Breaking Paragraphs into Lines

Presenter: Ananya

17th July 2018
Agenda

➔ Introduction
➔ Formulating the Problem of Line Breaking
➔ Desirability Criterias
  ◆ Adjustment Ratios
  ◆ Badness
  ◆ Demerits
➔ Intuition of Algorithm
➔ Improvement
➔ Complete Algorithm
➔ Computational Experience
Introduction

The most important operation when text materials are prepared is to divide long paragraphs into individual lines.

Text of the line should match the width it’s given as best as possible.

Some desirable properties -

- Avoid large differences between the interword spaces in lines, so that a loose line is not followed directly by a tight line.

- Hyphenate words, but do it sparingly.

- Avoid hyphenating two adjacent lines or second-to-last line of a paragraph.
Simple/Greedy Way

All of these things are desirable, but many line breaking implementations consider only a few, or none, of these qualities.

Standard Line Breaking Algorithm (Barnett)-

“Keep appending words to the current line, assuming the normal spacing, until reaching a word that does not fit. Break after this word, if it is possible to do so without compressing the spaces to less than the given minimum; otherwise break before this word, if it is possible to do so without expanding the spaces to more than the given maximum. Otherwise hyphenate the offending word, putting as much of it on the current line as will fit; if no suitable hyphenation points can be found, this may result in a line whose spaces exceed the given maximum.”
Better Way

• Choosing a particular line breaking opportunity affects all subsequent lines and can have a negative effect on the entire paragraph.

• Significant improvements are possible if the computer takes advantage of its opportunity to ‘look ahead’ at what is coming later in the paragraph, before making a final decision about where any of the lines will be broken.

• Instead of optimizing each line independently, it would be better if we tried to optimize the whole paragraph.
Formulating the Problem

Paragraph is a sequence $x_1, x_2 \ldots x_m$ of $m$ items, where each individual item $x_i$ is either a box, a glue, or a penalty specification.

- **Box** - Unbreakable piece of content with fixed width $w_i$.
- **Glue** - Represents space between boxes. A glue element consists of optimal width $w_i$, stretchability $y_i$ and shrinkability $z_i$.
- **Penalty** - Represents potential places to end one line and begin another. When item $x_i$ specifies a penalty, a number $p_i$ to decide whether or not to end a line at this point: positive $p_i$ cost suggests not to use that break point, a negative cost signals an appealing break point. Penalty specification consists of width $w_i$ (incase there is need to add typeset material) and flag $f_i$ (to detect two consecutive hyphenations)
Example

