Typesetting number sequences

FIFO and some more . . .

Kees van der Laan

Hunzeweg 57
9893 PB Garnwerd, The Netherlands
`cgl@rug.nl`

Abstract

Typesetting sequences of numerical values, represented via symbolic names which get their values on the fly, is dealt with. The sorting of the sequence is done by a linear sorting algorithm, of complexity O(n^2). Three or more consecutive numbers are typeset as a range.
The objective was to encode typesetting sequences of numbers as simple, concise, general, compatible, modular, orthogonal, and . . . , as possible in TEX.[1]

Keywords: Typesetting sequences, citation lists, lists of references, linear sorting, FIFO, plain TEX, macro writing, education.

1 Introduction

To think about typesetting sequences of numerical values looks a bit peculiar. I was pulled in this direction when thinking about the 'range notation' problem, which was posed at the tex-nl@hearn discussion list.[2] The idea is that for example the numbers 1, 2, 3 should appear in print as 1–3. This is a trifle when the numbers are known a priori. When symbolic names are used, which get their numerical values on the fly, one has to resort to macros. The macros must take care of ordering the sequence and proper typesetting the numbers.

For that purpose the macro `\typseq`—mnemonics: typeset sequence—was written.

There are various choices possible with respect to the TEX implementation of symbolic names for numbers. The most direct form is `\chardef\`$\langle symbolic\, name\rangle = \langle number\rangle$.[3]

The problem was solved in the Polya, 1957, way. First the kernel problem of typesetting an ordered sequence of numbers in range notation was considered. Collateral the independent problem of ordering a sequence had to be solved.[4] And finally the merging of both aspects and the TEX encoding.

The intended scope of readers consists of those who not only favor the use of LATEX and TEX, but also like to understand what is going on, and otherwise strive after keeping the encoding simple and concise.

Notations

`\ea`, `\nx`, and `\ag`, are used as shorthand for `\expandafter`, `\noexpand`, respectively `\aftergroup`.

2 Example of use

If we have:[5]

```
\chardef\dekker=5, \chardef\forsythe=3,
\chardef\reinsch=4, \chardef\knuth=11,
```

then

```
\typseq{\dekker,\knuth,\forsythe,\reinsch}
```

yields [3–5, 11].

3 Stepping stone

Suppose we have a non-descending sequence of numbers and we like to typeset these in range notation. For example 1, 2, 5, 7, 8, 9, 10 as [1, 2, 5, 7–10].

My solution is the invocation

`\cpr{1,2,5,7,8,9,10}`,

backboned by[6]

[1] Not blurred by safeguarding goodies or limited by a particular application.
[2] Not dealt with is the typesetting of large amount of data via tables or graphs, with or without statistical methods, or the typesetting of encoded data like van Wijngaarden's method to typeset millions of prime numbers on an A4 or two.
[3] This is restricted by 256. If needed one can use `\def\`$\langle symbolic\ name\rangle$`{...}`, or `\mathchardef`. . . .
[4] For sorting general sequences in TEX, via O$(n \log n)$ algorithms, see van der Laan, 1993.
[5] That these names stand for incomparable outstanding (numerical) mathematicians is a mere coincidence.
[6] Declarations are omitted. For FIFO see van der Laan, 1992b, or the listing of the file typseq.tex included below.

```
\def\cpr#1{{\def\process{\processc}\bs
 \fifo#1,\ofif,\prtfl\es}}
\def\processc#1{%
           \init{#1}\def\processc##1%
 {\ifnum##1=\lst\else\ifnum##1=\slst
  \lst=\slst\advance\slst1{}\else
  \prtfl\sepn\init{##1}\fi\fi}}
\def\init#1{\frst=#1\lst=#1\slst=#1{}%
   \advance\slst1 }
\def\prtfl{\the\frst\ifnum\frst<\lst
 \advance\frst1{}\ifnum\frst=\lst\sepn
 \else\nobreak--\nobreak\fi\the\lst\fi}
\def\bs{[}\def\es{]}\def\sepn{, }
```

Explanation

`\cpr` This is an independent macro for just typesetting a sequence in range notation.

