Lösungsskizzen zur Prüfung

(1) Laufzeit des ursprünglichen Algorithmus: $2n^3$.
Mögliche Verbesserungen der Laufzeit durch:

a) $\frac{2n^3}{6} = \frac{n^3}{3}$

b) $2n^2$

c) $3n + 2(\frac{n}{2})^3 = 3n + \frac{n^3}{4}$

Für welche Wertebereiche von n ist welche Optimierung besser?

a)=b):
$\frac{n^3}{3} = 2n^2$
$n_1 = 6$ ($n = 0$ ist nicht von Bedeutung)

a)=c):
$\frac{n^3}{3} = 3n + \frac{n^3}{4}$
$n^2 = 36, n_2 = 6$ ($n = 0$ und $n = -6$ sind nicht von Bedeutung)

b)=c):
$2n^2 = 3n + \frac{n^3}{4}$
$n^2 - 8n + 12 = 0$
$n_3 = 2, n_4 = 6$ ($n = 0$ ist nicht von Bedeutung)

Das Einsetzen der gefundenen Werte in obige Gleichungen ergibt das Ranking für die verschiedene Bereich von n:

$n = 1 \ldots$ a) besser als b) besser als c)
$n = 2 \ldots$ a) besser als b) gleich wie c)
$n = 3, 4, 5 \ldots$ a) besser c) besser als b)
$n = 6 \ldots$ a) gleich wie b) gleich wie c)
$n \geq 6 \ldots$ b) besser als c) besser als a)

Asymptotisch ($n \to \infty$) haben a) und c) eine Laufzeit von $O(n^3)$, b) hingegen hat eine Laufzeit von $O(n^2)$. D.h. für grössere Datenmengen ist in jedem Fall Variante b) zu bevorzugen! Daher fällt die Wahl auf Optimierung b).

(2)

(a) Richtig. Z.B. trifft dies auf alle Sortieralgorithmen zu, die wir in der Vorlesung kennengelernt haben: $O(n\ log\ n) \in O(n^2) \in O(n^4)$.

(b) Falsch. Die Interpolationssuche findet einen gesuchten Wert auch schon in $O(1)$ Zeit, z.B. wenn die gesuchten Werte wirklich exakt der Annahme der linearen Verteilung entsprechen.

(c) Falsch. Man muß sich nur einen nach rechts entarteten Baum mit ungerader Anzahl von Knoten vorstellen. Dessen symmetrische Reihenfolge hat nicht die Wurzel in der Mitte.

(d) Richtig. Z.B. haben die folgenden zwei Bäume eine idente SR: (1) B ... Wurzel, A ... linker Sohn, C ... rechter Sohn. (2) A ... Wurzel, B ... rechter Sohn, C ... rechter Sohn von B.

(e) Falsch. Wiederum hilft bei der Überlegung die Betrachtung eines entarteten Baumes.

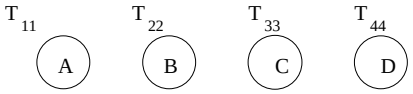
(3)

(a) Siehe Skriptum Seiten 88-89.

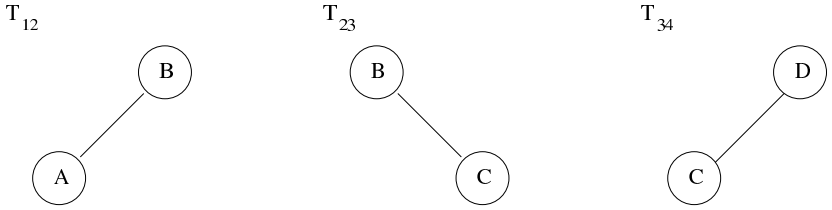
(b) Dynamisches Programmieren löst Probleme durch Kombination der Lösungen von Teilproblemen. Im Gegensatz zu Divide-and-Conquer sind die Teilprobleme dabei im Allgemeinen nicht unabhängig. Weiters wird im Gegensatz zum rekursiven D&C-Ansatz die Lösung für jedes Teilproblem gespeichert. Tritt also ein Teilproblem in mehreren "übergeordneten" Problemen auf, so wird es nur einmal berechnet und später lediglich wieder auf die schon berechnete Lösung zugegriffen. Bei D&C würde das Teilproblem dabei beliebig oft erneut berechnet werden!