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

<table>
<thead>
<tr>
<th>(x_1) = empty box for indentation</th>
<th>(w_1 = 18)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x_2) = box for ‘I’</td>
<td>(w_2 = 6)</td>
</tr>
<tr>
<td>(x_3) = box for ‘n’</td>
<td>(w_3 = 10)</td>
</tr>
<tr>
<td>(x_4) = glue for interword space</td>
<td>(w_4 = 6), (y_4 = 3)</td>
</tr>
<tr>
<td>(x_5) = box for ‘o’</td>
<td>(w_5 = 9)</td>
</tr>
<tr>
<td>(x_{309}) = box for ‘I’</td>
<td>(w_{309} = 5)</td>
</tr>
<tr>
<td>(x_{310}) = box for ‘i’</td>
<td>(w_{310} = 5)</td>
</tr>
<tr>
<td>(x_{311}) = box for ‘m’</td>
<td>(w_{311} = 15)</td>
</tr>
<tr>
<td>(x_{312}) = box for ‘e’</td>
<td>(w_{312} = 8)</td>
</tr>
<tr>
<td>(x_{313}) = box for ‘-’</td>
<td>(w_{313} = 6)</td>
</tr>
<tr>
<td>(x_{314}) = penalty for explicit hyphen</td>
<td>(w_{314} = 0), (\rho_{314} = 50), (f_{314} = 1)</td>
</tr>
<tr>
<td>(x_{315}) = box for ‘t’</td>
<td>(w_{315} = 7)</td>
</tr>
<tr>
<td>(x_{592}) = box for ‘y’</td>
<td>(w_{592} = 10)</td>
</tr>
<tr>
<td>(x_{593}) = penalty for optional hyphen</td>
<td>(w_{593} = 6), (\rho_{593} = 50), (f_{593} = 1)</td>
</tr>
<tr>
<td>(x_{594}) = box for ‘t’</td>
<td>(w_{594} = 7)</td>
</tr>
<tr>
<td>(x_{595}) = box for ‘h’</td>
<td>(w_{595} = 10)</td>
</tr>
<tr>
<td>(x_{596}) = box for ‘i’</td>
<td>(w_{596} = 5)</td>
</tr>
<tr>
<td>(x_{597}) = box for ‘n’</td>
<td>(w_{597} = 10)</td>
</tr>
<tr>
<td>(x_{598}) = box for ‘g’</td>
<td>(w_{598} = 9)</td>
</tr>
<tr>
<td>(x_{599}) = box for ‘’</td>
<td>(w_{599} = 5)</td>
</tr>
<tr>
<td>(x_{600}) = finishing glue</td>
<td>(w_{600} = 0), (y_{600} = \infty), (z_{600} = 0)</td>
</tr>
<tr>
<td>(x_{601}) = forced break</td>
<td>(w_{601} = 0), (\rho_{601} = -\infty), (f_{601} = 1)</td>
</tr>
</tbody>
</table>
Legal Breakpoint

This are the only places where a paragraph may be broken; \( x_i \) is a legal breakpoint if and only if:

1. it is a penalty item with penalty \(< +\infty\);
2. it is a glue item and \( x_{i-1} \) is a box item.

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

The example has 129 legal breakpoints giving \( 2^{129} \) ways of breaking

Most of the line breaking choices would be absurd.
Desirability Criteria - Adjustment Ratio

**Adjustment Ratio (r):** How much the line has to be shrinked \((r < 0)\) or stretched \((r > 0)\) to fit in the given length. \(r\) may not be \(-1\) or greater than a given threshold.

- If \(L_j = l_j\) (a perfect fit), let \(r_j = 0\).
- If \(L_j < l_j\) (a short line), let \(r_j = (l_j - L_j)/Y_j\), assuming that \(Y_j > 0\); the value of \(r_j\) is undefined if \(Y_j \leq 0\) in this case.
- If \(L_j > l_j\) (a long line), let \(r_j = (l_j - L_j)/Z_j\), assuming that \(Z_j > 0\); the value of \(r_j\) is undefined if \(Z_j \leq 0\) in this case.
First Fit Algorithm

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

**Badness ($\beta_j$):** It is related to the adjustment ratio of a line: It is nearly zero when $|r_j|$ is small but grows rapidly when $|r_j|$ takes values exceeding 1.

$$\beta_j = \begin{cases} \infty, & \text{if } r_j \text{ is undefined or } r_j < -1; \\ 100|r_j|^3, & \text{otherwise}. \end{cases}$$
Best Fit Algorithm

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

Each break is chosen so as to minimize the ‘badness plus penalty’ of that line.

Greedy Algorithm

Minimum possible value of $\beta_j + \pi_j$
Desirability Criteria - Demerits

**Demerits** ($\delta_j$): Optimum way to choose the breakpoints. It consists of Badness rating $\beta_j$ and penalty $\pi_j$. It also considers additional penalty $\alpha_j$ assessed for consecutive hyphenated lines.

\[
\delta_j = \begin{cases} 
(1 + \beta_j + \pi_j)^2 + \alpha_j, & \text{if } \pi_j \geq 0; \\
(1 + \beta_j)^2 - \pi_j^2 + \alpha_j, & \text{if } -\infty < \pi_j < 0; \\
(1 + \beta_j)^2 + \alpha_j, & \text{if } \pi_j = -\infty.
\end{cases}
\]
Optimum Fit Algorithm

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

The algorithm is globally optimum in the sense of having fewest total ‘demerits’ over all choices of breakpoints.