`\processc` The encoding makes use of the FIFO paradigm[7] to process each element. In order to perform the appropriate action we must look ahead, or postpone the typesetting, both to an unknown depth. It seems natural to postpone *as long as needed* the typesetting, with the first and last elements of the range so far—not necessarily different—stored in the counter variables `\frst`, respectively `\lst`.
The mechanism of redefinition is used to account for initialization. The first definition of `\processc` invokes `\init`—to give the counter variables `\frst` and `\lst` the value of the current list element—followed by a *redefinition* of `\processc`.[8] In the latter redefinition an element is skipped when it equals[9] the previous one. If it equals the successor of `\lst` then `\lst` gets the value of its successor. Otherwise 'the range' is typeset followed by the value of the separator, `\sepn`. The typesetting is handled by the macro `\prtfl`. The degenerate case, `\frst=\lst`, yields a single number. After having typeset the range we are in a pseudo initial state. `\frst` (and `\lst`) must get the value of the list element at hand. This is done again via the invocation of `\init`.

At the end of the list we must typeset appropriately the values of `\frst` and `\lst`. This is done via the invocation of `\prtfl`.

If we look at sorting as a CISO—Collective-In-Smallest-Out—process, then it is rather straightforward to combine sorting with `\processc`, because `\processc` handles the range typesetting independent of how the successive arguments are obtained.[10]

4 The macro \typseq

Purpose. The purpose of the macro `\typseq` is to typeset automatically a sequence in ascending order in range notation.

Input. The argument of `\typseq` is the sequence of *symbolic* names—representing the numerical values—separated by commas.

Result. The values of the sequence items are typeset in ascending order in range notation, separated by the value of `\sepn`, and delimited by the values of `\bs`, respectively `\es`.

Design. My encoding of automatic typesetting sequences of numbers comes down to
- Store the input sequence as a list with active list separators,
- Sort the list via appropriate definition of the active list separator,
- Typeset the list appropriately.

The file typseq.tex

```
%Shorthands
\let\ag=\aftergroup
\let\ea=\expandafter\let\nx=\noexpand
%Counters
\newcount\frst%First value of range
\newcount\lst %Last value of range
\newcount\slst%Successor \lst
%Newif-s
\newif\ifnoe%   Mnemonics: if not empty
%Parameters: separators
\def\sepn{, }%  Number separator
%Parameters: brackets
\def\bs{[}\def\es{]}
%FIFO with comma as separator
\def\fifo#1,{\ifx\ofif#1\ofif\fi%
 \process{#1}\fifo}\def\ofif#1\fifo{\fi}
%Store, sort and typeset
\def\typseq#1{{\strseqaslst{#1}\bs\srt\prtfl%
 \es}}% Local scope because of redef-s.
%Store sequence as list
\def\strseqaslst#1{\let\process=\processs
\xdef\list{\fifo#1,\ofif,}}
%ProcessS stores consecutive elements
%(preceded and) separated by \ls as a list.
\def\processs#1{\nx\ls\nx#1}
%Mod from Syntactic Sugar, MAPS92.1, p135
\def\srt{% Assumed is \list contains
%        symbolic names separated by \ls.
 \loop\ifx\empty\list\noefalse%
      \else\noetrue\fi%
%Test for NOEmpty list.
 \ifnoe \first\list%  \min=first element
```

[7] For FIFO and especially (variant) TEX encodings of the principle, see van der Laan, 1992b.
[8] This mechanism is general and elegant for coping with begin situations, where the first action is the only one different from the rest.
[9] For an ordered sequence 'less than' could be used. 'Equals' allows for non-ordered sequences too: just numbers which form a range are compressed.
[10] This prompts another approach for solving the problem: maintaining a priority queue. My case rests.

```
    \list%  Find minimum and store \min
    \processc\min%  Typset\min
%Delete minima from \list.
  {\def\ls####1{\ifx####1\min\else%
   \nx\ls\nx####1\fi}\xdef\list{\list}}%
 \repeat}%   end \srt
%List Separator.
\def\ls#1{\ifnum#1<\min\let\min=#1{}\fi}
%Pop up first element of list #1
\def\first#1{\def\lop\ls##1##2\pol{%
   \let\min=##1{}}\ea\lop#1\pol}

%Compressing sequences; it is assumed
%that elements are separated by commas.
\def\cpr#1{{\def\process{\processc}\bs%
 \fifo#1,\ofif,\prtfl\es}}
%Typeset element or keep track of range
\def\processc#1{\init{#1}\def\processc##1%
 {\ifnum\lst=##1{}\else\ifnum\slst=##1%
  \lst=\slst\advance\slst1{}\else%
  \prtfl\sepn\init{##1}\fi\fi}}
\def\init#1{\frst=#1\lst=#1\slst=#1{}%
   \advance\slst1{}}
%Print range: \frst-\lst (or \lst).
\def\prtfl{\the\frst\ifnum\frst<\lst{}%
 \advance\frst1{}\ifnum\frst=\lst\sepn%
 \else\nobreak--\nobreak\fi\the\lst\fi}
\endinput            %dec 92; cgl@rug.nl
%Test/example program. \tracingmacros=1
%Typeset sequence: \chardef\a=1
\chardef\b=27\chardef\c=134  %all <256!
\typseq{\c,\a,\b}.
\bye
```

Explanation

`\typseq` This composition macro invokes the macros for storing (`\strseqaslst`),[11] sorting (`\srt`), and typesetting (`\processc`).
`\strseqaslst` The arguments of the macro are the sequence elements separated by commas. First the data are stored in a list, via the use of the FIFO paradigm.[12] `\ls` is used as active list separator.
`\srt` We loop through the `\lists` until the `\list` is exhausted. In each step the minimum value is determined, typeset, and deleted from the list.[13]

- Finding the minimum.
  To initiate the process the first element is considered to be the minimum. Then the `\list` is invoked with the active list separator the function to find better minima.
- Delete minimum.
  The deletion of the minima from the list is TEX specific. We first redefine locally the list separator, `\ls`, with the function to delete the element if it equals the minimum value.[14] Then `\list` is `\xdef`-ed with `\list` as replacement text! This expands `\list` and provides in `\list` the non-minima, again separated by `\ls`-s.[15]
  Concise and elegant isn't it?
- Typesetting.
  We must account for the range notation—three or more consecutive numbers are represented as $\langle first\ number\rangle$–$\langle last\ number\rangle$—and other conventions like the separation symbol, and enclosing the total within square brackets. The latter have been parameterized into `\sepn`, `\bs`, respectively `\es`.

5 Variation

My earlier variant of `\storeseqaslst` made use of `\ag`, as follows

```
\def\strseqaslst#1{{\ag\def\ag\list\ag{%
 \let\process\processs\fifo#1,\ofif,}}}
%with
\def\processs#1{\ag\ls\ag#1}
```

When an application provides just numbers—for example page numbers after index items—the above `\strseqaslst` can be adapted to store these numbers in `\ls\1\ls\2\ls` . . . `\ls\`$\langle n\rangle$, via

```
\def\strseqaslst#1{{\ag\def\ag\list\ag{%
 \let\process\processs\global\n0
 \fifo#1,\ofif,}}}
%with
\def\processs#1{\global\advance\n1
 \ea\xdef\csname\the\n\endcsname{#1}%
 \ag\ls\ea\ag\csname\the\n\endcsname}
```

The encoding of `\strseqaslst` can be made robust with respect to redundant spaces. I refrained from this at the moment.
`\strseqaslst` is superfluous when the convention is adopted to separate and precede the arguments of `\typseq` by `\ls`.

For other representations of the numbers, for example in superscript, one can redefine `\prtfl`. Penalties can be inserted in the replacement text of `\sepn`, to inhibit line breaks. The question is: How much?

