Übungen zu Algorithmen | **Wintersemester 2020/21**
WSI für Informatik | **Universität Tübingen**
Kaufmann/Kuckuk/Förster | **16. Februar 2021**

Aufgabensammlung Teil 12

Thema: Dynamische Programmierung

Aufgabe 1: Maximale Pfadsumme

Dies ist eine alte Klausuraufgabe!

Gegeben sei ein Schema S von Zahlen, dass wie folgt aufgebaut ist:

$$
\begin{array}{ccccccccccc}
 & & & & & Z_{1,1} & & & & & \\
 & & & & Z_{2,1} & & Z_{2,2} & & & & \\
 & & & Z_{3,1} & & Z_{3,2} & & Z_{3,3} & & & \\
 & & Z_{4,1} & & Z_{4,2} & & Z_{4,3} & & Z_{4,4} & & \\
 & Z_{5,1} & & Z_{5,2} & & Z_{5,3} & & Z_{5,4} & & Z_{5,5} & \\
Z_{6,1} & & Z_{6,2} & & Z_{6,3} & & Z_{6,4} & & Z_{6,5} & & Z_{6,6} \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
Z_{n,1} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & Z_{n,n}
\end{array}
$$

Dabei hat ein Schema n Zeilen und Zeile i hat genau i Einträge. Ein *gültiger Pfad* in S ist eine Zahlenfolge $(Z_{1,i_1}, Z_{2,i_2}, Z_{3,i_3}, \ldots, Z_{n,i_n})$, wobei $i_1 = 1$ und $i_{k+1} \in \{i_k, i_k + 1\}$ für $k = 1, \ldots, n-1$ (von einem Eintrag im Schema ist es also nur erlaubt zu einem der beiden Einträge diagonal darunter zu gehen, zum Beispiel von $Z_{5,2}$ nach $Z_{6,2}$ oder $Z_{6,3}$, siehe Abbildung). Eine *Pfadsumme* ist die Summe aller n Zahlen auf einem gültigen Pfad.

a) Geben Sie einen Algorithmus in Pseudocode an, der mit Hilfe von **Dynamischer Programmierung** die maximale Pfadsumme für ein beliebiges Schema berechnet. Erklären Sie auch die Idee/Funktionsweise Ihres Algorithmus.

b) Begründen Sie kurz die Korrektheit und die Laufzeit Ihres Algorithmus.

Aufgabe 2: Längste Gemeinsame Substrings

Dies ist eine alte Klausuraufgabe!

Ein String $w = w_1 \cdots w_k$ ist ein *Substring* eines Strings $s = s_1 \cdots s_n$, wenn es eine streng monoton wachsende Folge $\langle i_1, \ldots, i_k \rangle$ gibt, so dass $s_{i_j} = w_j$ für alle $j = 1, \ldots, k$ (d. h. w entsteht aus s durch Streichen von Buchstaben). Gegeben seien zwei Strings $s = s_1 \cdots s_n$ und $t = t_1 \cdots t_m$. Ein *längster gemeinsamer Substring* $w = w_1 \cdots w_k$ von s und t ist wie folgt definiert:

- w ist sowohl Substring von s als auch von t und
- jeder andere gemeinsame Substring von s und t hat eine Länge von höchstens k.

Zum Beispiel sind 'OIT', 'LIT' und 'OME' längste gemeinsame Substrings von s = ALGORITHMEN und t = KOMPLEXITÄT.

a) Geben Sie einen Algorithmus in Pseudocode an, der mit Hilfe von **Dynamischer Programmierung** einen längsten gemeinsamen Substring von s und t berechnet.

b) Begründen Sie kurz die Korrektheit und die Laufzeit Ihres Algorithmus.

Aufgabe 3: fancyFunction

Dies ist eine alte Klausuraufgabe!

Sei M eine $n \times n$-Matrix mit natürlichen Zahlen als Einträge. Dabei ist $M_{i,j}$ der Eintrag der i-ten Zeile und j-ten Spalte. Betrachten Sie folgenden Pseudocode:

```
if i = n und j = n then
    return M_{i,j};
end
opt_1, opt_2 ⟵ 0;
if i < n then
    opt_1 ⟵ M_{i,j}+fancyFunction(M, i + 1, j);
end
if j < n then
    opt_2 ⟵ M_{i,j}+fancyFunction(M, i, j + 1);
end
return max(opt_1, opt_2);
```

Algorithm 1: fancyFunction(M, i, j)

Der Startaufruf ist fancyFunction$(M, 1, 1)$.

a) Erklären Sie mit Hilfe einer Skizze, was dieser Pseudocode berechnet.

b) Geben Sie eine ungefähre Laufzeit an.

c) Wie können Sie die Laufzeit mit Hilfe von Dynamischer Programmierung verbessern? Geben Sie einen Algorithmus in Pseudocode an und begründen Sie dessen Korrektheit und Laufzeit.

Lösungen:

1. a) Bestimme $\Sigma_{i,j}$ als maximal erreichbare Summe von $Z_{1,1}$ zu $Z_{i,j}$. Benutze dazu jeweils die Einträge $\Sigma_{i-1,j}$ und $\Sigma_{i-1,j-1}$.

   b) Durch zeilenweises Vorgehen sind die Werte der Zeile $i-1$ korrekt berechnet, wenn Zeile i betrachtet wird. Laufzeit ist $\mathcal{O}(n^2)$.

2. a) Bestimme $L_{i,j}$ als längster gemeinsamer Substring der ersten i Zeichen von s und der ersten j Zeichen von t. Benutze dazu jeweils den Eintrag $L_{i-1,j-1}$ (nur falls Zeichen i von s und j von t sind gleich) bzw. das Maximum von $L_{i-1,j}$ und $L_{i,j-1}$.

   b) Durch vorherige Berechnung von $L_{i-1,j-1}$, $L_{i-1,j}$ und $L_{i,j-1}$ kann $L_{i,j}$ korrekt bestimmt werden. Laufzeit ist $\mathcal{O}(n \cdot m)$.

3. a) Es wird die Pfadsumme wie bei 1. berechnet, wobei man von $M_{1,1}$ zu $M_{n,n}$ geht.

   b) $\Theta\left(\binom{2n}{n}\right)$. Jeder Pfad ist $2n$ lang, dabei wird n-mal eine Spalte weitergegangen.

   c) Die rekursiven Aufrufe in Zeile 6 bzw. 9 sollten nur dann ausgeführt werden, wenn die entsprechenden Aufrufe zum ersten Mal geschehen. In diesem ersten Aufruf, speichert man zwischen Zeile 10 und 11 das Ergebnis ab, das man dann bei späteren Aufrufen wieder abrufen kann.