Die Türme von Hanoi

Nach einer Legende standen einmal drei goldene Säulen vor einem Tempel in Hanoi; auf die erste waren 100 Scheiben geschichtet, stets eine kleinere Scheibe auf einer größeren. Ein alter Mönch erhielt die Aufgabe, den Turm aus Scheiben von Säule 1 nach Säule 2 zu transportieren unter folgenden Bedingungen:

- es darf stets jeweils nur die oberste Scheibe abgenommen werden,

- es darf niemals eine größere Scheibe auf einer kleineren liegen.

Wenn die Arbeit vollendet sei, so die Legende weiter, sei das Ende der Welt gekommen.

Der kluge Mönch erkannte bald, daß zur Lösung der Aufgabe die dritte Säule benötigt würde, und überlegte sich folgenden Algorithmus:

Sein ältester Schüler solle die Aufgabe erledigen, den Turm aus den 99 obersten Scheiben von Säule 1 nach Säule 3 unter Verwendung von Säule 2 als Zwischenablage zu transportieren; dann wolle er selbst die letzte und größte Scheibe von Säule 1 nach Säule 2 tragen; für den Transport der 99 Scheiben von Säule 3 nach Säule 2 unter Verwendung von 1 werde wiederum der älteste Schüler in Anspruch genommen.

Da der älteste Schüler seinem Meister an Gelehrsamkeit nicht nachstand, delegierte er einen Teil seines Auftrags an den zweitältesten Schüler; dieser delegierte wiederum an den drittältesten Schüler usw.

Die Scheiben seien, beginnend mit der kleinsten, von 1 bis 100 (bzw. 1 bis n) durchnummeriert.

Die Anweisung **"Transportiere den Turm von 100 Scheiben von Säule 1 nach Säule 2 unter Verwendung von Säule 3"** wird im folgenden mit **TRANSPORT(100, 1, 2, 3)**, die Anweisung **"Transportiere den Turm von n Scheiben von Säule s1 nach Säule s2 unter Verwendung von Säule s3"** mit **TRANSPORT(n, s1, s2, s3)** bezeichnet.

Damit besteht der Auftrag TRANSPORT(100, 1, 2, 3) des alten Mönchs aus drei Schritten:

1. TRANSPORT(99, 1, 3, 2) (delegiert an den ältesten Schüler)
2. Transportiere Scheibe 100 von Säule 1 nach Säule 2 (Arbeit des Meisters)
3. TRANSPORT(99, 3, 2, 1) (delegiert an den ältesten Schüler)

In gleicher Weise ruft TRANSPORT(99, 1, 3, 2) auf:

1. TRANSPORT(98, 1, 2, 3) (delegiert an den zweitältesten Schüler)
2. Transportiere Scheibe 99 von Säule 1 nach Säule 3 (Arbeit des ältesten Schülers)
3. TRANSPORT(98, 2, 3, 1) (delegiert an den zweitältesten Schüler)

Allgemein veranlaßt TRANSPORT(n, s1, s2, s3) folgende Anweisungen, falls n > 1 ist:

1. TRANSPORT(n- 1, s1, s3, s2)
2. Transportiere Scheibe n von Säule s1 nach Säule s2
3. TRANSPORT(n- 1, s3, s2, s1)

Die Pascal-Prozedur überrascht durch Einfachheit und Eleganz gleichermaßen:

```
Procedure TRANSPORT(n, s1, s2, s3: integer);
begin
    if n>1 then TRANSPORT(n-1, s1, s3, s2);
    writeln('Transportiere Scheibe ',n:2,' von Säule ',s1:2,' nach Säule ',s2:2);
    if n>1 then TRANSPORT(n-1, s3, s2, s1)
end;
```

Für n Scheiben werden n Mönche benötigt: der Meister transportiert die n-te, also die größte, Scheibe genau einmal von Säule 1 nach Säule 2; der 2. Mönch (ältester Schüler) ist für den Transport der (n–1)-ten, der zweitgrößten, Scheibe zuständig, der 3. Mönch (zweitältester Schüler) transportiert die (n–2)-te Scheibe, allgemein transportiert der i-te Mönch die (n–i+1)-te Scheibe für i = 1, 2, ... , n.

