Höhere Algorithmik
Mitschrift

Max Wisniewski

WS 2011/2012
21. Oktober 2011

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Ziel

Bisher haben wir einfache solcher Algorithmen betrachten (ALP1, ALP3, etc.). In dieser Vorlesung werden wir uns nun mit komplexeren Problemen beschäftigen. Wir wollen die Probleme unter folgenden Aspekten betrachten:

- Entwurf von Algorithmen
- Analyse dieser Algorithmen
- Bewertung dieser Algorithmen

1.2 Algorithmus

Def.: Ein Algorithmus ist ein endlich beschriebenes, effektives Verfahren, das eine Eingabe in eine Ausgabe überführt.

Zu Begin Betrachten wir ein einfaches Problem.

Kapitel 2

Repräsentant einer Menge

Gegeben sei folgendes statistisches Problem:
Es seien n Zahlen / Datensätze gegeben, wobei $n \ll 0$ gilt.
Gesucht ist ein Repräsentativer Wert für diese Menge.

2.1 Naiver Ansatz

Idee Wir verwenden den Durchschnitt / Mittelwert.
Die Laufzeit ist einfach, da wir nur einmal über alle Datensätze müssen. Setzen wir dabei eine konstante Zeit für Addition und Division vorraus, ist die Laufzeit $O(n)$.

Problem: Der Mittelwert ist Anfällig für Außreißer und daher nicht sehr aussagekräftig. Sind beispielsweise $n-1$ Werte zwischen 0 und 10 und ein $n$ter liegt bei 10.000.000 so wird das ganze Ergebnis zu diesem Wert hin verfälscht.

Dieser Repräsentant ist leicht zu berechnen, aber nicht sehr schön.
Betrachten wir daher einen anderen Ansatz.

2.2 K - SELECT

Def.: Ein Element s einer total geordneten Menge S hat den Rang k
$:\Leftrightarrow$ es gibt genau $(k-1)$ Elemente in S, die kleiner sind als s.

Man schreibt dafür $rg(s)$.

Def.: Sei S total geordnet mit $n = |\,S\,|$ und $s \in S$.

$$s \text{ heißt Median} :\Leftrightarrow rg(s) = \left\lceil \frac{n+1}{2} \right\rceil .$$

2.2.1 Das Problem

Gegeben Sei S, $|S| = n$ paarweise verschiedene Zahlen.
Nun wollen wir den Median s von S möglichst effizient finden.

2.2.2 Algorithmus I in $\Theta(n \cdot \log n)$

Was die Laufzeit schon nahe legt, bedienen wir uns hier eines Sortieralgorithmuses.

1. Soritere S. z. B. mit Heap - Sort .
   Benötigt $\Theta(n \cdot \log n)$ Schritte.

2. Gib das Element an der Stelle $\lceil \frac{n+1}{2} \rceil$ aus.
   Benötigt $\Theta(1)$ Schritte.

Laufzeit: $T(n) = \Theta(n \cdot \log n) + \Theta(1) = \Theta(n \cdot \log n)$.

Da für (vergleichsbasiertes) Sortieren jede Lösung mit $\Omega(n \cdot \log n)$ beschränkt ist, kann eine Lösung für das Medianproblem die Sortierung verwendet nicht schneller sein. Bleibt zu untersuchen, ob der Median ähnlich schwer ist, oder ob es einen Algorithmus gibt, der das Problem schneller lösen kann.

2.2.3 SELECT in O (n)

Angenommen es existiert eine Funktion SPLITTER(S), welche uns ein Element $q \in S$ liefert, so dass gilt:

$$rg(q) \geq \left\lfloor \frac{1}{4}\, n \right\rfloor \quad \wedge \quad rg(q) \leq \left\lceil \frac{3}{4}\, n \right\rceil .$$

Lemma: Angenommen wir können SPLITTER ohne weitere Kosten benutzen. Dann können wir den Median in $O(n)$ Zeit berechnen.

Beweis: Um diese Aussage zu beweisen lösen wir das allgemeinere Problem

$$\text{SELECT}(k, S)$$

finde Element mit Rang k. Dieses Problem wird Äuswahlproblem"genannt.

Idee: Nehme SPLITTER als PIVOT Element und teile die Menge der Daten daran auf.

Pseudocode:

```
SELECT( k , S )
  IF |S| < 100 THEN
    RETURN BRUTFORCE( k , S )  // z. B. Algorithmus I
  q ← SPLITTER( S )
  S_< ← { s ∈ S | s < q }
  S_> ← { s ∈ S | s > q}
  IF | S_< | ≥ k THEN
    RETURN  SELECT( k , S_< )
  ELSE IF |S_<| = k − 1 THEN
    RETURN q
  ELSE
    RETURN SELECT( k − |S_<| − 1 , S_>)
```

Laufzeitanalyse:

Da $rg(q) \in \left[\left\lfloor \frac{1}{4}\, n \right\rfloor , \left\lceil \frac{3}{4}\, n \right\rceil\right]$ gilt $|S_<|$, $|S_>| \leq \frac{3}{4}\, n$.

Also gilt:

$$T(n) \leq \begin{cases} O(1) & , n < 100 \\ O(n) + T\left(\frac{3}{4}\, n\right) & , \text{sonst} \end{cases}$$

Behauptung:

$$T(n) \in O(n)$$

Beweis:

$$\begin{aligned} T(n) &\leq c \cdot n + T\left(\tfrac{3}{4}\, n\right) \\ &\leq c \cdot n + c\left(\tfrac{3}{4}\, n\right) + T\left(\left(\tfrac{3}{4}\right)^2 n\right) \\ &\leq c \cdot n \sum_{i=0}^{\infty} \left(\tfrac{3}{4}^i\right) + O(1) \\ &\leq (4c) \cdot n + O(1) \\ &= O(n) \end{aligned}$$

□