

Algorithmen und Datenstrukturen

A7. Laufzeitanalyse: Bottom-Up-Mergesort

Marcel Lüthi and Gabriele Röger

Universität Basel

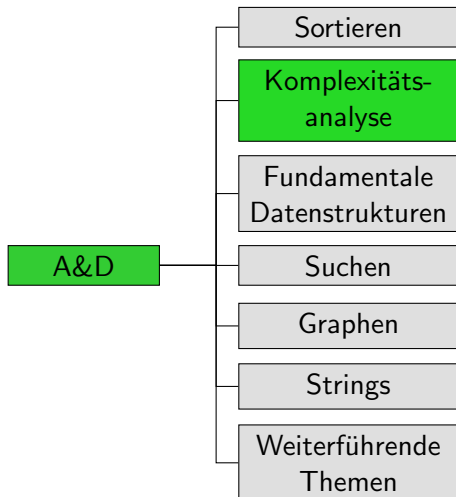
16. März 2022

A7.1 Laufzeitanalyse Bottom-Up-Mergesort

A7.2 Zusammenfassung

A7.1 Laufzeitanalyse Bottom-Up-Mergesort

Inhalt dieser Veranstaltung



Merge-Schritt

```
1 def merge(array, tmp, lo, mid, hi):
2     i = lo
3     j = mid + 1
4     for k in range(lo, hi + 1): # k = lo, ..., hi
5         if j > hi or (i <= mid and array[i] <= array[j]):
6             tmp[k] = array[i]
7             i += 1
8         else:
9             tmp[k] = array[j]
10            j += 1
11    for k in range(lo, hi + 1): # k = lo, ..., hi
12        array[k] = tmp[k]
```

Wir analysieren Laufzeit für $m := hi - lo + 1$

Merge-Schritt: Analyse

$$\begin{aligned}T(m) &= c_1 + c_2 m + c_3 m \\ &\geq (c_2 + c_3)m\end{aligned}$$

Für $m \geq 1$:

$$\begin{aligned}T(m) &= c_1 + c_2 m + c_3 m \\ &\leq c_1 m + c_2 m + c_3 m \\ &= (c_1 + c_2 + c_3)m\end{aligned}$$

Theorem

Der Merge-Schritt hat *lineare Laufzeit*, d.h. es gibt Konstanten $c, c', n_0 > 0$, so dass für alle $n \geq n_0$: $cn \leq T(n) \leq c'n$.

Bottom-Up-Mergesort

```
1 def sort(array):
2     n = len(array)
3     tmp = list(array)
4     length = 1
5     while length < n:
6         lo = 0
7         while lo < n - length:
8             mid = lo + length - 1
9             hi = min(lo + 2 * length - 1, n - 1)
10            merge(array, tmp, lo, mid, hi)
11            lo += 2 * length
12        length *= 2
```

Wir verwenden für die Abschätzung:

- c_1 Zeilen 2–4 **Annahme:** merge benötigt
- c_2 Zeilen 6 und 12 c_4 $(hi-lo+1)$ Operationen.
- c_3 Zeilen 8,9,11

Bottom-Up-Mergesort: Analyse I

Annahme: $n = 2^k$ für ein $k \in \mathbb{N}_{>0}$

Iterationen der äusseren Schleife (m für $hi-lo+1$):

- ▶ Iteration 1: $n/2$ mal innere Schleife mit Merge für $m = 2$
 $c_2 + n/2(c_3 + 2c_4) = c_2 + 0.5c_3n + c_4n$
- ▶ Iteration 2: $n/4$ mal innere Schleife mit Merge für $m = 4$
 $c_2 + n/4(c_3 + 4c_4) = c_2 + 0.25c_3n + c_4n$
- ▶ ...
- ▶ Äussere Schleife endet nach letzter Iteration ℓ .
- ▶ Iteration ℓ : 1 mal innere Schleife mit Merge für $m = n$
 $c_2 + n/n(c_3 + nc_4) = c_2 + c_3 + c_4n$

Insgesamt $T(n) \leq c_1 + \ell(c_2 + c_3n + c_4n) \leq \ell(c_1 + c_2 + c_3 + c_4)n$

Bottom-Up-Mergesort: Analyse II

Wie gross ist ℓ ?

- ▶ In Iteration i ist für den Merge-Schritt $m = 2^i$
- ▶ In Iteration ℓ hat Merge-Schritt $m = 2^\ell = n$
- ▶ Da $n = 2^k$ ist $\ell = k = \log_2 n$.

Mit $c := c_1 + c_2 + c_3 + c_4$ erhalten wir $T(n) \leq cn \log_2 n$.

Bottom-Up-Mergesort: Analyse III

Was, wenn n keine Zweierpotenz, also $2^{k-1} < n < 2^k$?

- ▶ Trotzdem k Iterationen der äusseren Schleife.
- ▶ Innere Schleife verwendet nicht mehr Operationen.
- ▶ $T(n) \leq cnk = cn(\lfloor \log_2 n \rfloor + 1) \leq 2cn \log_2 n$ (für $k > 2$)

Bottom-Up-Mergesort: Analyse IV

Ähnliche Abschätzung auch für untere Schranke möglich.

→ Übung

Theorem

*Bottom-Up-Mergesort hat **leicht überlineare Laufzeit**, d.h. es gibt Konstanten $c, c', n_0 > 0$, so dass für alle $n \geq n_0$ gilt $cn \log_2 n \leq T(n) \leq c'n \log_2 n$.*

Leicht überlineare Laufzeit

Leicht überlineare Laufzeit $n \log_2 n$:

→ doppelt so grosse Eingabe, etwas mehr als doppelt so lange Laufzeit

Was bedeutet das in der Praxis?

- ▶ Annahme: $c = 1$, eine Operation dauert im Schnitt 10^{-8} Sek.
- ▶ Bei 1 Tsd. Elementen warten wir $10^{-8} \cdot 10^3 \log_2(10^3) \approx 0.0001$ Sekunden.
- ▶ Bei 10 Tsd. Elementen ≈ 0.0013 Sekunden
- ▶ Bei 100 Tsd. Elementen ≈ 0.017 Sekunden
- ▶ Bei 1 Mio. Elementen ≈ 0.2 Sekunden
- ▶ Bei 1 Mrd. Elementen ≈ 299 Sekunden

Laufzeit $n \log_2 n$ nicht viel schlechter als lineare Laufzeit

Mergesort mit Kostenmodell I

Schlüsselvergleiche

- ▶ Werden nur in `merge` durchgeführt.
 - ▶ Mergen zweier Teilfolgen der Länge m und n benötigt bestenfalls $\min(n, m)$ und schlimmstenfalls $n + m - 1$ Vergleiche.
 - ▶ Bei zwei etwa gleich langen Teilfolgen sind das **linear** viele Vergleiche, d.h. es gibt $c, c' > 0$, so dass Anzahl Vergleiche zwischen cn und $c'n$ liegt.
- Anzahl der zum Sortieren einer Sequenz notwendigen Schlüsselvergleiche ist **leicht überlinear** in der Länge der Sequenz (analog zu Laufzeitanalyse).

Mergesort mit Kostenmodell II

Elementbewegungen

- ▶ Werden nur in `merge` durchgeführt.
- ▶ $2n$ Bewegungen für Sequenz der Länge n .
- ▶ Insgesamt für Mergesort **leicht überlinear** (analog zu Schlüsselvergleichen)

A7.2 Zusammenfassung

Zusammenfassung

- ▶ Mergesort hat leicht überlineare Laufzeit, Schlüsselvergleiche und Elementbewegungen.