(c) Berechnung siehe Skriptum Seite 89. Die Ergebnisse lauten:

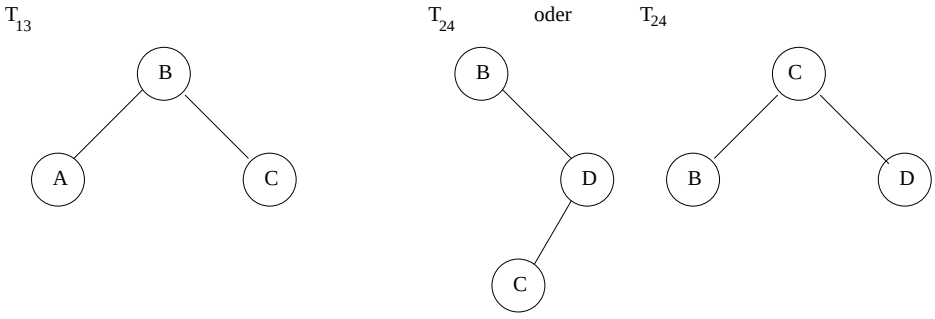
1 Knoten: $c_{11} = 1$, $c_{22} = 4$, $c_{33} = 2$, $c_{44} = 3$



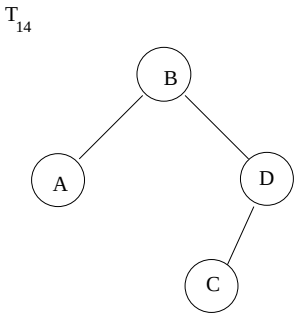
2 Knoten: $c_{12} = 6$, $c_{23} = 8$, $c_{34} = 7$



3 Knoten: $c_{13} = 10$, $c_{24} = 16$



4 Knoten: $c_{14} = 18$



(4)

a) Nein, es existiert kein präfix-freier Code mit Codewortlängen 2,2,3,3,3,4,4,5. Dies folgt z.B. aus folgender Beobachtung: Es gibt 2^k verschiedene präfix-freie Codes der Codewortlänge k. D.h. ein Codewort der Länge k ist ein $\frac{1}{2^k}$ aller möglichen Codes. Die Summe aller Anteile der verschiedenen Zeichen darf nie grösser als 1 werden! In diesem Fall:

$$\frac{1}{2^2} + \frac{1}{2^2} + \frac{1}{2^3} + \frac{1}{2^3} + \frac{1}{2^3} + \frac{1}{2^4} + \frac{1}{2^4} + \frac{1}{2^5} =$$

$$= \frac{2}{4} + \frac{3}{8} + \frac{2}{16} + \frac{1}{32} =$$

$$= \frac{16 + 12 + 4 + 1}{32} =$$

$$= \frac{33}{32} > 1$$

Hinweis: Es reicht nicht, einen Codebaum zu zeichen bei dem es sich nicht ausgeht und zu sagen es geht nicht. Das ist kein Beweis! Man müsste schon alle möglichen Codebäume zeichen oder sehr gut und schlüssig argumentieren warum es anhand dieses einen Baumes für keinen gehen kann. Das Finden eines negativen Beispieles ist kein Beweis für die Nicht-Existenz!

b) Es existiert ein präfix-freier Code mit den angegebenen Codewortlängen. Die Prüfung anhand der Anteile ergibt:

$$\frac{1}{2^1} + \frac{1}{2^2} + \frac{1}{2^3} + \ldots + \frac{1}{2^{2000}} + \frac{1}{2^{2001}} =$$

$$\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \ldots + \frac{1}{2^{2000}} + \frac{1}{2^{2001}} < \sum_{i=1}^{\infty} \frac{1}{2^i} = 1$$

Hinweis: In diesem Fall reicht es auch einen Code zu beschreiben der die Anforderungen erfüllt oder einen Codebaum zu zeichnen. Das Finden eines positiven Beispieles ist ein Beweis für die Existenz! z.B: