

Übungen zur Vorlesung  
**Datenstrukturen, Algorithmen und Programmierung 2 (DAP2)**  
 Sommersemester 2007

Blatt 4

**Aufgabe 4.1 (5 Punkte)**

Es sei die baumorientierte Datenstruktur für Union/Find-Befehle gegeben. Betrachte den Fall, dass ein vollständiger ausgeglichener Binärbaum der Tiefe  $d$  gegeben ist, dessen Blätter von links nach rechts die Nummern 1 bis  $n = 2^d$  haben (so ein Baum kann nicht durch eine Folge von Union- Befehlen entstehen).

Welche Kosten hat die Befehlsfolge  $\text{FIND}(1), \text{FIND}(2), \dots, \text{FIND}(n)$  für diesen Baum? Gib die Kosten für beide Varianten an, d.h. mit und ohne Pfadkomprimierung.

Hinweis: Bei der Betrachtung der Pfadkomprimierung kann die Buchhalter-Methode hilfreich sein.

**Aufgabe 4.2 (5 Punkte)**

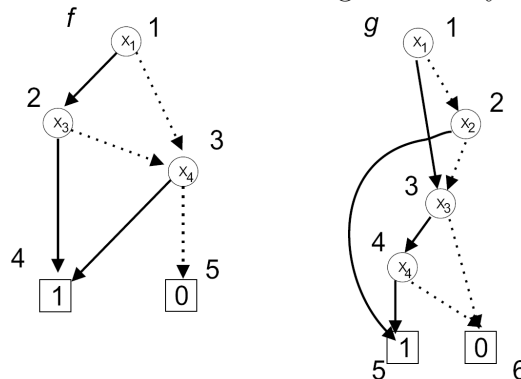
In der Vorlesung wurde darauf hingewiesen, dass die Größe eines  $\pi$ -OBDDs von der Variablenordnung  $\pi$  abhängt. Sichtbar wird dies z.B. bei der disjunkten quadratischen Form:

$$DQF(x_1, x_2, x_3, x_4, x_5, x_6) = (x_1 \wedge x_2) \vee (x_3 \wedge x_4) \vee (x_5 \wedge x_6).$$

Bilde bitte zwei möglichst kleine OBDDs für diese Funktion, einmal mit der Variablenordnung  $\pi_1 = (x_1, x_2, x_3, x_4, x_5, x_6)$  und einmal mit  $\pi_2 = (x_1, x_3, x_5, x_2, x_4, x_6)$ . Welches OBDD ist größer? Hätte man dies schon an der Funktion erkennen können? Begründe.

**Aufgabe 4.3 (5 Punkte)**

Sei  $\pi = (x_1, x_2, x_3, x_4)$ . Betrachte die folgenden  $\pi$ -OBDDs für Funktionen  $f$  und  $g$  (die Knotennummern dienen nur zur leichteren Beschreibung bei der Synthese):



Bestimme das reduzierte  $\pi$ -OBDD für die Funktion  $h = f \oplus g$  mittels des Synthesealgorithmus aus der Vorlesung (dabei sei  $\oplus$  das Exklusiv-Oder, d. h.  $x_1 \oplus x_2 = 1$  genau dann, wenn  $x_1 + x_2 = 1$ ). Gib dabei die Einträge der Computed- und Unique-Table an und beschreibe, wie der Algorithmus auf diesem Beispiel arbeitet.

**Aufgabe 4.4 (5 Punkte)**

Betrachte für geschlossenes Hashing ein Array der Größe  $M = 19$ , in dem die Positionen 2, 5, 6, 9, 10, 11, 12 und 17 belegt sind. Berechne für alle  $x \in \{5, 9, 11, 17\}$  die Plätze im Array, die bei quadratischem Sondieren, multiplikativem Sondieren bzw. doppeltem Hashing sondiert werden. Dabei ist keine Hintereinanderausführung dieser Operationen gemeint und es sollen die im Skript genannten Hashfunktionen für die jeweiligen Sondierungsstrategien benutzt werden.