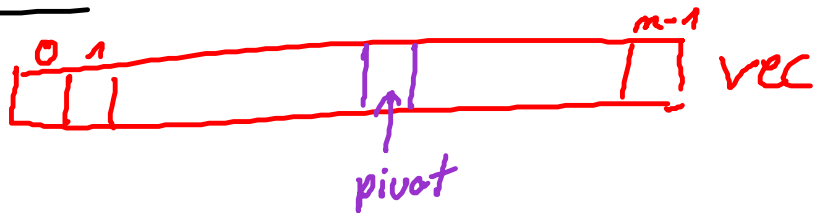
Ansagen:
- Ab morgen liegen die Listen zum Eintragen für Rücksprachen im Sekretariat MA 501.
- Unbedingt Uhrzeit und Terminnummer merken!
- Zugehörige Räume werden nächste Woche auf der Homepage bekanntgegeben.
- Nächsten Mittwoch (6.02.) CoMa-Umtrunk im A-Café.

QuickSort:

- Geg.



- Zerlege vec in Teilarrays s.d.
  - vec[i].key ≤ vec[pivot].key für $i = 0, \ldots, k-1$
  - vec[k].key = vec[pivot].key
  - vec[j].key > vec[pivot].key für $j = k+1, \ldots, n-1$
- 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:

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

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:

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

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

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

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

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

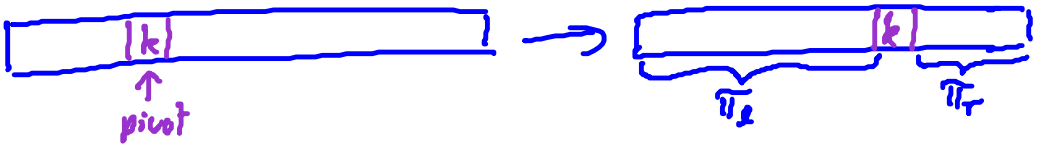
beachte: die Π_k sind paarweise disjunkt

$\Rightarrow \Pi = \dot{\bigcup_{k=1}^{n}} \Pi_k \qquad |\Pi_k| = (n-1)!$

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 $k+1, \dots, n$.



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

$$\sum_{\pi \in \Pi_k} C(\pi) = \underbrace{\sum_{\pi \in \Pi_k} z(\pi)}_{=: S_1} + \underbrace{\sum_{\pi \in \Pi_k} C(\pi_\ell)}_{=: S_2} + \underbrace{\sum_{\pi \in \Pi_k} C(\pi_r)}_{=: S_3}$$

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

Vergleiche für π_ℓ

Vergleiche für π_r

$$S_1 \overset{\text{Lemma}}{=} \sum_{\pi \in \Pi_k} n = |\Pi_k| \cdot n = (n-1)! \cdot n = n!$$

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

π_L entsteht aus π bei der Zerlegung im ersten Schritt von QuickSort.

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

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

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

analog dazu kann man zeigen, dass

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

Damit haben wir erhalten:

$$\bar{C}(n) = \frac{1}{n!} \sum_{k=1}^{n} (S_1 + S_2 + S_3)$$

$$\le \frac{1}{n!} \cdot \sum_{k=1}^{n} \left(n! + (n-1)!\, \bar{C}(k-1) + (n-1)!\, \bar{C}(n-k) \right)$$

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

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

$$= n + \frac{2}{n} \sum_{k=2}^{n-1} \bar{C}(k) \qquad \text{da } \bar{C}(0) = \bar{C}(1) = 0$$

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

Rekursionsungleichung

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

ogenauer: $\bar{C}(n) \leq \underbrace{2}_{c} \cdot n \cdot \ln(n)$

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

Ind.-Anf.: $n = 2 \qquad \bar{C}(2) = 1$

$c \cdot 2 \cdot \ln(2) = c \cdot 1.39\ldots \geq 1$ für $c \geq 1$

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

Ind.-Schluss: Zeigen Beh. für n:

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

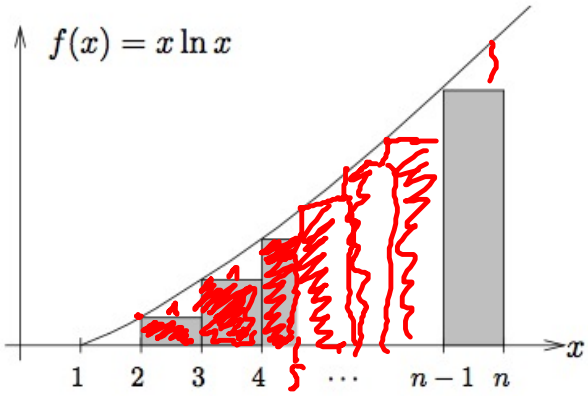
$$\overset{\text{I. Ann.}}{\leq} n + \frac{2}{n} \cdot \sum_{k=2}^{n-1} c \cdot k \cdot \ln(k)$$

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

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

Dann ist

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



$$\int_2^n f(x)\,dx = \int_2^n x \ln(x)\,dx$$

$$\text{part. Int.} \quad \frac{x^2}{2} \cdot \ln(x)\Big|_2^n - \int_2^n \frac{x^2}{2} \cdot \frac{1}{x}\,dx$$

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

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

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

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

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

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

da $\ln(n) = \frac{\log n}{\log e}$:

Wir haben also gezeigt, dass die mittlere # Vergleiche für QuickSort $\overline{C}(n) \in O(n \log n)$