Höhere Algorithmik, WS 2014/15 — 6. Übungsblatt

20 Gummipunkte. Schriftliche Einzelabgabe bis Dienstag, 25. 11. 2014, bzw. Mittwoch

31. Bestimmen der Mehrheit, 0 Punkte

Untersuchen Sie den nebenstehenden Teile-und-herrsche-Algorithmus zur Bestimmung der Mehrheit (Aufgabe 23 vom 4. Übungsblatt). Beweisen Sie, dass der Algorithmus für jede Eingabefolge das Mehrheitselement findet, sofern es existiert, oder finden Sie ein Gegenbeispiel. Kann man sagen, welche Bedeutung der zweite zurückgegebene Parameter hat? Wie ist das Ergebnis, wenn es nur zwei KandidatiX zur Auswahl gibt?

def $Mehrheit(a_1, \ldots, a_n)$:
  if $n = 1$: **return** $(a_1, 1)$
  $(x, i) := Mehrheit(a_1, \ldots, a_{\lfloor n/2 \rfloor})$
  $(y, j) := Mehrheit(a_{\lfloor n/2 \rfloor + 1}, \ldots, a_n)$
  if $x = y$: **return** $(x, i + j)$
  if $i > j$: **return** $(x, i - j)$
  else: **return** $(y, j - i)$

32. Rekursionen und Mediansuche, 0 Punkte

Der deterministische Algorithmus zur Mediansuche aus der Vorlesung zerteilt die Eingabe in Fünferblöcke und führt zur Rekursionsgleichung

$$T(n) = T(\lfloor \tfrac{n}{5} \rfloor) + T(\lfloor \tfrac{7n}{10} \rfloor + 3) + O(n),$$

für hinreichend große n.

(a) Wie sieht die Rekursionsformel aus, wenn die Eingabe in Blöcke der Größe k aufgeteilt wird, für allgemeine ungerade k? Vernachlässigen Sie dabei zunächst den konstanten Restterm (+3 in der obigen Rekursion), der durch die Teilbarkeit entsteht.

(b) Für welche Werte von k ist die Laufzeit linear?

(c) Bestimmen Sie die Laufzeit für alle ungeraden $k > 1$, für die die Laufzeit *nicht* linear ist. Nehmen Sie der Einfachheit halber für diese Teilaufgabe zunächst an, dass n eine Potenz von k ist.

33. Medianauswahl, 0 Punkte

(a) Wieviele Vergleiche benötigt Sortieren durch Einfügen für 5 Elemente?

(b) Wieviele Vergleiche benötigt Mergesort für 5 Elemente?

(c) Zeigen Sie, wie man den Median von 5 Elementen mit höchstens 7 Vergleichen bestimmen kann. (Für jeden zusätzlich eingesparten Vergleich gibt es 1 Luftpunkt;-)

34. Fibonacci, 0 Punkte

Bestimmen Sie die Anzahl der rekursiven Aufrufe der nebenstehenden Funktion $f(n)$ in Abhängigkeit von n.

def $f(n)$:
  if $n \leq 1$: **return** 1
  elseif $n = 2$: **return** 3
  else: **return** $2 \cdot f(n-1) + 3 \cdot f(n-2)$

35. Wechselgeld, 10 Punkte

Entwerfen Sie einen effizienten Algorithmus, der berechnet, wie ein gegebener Geldbetrag mit der kleinstmöglichen Anzahl von Münzen ausgezahlt werden kann, und analysieren Sie die Laufzeit und den Speicherbedarf. Die Eingabe ist die zu zahlende Summe sowie die Werte der zur Verfügung stehenden Münzen. Nehmen Sie an, dass von jeder Münzart ein genügend großer Vorrat vorhanden ist.

