Datenstrukturen und Algorithmen, SS 2024
Prof. Dr. P. Rossmanith
D. Cloerkes, M. Gehnen, D. Mock, N. Speetzen

Theoretische Informatik | RWTH AACHEN UNIVERSITY

Abgabe: 12.07.2024

Übungsblatt 12

Aufgabe T41

Gegeben ist ein Baum mit n Knoten. Jeder Knoten v hat ein nicht-negatives Gewicht $w(v)$. Entwerfen Sie einen effizienten Algorithmus, welcher eine unabhängige Knotenmenge (paarweise keine Kanten, auch Independent Set genannt) mit maximalem Gewicht findet.

Das Gewicht einer Knotenmenge ist die Summe der Gewichte der einzelnen Knoten.

Wie schnell ist Ihr Algorithmus und wieviel Speicher benötigt er?

Aufgabe T42

Gegeben ist ein sehr langer Text w und sehr viele kurze Wörter $u_1, \ldots, u_n$. Ziel ist es herauszufinden, wie oft jedes der Wörter u_i im Text w vorkommt.

Entwerfen Sie einen effizienten Algorithmus, der diese Aufgabe in einer Laufzeit von $O\big((|w| + |u_1| + \cdots + |u_n|)\log(|w|)\big)$ Schritten löst.

Aufgabe T43

```
Gegeben sei ein Text, den wir uns als von Leerzeichen und Zeilenumbrüchen
getrennte Folge von Wörtern vorstellen. Der Text soll nun linksbündig
auf einer Schreibmaschine getippt werden. (Auf einer Schreibmaschine hat
jeder Buchstabe die gleiche Breite, wie man es auch von Texteditoren
kennt.) Dabei sollen aber keine Wörter getrennt werden, sodass am rechten
Seitenrand mit Leerzeichen aufgefüllt werden muss. Der Seitenrand befindet
sich nach der 75. Position. Die ideale Länge einer Zeile ist also auch
75. Ist eine Zeile aber am rechten Rand mit k Leerzeichen aufgefüllt,
dann ist ihre Hässlichkeit genau k*k und die Hässlichkeit eines gesetzen
Textes ist die Summe der Hässlichkeiten aller Zeilen. Entwerfen Sie
einen Algorithmus, der die Wörter so auf die Zeilen verteilt, dass die
Hässlichkeit minimal ist. Wie könnte man das Problem alternativ lösen?
```

Die Hässlichkeit des obigen Textes ist 187 und besser geht es auch nicht.

Aufgabe H38 (10 Punkte)
Konstruieren Sie einen optimalen Suchbaum für die Schlüssel A, B, C und D. Auf diese wird mit den Wahrscheinlichkeiten 0.2, 0.3, 0.1 und 0.4 zugegriffen.
Erstellen Sie dazu die Tabellen für $w_{i,j}$ und $e_{i,j}$.

Aufgabe H39 (20 Punkte)
Sie möchten in einer längeren Zeichenkette w ein Wort u finden, so dass uuu ein Unterwort von w bildet. Insbesondere interessiert es Sie, wie lang so ein u maximal sein kann.
Entwerfen Sie einen halbwegs effizienten Algorithmus, der dieses Problem lösen kann und implementieren Sie ihn in einer vernünftigen Programmiersprache. Ihr Verfahren sollte so effizient sein, dass es mit $|w| = 50.000$ fertig wird, ohne dass man allzu lange warten muss.
Erläutern Sie Ihre Vorgehensweise und reichen Sie Ihr Programm ein. Im Moodle-Lernraum finden sie die drei Dateien `input1.txt`, `input2.txt` und `input3.txt`. Jede umfasst zehn Beispieleingaben in den ersten zehn Zeilen, d.h. in jeder dieser Zeilen steht ein w. Lösen Sie diese Beispiele. Für die volle Punktzahl sollten Sie wenigstens die ersten beiden Beispieleingaben vollständig lösen. Geben Sie die entsprechenden maximalen Längen Ihrer Lösungen an.

Aufgabe H40 (10 Punkte)
Entwerfen Sie einen Algorithmus, der die Swap-Edit-Distance zwischen zwei gegebenen Strings u und v berechnet. Diese ist definiert als die minimale Anzahl von Swap-Edit-Operationen, die man benötigt, um aus dem String u den String v zu machen. Es gibt drei Swap-Edit-Operationen:

1. Einfügen: Man kann an einer beliebigen Stelle einen Buchstaben einfügen.
2. Löschen: Man kann an einer beliebigen Stelle einen Buchstaben löschen.
3. Tauschen: Man kann zwei benachbarte Buchstaben tauschen.

Beispiel: Die Swap-Edit-Distance von $u = abba$ und $v = ababb$ beträgt 2, etwa indem man zuerst die beiden letzten Buchstaben von u vertauscht und dann am Ende ein b einfügt.