Eidgenössische
Technische Hochschule
Zürich

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

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

24. November 2016

Algorithmen & Datenstrukturen Lösungen zu Blatt P9 HS 16

Lösung P9.1 *Zeilenumbrüche.*

Wir nutzen für diese Aufgabe wieder Dynamische Programmierung: Für jedes $i \in \{0, \dots, n\}$ seien m_i die Gesamtkosten, wenn die letzte Zeile genau vor Wort w_i umgebrochen wird (wir ignorieren den Spezialfall des letzten Wortes eines Paragraphen). Wir setzen $m_0 = 0$, da ein Umbruch vor dem Wort w_0 der Anfang des Textes ist. Beachten Sie, dass die eigentlichen Buchstaben der Wörter nicht von Bedeutung sind, sondern lediglich die Wortlänge.

Wir können m_i berechnen, falls wir alle vorherigen Werte von $m_{j<i}$ kennen, indem wir uns überlegen, wie eine Lösung, die genau vor w_i umgebrochen wird, aussehen kann: Für ein $x > 0$ wird die Lösung aus der besten Lösung mit einem Zeilenumbruch genau vor w_{i-x} bestehen und dann aus den x Wörtern $w_{i-x}, w_{i-x+1}, \dots, w_{i-1}$ in einer Zeile. Die Kosten der besten Lösung mit einem Zeilenumbruch genau vor w_{i-x} sind bereits bekannt (der Wert von m_{i-x}) und man kann die Kosten für eine Zeile mit den Wörtern $w_{i-x}, w_{i-x+1}, \dots, w_{i-1}$ leicht berechnen. Wir berechnen nun die Kosten für alle möglichen Werte von $0 < x \leq i$, sodass die letzte Zeile höchstens Länge w hat, und setzen m_i auf die minimalen Kosten über alle solche x. Wir müssen bis zu $\mathcal{O}(w)$ Werte von x überprüfen, da $x \leq (w+1)/2$ (es gibt höchstens $(w+1)/2$ durch Leerzeichen getrennte Wörter in einer Zeile).

Für die letzte Zeile eines Paragraphen müssen wir lediglich die Berechnung von m_n ändern: Wir betrachten wiederum alle möglichen Werte von x (sodass die letzte Zeile höchstens Länge w hat) und verwenden das Minimum von m_{n-x} ohne aber die Kosten für die letzte Zeile hinzuzufügen.

Wir haben also eine Lösung mit Laufzeit in $\mathcal{O}(nw^2)$, da wir $\mathcal{O}(w)$ Werte von x für jedes der n Wörter betrachten und für jeden Wert in $\mathcal{O}(w)$ Zeit die Kosten für die Zeile mit den Wörtern $w_{i-x}, w_{i-x+1}, \dots, w_{i-1}$ berechnen können.

Wir können diese Lösung verbessern indem wir die Summe der Wortlängen $s_i = \sum_{j=0}^{i} |w_j|$ vorausberechnen. Dann ist die Länge der Zeile mit Wörtern $w_a, \dots, w_b$ genau $s_b - s_a + (b - a - 1)$ (der letzte Term zählt die Leerzeichen). Eine andere Lösung startet mit $x = 1$. Wenn x inkrementiert wird, merken wir uns in jedem Schritt die bisherige Länge der letzten Zeile und inkrementieren sie um die Länge des zusätzlichen Wortes. Die Kosten können dann in Laufzeit $\mathcal{O}(1)$ berechnet werden.

Diese beiden Lösungen haben eine Laufzeit in $\mathcal{O}(nw)$.

Lösungen

Auf der Vorlesungswebseite finden Sie eine Lösung, welche eine Laufzeit von $\mathcal{O}(nw)$ benötigt. Die Lösung enthält weitere Kommentare zur Implementierung.

Daten

Die Testdaten enthalten grossteils zufällige Wortlängen mit w zwischen 20 und 1000 und n zwischen 100 und 60000 in verschiedenen Kombinationen. Jeder Test im Judge beinhaltet 1-3 kleine Spezialfälle, welche dem Beispiel und den zur Verfügung gestellten Testdaten ähneln.

Hinweise zu den Abgaben. Manche Abgaben verwendeten ein zweidimensionales Array für Teilprobleme der Art $m_{i,j}$ ="minimale Kosten für Wörter $w_i, \ldots, w_{i+j}$" oder ähnliche Ansätze. Obwohl solche Ansätze grundsätzlich funktionieren können, hat ein solches Array $n \times n$ Einträge und passt für grosse Werte von n nicht in den Speicher. Eine kleine Verbesserung wäre, ein Array der Grösse $n \times w$ zu verwenden für Wörter in einer Zeile. Einige dieser Lösungen könnten Laufzeit $\mathcal{O}(nw^2)$ erreichen oder sogar $\mathcal{O}(nw)$. Diese Ansätze sind allerdings komplizierter als die vorgeschlagene Lösung mit einem eindimensionalen Array.