



Untere Schranke für Sortieralgorithmen

Weil unter den Beispielen zur “Anwendung von Satz A” in der Vorlesung gleich als erstes das schwierigste Beispiel, das Sortieren von Zahlen, angeführt wurde¹ und diese Anwendung vielleicht auf den ersten Blick nicht unproblematisch scheint, hier ein paar Erklärungsversuche bzw. Antworten auf *frequently asked questions*:

Hat das mit Problemreduktion zu tun? Nicht wirklich. Es geht nicht darum, mithilfe der hier definierten Funktion

$$f(x_1, \dots, x_n) := x_1^{j_1} + \dots + x_n^{j_n}$$

wirklich das Sortieren von n Zahlen x_i zu bewerkstelligen – wie sollte das auch gehen? Die Ausgabe soll ja eine Permutation der Folge $(1, \dots, n)$ sein, und zwar genau das Tupel (j_1, \dots, j_n) . Und nicht der skalare Wert $f(\dots) \in \mathbb{R}$.

(Wir können also nicht Zahlen sortieren, indem wir f berechnen; wir können aber – umgekehrt – mit einem Aufwand ohne Vergleiche $f(\dots)$ berechnen, wenn wir vorher die Zahlen x_i sortiert haben.)

Wozu wird f definiert? Wir können Satz A (das Lower-Bound-Theorem) nur anwenden, wenn für jeden möglichen Programmlauf (und das heißt hier: für jede mögliche Permutation der Indizes $1, \dots, n$) eine andere Zielfunktion Q_i herauskommt.

Genau so ist f auch definiert: Es gibt $n!$ verschiedene Permutationen, die alle prinzipiell als Ergebnis eines Sortiervorgangs möglich sind. Je nach (j_1, \dots, j_n) gibt es auch $n!$ verschiedene Funktionen $Q_i(x_1, \dots, x_n)$, die wir als Ausgabe A_i in jedes Blatt des Ablaufbaumes für unser RAM-Programm stecken.

Wenn die einzelnen Werte x_i nun gering variieren, sodaß aber nicht ihre Sortierordnung (j_1, \dots, j_n) gestört wird, beschreibt tatsächlich jede Teilfunktion Q_i (mit $i = 1, \dots, n!$) einen anderen Verlauf.

Gibt es dazu ein kleines Beispiel? Ja. Für $n = 4$ ergeben sich bspw. die folgenden $n! = 24$ Teilfunktionen Q_i bzw. A_i :

(nur der besseren Lesbarkeit wegen sei der Eingabevektor hier mit $(a, b, c, d) \equiv (x_1, x_2, x_3, x_4)$ notiert)

$$Q_1(a, b, c, d) := a^1 + b^2 + c^3 + d^4$$

$$Q_{13}(a, b, c, d) := a^3 + b^1 + c^2 + d^4$$

$$Q_2(a, b, c, d) := a^1 + b^2 + c^4 + d^3$$

$$Q_{14}(a, b, c, d) := a^3 + b^1 + c^4 + d^2$$

$$Q_3(a, b, c, d) := a^1 + b^3 + c^2 + d^4$$

$$Q_{15}(a, b, c, d) := a^3 + b^2 + c^1 + d^4$$

$$Q_4(a, b, c, d) := a^1 + b^3 + c^4 + d^2$$

$$Q_{16}(a, b, c, d) := a^3 + b^2 + c^4 + d^1$$

¹Skriptum, Abschnitt 1.5, Beispiel 1

$$\begin{array}{ll}
Q_5(a, b, c, d) := a^1 + b^4 + c^2 + d^3 & Q_{17}(a, b, c, d) := a^3 + b^4 + c^1 + d^2 \\
Q_6(a, b, c, d) := a^1 + b^4 + c^3 + d^2 & Q_{18}(a, b, c, d) := a^3 + b^4 + c^2 + d^1 \\
Q_7(a, b, c, d) := a^2 + b^1 + c^3 + d^4 & Q_{19}(a, b, c, d) := a^4 + b^1 + c^2 + d^3 \\
Q_8(a, b, c, d) := a^2 + b^1 + c^4 + d^3 & Q_{20}(a, b, c, d) := a^4 + b^1 + c^3 + d^2 \\
Q_9(a, b, c, d) := a^2 + b^3 + c^1 + d^4 & Q_{21}(a, b, c, d) := a^4 + b^2 + c^1 + d^3 \\
Q_{10}(a, b, c, d) := a^2 + b^3 + c^4 + d^1 & Q_{22}(a, b, c, d) := a^4 + b^2 + c^3 + d^1 \\
Q_{11}(a, b, c, d) := a^2 + b^4 + c^1 + d^3 & Q_{23}(a, b, c, d) := a^4 + b^3 + c^1 + d^2 \\
Q_{12}(a, b, c, d) := a^2 + b^4 + c^3 + d^1 & Q_{24}(a, b, c, d) := a^4 + b^3 + c^2 + d^1
\end{array}$$

