Übung 13: Dynamische Programmierung

Aufgabe 1: Rod Cutting

Das Beispiel der Vorlesung für das Rod-Cutting-Problem wird erweitert. Zusätzlich können nun auch Stäbe der Länge $i = 9$ und $i = 10$ zugeschnitten werden und mit den Preisen $p_9 = 24$ und $p_{10} = 30$ verkauft werden. Es gelten die Bezeichnungen der Vorlesungen, die komplette Preistabelle für Stäbe der Länge *i* ist wie folgt vorgegeben:

Länge i	1	2	3	4	5	6	7	8	**9**	**10**
Preis p_i	1	5	8	9	10	17	17	20	**24**	**30**

Für die maximale Stablänge $i = 8$ sei die Ergebnistabelle bereits vorgegeben.

Index i	0	1	2	3	4	5	6	7	8	9	10
r[i]	0	1	5	8	10	13	17	18	22		
s[i]	0	1	2	3	2	2	6	1	2		

a) Was ist der maximale Erlös $r[9]$? Welcher Wert ergibt sich für $s[9]$?
Führen Sie zur Begründung den Pseudocode `EXTENDED-BOTTOM-UP-CUT-ROD(p,n)` aus und geben Sie den Wert von **`q`** und **`s[9]`** am Ende ***jeder(!)*** Iteration der ***inneren*** `for`-Schleife an (Zeile 8, Folie 20). Starten Sie also gleich bei $j = 9!$.

b) Was ist der maximale Erlös $r[10]$? Welcher Wert ergibt sich für $s[10]$? Eine ausführliche Begründung wie in a) ist hier nicht notwendig.

c) Was ist die asymptotische Laufzeit von `EXTENDED-BOTTOM-UP-CUT-ROD(p,n)` in Bezug auf die Eingabe *n* (= Stablänge)? Könnte man dynamische Programmierung auch anwenden, falls der Stab an ***beliebigen*** Stellen (nicht nur bei ganzzahligen Werten) unterteilt werden kann und der Preis als Funktion der Länge gegeben ist?

Aufgabe 2: Längste Gemeinsame Teilfolge / Longest Common Subsequence (LCS)

Was ist die Längste Gemeinsame Teilfolge der beiden Folgen:

- $X =< B, A, A, B >$ und
- $Y =< A, B, A, B, B, A, B >$?

Verwenden Sie dazu ***exakt*** den Algorithmus der Vorlesung und vervollständigen Sie die Tabelle in ähnlicher Weise wie auf Folie 34.

	j	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
i		y_i	**A**	**B**	**A**	**B**	**B**	**A**	**B**
0	x_i	0	0	0	0	0	0	0	0
1	**B**	0	0 ↑						
2	**A**	0							
3	**A**	0							
4	**B**	0							

Aufgabe 3: Levenshtein - Editierdistanz

Der *Levenshtein-Algorithmus* errechnet die Mindestanzahl von Editieroperationen (Einfügen, Löschen, Substitution), um eine bestimmte Zeichenkette in eine andere Zeichenkette zu überführen. Das Problem ist eng verwandt mit der *Längsten Gemeinsamen Teilfolge*. Die Grundidee wurde kurz in der Vorlesung vorgestellt, beachten Sie die Folien. Gegeben seien die folgenden 2 Zeichenketten:

$X = L\ E\ V\ E\ N\ S\ H\ T\ E\ I\ N$
$Y = M\ E\ I\ L\ E\ N\ S\ T\ E\ I\ N$

a) Ergänzen Sie die untenstehende Tabelle, um die Levenshtein-Distanz für die Überführung der Zeichenkette *X* in die Zeichenkette *Y* zu berechnen. Was ist die Editierdistanz?

b) Ergänzen Sie den Konstruktor der vorgegebenen Klasse `Levenshtein.java`. Der Konstruktur soll die benötigte Tabelle errechnen. Testen Sie mit der JUnit-Klasse.

c) Welche Laufzeit hat der Levenshtein Algorithmus für 2 Zeichenketten der Länge *m* und *n*?

d) Schauen Sie sich berechnete Tabelle Ihres Programmes an, die von der JUnit Testklasse ausgegeben wird. Geben Sie an, welche Editieroperationen man auf die Zeichenkette „LEVENSHTEIN“ anwenden muss, um diese in „MEILENSTEIN“ zu überführen. **WICHTIG**: Die Anzahl der Editieroperationen muss ***minimal*** sein!

	j	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
i		y_i	**M**	**E**	**I**	**L**	**E**	**N**	**S**	**T**	**E**	**I**	**N**
0	x_i	0	1	2	3	4	5	6	7	8	9	10	11
1	**L**	1											
2	**E**	2											
3	**V**	3											
4	**E**	4											
5	**N**	5											
6	**S**	6											
7	**H**	7											
8	**T**	8											
9	**E**	9											
10	**I**	10											
11	**N**	11											