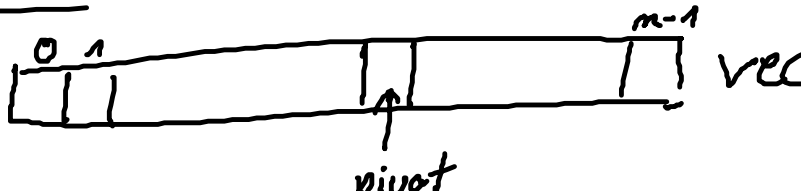


Aussagen: • Ab morgen liegen die Listen zum Eintragen für Rückgesprächen im Sekretariat MA 501.

- Unbedingt Uhrzeit und Terminnummer merken!
 - Zugehörige Räume werden nächste Woche auf der Homepage bekanntgegeben.
 - Nächsten Mittwoch (6.02.) Colla-Undruck im A-Café.
-

QuickSort:

• beg.  vec

• Zerlege vec in Teilarrays s. d.

- $vec[i].key \leq vec[pivot].key$ für $i = 0, \dots, k-1$
- $vec[k].key = vec[pivot].key$
- $vec[j].key > vec[pivot].key$ für $j = k+1, \dots, m-1$

• Wende QuickSort rekursiv auf Teilarrays an, sofern diese mind. 2 Komponenten aufweisen.

Beob: Vorteil gegenüber MergeSort: Es müssen keine zusätzlichen Kopien von Teilarrays angelegt werden.

Haben gesehen: WorstCase Laufzeit von $\Omega(n^2)$ ← schlechter als MergeSort

Aber: Empirische Ergebnisse zeigen: QuickSort ist in der Praxis am schnellsten! ✓

Grund: Im Mittel ist QuickSort viel schneller als im Worst Case.

Im Folgenden geben wir eine theoretische Analyse des mittleren Aufwands von QuickSort:

Dabei sei Π die Menge aller Permutationen der Zahlen $1, 2, \dots, n$.
($|\Pi| = n!$)

Für jedes $\pi \in \Pi$ ermitteln wir # Vergleiche $C(\pi)$, die QuickSort benötigt.
Dann bilden wir den Durchschnitt $\frac{1}{n!} \sum_{\pi \in \Pi} C(\pi)$

Annahme: Jede Permutation tritt mit gleicher Wahrscheinlichkeit $\frac{1}{n!}$ auf.
("gleichverteilt").

$\bar{C}(n) :=$ mittlere Anzahl von Vergleichen zum Sortieren eines Arrays der Länge n (gemittelt über alle Permutationen)
 $= \frac{1}{n!} \sum_{\pi \in \Pi} C(\pi)$ "erwartete # Vergleiche"

Wir werden zeigen: $\bar{C}(n) \in O(n \log n)$

Dazu zerlegen wir Π in n Klassen:

$\Pi_k := \{ \pi \in \Pi \mid \text{erstes Pivotelement unseres QuickSort-Alg. hat Wert } k \}$

Bsp: $m=3$ $\Pi = \{123, 132, 213, 231, 312, 321\}$

$\uparrow \quad \uparrow \quad \uparrow \quad \uparrow \quad \uparrow \quad \uparrow$

$$\Pi_1 = \{213, 312\}$$

$$\Pi_2 = \{123, 321\}$$

$$\Pi_3 = \{132, 231\}$$

beachte: die Π_k sind paarweise disjunkt

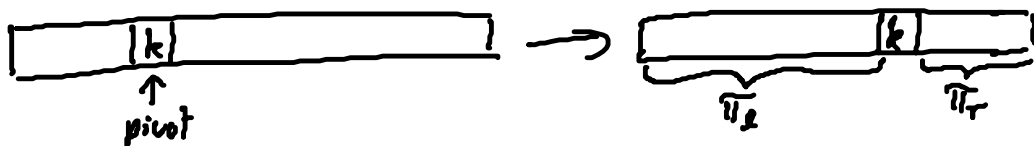
$$\Rightarrow \Pi = \bigcup_{k=1}^m \Pi_k$$

$$|\Pi_k| = (m-1)!$$

QuickSort macht als erstes die Aufteilung in 2 Teilarrays Π_L, Π_R .

Dabei gilt:

$\forall \pi \in \Pi_k$ besteht π_L aus einer Permutation der Zahlen $1, \dots, k-1$
 und π_R aus einer Permutation der Zahlen $k+1, \dots, m$.



$$\Rightarrow \bar{C}(m) = \frac{1}{m!} \cdot \sum_{k=1}^m \sum_{\pi \in \Pi_k} C(\pi)$$

$$\sum_{\pi \in \Pi_k} C(\pi) = \sum_{\pi \in \Pi_k} z(\pi) + \sum_{\pi \in \Pi_k} C(\pi_L) + \sum_{\pi \in \Pi_k} C(\pi_R)$$

$\leftarrow S_1 \qquad \qquad \qquad \leftarrow S_2 \qquad \qquad \qquad \leftarrow S_3$

wobei $z(\pi) = \#$ Vergleiche zum Zerlegen von π in π_L, k, π_R

$\#$ Vergleiche für π_L

$\#$ Vergleiche für π_R

$$S_1 \stackrel{\text{Lemma}}{\leq} \sum_{\pi \in \Pi_k} m = |\Pi_k| \cdot m = (m-1)! \cdot m = m!$$

$$S_2 := \sum_{\pi \in \Pi_k} C(\pi_k)$$

π_k entsteht aus π bei der Zerlegung im ersten Schritt von Gleichheit.

Beh: In dieser Summe taucht jede Permutation π_k von $1, \dots, k-1$ gleich oft auf.

