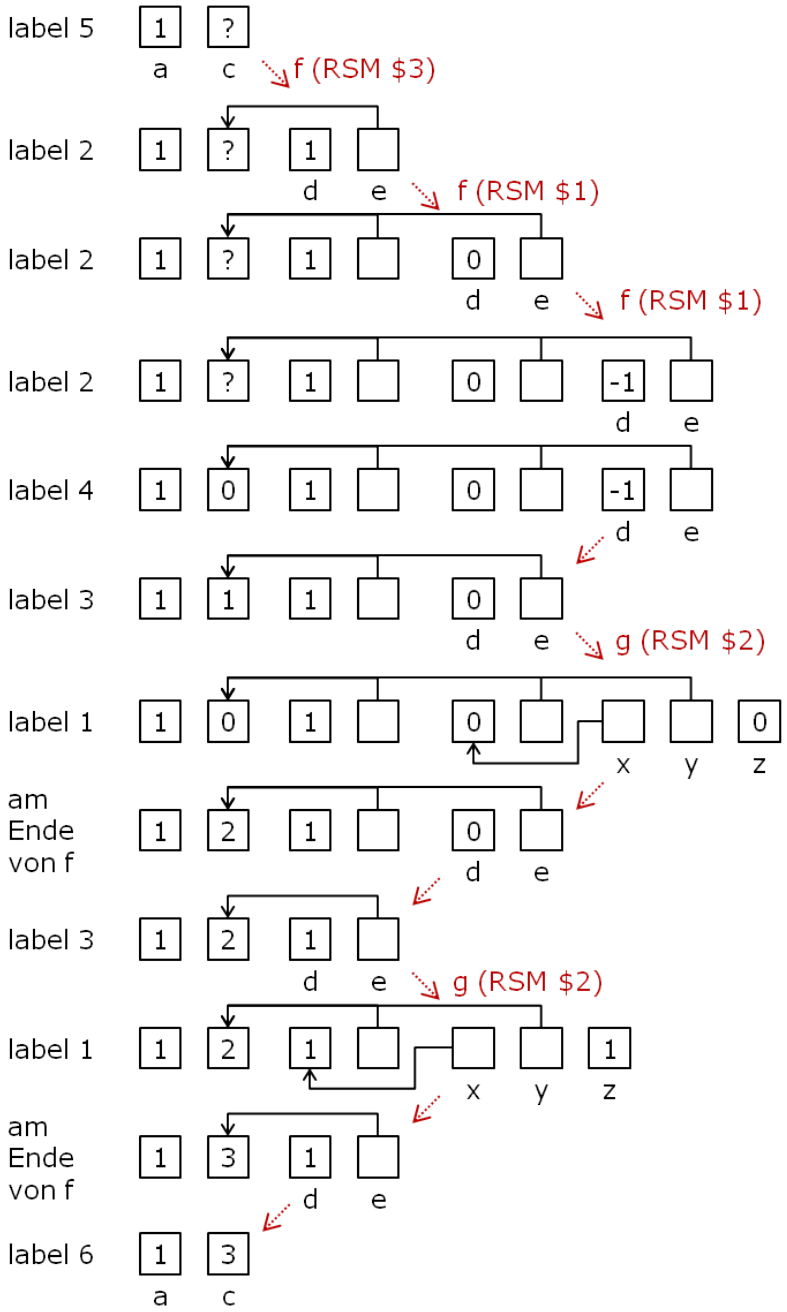
Lösung der 5. Übung – Informatik I für VIW

