technische universität
dortmund

Fakultät für Informatik
Lehrstühle 2, 11

Prof. Dr. C. Sohler, A. Krivošija
H. Blom, N. Kriege, H. Timm, B. Zey
http://tiny.cc/dap2p

Sommersemester 2014

DAP2 Praktikum – Blatt 8

Ausgabe: 23. Mai — Abgabe: 02.–06. Juni

Studienleistung (Scheinkriterien)

- Zum Bestehen des Praktikums muss jeder Teilnehmer die folgenden Leistungen erbringen:
  - Es müssen mindestens 50 Prozent der Punkte in den Kurzaufgaben erreicht werden.
  - Es müssen mindestens 50 Prozent der Punkte in den Langaufgaben erreicht werden.
  - Man muss an mindestens 12 Terminen anwesend sein und aktiv mitarbeiten (nach Absprache mit den Tutoren ist es auch möglich, Fehltermine in anderen Gruppen nachzuholen, sofern dort Platz ist).
- An Feiertagen suchen Sie sich bitte einen Ersatztermin, ansonsten wird dies als "nicht erschienen"gewertet.

Hinweis zu den Langaufgaben

Für diese Aufgaben stehen Ihnen ca. zwei Wochen Bearbeitungszeit zur Verfügung. In den Praktika dieser Woche können (und sollen) Sie mit der Bearbeitung beginnen. Den Rest müssen Sie außerhalb des Praktikums erledigen. Neben den Lernräumen können Sie dafür auch die Poolräume (z.B. in einer Freistunde) nutzen, wenn dort gerade keine Veranstaltungen stattfinden. Im Praktikum in zwei Wochen präsentieren Sie Ihrem Tutor dann Ihre Lösungen auf Ihrem Praktikumsrechner im Pool und bekommen Punkte dafür (nach Absprache mit dem Tutor ist es auch möglich, die Lösung auf Ihrem eigenen Rechner zu präsentieren). **Wichtig:** Sie müssen die Aufgaben **vor** Beginn des Praktikums (in dem Sie die Lösungen präsentieren) fertig haben. `Sie dürfen die Langaufgaben in Gruppen von bis zu drei Studierenden bearbeiten und präsentieren.`

Ebenfalls wichtig: Der Quellcode ist natürlich mit sinnvollen Kommentaren und die Schleifen mit zugehörigen Invarianten (inkl. Bereich in dem die Invariante gilt) zu kommentieren.

Langaufgabe 8.1 **(4 Punkte)**

Lernziel: Editierdistanz (Dynamische Programmierung)

Implementieren Sie einen Algorithmus zur Bestimmung der sogenannten minimalen Editierdistanz zwischen zwei Sequenzen von Buchstaben, die auch als "Levenshtein-Distanz" bekannt ist. Unter der minimalen Editierdistanz versteht man die minimal benötigte Anzahl von Einfüge-, Lösch- und Ersetzungs-Operationen einzelner Buchstaben, die benötigt werden, um eine Sequenz von Buchstaben in eine andere Sequenz von Buchstaben zu transformieren.

So ist beispielsweise 4 die minimale Editierdistanz zwischen den beiden Sequenzen `baacda` und `abace`, weil sich `baacda` mit Hilfe von 4 Einfüge-, Lösch- und Ersetzungs-Operationen in `abace` transformieren lässt:

Anzuwendende Operation	Sequenz nach Operation
Füge 'a' an Position 1 ein	`abaacda`
Lösche 'a' an Position 4	`abacda`
Ersetze 'd' durch 'e' an Position 5	`abacea`
Lösche 'a' an Position 6	`abace`

Weitere Erläuterungen zum Problem der Bestimmung der minimalen Editierdistanz sowie der zu implementierende, auf dynamischer Programmierung basierende Algorithmus zur Lösung dieses Problems lassen sich in der Literatur oder durch eine Internet-Recherche finden.

Konkret ist Folgendes zu erfüllen:

- Legen Sie eine Klasse `EditDistance` an, die die Methode `int distance(String a, String b)` enthält.
- Implementieren Sie die Methode `int distance(String a, String b)` mit Hilfe des auf dynamischer Programmierung basierenden Algorithmus zur Bestimmung der minimalen Editierdistanz.
- Fügen Sie weiterhin eine Methode ein, die aus einer gegebenen Datei – in der in jeder Zeile genau eine Sequenz von Buchstaben steht – alle Zeilen ausliest und jede Zeile als eine Sequenz von Buchstaben (d.h., als einen `String`) interpretiert.
- Das Programm soll entweder einen oder zwei Parameter übergeben bekommen:
  - Falls ein Parameter übergeben wird, soll dies der Name einer einzulesenden Datei sein, für die für jede Kombination von jeweils zwei verschiedenen Zeilen/Sequenzen aus dieser Datei die minimale Editierdistanz bestimmt werden soll.
    Beispielaufruf: `java EditDistance datei.txt`
  - Falls zwei Parameter übergeben werden, sollen diese als Sequenzen interpretiert werden und es soll die minimale Editierdistanz für die Transformation der ersten Sequenz (erster Parameter) in die zweite Sequenz (zweiter Parameter) berechnet werden.
    Beispielaufruf: `java EditDistance Informatik Interpolation`

Langaufgabe 8.2 **(4 Punkte)**

Lernziel: Ausgabe der Editieroperationen

Das Programm aus Aufgabenteil 1 soll nun so erweitert werden, dass nicht nur die benötigte Anzahl der Operationen, sondern auch die einzelnen Operationen selbst sowie die Zwischenergebnisse der Transformation ausgegeben werden.

- Schreiben Sie eine Methode `printEditOperations`, die schrittweise die konkreten Operationen und deren jeweilige Kosten ausgibt, die für eine optimale Transformation einer Ausgangssequenz in eine Zielsequenz benötigt werden.
- Dabei soll für jeden (Zwischen-)Schritt die Sequenz mit ausgegeben werden, die nach Anwendung der Operation dieses Schrittes entsteht.
- Die Ausgabe soll dabei so wie in dem folgenden Beispiel formatiert sein:

```
Loesung fuer "baacda" --> "abace" mit Gesamtkosten 4:
=====================================================
1) Kosten 1: Fuege a an Position 1 ein --> abaacda
2) Kosten 0: Ersetze b durch b an Position 2 --> abaacda
3) Kosten 0: Ersetze a durch a an Position 3 --> abaacda
4) Kosten 1: Loesche a an Position 4 --> abacda
5) Kosten 0: Ersetze c durch c an Position 4 --> abacda
6) Kosten 1: Ersetze d durch e an Position 5 --> abacea
7) Kosten 1: Loesche a an Position 6 --> abace
```

- Erweitern Sie Ihr Programm derart, dass als jeweils letzter Parameter "-o" übergeben werden kann. In diesem Fall soll an Stelle der einfachen Ausgabe in Form der minimalen Editierdistanz die detaillierte Ausgabe mit Hilfe der Methode `printEditOperations` erfolgen.
  Beispielaufruf: `java EditDistance datei.txt -o`
  Beispielaufruf: `java EditDistance Informatik Interpolation -o`