Eidgenössische
Technische Hochschule
Zürich

Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Federal Institute of Technology at Zurich

Department Informatik
Markus Püschel
Peter Widmayer
Thomas Tschager
Tobias Pröger
Tomáš Gavenčiak

17. November 2016

Algorithmen & Datenstrukturen — Blatt P9 — HS 16

Abgabe: Bis zum 24. November 2016 um 10 Uhr auf dem Judge (ausschliesslich Quellcode).

Aufgabe P9.1 *Zeilenumbrüche.*

Eines grundlegendes Problem von Textsatzsystemen ist das Verteilen von Wörtern auf mehrere Zeilen und das Verteilen der Zeilen auf Seiten mit dem Ziel ein möglichst schönes und lesbares Layout zu erhalten. Wir beschränken uns hier auf eine (sehr vereinfachte) Variante des ersten Problems, dem Positionieren von Zeilenumbrüchen in einem gegebenen Text.

Der gegebene Text T ist eine Abfolge von Paragraphen $P_1, \ldots, P_k$, welche unabhängig von einander bearbeitet werden. Jeder Paragraph P_i ist eine Liste von n_i Wörtern und hat eine gegebene Seitenbreite $w_i > 0$. Jedes Wort in P_i ist höchstens w_i lang.

Die Ausgabe soll aus demselben Text mit einem oder mehreren aufeinanderfolgenden Wörtern in einer Zeile bestehen, wobei zwei aufeinanderfolgende Wörter in einer Zeile durch genau ein Leerzeichen getrennt werden müssen. Alle Zeichen (inkl. Leerzeichen) haben dieselbe Breite und die Länge $\text{len}(L)$ einer Zeile L ist die Anzahl der Zeichen der Wörter und der dazwischenliegenden Leerzeichen. Zum Beispiel hat die Zeile "`This is an example.`" Länge $4+1+2+1+2+1+8 = 19$.

Das Ziel dieser Aufgabe ist es, den Text so zu formatieren, dass jede Zeile eines Paragraphen P_i höchstens Länge w_i hat, aber gleichzeitig möglichst schön aussieht: Wir möchten also, dass die Länge jeder Zeile möglichst nahe an w_i herankommt. Daher weissen wir jeder Zeile Kosten $(w_i - len(L))^2$ zu, d.h. die quadrierte Anzahl der Leerzeichen, die am Ende eingefügt werden müssten, damit die Zeile genau Länge w_i hat. Zum Beispiel hat eine Zeile mit Länge genau w Kosten 0 und eine Zeile mit nur einem Wort bestehend aus einem einzigen Zeichen hat Kosten $(w_i - 1)^2$. Die letzte Zeile eines Paragraphen ist eine Ausnahme – die Kosten dieser Zeile sind immer 0, da die Länge der letzten Zeile nicht wichtig ist.

Ziel dieser Aufgabe ist es, für jeden Paragraphen die optimalen Zeilenumbrüche zu finden, sodass die Summe der Kosten der resultierenden Zeilen minimiert wird. Beachten Sie dass ein *"Greedy"*-Ansatz, der jede Zeile so lange wie möglich macht bevor ein Zeilenumbruch eingefügt wird, im Allgemeinen nicht optimal bist – siehe das Beispiel unten.

Eingabe Die Eingabe besteht aus mehreren Paragraphen. Die erste Zeile enthält die Ganzzahl $k > 0$, die Anzahl der Paragraphen, die folgen.

Jeder Paragraph P_i ist unabhängig von den anderen und besteht aus mehreren Zeilen: Die erste Zeile enthält die Ganzzahlen $n_i > 0$, die Anzahl der Wörter des Paragraphen, und $w_i > 0$,

die Seitenbreite, durch Leerzeichen getrennt. Die folgenden n_i Zeilen enthalten die Wörter des Paragraphen, ein Wort pro Zeile.

Die Wörter bestehen aus Buchstaben aus dem englischen Alphabet, Zahlen und den folgenden Zeichen[1]: `-.,?!:;'"()[]{}`

Ausgabe Für jeden Paragraphen sollte die Ausgabe aus einer einzigen Zeile bestehen, die die kleinstmöglichen Kosten enthalten.

Bonus Sie erhalten einen Bonuspunkt pro 100 Punkte auf dem Judge (abgerundet). Insgesamt können Sie bis zu 200 Punkte auf dem Judge erhalten. Damit alle Tests auf dem Judge erfolgreich sind, sollte die Laufzeit Ihres Programms in $O(w_1 n_1 + \cdots + w_k n_k)$ liegen (eine ausreichend effiziente Implementierung vorausgesetzt).

Senden Sie Ihr `Main.java` unter folgendem Link ein: `https://judge.inf.ethz.ch/team/websubmit.php?cid=18985&problem=DA_P9.1`. Das Passwort für die Einschreibung ist "`quicksort`".

Beispiele

Eingabe

```
2
9 9
A
B
C
D
E
F
GGGGGGGGG
HHH
I.
8 18
Lorem
ipsum
dolor
sit
amet,
consectetur
adipiscing
elit.
```

Ausgabe

```
32
107
```

Beispiel für eine optimale Formatierung mit gekennzeichneter Seitenbreite und Zeilenkosten.

```
A B C    | 16
D E F    | 16
GGGGGGGGG| 0
```

[1] Die Menge der verwendeten Zeichen sollte für Ihr Programm allerdings nicht relevant sein.

```
HHH I.   | 0 (letzte Zeile)

Lorem ipsum       | 49
dolor sit amet,   | 9
consectetur       | 49
adipiscing elit.  | 0 (letzte Zeile)
```

Hinweise Wir stellen für diese Aufgabe eine Programmvorlage als Eclipse Projektarchiv auf der Vorlesungswebseite zur Verfügung. In der Vorlage wird die Eingabe bereits eingelesen. Das Archiv enthält weitere Testdaten, damit Sie lokal testen können. Ausserdem stellen wir zusätzlich ein `Judge.java` Programm zur Verfügung, das Ihr Programm `Main.java` mit allen verfügbaren Testdaten testet – öffnen und starten sie dazu einfach `Judge.java` in derselben Weise wie Sie `Main.java` starten würden. Die Art und Weise, wie Ihr Programm `Main.java` arbeitet, wird dadurch nicht beeinflusst. `Judge.java` soll lediglich das lokale Testen erleichtern. Die zur Verfügung gestellten Testdaten sind nicht die Testdaten, welche der Judge verwendet, und im Vergleich nicht so umfangreich.

Wir freuen uns über Rückmeldung zum lokalen Judge `Judge.java`.