Fakultät Verkehrswissenschaften

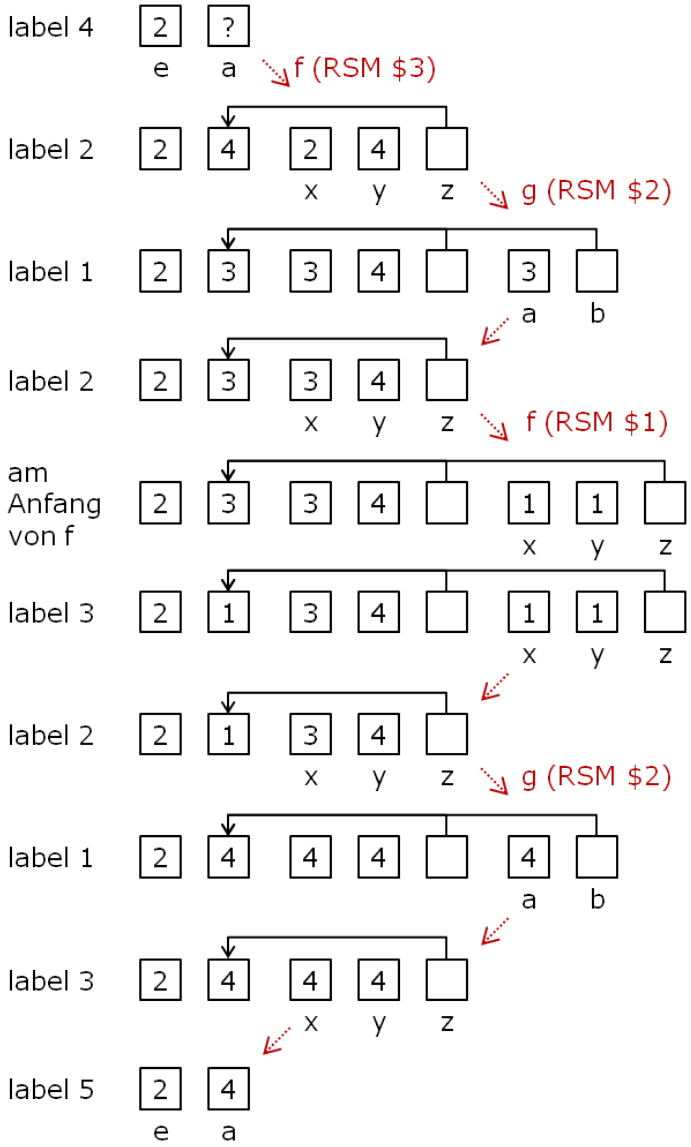
Fachrichtung Verkehrsingenieurwesen

Zeitraum: 13.12.2010 bis 07.01.2011 (WS 2010/11)

Aufgabe 1



Aufgabe 2



Aufgabe 3

(a)

- $\mathcal{O}(2^n)$, weil die Funktion $f(n) = 2^n$ die Funktion $g(n) = n^4$ dominiert.
- $1/2 \cdot n \cdot (1 + \log n) = 1/2 \cdot n + 1/2 \cdot n \cdot \log n$; deshalb hat diese Laufzeit die Komplexität $\mathcal{O}(n \cdot \log n)$, weil die Funktion $f(n) = n \cdot \log n$ die Funktion $g(n) = n$ dominiert.
- $2 + 4 + 6 + \cdots + (n-2) + n = 2 \cdot (1 + 2 + 3 + \cdots + n/2) = 2^{\frac{n/2 \cdot (n/2+1)}{2}}$ (das gilt wegen der Summenformel $\sum_{i=1}^{m} i = \frac{m \cdot (m+1)}{2}$). Weiterhin gilt $2^{\frac{n/2 \cdot (n/2+1)}{2}} = n/2 \cdot (n/2+1) = 1/4 \cdot n^2 + 1/2 \cdot n$. Die Laufzeit hat also die Komplexität $\mathcal{O}(n^2)$.

- Hier muss man die beiden folgenden Logarithmengesetze anwenden:

$$\log(a \cdot b) = (\log a) + (\log b)\ , \qquad \log(a^b) = b \cdot \log a\ .$$

Deshalb ist $\log(n^2 \cdot 2^n) = \log(n^2) + \log(2^n) = 2 \cdot \log n + n \cdot \log 2 = 2 \cdot \log n + (\log 2) \cdot n$. Die Zahl $\log 2$ ist nur ein konstanter Faktor und die Funktion $f(n) = n$ dominiert die Funktion $g(n) = \log n$. Deshalb hat die Laufzeit die Komplexität $\mathcal{O}(n)$.

