Tower of Hanoi, revisited

Kees van der Laan

December 1991

Abstract

Another version of programming 'The Tower of Hanoi' in TEX is provided.[1] No nodding knowledge of Lisp is required; just plain TEX. There is no restriction on the number of disks, apart from the installed limits of TEX. Generalized disks can be moved as well.

1 Introduction

At the Dedham TUG91 conference, I attended David Salomon's advanced TEX course. Instead of redoing his clear and ample exercises I decided to rework Leban(1985). The more so because elaborating a classic example might bring you to fundamental issues. In courses on programmming the Tower of Hanoi problem is used to illustrate paradigms. I was pleased to encounter some paradigms of TEX programming while revisiting the tower.

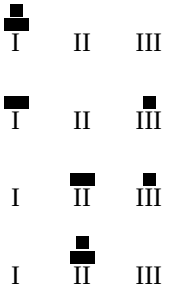
2 The Tower of Hanoi problem

A pyramid of disks—meaning a tower with implicit ordering of the disks—has to be moved under the restrictions that only one disk at a time can be moved and that each intermediate state consists of pyramids, obeying the original implicit ordering. In total three places for (intermediate) pyramids are allowed. For a pyramid of n disks, the solution needs $2^n - 1$ moves. For an introduction to the problem see the first paragraphs in Graham c.s.(1989).

3 Example

```
\input hanoi.tex \hanoi2 \bye
```

will yield the process of replacements for 2 disks from tower I to tower II



[1] For an earlier article on the issue, see Leban(1985).

4 The file hanoi.tex

This file contains all the macros: the top `\hanoi`, the version with more parameters `\Hanoi`, the macro for the moves `\movedisk`, along with the auxiliary macro to prefix a string, `\logoffx...\logoffx...`, and the auxiliaries for printing `\showtowers`, and `\pt`.

As data structure a simplified version of Knuth's list, see TEXbook Appendix D.2, is used. A tower has the replacement text $\backslash\backslash\langle\text{item}_1\rangle\ \backslash\backslash\langle\text{item}_2\rangle\ \ldots\ \backslash\backslash\langle\text{item}_n\rangle$, where in this case each item is a control sequence. The separators, `\\`, '. . . are enormously useful, because we can define `\\` to be any desired one-argument macro and then we can *execute* the list!'

```
%hanoi.tex version 18 dec 91
\newcount\n   %The number of disks
\newcount\brd %Breadth of towers
\newcount\hgt %Height of maximum tower
\newcount\dskhgt %Height of each disk
\let\ea=\expandafter %Shorthand
\let\ag=\aftergroup  %Shorthand
\def\preloop{%To create loopcnt, a
             %local loopcounter
             %(see also loopy.TeX).
   \bgroup \advance\count10 by 1
   \countdef\loopcnt=\count10
            %Symbolic name
   \loopcnt=1 %(default)
 }%end \preloop
\def\postloop{\loopcnt=0 %Restore
   \egroup}%end \postloop
%
%Hanoi macros, top level
\def\hanoi#1{%Argument can be digit(s)
             %or a counter (numeric)
 \n=#1 %Assign argument value to \n
 \def\II{}\def\III{}%Empty towers
 %Next is inspired by the TeXbook,
 %p374, 378
 %The initial tower for \I is created
 %The initial tower for \I is created
```

