1 Längste aufsteigende Teilfolge – Algorithmenentwicklung durch Induktion

Gegeben sei eine Folge $S = x_1, x_2, \ldots, x_n$ von paarweise verschiedenen Zahlen. Eine *Teilfolge* von S ist eine Folge $x_{i_1}, x_{i_2}, \ldots, x_{i_k}$ mit $i_1 < i_2 < \ldots < i_k$. Dementsprechend bezeichnen wir als *aufsteigende Teilfolge* von S eine Teilfolge mit $x_{i_j} < x_{i_{j+1}}$ für alle $1 \leq j < k$. Wir führen ferner die Bezeichnung $|S|$ für die Länge n der Folge S ein.

> **Problem:** Bestimme eine aufsteigende Teilfolge maximaler Länge für eine Folge S paarweise verschiedener Zahlen.

Wir erläutern am Beispiel dieses Problems die Entwicklung eines Algorithmus per Induktion, ausgehend von folgendem einfachen Ansatz:

Induktionsannahme: Für eine Folge der Länge $< m$ kann eine längste aufsteigende Teilfolge gefunden werden.

Der Induktionsbeginn (Länge 1) ist trivial, und für den Induktionsschritt können wir annehmen, eine längste wachsende Teilfolge T von $x_1, x_2, \ldots x_{m-1}$ zu kennen. Bei Hinzunahme des Elements x_m ist T eventuell erweiterbar, aber nur dann, wenn das letzte Element von T kleiner als x_m ist.

Dies ist aber offensichtlich ein zu schwacher Ansatz, denn wenn nicht T um x_m erweiterbar ist, dann vielleicht eine andere längste Teilfolge T' mit $|T'| = |T|$, die aber in einem kleineren Element als T endet. Der Algorithmus findet also nicht notwendig eine maximale Lösung.

Wir können versuchen, dieses Problem zu umgehen, indem wir *sämtliche* Teilfolgen betrachten:

Induktionsannahme: Für eine Folge der Länge $< m$ können *alle* längsten aufsteigenden Teilfolgen gefunden werden.

Wieder ist der Induktionsbeginn trivial. Aber wie beim ersten Ansatz ergibt sich ein grundsätzliches Problem: Angenommen, keine der gespeicherten längsten wachsenden Teilfolgen T von $x_1, x_2, \ldots, x_{m-1}$ ist durch das neue Element x_m erweiterbar. Die Induktionsannahme verlangt aber, daß wir *alle* längsten Teilfolgen kennen. Diese Teilfolgen haben nach wie vor die Länge $|T|$, können nun aber auch mit x_m enden. Wir müssen also zu den Teilfolgen aus der Induktionsannahme noch die mit x_m endenden hinzufügen, um für den nächsten Schritt eine vollständige Menge an längsten aufsteigenden Teilfolgen zu haben. Dafür benötigen wir wiederum alle wachsenden Teilfolgen der Länge $|T| - 1$, u.s.w.

Die Speicherung mehrerer wachsender Teilfolgen maximaler Länge führt also offensichtlich zu einem nicht beherrschbaren Aufwand. Daher versuchen wir, selektiver vorzugehen: Wir stellen fest, daß die Speicherung einer Teilfolge $T = x_{i_1}, x_{i_2}, \ldots, x_{i_k}$ von $x_1, x_2, \ldots, x_{m-1}$ offenbar redundant ist, wenn wir eine Teilfolge $T' = x_{i'_1}, x_{i'_2}, \ldots, x_{i'_k}$ kennen mit der Eigenschaft $x_{i'_k} < x_{i_k}$. Wir können uns also auf folgende Forderung beschränken:

Induktionsannahme: Für eine Folge der Länge $< m$ kann eine längste aufsteigende Teilfolge mit minimalem letzten Element gefunden werden.