Begründung: Bei der Zerlegung des Arrays in $\boxed{\pi_k} \mid k \mid \pi_k$ werden nur Vergleiche mit dem Pivotelement angestellt. Der genaue Wert der Elemente $\leq k-1$ spielt dabei keinerlei Rolle. Daher tauchen alle möglichen Permutationen dieser Elemente aus Symmetriegründen gleich oft auf, nämlich genau $\frac{(n-1)!}{(k-1)!}$ mal.

$$\Rightarrow S_2 = \sum_{\pi \in \Pi_k} C(\pi_k) = \frac{(n-1)!}{(k-1)!} \cdot \sum_{\substack{\text{Permutation} \\ \pi_k \text{ von } 1, \dots, k-1}} C(\pi_k) = (n-1)! \cdot \bar{C}(k-1)$$

Analog dazu kann man zeigen, dass

$$S_3 = \sum_{\pi \in \Pi_k} C(\pi_r) = \frac{(n-1)!}{(n-k)!} \sum_{\substack{\text{Perm. Typ} \\ \text{von } k+1, \dots, n}} C(\pi_r) = (n-1)! \cdot \bar{C}(n-k)$$

Damit haben wir erhalten:

$$\begin{aligned} \bar{C}(n) &= \frac{1}{n!} \sum_{k=1}^n (S_1 + S_2 + S_3) \\ &\leq \frac{1}{n!} \cdot \sum_{k=1}^n (n! + (n-1)! \bar{C}(k-1) + (n-1)! \bar{C}(n-k)) \\ &= n + \frac{1}{n} \sum_{k=1}^n \bar{C}(k-1) + \frac{1}{n} \cdot \sum_{k=1}^n \bar{C}(n-k) \\ &= n + \frac{2}{n} \sum_{k=1}^n \bar{C}(k-1) \end{aligned}$$

$$= n + \frac{2}{n} \sum_{k=2}^{n-1} \bar{C}(k)$$

$$\text{da } \bar{C}(0) = \bar{C}(1) = 0$$

$$\boxed{\bar{C}(n) \leq n + \frac{2}{n} \cdot \sum_{k=2}^{n-1} \bar{C}(k)}$$

Rekursionsungleichung

Beh: $\bar{C}(n) \in O(n \log n)$

$$\text{spracher: } \bar{C}(n) \leq 2 \cdot n \cdot \underset{\substack{\uparrow \\ c}}{\ln(n)}$$

Beweis: Vollst. Induktion nach n für allg. c .

$$\text{Ind.-Auf. : } n=2 \quad \bar{C}(2) = 1$$

$$c \cdot 2 \cdot \ln(2) = c \cdot 1.39 \geq 1 \quad \text{für } c \geq 1$$

Ind.-Ann: Beh. gilt für alle Werte $1, \dots, n-1$ für ein festes $n \in \mathbb{N}$.

Ind.-Schluss: Zeigen Beh. für n :

$$\bar{C}(n) \stackrel{\text{Rek. Formel}}{\leq} n + \frac{2}{n} \cdot \sum_{k=2}^{n-1} \bar{C}(k)$$

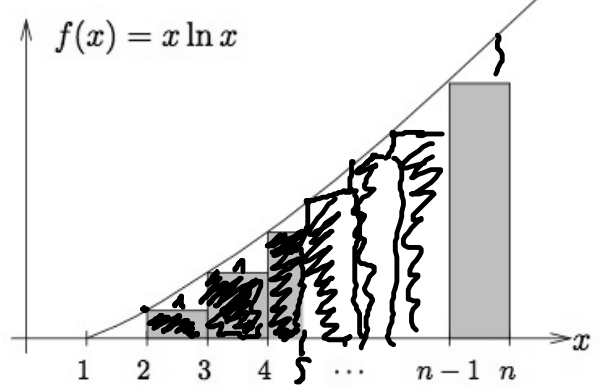
$$\stackrel{\text{i. Ann.}}{\leq} n + \frac{2}{n} \cdot \sum_{k=2}^{n-1} c \cdot k \cdot \ln(k)$$

$$= n + \frac{2 \cdot c}{n} \cdot \sum_{k=2}^{n-1} k \cdot \ln(k)$$

Betrachte die Funktion $f(x) = x \cdot \ln(x)$ ($f: \mathbb{R}_{>0} \rightarrow \mathbb{R}$)

Dann ist

$$\sum_{k=2}^{m-1} k \cdot \ln(k) \leq \int_2^m f(x) dx$$



$$\int_2^m f(x) dx = \int_2^m x \ln(x) dx$$

$$\underline{\text{part. Int.}} \quad \left. \frac{x^2}{2} \cdot \ln(x) \right|_2^m - \int_2^m \frac{x^2}{2} \cdot \frac{1}{x} dx$$

$$= \frac{m^2}{2} \cdot \ln(m) - 2 \cdot \ln(2) - \left(\frac{m^2}{4} - 1 \right)$$

$$= \frac{m^2}{2} \cdot \ln(m) - \frac{m^2}{4} + \underbrace{1 - 2 \cdot \ln(2)}_{< 0}$$

$$\leq \frac{m^2}{2} \cdot \ln(m) - \frac{m^2}{4}$$

$$\Rightarrow \bar{C}(n) \leq n + \frac{2c}{n} \cdot \left(\frac{m^2}{2} \cdot \ln(m) - \frac{m^2}{4} \right)$$

$$= n + c \cdot n \cdot \ln(m) - \frac{c}{2} n$$

$$\leq c \cdot n \cdot \ln(m) \quad \text{für } c \geq 2$$

$$\text{da } \ln(m) = \frac{\log m}{\log 2} :$$

Wir haben also gezeigt, dass die mittlere ~~W~~ Vergleichs-
für QuickSort $\bar{C}(n) \in O(n \log n)$