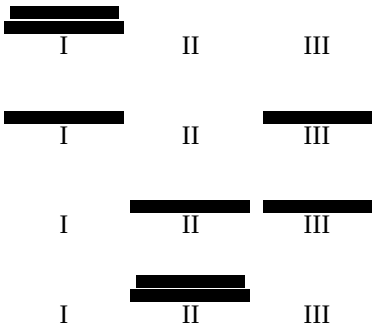
```
 %   \def\I{\\\i\\\ii\\\iii...\\\'n'}
 %next to the defs for \i,\ii,...\'n'.
 \preloop\ag\def\ag\I\ag{%
  \loop
    \ea\xdef\csname\romannumeral\loopcnt
            \endcsname{\the\loopcnt}
    \ag\\%separator
    \ea\ag\csname
         \romannumeral\loopcnt\endcsname
    \ifnum\loopcnt<\n
    \advance\loopcnt by 1
  \repeat   \ag}
 \postloop
 %For printing, values are needed for
 \brd=\n %Breadth of largest disk
 \advance\brd by 3 %Little room extra
 \dskhgt=1 %Height of disks
 \hgt=\n\multiply\hgt by2 %\hgt is height
       \advance\hgt by1 %of towers
 \showtowers %Print initial state
 \Hanoi\I\II\III\n
 }%end \hanoi
%
\def\Hanoi#1#2#3#4{%Moves from #1 to #2,
                   %with aid of tower #3.
 %The number of disks is #4, in a counter.
 \ifnum#4=1 %For Tower of 1 disk,
            %just move the disk
   \movedisk\from#1\to#2%
   \showtowers%Print towers after move
 \else%Problem of #4 disks is solved by
      %- problem of (#4-1) disks,
      %- a move, and
      %- a problem of (#4-1) disks.
   {\advance#4 by-1 \Hanoi#1#3#2#4}%
   \movedisk\from#1\to#2%
   \showtowers%Print towers after move
   {\advance#4 by-1 \Hanoi#3#2#1#4}%
 \fi}%end \Hanoi
%
%Moving of the disks, TeXbook, App. D.2
%Slightly adapted versions of \lop (
%called \movedisk with function that
%first element of #1 is prefixed to #2)
%and \lopoff modification
\def\movedisk\from#1\to#2{%Move disk from
%tower #1 to tower #2
  \ea\lopoffx#1\lopoffx#1#2}
\def\lopoffx\\#1#2\lopoffx#3#4{\ea\gdef%
  \ea#4\ea{\ea\\\ea#1#4}
  \gdef#3{#2}%restore stub}%end\lopoffx
 }%end \movedisk
%
%Printing tower status
\def\showtowers{%Display pyramids
 \par\quad\hbox{\pt\I\ \pt\II\ \pt\III
              }\par
 }%end \showtowers
%
%Auxiliaries
\def\gobble#1{}%To eat character
%
\def\\#1{\hbox to\brd ex{\hss%
  \vrule width#1ex height\dskhgt ex%
                         \hss}%
 }%end \\
%
\def\pt#1{%Print Tower.
%#1 is \I, \II, or \III
\vbox to\hgt ex{\baselineskip=.2ex\vss%
      #1%
      %Format pointer underneath
      \hbox to\brd ex{\hss%
              \ea\gobble\string#1\hss}%
      }%end vbox
 }%end \pt
```

5 Disks not restricted to one digit

One could invoke `\hanoi{10}`[2] at the expense of ample time and use of paper. In order to illustrate the possibility of the macros to cope properly with disks denoted by more than one digit, the pyramid $\begin{smallmatrix}/\,9\,\backslash\\ /\,10\,\backslash\end{smallmatrix}$ can be handled via

```
\n=2    %Number of disks
\def\ix{9}\def\x{10}%Disks
\def\I{\\\ix\\\x}\def\II{}\def\III{}
\brd=10 %Breadth of largest disk
\hgt= 6 %Height of tower
\dskhgt=1 %Height of disks
\def\\#1{\hbox to\brd ex{\hss%
  \vrule width#1ex height\dskhgt ex%
                         \hss}}%
\showtowers%Initial state
\Hanoi\I\II\III\n
```

with result



[2] Or an invoke with an even larger argument. I allowed my modestly equipped MS-DOS PC to loop for $\backslash n = 1, 2, 3, \ldots$ in order to experience TEX's limits with respect to the practical upper bound for the size of the tower. A tower of 10 disks did take ≈ 1.5 minutes to TEX, with empty `\showtowers`. I had the patience to await the TEXing up to the tower of size 16, and concluded that there are no limits for reasonable usage. (Note that every increase of the size by 1 will double the time needed.) So, I don't know when my PC will crash or, to paraphrase the monks, 'when the world will end.' I also modified the macros into versions with `\I`, `\II`, and `\III` defined as token variables. I was surprised to see *that* version run 4 times slower. Whatever the value, the hanoi macros can be used as a program to compare the efficiency of TEX implementations.