Offenbar sind diese Q_i **verschiedene rationale Funktionen**.²

Wie wird jetzt genau Satz A angewendet? Da wir für die Eingabe $X = (x_1, \dots, x_n)$ den stetigen Definitionsraum \mathbb{R}^n betrachten, auf dem die Q_i definiert sind,³ muß der Umgebungsparameter ε vorsichtig genug gewählt sein: Wenn wir als Mittelpunkte X_i der ε -Kugeln bequemerweise Tupel aus ganzzahligen Werten wählen, also für unser Beispiel $n = 4$ die $q := n! = 24$ Ortsvektoren

$$\begin{array}{llll}
X_1 = (1, 2, 3, 4) & X_7 = (2, 1, 3, 4) & X_{13} = (3, 1, 2, 4) & X_{19} = (4, 1, 2, 3) \\
X_2 = (1, 2, 4, 3) & X_8 = (2, 1, 4, 3) & X_{14} = (3, 1, 4, 2) & X_{20} = (4, 1, 3, 2) \\
X_3 = (1, 3, 2, 4) & X_9 = (2, 3, 1, 4) & X_{15} = (3, 2, 1, 4) & X_{21} = (4, 2, 1, 3) \\
X_4 = (1, 3, 4, 2) & X_{10} = (2, 3, 4, 1) & X_{16} = (3, 2, 4, 1) & X_{22} = (4, 2, 3, 1) \\
X_5 = (1, 4, 2, 3) & X_{11} = (2, 4, 1, 3) & X_{17} = (3, 4, 1, 2) & X_{23} = (4, 3, 1, 2) \\
X_6 = (1, 4, 3, 2) & X_{12} = (2, 4, 3, 1) & X_{18} = (3, 4, 2, 1) & X_{24} = (4, 3, 2, 1),
\end{array}$$

dann sind diese Punkte X_i nicht nur paarweise verschieden, sondern sie liegen auch noch jeweils mindestens $\sqrt{2}$ Längeneinheiten⁴ auseinander. Wir können den Radius der die X_i umgebenden 4-dimensionalen Kugeln also getrost auf $\varepsilon := 1/2 < \sqrt{2}/2$ setzen.

Wir haben die X_i bewußt derart gewählt, daß⁵ **in der ε -Umgebung eines jeden X_i die Funktion f lokal identisch einer Funktion Q_j** ist, wobei alle $i, j \in \{1, \dots, n!\}$ vorkommen. (Wir haben die Q_j und ihre Stützstellen X_i sogar so symmetrisch definiert, daß die Zuordnung jeweils für $i \equiv j \in \{1, \dots, n!\}$ hin- und rückwärts, wie leicht nachzuprüfen ist.)

Nun, da alle Voraussetzungen erfüllt sind, kann das Lower-Bound-Theorem angewendet werden:

Die Berechnung von f mit einer Random Access Machine benötigt im allgemeinen, d. h. schlechtesten Fall

$$R \geq \log_2 q = \log_2 n! \in \Theta(n \log n)$$

Vergleichsoperationen.

Dieses Ergebnis stimmt mit der herkömmlichen **unteren Schranke fürs Sortierproblem** $R \in \Omega(n \log n)$ überein, weil die Berechnung von $f(X) = \sum_i x_i^{j_i}$ überhaupt keine Vergleiche benötigt und die korrekt sortierten Indizes (j_1, \dots, j_n) dem Programm bekannt sein müssen, um einen Wert $f(X_0)$ zu ermitteln.

²auch wenn sie an Punkten, die auf Diagonalen liegen (wie bspw. $b = c$), gleiche Werte liefern können

³und nicht – wie im Skriptum nahegelegt – den diskreten Raum der Permutationen X des Tupels $(1, \dots, n)$

⁴ $= \sqrt{1^2 + 1^2} \leq |X_i - X_j|$ (für $i \neq j$) nach dem euklidischen Abstandsmaß

⁵wie es auch die Voraussetzung von Satz A verlangt