Der Induktionsanfang ist nach wie vor trivial. Angenommen, die Induktionsvoraussetzung liefert für $|S| = m-1$ eine "beste" längste aufsteigende Teilfolge T. Ist T durch x_m erweiterbar, so sind wir fertig. Denn die Lösung für $|S| = m$ ist in diesem Fall eindeutig. Wenn aber T nicht erweiterbar ist, so ist (ähnlich wie im letzten Versuch) T möglicherweise nicht mehr die aktuell beste Sequenz. Das kann dann der Fall sein, wenn x_m kleiner ist als das letzte Element von T. Wir brauchen also die längsten aufsteigenden Teilfolgen der Längen $|T|, |T|-1, \ldots, 1$ von S, um bei Hinzunahme von x_m ein maximales Ergebnis garantieren zu können. Ist dieser Aufwand nötig? – Auch ohne Induktion sieht man leicht ein, daß kürzere Teilfolgen im Laufe des Algorithmus nicht einfach verworfen werden können, denn jede von ihnen könnte schlußendlich der Beginn der Ergebnis-Sequenz sein.

Induktionsannahme: *Für eine Folge der Länge $< m$ kann (für jedes $k < m$) eine Teilfolge BIS(k) der Länge k mit minimalem letzten Element bestimmt werden, falls sie existiert.*

Wir führen die Bezeichnung $BIS(k).last$ ein für das letzte Element von $BIS(k)$, und es sei s die Länge der aktuellen LIS (also der größte Wert, für den $BIS(s)$ definiert ist). Für den Induktionsschritt haben wir nun also $BIS(1)$,$BIS(2)$,...,$BIS(s)$ zur Verfügung und müssen diejenigen Sequenzen bestimmen, die sich nach Hinzunahme von x_m ändern. Das Element x_m kann eine Sequenz $BIS(k)$ nur dann verlängern und damit ein neues $BIS(k+1)$ liefern, wenn $BIS(k).last < x_m$ *und* $x_m < BIS(k+1).last$. Wie läßt sich dies effizient testen?

Offensichtlich gilt $BIS(1).last < BIS(2).last < \ldots < BIS(s).last$. Somit können wir der Reihe nach die Werte von $BIS(i).last$ für absteigende i betrachten, bis zum erstenmal ein $BIS(j).last < x_m$ auftritt.

- Falls alle $BIS(j).last \geq x_m$, so wird $BIS(1) := x_m$.
- Ergibt sich $j = s$, so setzen wir $BIS(s+1) := BIS(s)x_m$ und $s = s+1$.
- Ansonsten gilt $BIS(j).last < x_m < BIS(j+1).last$, und wir können $BIS(j+1)$ ersetzen durch $BIS(j)x_m$.

Nun bleibt nur noch die Suche nach j effizient zu implementieren. Nachdem wir bereits erkannt haben, daß die $BIS(i).last$ sortiert sind, geht dies leicht durch binäre Suche. Somit kann der richtige Wert von j immer in Zeit $O(\log n)$ gefunden werden.

Um insgesamt einen effizienten Algorithmus zu erhalten, muß man außerdem darauf verzichten, die $BIS(j)$ komplett zu speichern und in jedem Schritt zu aktualisieren. (Das Ersetzen von $BIS(j+1)$ durch $BIS(j)x_m$ würde sonst zum Beispiel zu lange dauern.) Stattdessen speichert man für jedes j, $1 \leq j \leq s$, lediglich den Index i des Elementes x_i, das $BIS(j).last = x_i$ erfüllt. Somit können alle Aktualisierungen nach dem Finden des richtigen Index j in konstanter Zeit erfolgen, und es ergibt sich eine Komplexität von $O(n \log n)$.

Um am Ende auch eine längste aufsteigende Teilsequenz ausgeben zu können, genügt es, sich beim Setzen von $BIS(j).last = x_i$ den Index von $BIS(j-1).last$ als Vorgänger $pred(x_i)$ von x_i zu merken. Am Ende kann dann durch Zurückverfolgen der $pred$-Werte von $BIS(s).last$ aus eine längste aufsteigende Teilsequenz rekonstruiert werden. Durch diese Erweiterung verschlechtert sich die Zeitkomplexität des Algorithmus nicht.

