

Übungen zur Vorlesung
Effiziente Algorithmen
SS 2002
Blatt 6

Aufgabe 6.1 (4 Punkte)

Für das Stringmatchingproblem sind auch randomisierte Algorithmen mit erwarteter Laufzeit $O(n + m)$ bekannt. Ein Ansatz ist in den Abschnitten 7.4 und 7.6 des Buchs „Randomized Algorithms“ von Motwani und Raghavan, Cambridge University Press (1995), beschrieben. Im letzten Absatz auf Seite 171 wird angerissen, wie ein zuvor beschriebener, so genannter „Monte-Carlo-Algorithmus“, d. h. ein nicht unbedingt irrtumsfreier randomisierter Algorithmus, für das Stringmatchingproblem in einen „Las-Vegas-Algorithmus“, d. h. einen irrtumsfreien Algorithmus, umgewandelt werden kann.

Beschreiben Sie den erwähnten Las-Vegas-Algorithmus mit eigenen Worten. Geben Sie die wesentlichen Ideen des Algorithmus wieder und erläutern Sie, warum er funktioniert. Können Sie eine obere Schranke für die Rechenzeit des Algorithmus, die für jede Ausprägung der dort verwandten Zufallszahlen gültig ist, angeben?

Aufgabe 6.2 (4 Punkte)

Wir betrachten das NP-vollständige Problem PARTITION. Dieses besteht darin, für vorgegebene natürliche Zahlen b_1, \dots, b_n zu entscheiden, ob es eine Teilmenge $I \subseteq \{1, \dots, n\}$ gibt, sodass $\sum_{i \in I} b_i = \sum_{i \notin I} b_i$ gilt. Geben Sie einen so genannten pseudopolynomiellen Algorithmus für PARTITION an, d. h. einen Algorithmus, dessen Rechenzeit polynomiell in den Zahlen n und $\sum_{i=1}^n b_i$ ist. (Damit zeigen Sie, dass PARTITION vermutlich nicht stark NP-vollständig ist.) Ihr Algorithmus soll eine Teilmenge I mit den geforderten Eigenschaften berechnen, falls mindestens eine solche existiert, und sonst „Nein“ ausgeben.

Aufgabe 6.3 (4 Punkte)

Als ebenfalls in der Bioinformatik wichtiges Problem gilt *Longest Common Subsequence (LCS)*. Ein String (z_1, \dots, z_k) heißt Subsequenz eines Strings (x_1, \dots, x_n) , wenn eine streng monoton wachsende Folge i_1, \dots, i_k von Indizes existiert, sodass $z_j = x_{i_j}$ für $j \in \{1, \dots, k\}$ gilt. Bei LCS besteht die Aufgabe darin, für zwei Strings $x = (x_1, \dots, x_n)$ sowie $y = (y_1, \dots, y_m)$ die maximale Länge k eines Strings (z_1, \dots, z_k) zu finden, der sowohl Subsequenz von x als auch Subsequenz von y ist.

Entwerfen Sie mit dem Ansatz der dynamischen Programmierung einen effizienten Algorithmus, der das Problem LCS löst.

Aufgabe 6.4 (4 Punkte)

Gegeben seien Daten $x_1 < \dots < x_n$ sowie Abfragewahrscheinlichkeiten p_1, \dots, p_n für diese Daten ($\sum_{i=1}^n p_i = 1$). Für einen Suchbaum T bezeichne $\text{depth}_T(x_i)$ die Tiefe des Knotens, an dem das Datum x_i in T abgespeichert ist. Wir bewerten die Zeit für die Suche nach x_i durch die Anzahl der Knoten auf dem Weg von der Wurzel zum Knoten, an dem x_i abgespeichert

ist, also durch $\text{depth}_T(x_i) + 1$. Die erwartete Suchzeit für T ist dann die Summe der durch die Abfragewahrscheinlichkeiten gewichteten Suchzeiten:

$$E_T := \sum_{i=1}^n p_i \cdot (\text{depth}_T(x_i) + 1).$$

Geben Sie einen Algorithmus an, der einen binären Suchbaum mit minimaler erwarteter Suchzeit in Zeit $O(n^3)$ berechnet.

Hinweis: Die Aufgabe kann als ein Problem vom „Intervalltyp“ aufgefasst werden.