Algorithmen & Datenstrukturen

Woche 6

Marius Tomek, Nicolas Wehrli, Tim Rieder

31. Oktober 2022

ETH Zürich

Inhalt

Kurze Kommentare zur letzten Serie

Dynamische Programmierung

Longest Increasing Subsequence

Longest Increasing Subsequence

Wir betrachten ein Array von beliebigen Werten mit Länge $n \in \mathbb{N}$:

Wir wollen die längste strikt wachsende Subsequenz finden

Hierzu verwenden wir DP:

Longest Increasing Subsequence

Bottom-up design:

Längste strikt wachsende Subsequenz welche in k endet:

$$LIS(k) = \max(LIS(i) + 1)\ \forall i\ (i < k) \wedge (A[i] < A[k])$$

Aus $LIS(k)\ \forall k < n$ folgt längste strikt wachsende Subsequenz:

$$\max_{\forall i < n}(LIS(k))$$

Longest Increasing Subsequence

Algorithm 1 LIS(A, n)

1: $B \leftarrow$ **new Array**$[n]$ $\{1 \ldots 1\}$
2: **for** $k \leftarrow 1$ to n **do**
3: **for** $j \leftarrow 1$ to k **do**
4: **if** $A[k] > A[j]$ **then** ▷ Change B[k] only if A[k] > A[j]
5: $B[k] \leftarrow \max(B[k], B[j] + 1)$ ▷ Set B[k] to B[j] + 1 if larger than B[k]
6: **return** $\max_{1 \leq i \leq n}(B[i])$ ▷ Return maximum in array B

Longest Common Subsequence

Min Edit Distance

Homework 05

Peergrading 5.5