Algorithmen und Berechnungskomplexität I, WS 12/13
Aufgabenblatt 5
Universität Bonn, Institut für Informatik, Abteilung I

- *Die Lösungen können bis Dienstag, 20.11., 12:15 Uhr in den Postkasten im AVZ III eingeworfen werden.*

Aufgabe 17: Platzwahl in Seminarvorträgen (4 Punkte)

Bei einem gut besuchten (alle Plätze sind besetzt) Seminarvortrag haben sich die Zuhörer rücksichtsvollerweise so platziert, dass kein Vordermann größer ist als der hinter ihm Sitzende (in jeder Reihe sind gleich viele Plätze vorhanden). Damit hat jeder einen hervorragenden Blick nach vorn. An der linken Wand befindet sich nun eine ebenfalls beim Vortrag benutzte Tafel, die unter Umständen immer noch von größeren Personen verdeckt wird. Deshalb verständigen sich alle Teilnehmer darauf, sich jeweils innerhalb einer Reihe aufsteigend nach der Körpergröße von links nach rechts zu platzieren. Jetzt kann jeder problemlos nach links schauen, aber ist für alle der Blick nach vorne noch frei? Beweisen Sie Ihre Vermutung!

Aufgabe 18: Datenstrukturen: Bäume (4 Punkte)

Sei B ein Binärbaum der Höhe h. Wie in der Vorlesung definiert, sei die Höhe eines Baums die maximale Tiefe über alle seine Knoten. Der Klarheit halber sei noch einmal definiert: Die Wurzel liege in Tiefe 0. Zeigen Sie via Induktion: In Tiefe i gibt es maximal 2^i viele Knoten.

Bitte wenden!

Aufgabe 19: Datenstrukturen: Stacks (4 Punkte)

a) Wie kann ein Stack durch eine einfach verkettete Liste realisiert werden, so dass die Operationen PUSH und POP konstante Zeit ($O(1)$) benötigen?

b) Beschreiben Sie ein Verfahren, um mit Hilfe *nur eines* n-elementigen Arrays $A[1, \ldots, n]$ *zwei* Keller zu implementieren. Die Fehlermeldung „Keller voll“ soll erst dann erscheinen, wenn die Gesamtzahl der Elemente in beiden Kellern zusammen den Wert n übersteigt. Dabei sollen die Operationen PUSH(*keller*, *element*) und POP(*keller*) mit *keller* $\in \{1, 2\}$ in konstanter Zeit laufen.

Aufgabe 20: Huffman-Codes (4 Punkte)

Ein *Huffman Code* ist ein optimaler Präfixcode, der durch den folgenden Algorithmus berechnet werden kann (siehe auch im Buch von Cormen):

Algorithmus 1 HUFFMAN(C)

$n := |C|$;
$Q := C$;
for $i = 1$ **to** $n - 1$ **do**
  Erzeuge einen neuen Knoten z;
  $z.left := x :=$ entnimm aus Q das Element mit der geringsten Häufigkeit;
  $z.right := y :=$ entnimm aus Q das Element mit der geringsten Häufigkeit;
  $z.freq := x.freq + y.freq$;
  Füge z in Q ein;
end for
return Element aus Q mit geringster Häufigkeit.

Der Algorithmus erzeugt für einen Text über einer Symbolmenge C einen Binärbaum T. Dabei bezeichne $c.freq$ die Häufigkeit des Symbols c im Text. Aus dem Baum ist der Code folgendermaßen abzulesen: Die Blätter des Baumes enthalten die zu codierenden Symbole. Jede Kante zu einem linken Nachfolger wird mit einer 0 beschriftet, jeder Kante zu einem rechten Nachfolger

mit einer 1. Das Codewort für ein Symbol s ergibt sich dann aus der Konkatenation der Kantenbeschriftungen auf dem Pfad von der Baumwurzel zu dem Blatt, das zu s korrespondiert.

Der Algorithmus ist ein gutes Beispiel für die Nützlichkeit von Datenstrukturen wie Queues bzw. Heaps, die ein schnelles Auffinden des kleinsten Elements einer Menge ermöglichen.

a) Angenommen, der Algorithmus HUFFMAN wurde mit dem Ziel entworfen Symbolen mit hoher Häufigkeit einen möglichst kurzen Huffmann-Code zuzuweisen: welches der Paradigmen zum Algorithmenentwurf, die Sie bisher in der Vorlesung kennengelernt haben, wurde hier verwendet?

b) Erzeugen Sie einen Huffman-Code des folgenden Texts:

> In the beginning the Universe was created. This has made a lot of people very angry and been widely regarded as a bad move.

Gehen Sie dabei davon aus, dass das Alphabet nur genau die Zeichen enthält, die auch im Text auftreten, inklusive Leerzeichen. Ignorieren Sie Groß-/Kleinschreibung.

Hinweis: Erstellen Sie als erstes eine Häufigkeitstabelle, in der Sie für jedes Symbol angeben, wie oft es im Text vorkommt.