Introduction

Victor Eijkhout

Notes for CS 594 – Fall 2004

Ancient typesetting systems

- Input was compiled to printable form; not ‘wysiwyg’
- Sequential processing of text input file
- Commands for font choice and other layout
- Macros with replacement text

  ```
  $ADAM$ --> From our correspondent in Amsterdam
  ```

Computer typesetting systems

- Same idea: document compilation, macros for text replacement and formatting

  ```
  \TeX => T\kern -.1667em\lower .5ex\hbox {E}\kern -.125e
  ```

  gives ‘TeX’.
- Macro replacement language
- Turing equivalent

Logical markup

- Use macro names to indicate structure:

```
\begin{theorem}
\TeX\ is pretty cool.
\end{theorem}
\begin{proof}
See for yourself
\end{proof}
```

> **Theorem**
>
> *TEX is pretty cool.*
>
> *Proof:* *See for yourself* ●

Logical markup (2)

- Layout is determined by style declaration

  ```
  \documentclass{article}
  \documentclass{IEEEproc}
  \documentclass[twoside,a4paper]{artikel1}
  ```

The name of the games

- 'TEX' is "tek", 'LATEX' is "lay-tek" or "lah-tek"
- TEX is the basic system
- LATEX is a macro package on top of it

Aims of LATEX

- Foremost: scientific documents
- But also classes for letters, vitae, plays
- Excellent math typesetting
- Customizable, extendable
- Not all that great for fancy layouts

LATEX styles

- Commands for document structuring

  ```
  \section{Introduction}
  \subsection{Prior research}
  ```

- Tools for making new structure constructs

  ```
  \newtheorem{corollary}
  ```

- Lower level tools

Math typesetting

- Sophisticated algorithms
- Math fonts with many parameters

$$\sqrt[3]{\frac{B}{1-A^2_{j_1,j_2}}}+\cdots+1=\int_1^\infty \widehat{\sin t}\,\mathrm{d}t$$

- Large number of weird symbols

You will learn

- LATEX commands for everyday use
- Ways of customizing LATEX

Short history of TEX

- Thought up by Donald Knuth in 1978
- as a summer project for his students
- First real implementation in 1981
- TEX2 in 1985, revision TEX3 in 1991, now frozen
- Omega, pdftex

Features of the TEX language

- Macro language

  ```
  \def\theorem{\newline \bold Theorem \theoremcounter}
  \theorem This is good
  ```

- Dynamically changable syntax
- Many low-level constructs

You will learn

- Boxes, glue, paragraph parameters
- Fancy macro programming

Lexical analysis

- Recognize words, numbers and such
- Used as basic blocks for grammar
- Finite State Automaton usually sufficient

Syntactical analysis

- Recognize statements and constructs
- Translate ‘meaning’ into internal representation
- Pushdown Automaton usually sufficient

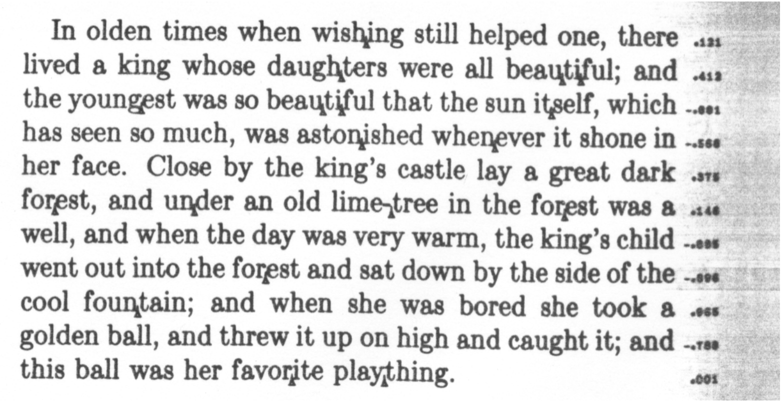
You will learn

- Recap of automata theory (FSA, PDA)
- Applications of automata in programming language parsing
- *lex* and *yacc* unix tools
- Hashing

Paragraph breaking

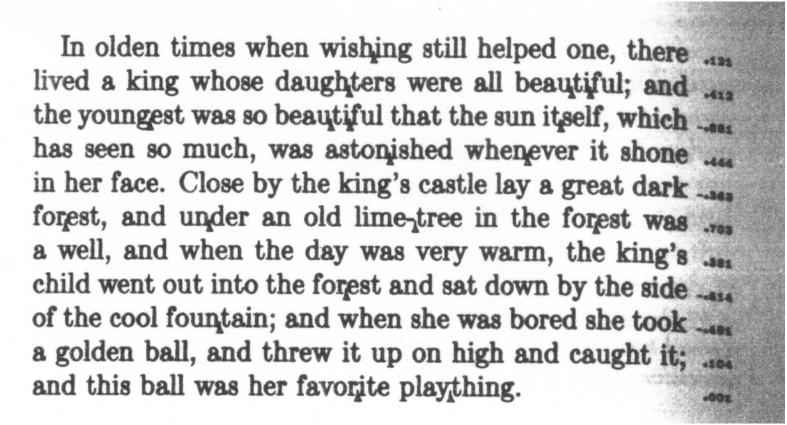
- For right-justified paragraph:
- Compress some lines, stretch others, use hyphenation
- Aim for even ‘colour’, avoid consecutive hyphens, rivers, et cetera
- With n words, 2^n breakpoints: efficient algorithm needed

Naive ‘first fit’ breaking



In olden times when wishing still helped one, there .121
lived a king whose daughters were all beautiful; and .412
the youngest was so beautiful that the sun itself, which -.881
has seen so much, was astonished whenever it shone in -.566
her face. Close by the king’s castle lay a great dark .375
forest, and under an old lime-tree in the forest was a .146
well, and when the day was very warm, the king’s child -.895
went out into the forest and sat down by the side of the -.896
cool fountain; and when she was bored she took a .965
golden ball, and threw it up on high and caught it; and -.789
this ball was her favorite plaything. .001

Sophisticated line breaking



In olden times when wishing still helped one, there .131
lived a king whose daughters were all beautiful; and .412
the youngest was so beautiful that the sun itself, which −.881
has seen so much, was astonished whenever it shone .444
in her face. Close by the king's castle lay a great dark −.363
forest, and under an old lime-tree in the forest was .703
a well, and when the day was very warm, the king's .381
child went out into the forest and sat down by the side −.814
of the cool fountain; and when she was bored she took −.481
a golden ball, and threw it up on high and caught it; .104
and this ball was her favorite plaything. .001

TEX's paragraph algorithm

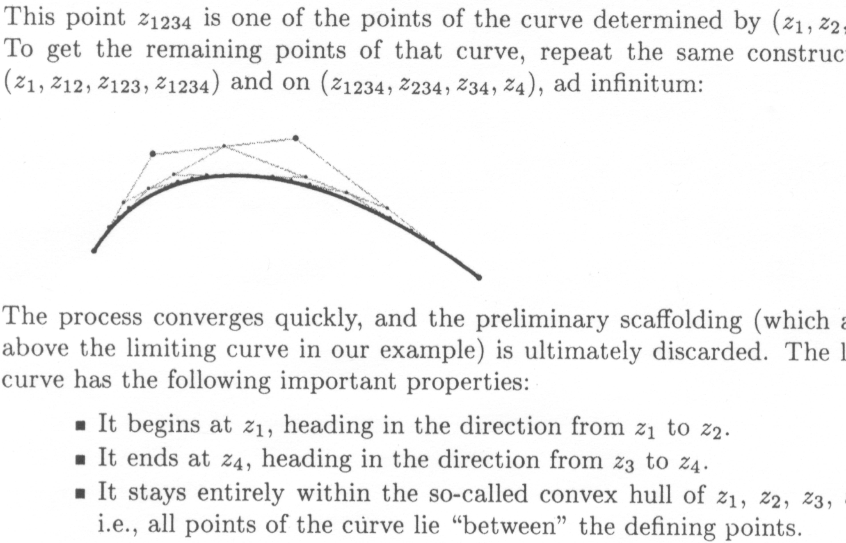
- Dynamic programming
- Small number of possibilities considered
- Fast running time

You will learn

- Dynamic programming
- NP-completeness
- Python

Metafont

- Knuth also wrote a program to design fonts with: Metafont
- Based on splines

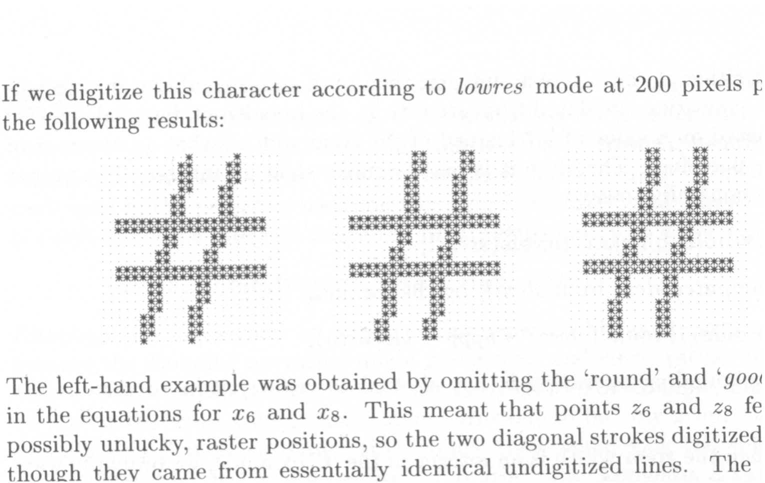


This point z_{1234} is one of the points of the curve determined by $(z_1, z_2$
To get the remaining points of that curve, repeat the same construc
$(z_1, z_{12}, z_{123}, z_{1234})$ and on $(z_{1234}, z_{234}, z_{34}, z_4)$, ad infinitum:

The process converges quickly, and the preliminary scaffolding (which
above the limiting curve in our example) is ultimately discarded. The
curve has the following important properties:

- It begins at z_1, heading in the direction from z_1 to z_2.
- It ends at z_4, heading in the direction from z_3 to z_4.
- It stays entirely within the so-called convex hull of z_1, z_2, z_3,
  i.e., all points of the curve lie "between" the defining points.

Raster graphics



If we digitize this character according to *lowres* mode at 200 pixels p the following results:

The left-hand example was obtained by omitting the 'round' and '*goo* in the equations for x_6 and x_8. This meant that points z_6 and z_8 fe possibly unlucky, raster positions, so the two diagonal strokes digitized though they came from essentially identical undigitized lines. The

You will learn

- Interpolation theory
- Splines
- Issues in raster graphics

TEX's expansion mechanism

- TEX commands are of two kinds: expansion and execution
- The expansion mechanism is strong enough to implement lambda calculus

You will learn

- Some foundations of mathematics

What's in an input file

- Plain Ascii?
- The problem with funny languages

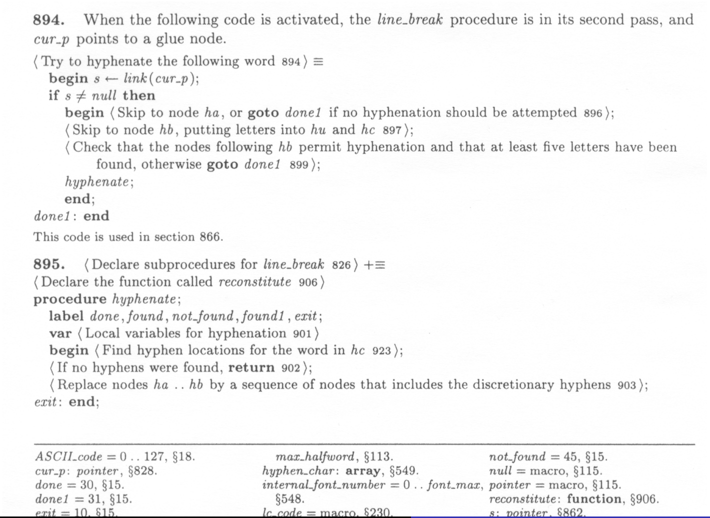
You will learn

- A history of character encodings
- Unicode
- Font organization

Yet another Knuth product

- The WEB system for literate programming
- Write code and documentation together

Source pretty printing



894. When the following code is activated, the *line_break* procedure is in its second pass, and *cur_p* points to a glue node.

```
⟨Try to hyphenate the following word 894⟩ ≡
   begin s ← link(cur_p);
   if s ≠ null then
      begin ⟨Skip to node ha, or goto done1 if no hyphenation should be attempted 896⟩;
      ⟨Skip to node hb, putting letters into hu and hc 897⟩;
      ⟨Check that the nodes following hb permit hyphenation and that at least five letters have been
            found, otherwise goto done1 899⟩;
      hyphenate;
      end;
done1: end
```

This code is used in section 866.

895. ⟨Declare subprocedures for *line_break* 826⟩ +≡

```
⟨Declare the function called reconstitute 906⟩
procedure hyphenate;
   label done, found, not_found, found1, exit;
   var ⟨Local variables for hyphenation 901⟩
   begin ⟨Find hyphen locations for the word in hc 923⟩;
   ⟨If no hyphens were found, return 902⟩;
   ⟨Replace nodes ha . . hb by a sequence of nodes that includes the discretionary hyphens 903⟩;
exit: end;
```

ASCII_code = 0 . . 127, §18.
cur_p: *pointer*, §828.
done = 30, §15.
done1 = 31, §15.
exit = 10, §15.
max_halfword, §113.
hyphen_char: **array**, §549.
internal_font_number = 0 . . *font_max*, §548.
lc_code = macro, §230.
not_found = 45, §15.
null = macro, §115.
pointer = macro, §115.
reconstitute: **function**, §906.
s: *pointer*, §862.

You will learn

- Literate programming
- WEB and noweb

You will learn

- History of TEX
- Knuth's notions of development
- The 'torture test' idea
- Competing notions