8. Übungsblatt
Thema: Dynamische Programmierung

Aufgabe 33

Schreiben Sie ein Programm, das das Geldwechselproblem aus Aufgabe 12 (Bestimmung der Anzahl der Möglichkeiten zum Wechseln eines gegebenen Geldbetrages) mit Hilfe dynamischer Programmierung löst.
Hinweis: Erinnern Sie sich an die Lösung des Rucksackproblems mit Hilfe dynamischer Programmierung und versuchen Sie, eine ähnliche Rekursionsformel aufzustellen.

Aufgabe 34

In vielen Anwendungen tritt das Problem auf, zu bestimmen, wie ähnlich zwei Zeichenketten sind; z.B. wenn in einem Text schreibweisen- und tippfehlertolerant gesucht werden soll. Ein bekannter Lösungsansatz für dieses Problem ist der sogenannte Editierabstand, auch Levenshtein-Distanz genannt [1]. Dieses Maß ist definiert als die minimale Anzahl von Editieroperationen, d.h. von Einfügungen, Löschungen und Ersetzungen von Buchstaben, durch die sich die eine Zeichenkette in die andere umwandeln läßt. Gegeben seien z.B. die Zeichenketten s_1 = "Regen" und s_2 = "Reise". Dann ist der Editierabstand 3, da ein Buchstabe eingefügt (das 'i' oder das 's'), ein Buchstabe geändert (das 'g' in ein 'i' oder ein 's') und ein Buchstabe gelöscht werden muß (das 'n').

Wir betrachten in dieser Aufgabe eine verfeinerte Version dieses Maßes, die sogenannte gewichtete Levenshtein-Distanz. In dieser Version wird jeder Editieroperation ein (nicht negatives) Gewicht zugeordet, und zwar dem Einfügen eines Buchstabens das Gewicht w_{ins} (insertion), dem Löschen eines Buchstabens das Gewicht w_{del} (deletion), dem Ändern eines Buchstabens von Groß- in Kleinschreibung oder umgekehrt das Gewicht w_{case} (case change), und dem Ersetzen eines Buchstabens durch einen anderen das Gewicht w_{subst} (substitution). Der Abstand zweier Zeichenketten ist dann das minimale Gewicht einer Folge von Editieroperationen, die die eine Zeichenkette in die andere überführt.

Die gewichtete Levenshtein-Distanz zweier Zeichenketten $a_1 \dots a_n$ und $b_1 \dots b_m$ kann folgendermaßen rekursiv definiert werden (man vergleiche das in der Vorlesung behandelte Problem der längsten gemeinsamen Teilfolge):

$$d(a_1 \dots a_n, b_1 \dots b_m) = \begin{cases} m \cdot w_{\text{ins}}, & \text{falls } n = 0, \\ n \cdot w_{\text{del}}, & \text{falls } m = 0, \\ \min\{d(a_1 \dots a_n, b_1 \dots b_{m-1}) + w_{\text{ins}}, & \\ \qquad d(a_1 \dots a_{n-1}, b_1 \dots b_m) + w_{\text{del}}, & \text{sonst.} \\ \qquad d(a_1 \dots a_{n-1}, b_1 \dots b_{m-1}) + w(a_n, b_m)\}, & \end{cases}$$

wobei

$$w(a, b) = \begin{cases} 0, & \text{falls } a = b, \\ w_{\text{case}}, & \text{falls uppercase}(a) = \text{uppercase}(b), \\ w_{\text{subst}}, & \text{sonst.} \end{cases}$$

Schreiben Sie ein Programm, das für zwei Zeichenketten die gewichtete Levenshtein-Distanz mit Hilfe dynamischer Programmierung berechnet. Sehen Sie eine Möglichkeit vor, die Gewichte der Editieroperationen anzugeben.

Literatur

[1] V.I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics Doklady 10(8), 707–710, 1966, and Doklady Akademii Nauk SSSR 163(4), 845-848, 1965

Aprilscherzaufgabe (für Informatiker und Wirtschaftsinformatiker)

Thema: Graphische Programmierung
Was berechnet das folgende C-Programm und wie tut es das?
Was passiert, wenn man den Kreis größer macht?

```
#define _ ++O<OO||O>OO++;
int O=OO,OO=O;main(){o();printf("%1.3f\n",4.*O/OO/OO);}o()
{
            _-_-_-_
       _-_-_-_-_-_-_-_-_
    _-_-_-_-_-_-_-_-_-_-_-_
  _-_-_-_-_-_-_-_-_-_-_-_-_-_
 _-_-_-_-_-_-_-_-_-_-_-_-_-_-_
 _-_-_-_-_-_-_-_-_-_-_-_-_-_-_
_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_
_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_
_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_
_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_
 _-_-_-_-_-_-_-_-_-_-_-_-_-_-_
 _-_-_-_-_-_-_-_-_-_-_-_-_-_-_
  _-_-_-_-_-_-_-_-_-_-_-_-_-_
    _-_-_-_-_-_-_-_-_-_-_-_
        _-_-_-_-_-_-_-_
            _-_-_-_
}
```

Die Originalversion dieses Programm stammt von Brian Westley. Es gewann im International Obfuscated C Code Contest 1988 (`http://reality.sgi.com/csp/ioccc/index.html`) den Preis für das beste Layout.

Das Programm wurde leicht verändert, um es auch mit einem streng dem ANSI-Standard entsprechenden C-Compiler (z.B. dem Gnu C-Compiler) — wenn auch nicht ohne Warnungen — übersetzen zu können. (Unter `http://reality.sgi.com/csp/ioccc/1988/westley.c` ist die Originalversion des Programms verfügbar.)