(b) In der Funktion `f1` läuft die Schleife $\approx n/2$ mal ab und in jedem Schleifendurchlauf wird ein Befehl ausgeführt. Es werden also $\approx n/2$ Befehle ausgeführt. Die Laufzeitkomplexität ist also $\mathcal{O}(n)$.

In der Funktion `f2` wird in jedem Schleifendurchlauf die Variable `i` verdoppelt. Angefangen mit dem Wert 1 läuft das solange ab, bis `i` größer oder gleich `n` ist. Wie oft muss man (angefangen bei 1) verdoppeln, bis man einen Wert größer oder gleich `n` erreicht hat? Ungefähr $\log(n)$ mal! Die Schleife läuft also $\log n$ mal ab. In jedem Schleifendurchlauf wird ein Befehl ausgeführt. Also hat man $\approx \log(n)$ Befehle auszuführen. Die Laufzeitkomplexität ist $\mathcal{O}(\log n)$.

Die Laufzeit der Funktion `f3` ist schwerer zu bestimmen. Nehmen wir an, n hat den Wert 128 (das ist 2^7). Dann nimmt i in der äußeren Schleife die Werte $1, 2, 4, 8, 16, 32, 64$ an (das sind also $7 = \log 128$ Schleifendurchläufe, so haben wir das schon für die Funktion `f2` bestimmt). Die innere Schleife wird immer `i`-mal ausgeführt. Im ersten Durchlauf der äußeren Schleife also 1 Mal, im nächsten Durchlauf 2 Mal, danach 4 mal, usw. Der Befehl `feld[j]++` wird also am Ende genau $1 + 2 + 4 + 8 + 16 + 32 + 64$ mal ausgeführt werden. $1 + 2 + 4 + 8 + 16 + 32 + 64$ ist aber genau 127, also $n - 1$.

Das ist kein Zufall. Addiert man die ersten m Zweierpotenzen (also $2^0, 2^1, 2^2, 2^3, \ldots, 2^{m-1}$), dann erhält man

$$2^0 + 2^1 + \cdots + 2^{m-1} = \sum_{i=0}^{m-1} 2^i = 2^m - 1\ .$$

Das lässt sich leicht beweisen. Der innerste Befehl `feld[j]++` wird also immer $2^m - 1$ mal ausgeführt, wobei m die Anzahl der Durchläufe der äußeren Schleife ist (im Beispiel $n = 128$ ist also $m = 7$). Wir haben für die Funktion `f2` schon bestimmt, dass die äußere Schleife $\approx \log n$ mal ausgeführt wird. Also ist $m \approx \log n$ und die Gesamtlaufzeit ist

$$\approx 2^{\log n} - 1 = n - 1\ .$$

Die Laufzeitkomplexität von `f3` ist also $\mathcal{O}(n)$.

Aufgabe 4 (Zusatz)

```
#include <stdio.h>

void calcNext(int oldSil[], int newSil[], int oldLength) {
   ... /* siehe Vorlesung 8, Folie 32 */
}

int main() {
   int feld1[1023], feld2[1023], int curLength=1, i, nextArray = 2;

   feld1[0] = 0; /* erste Silhouette direkt erzeugen */

   for (i=2; i<=10; i++) {
      if (nextArray = 2) {
         /* naechste Silhouette aus feld1 berechnen und in feld2 speichern */
         calcNext(feld1, feld2, curLength);
         nextArray = 1;
      }
      else {
         /* naechste Silhouette aus feld2 berechnen und in feld1 speichern */
         calcNext(feld2, feld1, curLength);
         nextArray = 2;
      }

      curLength = 2*curLength + 1;   /* Laenge der naechsten Silhouette berechnen */
   }

   /* Ausgabe von feld2 */

   return 0;
}
```