = c_i + \phi(T_i) - \phi(T_{i-1})$

Fall 1: $\alpha_i < \frac{1}{k}$

Das Array wächst nicht $\Rightarrow g_i = g_{i-1} \land c_i = 1$

$$\begin{aligned}
\frac{n_i}{g_i} &= \alpha_i < \frac{1}{k} \\
k \cdot n_i &< g_i \\
k \cdot n_i &< g_i + k
\end{aligned} \tag{11}$$

$$\begin{aligned}
c_i' &= 1 + \left|k \cdot n_i - g_i\right| - \left|k \cdot n_{i-1} - g_{i-1}\right| \\
&= 1 + \left|k \cdot n_i - g_i\right| - \left|k \cdot n_{i-1} - g_{i-1}\right| \\
&= 1 + \left|k \cdot n_i - g_i\right| - \left|k \cdot (n_i - 1) - g_i\right| \\
&= 1 + \left|k \cdot n_i - g_i\right| - \left|k \cdot n_i - (g_i + k)\right| \\
&= 1 + g_i - k \cdot n_i - g_i + k + k \cdot n_i \\
&= 1 + k
\end{aligned} \tag{12}$$

Fall 2: $\alpha_{i-1} < \frac{1}{k} \land \alpha_i \geq \frac{1}{k}$

Das Array wächst nicht $\Rightarrow g_i = g_{i-1} \land c_i = 1$

$$\begin{aligned}
\frac{n_i}{g_i} &= \alpha_i \geq \frac{1}{k} \\
&\Rightarrow k \cdot n_i \geq g_i \\
&\Rightarrow k \geq \frac{g_i}{n_i} \\
\frac{n_{i-1}}{g_{i-1}} &= \alpha_{i-1} < \frac{1}{k} \\
&\Rightarrow k \cdot n_{i-1} < g_{i-1}
\end{aligned} \tag{13}$$

$$\begin{aligned}
c_i' &= 1 + \left|k \cdot n_i - g_i\right| - \left|k \cdot n_{i-1} - g_{i-1}\right| \\
&= 1 + k \cdot n_i - g_i - \left(g_{i-1} - k \cdot n_{i-1}\right) \\
&= 1 + k \cdot n_i - g_i - g_{i-1} + k \cdot n_{i-1} \\
&= 1 + 2 \cdot \left(k \cdot n_i - g_i\right) \\
&< 1 + 2 \cdot \left(\frac{g_i}{n_i} \cdot n_i - g_i\right) \\
c_i' &< 1
\end{aligned} \tag{14}$$

Fall 3: $\frac{1}{k} \leq \alpha_{i-1} < 1$

Das Array wächst nicht $\Rightarrow g_i = g_{i-1} \land c_i = 1$

$$\begin{aligned}
\frac{n_{i-1}}{g_{i-1}} &= \alpha_{i-1} < 1 \\
&\Rightarrow g_i = g_{i-1} < k \cdot n_{i-1}
\end{aligned} \tag{15}$$

$$
\begin{aligned}
c_i' &= 1 + \left|k \cdot n_i - g_i\right| - \left|k \cdot n_{i-1} - g_{i-1}\right| \\
&= 1 + k \cdot n_{i-1} + k - g_i - k \cdot n_{i-1} + g_{i-1} \\
&= 1 + k
\end{aligned} \tag{16}
$$

Fall 4: $\alpha_{i-1} = 1$

Das Array wächst $\Rightarrow g_i = 2 \cdot g_{i-1} \land c_i = n_i$

$$
\begin{aligned}
\frac{n_{i-1}}{g_{i-1}} &= 1 \\
\Rightarrow n_{i-1} &= g_{i-1} \\
\frac{n_i}{g_i} &> \frac{1}{k} \\
\Rightarrow k \cdot n_i &> g_i
\end{aligned} \tag{17}
$$

$$
\begin{aligned}
c_i' &= n_i + \left|k \cdot n_i - g_i\right| - \left|k \cdot n_{i-1} - g_{i-1}\right| \\
&= n_{i-1} + 1 + k + k \cdot n_{i-1} - 2g_{i-1} - k \cdot n_{i-1} + g_{i-1} \\
&= n_{i-1} + 1 + k + k \cdot n_{i-1} - 2n_{i-1} - k \cdot n_{i-1} + n_{i-1} \\
&= 1 + k
\end{aligned} \tag{18}
$$

Ergebnis

In allen 4 Fällen ist die amortisierte Laufzeit nur abhängig von dem konstanten Wachstumsfaktor $k \geq 1$. Wir können diesen also frei (aber fest, nicht in Abhängigkeit von n) wählen, ohne die amortisiert konstante worst-case Laufzeit der INSERT Operation zu verlieren.