Datenstrukturen, Algorithmen und Programmierung 2

Amin Coja-Oghlan

June 10, 2022

Lehrstuhl Informatik 2
Fakultät für Informatik

Randomisiertes Quicksort

Ziel

Eine Liste $L = (\ell_1, \ldots, \ell_n)$ vergleichbarer Elemente aufsteigend sortieren.

Vergleichbarkeit

- für je zwei Element ℓ_i, ℓ_j gilt entweder $\ell_i < \ell_j$, $\ell_i = \ell_j$ oder $\ell_i > \ell_j$
- die Ordnung ist transitiv: $\ell_h \le \ell_i$ und $\ell_i \le \ell_j \Rightarrow \ell_h \le \ell_j$
- die Ordnung ist antisymmetrisch: $\ell_i \le \ell_j$ und $\ell_j \le \ell_i \Rightarrow \ell_i = \ell_j$
- wir haben Zugriff auf eine Funktion, die zwei Element ℓ_i, ℓ_j vergleicht

Randomisiertes Quicksort

Algorithmus `Quicksort`

1. Für $i = 1, \ldots, n$
2. falls $\ell_i < \ell_1$, füge ℓ_i der Liste K hinzu.
3. falls $\ell_i > \ell_1$, füge ℓ_i der Liste G hinzu.
4. falls $\ell_i = \ell_1$, füge ℓ_i der Liste M hinzu.
5. Wende `Quicksort` rekursiv an, um K und G zu sortieren.
6. Gib K, M, G aus.

Randomisiertes Quicksort

Laufzeit nochmal

- auf einer *sortierten* Eingabe hat Quicksort Laufzeit $\Theta(n^2)$
- dennoch ist Quicksort “in der Praxis” beliebt
- um das “reale” Verhalten von Quicksort besser zu verstehen, analysieren wir den Algorithmus auf *zufälligen* Permutationen

Randomisiertes Quicksort

Permutationen

- eine n-Permutation ist eine bijektive Abbildung

$$\{1,\ldots,n\} \to \{1,\ldots,n\}$$

- wir schreiben $[n] = \{1,\ldots,n\}$
- die Menge aller n-Permutationen wird mit $\mathbb{S}_n$ bezeichnet
- insgeamt gibt es

$$|\mathbb{S}_n| = n! = \prod_{i=1}^{n} i$$

n-Permutationen

Randomisiertes Quicksort

Permutationen

- mit $\boldsymbol{\sigma} \in \mathbb{S}_n$ wird eine zufällige Permutation bezeichnet
- wir interessieren uns für die *erwartete* oder *durchschnittliche* Laufzeit
- dazu führen wir die n-te harmonische Zahl ein:

$$H_n = \sum_{i=1}^{n} \frac{1}{i}$$

- wir erinnern uns auch an die Definition des natürlichen Logarithmus':

$$\log(x) = \int_1^x \frac{1}{z} \mathrm{d}z$$

Randomisiertes Quicksort

Satz

Angewandt auf die zufällige Permutation $\boldsymbol{\sigma}$ hat Quicksort eine erwartete Laufzeit von $\leq 2(n+1)H_n$ Vergleichen.

Randomisiertes Quicksort

Die Euler-Mascheroni-Zahl

Ein wenig Analysis zeigt, daß

$$\lim_{n\to\infty} H_n - \log n = \gamma \approx 0.57721\ldots$$

Aus dem Satz folgt also, daß Quicksort auf Eingabe $\boldsymbol{\sigma}$ nur $O(n \log n)$ Vergleiche in Erwartung benötigt

Randomisiertes Quicksort

Zufällige Wahl des Pivots

- die Analyse von Quicksort auf zufälligen Permutationen hat *nur* verwendet, daß das Pivot zufällig ist
- wenn wir statt des ersten Elements also ein zufälliges Element als Pivot verwenden, erzielen wir auf *jeder* Eingabe eine Laufzeit von $O(n \log n)$
- man spricht von einem Las Vegas-Algorithmus

Randomisiertes Quicksort

Zusammenfassung

- wir haben Quicksort auf *zufälligen* Permutationen analysiert
- die Laufzeit hat sich dabei auf $O(n \log n)$ (erwartet) verbessert
- die Analyse zeigt, daß eine *randomisierte* Version dieselbe erwartete Laufzeit auf *beliebigen* Eingaben hat