Prof. Andreas Goerdt
M. Pippig, M. Riedel, U. Schönke

TU Chemnitz
Wintersemester 2003/2004

Algorithmen und Programmierung

12. Übung – Lösungsvorschläge

1. Aufgabe:

```
public static int binSucheRek(int[] f, int x,
                              int links, int rechts) {

  if (links == rechts) {
    if (f[links]==x) {
      return links;
    }
    else {
      // nicht enthalten
      return -1;
    }
  }
  else {
    int mitte = (links + rechts) / 2;
    if (x > f[mitte]) {
      return binSucheRek(f,x,mitte+1,rechts);
    }
    else {
      return binSucheRek(f,x,links,mitte);
    }
  }
}

// Aufruf:

position = binSucheRek(feld,gesucht,0,feld.length-1);
```

2. Aufgabe:

```
public static void partRek1(int[] f, int x, int i, int j) {

  int temp;

  if (i>j) {
    // fertig, also nichts tun
    System.out.println("Indexueberschneidung bei i = "
                       + i +  " und j = " + j);
    return;
  }

  if (f[i]<x) {
    i = i + 1;
    partRek1(f,x,i,j);
    return;
  }

  if (f[j]>x) {
    j = j - 1;
    partRek1(f,x,i,j);
    return;
  }

  if (i<=j) {
    //tauschen
    temp = f[i];
    f[i] = f[j];
    f[j] = temp;
    i = i + 1;
    j = j - 1;
    partRek1(f,x,i,j);
    return;
  }
}


// Andere Variante

public static void partRek2(int[] f, int x, int i, int j) {

  int temp;

  if (i>j) {
    // fertig, also nichts tun
    System.out.println("Indexueberschneidung bei i = "
```

```
                        + i + "␣und␣j␣=␣" + j);
  }
  else {
    if (f[i]<x) {
      i = i + 1;
      partRek2(f,x,i,j);
    }
    else {
      if (f[j]>x) {
        j = j - 1;
        partRek2(f,x,i,j);
      }
      else {
        if (i<=j) {
          //tauschen
          temp = f[i];
          f[i] = f[j];
          f[j] = temp;
          i = i + 1;
          j = j - 1;
          partRek2(f,x,i,j);
        }
      }
    }
  }
  return;
}


// Aufruf:

partRek1(feld,feld[k],0,feld.length-1);

partRek2(feld,feld[k],0,feld.length-1);
```

3. Aufgabe:

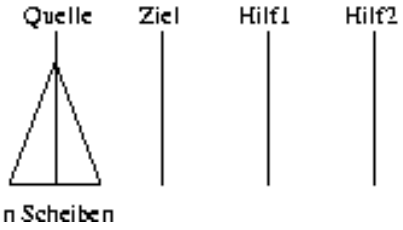
a) Wir wissen, dass $T_1 = 1$ ist (Trivialfall: eine Scheibe – ein Zug). Falls die Anzahl der Scheiben größer als 1 ist, machen wir zunächst einen rekursiven Aufruf mit $n-1$ Scheiben, transportieren dann eine Scheibe auf den Zielstab und machen noch einen rekursiven Aufruf mit $n-1$ Scheiben. T_n ergibt sich folglich mit

$$\begin{aligned} T_n &= T_{n-1} + 1 + T_{n-1} \\ &= 2 \cdot T_{n-1} + 1 \end{aligned}$$

Bei der genaueren Untersuchung dieser rekursiven Berechnungsvorschrift erhalten wir Folgendes:

$$\begin{aligned} T_n &= 2 \cdot T_{n-1} + 1 \\ &= 2 \cdot \left(2 \cdot T_{n-2} + 1\right) + 1 \\ &= 2 \cdot \left(2 \cdot \left(2 \cdot T_{n-3} + 1\right) + 1\right) + 1 \\ &= 2 \cdot \left(2 \cdot \ldots \left(2 \cdot T_1 + 1\right) \ldots + 1\right) + 1 \\ &= 2^{n-1} + 2^{n-2} + 2^{n-3} + \ldots + 1 \\ &= 2^n - 1 \end{aligned}$$

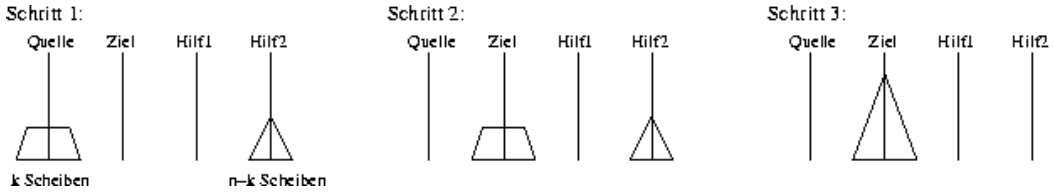
b) Die Ausgangsposition bei „Hanoi mit 4 Stäben“ sieht folgendermaßen aus:



Eine denkbare Strategie zum Umsetzen der Scheiben wäre nun folgende:

1) Transportiere oberen Teil der Scheiben auf einen Hilfsstab, so dass noch k Scheiben auf dem Quellstab verbleiben.
2) Transportiere diese k Scheiben vom Quell- zum Zielstab. Der eine Hilfsstab, auf dem sich die $n-k$ oberen Scheiben befinden, ist dafür natürlich nicht nutzbar, so dass es sich bei diesem Schritt wieder um ein „Hanoi mit 3 Stäben“ handelt!
3) Transportiere die $n-k$ oberen Scheiben vom Hilfsstab auf den Zielstab. Hierbei sind wieder alle 4 Stäbe nutzbar!

Veranschaulichung:



W_n ergibt sich daher mit

$$\begin{aligned} W_n &= W_{n-k} + T_k + W_{n-k} \\ &= 2 \cdot W_{n-k} + T_k \end{aligned}$$

c) Wir betrachten jetzt nur noch Werte für n der Form $n = \frac{m(m+1)}{2}$ und k ist mit $k = m$ fest gesetzt. Also gilt nach obiger Formel

$$\begin{aligned} W_{\frac{m(m+1)}{2}} &= 2 \cdot W_{\frac{m(m+1)}{2}-m} + T_m \\ &= 2 \cdot W_{\frac{m(m+1)-2m}{2}} + T_m \\ &= 2 \cdot W_{\frac{m^2+m-2m}{2}} + T_m \\ &= 2 \cdot W_{\frac{m^2-m}{2}} + T_m \\ &= 2 \cdot W_{\frac{(m-1)m}{2}} + T_m \end{aligned}$$

d) Wir wissen, dass für $n = 1$ (mit $m = 1$) gilt: $W_1 = 1$ (Trivialfall: eine Scheibe – ein Zug). Wir untersuchen nun obige Formel analog wie bei a):

$$\begin{aligned} W_{\frac{m(m+1)}{2}} &= 2 \cdot W_{\frac{(m-1)m}{2}} + T_m \\ &= 2 \cdot \left(2 \cdot W_{\frac{(m-2)(m-1)}{2}} + T_{m-1} \right) + T_m \\ &= 2 \cdot \left(2 \cdot \left(2 \cdot W_{\frac{(m-3)(m-2)}{2}} + T_{m-2} \right) + T_{m-1} \right) + T_m \\ &= 2 \cdot \left(2 \cdot \ldots \left(2 \cdot W_{\frac{(m-(m-1))(m-(m-2))}{2}} + T_2 \right) \ldots + T_{m-1} \right) + T_m \end{aligned}$$

Da

$$W_{\frac{(m-(m-1))(m-(m-2))}{2}} = W_1 = 1 = 2^1 - 1 = T_1$$

gilt

$$\begin{aligned} W_{\frac{m(m+1)}{2}} &= 2^{m-1} T_1 + 2^{m-2} T_2 + 2^{m-3} T_3 + \ldots + 2^0 T_m \\ &= 2^{m-1}(2^1 - 1) + 2^{m-2}(2^2 - 1) + \ldots + 2^0 (2^m - 1) \\ &= 2^m - 2^{m-1} + 2^m - 2^{m-2} + \ldots + 2^m - 2^0 \\ &= m \cdot 2^m - (2^m - 1) \\ &= (m-1) \cdot 2^m + 1 \end{aligned}$$

4. Aufgabe:

   Es gibt sehr viele verschiedene Lösungsmöglichkeiten für diese Aufgabenstellung, sowohl iterative als auch rekursive. Beispielhaft sei hier eine relativ leicht verständliche und sehr kurze rekursive Variante angegeben. Sie setzt folgende Idee um: An eine bereits erzeugte SMW-Zahl können nur noch solche Ziffern angehängt werden, die größer als die Einerstelle sind. Der Algorithmus erzeugt also für die aktuelle SMW-Zahl (Parameter) durch Anhängen einer Ziffer neue SMW-Zahlen und ruft sich mit diesen rekursiv auf. Der Startwert für den Parameter ist die kleinste SMW-Zahl 0.

```
public static void smw_rekursiv (int zahl){

  int ziffer;

  System.out.println(zahl);

  for (ziffer = zahl % 10 + 1; ziffer <= 9; ziffer++) {
    smw_rekursiv(10 * zahl + ziffer);
  }
  return;
}

// Aufruf:

smw_rekursiv(0);
```