Beispiele:

```
64 1 2 5 10 20 50 100 200   → Ausgabe:  4 Münzen: 64=50+10+2+2
22 12 1 18 3 5              → Ausgabe:  3 Münzen: 22=12+5+5
11 7 3 10 99                → Ausgabe:  keine Lösung
```

36. Wechselgeld und Rucksack, 0 Punkte

    (a) Wenn es keine Münze mit Wert 1 gibt, dann ist das Wechselgeldproblem der vorigen Aufgabe nicht immer lösbar.
    Nehmen Sie an, Sie hätten ein Programm zur Lösung des (Standard-)Rucksackproblems, also des 0-1-Rucksackproblems, wo jeder Gegenstand nur einmal genommen werden kann. Wie könnten Sie mit Hilfe dieses Programms testen, ob das Wechselgeldproblem eine Lösung hat?

    (b) Wie könnten Sie mit Hilfe dieses Programms das Wechselgeldproblem von Aufgabe 35 lösen?

37. Exponentielle Suche, 0 Punkte

    In einem sortierten Feld $(a_1, a_2, \ldots, a_n)$ soll ein Wert b gefunden werden.

    (a) Zeigen Sie, dass man b in $O(\log i)$ Zeit finden kann, wenn b das i-kleinste Element ist. Man vergleicht dazu b mit den Elementen an Position 1,2,4,8,..., und am Schluss fährt man mit binärer Suche fort.

    (b) Nehmen wir an, wir haben eine Anfangsschätzung, dass sich b ungefähr an Position j befindet. Zeigen Sie, wie man b in $O(\log(2 + |i - j|))$ Vergleichen findet.

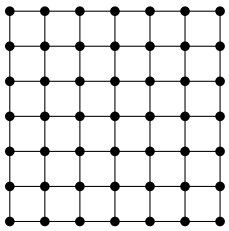
38. Beste unabhängige Knotenmenge in Bäumen, 10 Punkte

    (a) Gegeben ist ein binärer Baum mit beliebigen Werten in den Knoten. (Es muss kein Suchbaum sein.) Eine *unabhängige Knotenmenge* ist eine Teilmenge der Knoten, die keinen Knoten gemeinsam mit seinem Elternknoten enthält. Entwerfen Sie einen effizienten Algorithmus, der die unabhängige Knotenmenge mit größtem Gesamtgewicht bestimmt, und analysieren Sie die Laufzeit.

    (b) Geben Sie an, welche Merkmale des *teile-und-herrsche*-Prinzips und welche Merkmale der *dynamischen Programmierung* Ihr Algorithmus hat.

39. Lokales Maximum in Gittergraphen, 0 Punkte

    Gegeben ist ein $(n \times n)$-Gittergraph, in dessen Knoten Werte stehen. Wie in Aufgabe 25 ist ein lokales Maximum ein Knotenwert, der mindestens so groß ist wie die Werte in den Nachbarknoten. Wieviele Knotenwerte muss man im schlimmsten Fall anschauen, um eine lokales Maximum zu finden?



ein 7×7-Gitter

40. Paarweise Kommunikation in Runden. Knobelaufgabe, 0 Punkte

    Acht verteilte Sensoren messen jeweils für sich die Niederschlagsmenge. Am Ende des Tages um 24 Uhr treten sie miteinander in Kontakt, mit dem Ziel, dass die gesamte Niederschlagsmenge, also die Summe der Messwerte, *allen* Sensoren bekannt wird. Sie können dazu paarweise miteinander kommunizieren: Zwei Sensoren können eine Punkt-zu-Punkt-Funkverbindung aufbauen und dann beliebig viele Daten austauschen. Dieser Austausch dauert eine Minute, inklusive Auf- und Abbau der Verbindung und Sicherung des Austauschs. Die betroffenen Sensoren können in dieser Zeit nicht mit anderen Sensoren kommunizieren, aber andere Sensoren können gleichzeitig paarweise untereinander Daten austauschen.

    (a) Wieviele Kommunikationsrunden (Minuten) sind zum vollständigen Austausch der Informationen notwendig?

    (b) Wieviele Runden braucht man für 5 Sensoren?

    (c) Wieviele Runden braucht man für 7 Sensoren? Für 10 Sensoren?