Abschließend soll die Berechnung einer längsten aufsteigenden Teilfolge durch den beschriebenen Algorithmus am Beispiel der Folge $10, 19, 13, 17, 14, 15, 12$ (d.h. $x_1 = 10$, $x_2 = 19, \ldots, x_7 = 12$) vorgeführt werden. Die folgende Tabelle zeigt für $m = 1, \ldots, 7$ die Werte der relevanten Variablen nach Bearbeitung von x_m, wobei jede Spalte ℓ_j den Wert $BIS(j).last$ zeigt und jede Spalte p_j den Index von $pred(x_j)$.

m	ℓ_1	ℓ_2	ℓ_3	ℓ_4	ℓ_5	ℓ_6	ℓ_7	p_1	p_2	p_3	p_4	p_5	p_6	p_7
1	$x_1=10$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$
2	$x_1=10$	$x_2=19$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	1	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$
3	$x_1=10$	$x_3=13$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	1	1	$\perp$	$\perp$	$\perp$	$\perp$
4	$x_1=10$	$x_3=13$	$x_4=17$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	1	1	3	$\perp$	$\perp$	$\perp$
5	$x_1=10$	$x_3=13$	$x_5=14$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	1	1	3	3	$\perp$	$\perp$
6	$x_1=10$	$x_3=13$	$x_5=14$	$x_6=15$	$\perp$	$\perp$	$\perp$	$\perp$	1	1	3	3	5	$\perp$
7	$x_1=10$	$x_7=12$	$x_5=14$	$x_6=15$	$\perp$	$\perp$	$\perp$	$\perp$	1	1	3	3	5	1

Am Ende gilt $s = 4$, d.h. die längste aufsteigende Teilsequenz hat Länge 4. Ausgehend von $BIS(4).last = x_6 = 15$ liefert das Zurückverfolgen der $pred$-Werte wegen $pred(x_6) = x_5$, $pred(x_5) = x_3$, $pred(x_3) = x_1$ und $pred(x_1) = \perp$ die längste aufsteigende Teilsequenz $x_1 = 10$, $x_3 = 13$, $x_5 = 14$, $x_6 = 15$.

2 Erweiterung auf Folgen von Zahlenpaaren

Wir betrachten die folgende Erweiterung des LIS-Problems. Statt einer Folge von Zahlen sei jetzt eine Folge von Zahlenpaaren gegeben. Aufeinanderfolgende Paare (x_i, y_i) und (x_j, y_j) einer aufsteigenden Teilfolge müssen dann $x_i < x_j$ und $y_i < y_j$ erfüllen. Gesucht ist wieder eine längste aufsteigende Teilfolge der gegebenen Folge.

Für dieses erweiterte LIS-Problem auf Zahlenpaaren geht die im vorigen Abschnitt verwendete Induktion nicht ohne Weiteres durch: $BIS(k)$ ist i.a. nicht eindeutig. Man muß daher eine schwächere Induktionsvoraussetzung wählen, sich also mehr als eine einzige "beste" aufsteigende Sequenz jeder Länge merken. Naheliegende Lösungen führen dabei aber sofort zu (mindestens) quadratischer Laufzeit. Bei genauerer Überlegung lassen sich aber einige der Ideen, die zu dem $O(n \log n)$-Algorithmus für einfache Zahlenfolgen geführt haben, auf das erweiterte Problem übertragen.

Die beste uns bekannte Lösung für das erweiterte Problem hat eine Worst-Case-Komplexität $O(n \log^2 n)$. Aber auch Lösungen mit schlechterer (z.B. quadratischer) Worst-Case-Laufzeit lassen sich mit mäßigem Aufwand so optimieren, daß sie bei zufällig generierten Folgen von Zahlenpaaren einer Laufzeit $O(n \log n)$ recht nahe kommen. Der ursprüngliche Autor dieser Aufgabe, sowie der Autor dieses Textes, sind sich nicht sicher, ob auch das erweiterte LIS-Problem in Worst-Case-Zeit $O(n \log n)$ zu lösen ist.