6 Looking back

The data structure `\list` is peculiar. Not only obeys it to the queue access method, that is from left to right via the execution of `\list` with active list separator `\ls`, but individual elements can be accessed via their

[11] Mnemonics: store-sequence-as-list.
[12] This is in the spirit of the generation of n-stars, TEXbook Appendix D.1 p.374. No `\aftergroup` is necessary in this particular case, however.
[13] This has been published earlier and perhaps a bit hidden, in Syntactic Sugar, van der Laan, 1992a.
[14] Note that multiple occurrences of the minimum is accounted for.
[15] The four #-s in the definition of `\ls`—redefinition within `\srt`, line 36–37—look peculiar. We must account for the hidden definition of the loop `\body`, which makes that the definition of `\ls` is nested two levels deep!

names too. During the invocation of the example the data structure is modified as follows[16]

```
sequence :\d,\k,\f,\r
list     :\ls\d\ls\k\ls\f\ls\r ,
```

and gradually into `\list` equals `\empty` while sorting. Neat![17]

7 Epilogue

Typesetting of sequences on the fly occur when dealing with citation lists, or a list of references to figures and the like. LaTeX's `\cite` doesn't automatically order sequences or typeset ranges.

The range representation was posed as a problem at the tex-nl@hearn discussion list. A (LaTeX) style was offered in reply.[18] This style was composed by Ronald Kappert out of the work of Arseneau and Green. Kappert reported that Arseneau has improved upon the style. The improved style is available on file server niord.shsu.edu. According to Kappert it is in general use, especially by those who have long lists of citations. I myself don't use long citation lists. I am happy with the system of 'name followed by year.' As author, and also as reader, I know by heart the name and year of most of the works to refer to. This is structurally simple. It also makes the usual multi-pass processing of a list of references superfluous, because there is no information needed which will be created later. It is already there. However, in practice authors are restricted by journal conventions.

The hardest thing was not to introduce bells-and-whistles and to keep it as straight as possible. I refrained from introducing robustness with respect to arguments with unnecessary spaces.
Ronald Kappert is kindly acknowledged for his remarks and suggestions in proofing the article.

8 TeXniques used

- Creating a list of dynamical length via FIFO (and as alternative with `\aftergroup`).
- Sorting a list via repeated execution of `\xdef\list{ \list}`, and appropriate use of the active list separator.
- Handling recursion (loop) initialization, such that some action on first traversal is different from the action via the same name, on later traversals.

References

[1] Jeffreys, A (1990): Lists in TeX's mouth. *TUGboat* 11, no. (2), 237–244.

[2] Kappert, R (1992): scite.sty. (From the file server; it is compiled of work from D. Arseneau and I. Green: overcite.sty, drftcite.sty, cite.sty. Actually, I. Green made the style citesort.sty, which has been assimilated by D. Arseneau in his styles.)

[3] Knuth, D.E (1984): *The TeXbook*, Addison-Wesley.

[4] Laan, C.G. van der (1992a): Syntactic Sugar. MAPS92.2, 130–136. (Submitted TUG '93.)

[5] Laan, C.G. van der (1992b): FIFO & LIFO sing the BLUes. MAPS92.2, 139–144. (Submitted TUGboat.)

[6] Laan, C.G. van der (1992c): Tower of Hanoi, revisited. *TUGboat* 13, no. (1), 91–94. Also in MAPS92.1, 125–127.

[7] Laan, C.G. van der (1993): Sorting in BLUe, MAPS93.1. (Submitted TUG '93.)

[8] Lamport, L (1986): LaTeX, user's guide & reference manual. Addison-Wesley.

[9] Polya, G (1957): How to solve it. Anchor.

[16] Symbolic names abbreviated to their first letter.

[17] Instead of emptying the `\list` I could have maintained a property list, for example a permutation array. Another approach. My case rests.

[18] My code consists of the modules: storing, sorting, and typesetting in range notation (each ≈ 10 lines), next to the composition (1 line).