6 Generalized disks

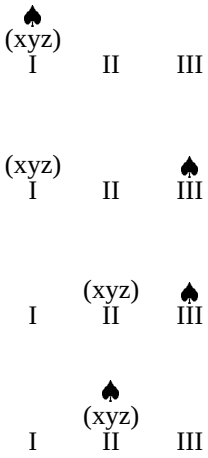
What about for example `(xyz)` as disk? Let us assume for printing that the contents of the pyramids—the strings—will do, in the implicit provided order. This can be obtained via a modified `\\` definition.

7 Example

The tower can be moved, with the states printed via

```
\n=2\def\i{$\spadesuit$}\def\xyz{(xyz)}
\def\I{\\\i\\\xyz}\def\II{}\def\III{}
\brd=6 \hgt=7
\def\\#1{\hbox to \brd ex{\hss%
                          #1\hss}}%
\showtowers%Initial state
\Hanoi\I\II\III\n
```

with results



8 Interactivity

Downes(1991) inspired me to think about direct communication with the user. What about the modification of `\showtowers` into an appropriate `\immediate\write16{...}` command, such that the moves will appear on the screen? No previewing nor printing![3]

```
%Direct screen \showtowers
\def\showtowers{
\immediate\write16{
   \ea\gobble\string\I:   \I
   \ea\gobble\string\II:  \II
   \ea\gobble\string\III: \III}}
```

9 Conclusion

Is this just for fun? It was appropriate for 'Fun with TEX,' NTG's 91 fall meeting at Eindhoven. Furthermore, I experienced the following fundamental (TEX) programming issues

- recursion in solving the problem
- the use of the list data structure and separators to execute the list, see the TEXbook Appendix D.2
- creating and using a local loop counter[4]
- creation of a dynamic number of command names and a string of dynamic length
- generalizing the problem (not only numbers can denote disks; note that no comparison of disks is done, the ordering of the subtowers is maintained implicitly)
- direct communication on screen.

TEXnically `\aftergroup`, `\countdef`, `\csname`, `\expandafter`, `\ifnum`, `\ifx`, `\immediate\write16...`, `\loop`, `\romannumeral`, and `\string` are exercised.

The hardest thing was to get the towers aligned when formatting commands were split over several lines, due to the two-column format. Several % symbols were needed to annihilate the effect of spurious spaces, especially those created by some ⟨cr⟩'s.

It did take some time to realize the benefits of Knuth's list macros, not to say that I wandered around quite a bit.

References

[1] Downes, M.J. (1991): Dialogue with TEX. Proceedings TUG91.
[2] Graham, R.L, D.E. Knuth, O. Pastashnik (1989): Concrete Mathematics. Addison-Wesley.
[3] Knuth, D.E. (1984): The TEXbook. Addison-Wesley.
[4] Leban, B. (1985): A solution to the Tower of Hanoi problem using TEX. *TUGboat* 6, no. (3), 151–154.
[5] Pittman, J.E. (1988): Loopy.TeX. *TUGboat* 9, no. (3), 289–291.
[6] Salomon, D. (priv.comm.)

[3] Alas, no control over the format on the screen either.

[4] This was mentioned before by Pittman(1988). It is a matter of taste and programming style whether one prefers this next best to the hidden counter idea, above the use of a global counter, for counting the number of times a loop is traversed. The difference in efficiency is negligible.