Bei 3 Scheiben veranlaßt TRANSPORT(3, 1, 2, 3) folgende Aufrufe (Book antiqua) und Anweisungen (Arial), wobei in der 1. Spalte die Nummer des Aufrufs steht:

1. TRANSPORT(3, 1, 2, 3)
2. TRANSPORT(2, 1, 3, 2)
3. TRANSPORT(1, 1, 2, 3)
   transportiere Scheibe 1 von Säule 1 nach Säule 2
   transportiere Scheibe 2 von Säule 1 nach Säule 3
4. TRANSPORT(1, 2, 3, 1)
   transportiere Scheibe 1 von Säule 2 nach Säule 3
   transportiere Scheibe 3 von Säule 1 nach Säule 2
5. TRANSPORT(2, 3, 2, 1)
6. TRANSPORT(1, 3, 1, 2)
   transportiere Scheibe 1 von Säule 3 nach Säule 1
   transportiere Scheibe 2 von Säule 3 nach Säule 2
7. TRANSPORT(1, 1, 2, 3)
   transportiere Scheibe 1 von Säule 1 nach Säule 2

Aufgaben:

a) Begründe, daß der Algorithmus terminiert!

b) Fertige wie oben eine Aufstellung der Aufrufe für n=4 an!

c) Welche Arbeit verrichtet der i-te Mönch?

Lösung zu c):

Der 1. Mönch transportiert die n-te Scheibe 1-mal, der 2. Mönch die (n-1)-te Scheibe 2-mal, der 3. Mönch die (n-3+1)-te Scheibe 4-mal, also 2^{3-1}-mal; der i-te Mönch transportiert die (n-i+1)-te Scheibe 2^{i-1}-mal (strenger Beweis: vollständige Induktion über i).
Von 100 Mönchen plagt sich der jüngste an der 1. Scheibe $2^{100-1} = 2^{99} = 6{,}34 * 10^{29}$- mal!

d) Wie verhält sich der Speicheraufwand in Abhängigkeit von der Anzahl n der Scheiben?

Lösung zu d):
Da TRANSPORT(n, s1, s2, s3) 2^{n-2} weitere TRANSPORT-Aufrufe veranlaßt, umfaßt TRANSPORT(n, s1, s2, s3) insgesamt 2^{n-1} Aufrufe (strenger Beweis: vollständige Induktion über n), deren Anzahl wächst folglich

exponentiell. Es sind allerdings stets höchstens n Aufrufe aktiv, wie folgende Überlegung zeigt:
Für n=3 existieren höchstens 3 Aufrufe der Prozedur TRANSPORT gleichzeitig, wie man an obiger Aufstellung sieht. Entsprechend mache man sich klar, daß bei n Scheiben höchsten n Aufrufe der Prozedur TRANSPORT gleichzeitig aktiv sind. Der Speicheraufwand wächst lediglich linear mit n, so daß keine Fehlermeldung wie "Memory Overflow" oder "Stack Overflow" während der Laufzeit zu erwarten ist!

e) Wie verhält sich der Zeitaufwand in Abhängigkeit von der Anzahl n der Scheiben?

Lösung zu e):

Da der i-te Mönch seine ihm zugewiesene Scheibe 2^{i-1} -mal zu tragen hat, gibt es insgesamt $1 + 2 + 4 + 2^{4-1} + 2^{5-1} + \ldots\ldots + 2^{n-1}$ Transportvorgänge bzw. writeln-Anweisungen; dies ist eine geometrische Reihe $\{ s_n \}$ mit a = 1, q = 2:

$$s_n = \Sigma\, a_i = \Sigma a\, q^{i-1} = (1 - 2^n) / (1 - 2) = 2^n - 1$$

$$s_{100} = 2^{100} - 1 \approx 1{,}268 * 10^{30}$$

Unterstellt man, daß ein äußerst leistungsfähiger Computer für einen Transportauftrag, also die Ausführung der writeln-Anweisung, einen Zeitbedarf von 1 mikro-Sekunde hat, ergibt sich für das vollständige Umschichten von 100 Scheiben die Zeitspanne

$t = 1{,}268 * 10^{30}$ s / 10^6 = $1{,}268 * 10^{24}$ s = $1{,}268 * 10^{24}/(3600 * 24 * 365)$ a
= $4{,}02 * 10^{16}$ a = 2.000.000 Weltalter,

wenn man das Alter des Weltalls zu 20 Milliarden Jahren annimmt!
Dieses Beispiel zeigt, daß die Ausführung eines Algorithmus an expontiell wachsendem Zeitbedarf in der Praxis scheitert.

f) Ersetze die writeln-Anweisungen durch entsprechende Graphik-Routinen!

g) Hier das vollständige Pascal-Programm (mit Eingabe von n):

```
program hanoi;

uses crt, printer;

var n: integer;

procedure transport(n, s1, s2, s3: integer);

begin

    if n>1 then transport(n-1, s1, s3, s2);

    writeln('Transportiere Scheibe ',n:2,' von Säule ',s1:2,' nach Säule ',s2:2);

    if n>1 then transport(n-1, s3, s2, s1)

end;
```

```
begin
        clrscr;
        write(' Wieviele Scheiben? ');
        readln(n);
        transport(n,1,2,3);
        while not keypressed do
end.
```