Finds the best choice of breakpoints by minimizing the sum of $\delta_j$ over all lines $j$. 
All 3 algorithms with line width 500 units

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

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

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

a) First Fit
b) Best Fit
c) Optimum Fit
## Demerit per line

<table>
<thead>
<tr>
<th>First fit</th>
<th>Best fit</th>
<th>Optimum fit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1734</td>
<td>1734</td>
<td>2357</td>
</tr>
<tr>
<td>4692</td>
<td>4692</td>
<td>6</td>
</tr>
<tr>
<td>3440</td>
<td>3440</td>
<td>938</td>
</tr>
<tr>
<td>3066</td>
<td>9</td>
<td>212</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>22</td>
<td>2</td>
</tr>
<tr>
<td>276</td>
<td>210</td>
<td>27</td>
</tr>
<tr>
<td>5</td>
<td>24</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>476</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>13218</td>
<td>10143</td>
<td>4030</td>
</tr>
</tbody>
</table>
Intuition of Algorithm

FEASIBLE BREAKPOINT - a place where the text of the paragraph from the beginning to this point can be broken into lines whose adjustment ratio does not exceed a given tolerance.

The Algorithm

● locates all of the feasible breakpoints and remembers the best way to get to each one, in the sense of fewest total demerits.
● Maintains list of ‘active’ breakpoints, representing all of the feasible breakpoints.
● A potential breakpoint \( b \) is considered feasible if there exists any active breakpoint \( a \) such that the line from \( a \) to \( b \) has an acceptable adjustment ratio.
● \( b \) is appended to the active list.
● Remembers the identity of the breakpoint \( a \) that minimizes the total demerits.
● Breakpoint \( a \) is removed from the active list for which the line from \( a \) to \( b \) has an adjustment ratio less than -1.
In olden times when wishing still helped one, there lived a king whose daughters were all beautiful; and the youngest was so beautiful that the sun itself, which has seen so much, was astonished whenever it shone in her face. Close by the king’s castle lay a great dark forest, and under an old lime-tree in the forest was a well, and when the day was very warm, the king’s child went out into the forest and sat down by the side of the cool fountain; and when she was bored she took a golden ball, and threw it up on high and caught it; and this ball was her favorite plaything.
Examples of line breaking with different line sizes

The area of a circle is a mean proportional between any two regular and similar polygons of which one circumscribes it and the other is isometric with it. In addition, the area of the circle is less than that of any circumscribed polygon and greater than that of any isometric polygon. And further, of these circumscribed polygons, the one that has the greater number of sides has a smaller area than the one that has a lesser number; but, on the other hand, the isometric polygon that has the greater number of sides is the larger.

I turn, in the following treatises, to various uses of those triangles whose generator is unity. But I leave out many more than I include; it is extraordinary how fertile in properties this triangle is. Everyone can try his hand.
Further Improvement

Contrast between tight and loose lines can make lines appear worse.

\TeX\'S algorithm for line breaking recognizes four kinds of lines:

- Class 0 (tight lines), where $-1 \leq r < -0.5$;
- Class 1 (normal lines), where $-0.5 \leq r < +0.5$;
- Class 2 (loose lines), where $+0.5 \leq r < +1$;
- Class 3 (very loose lines), where $r \geq +1$.

Additional demerits are added when adjacent lines are not of the same or adjacent class.
Further Improvement

Looseness Parameter (q) - an integer such that the total number of lines produced for the paragraph is as close as possible to q plus the optimum number, without violating the conditions of feasibility.

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

q=-1

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

q=0

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

q=+1

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

q=+2
We want to have a data structure that makes this algorithm efficient.
Data Structure Fields

position(a) = index of breakpoint represented by this node (0 = start of paragraph)
line(a) = number of the line ending at this breakpoint;
fitness(a) = fitness class of the line ending at this breakpoint;
totalwidth(a) = (Σw)_{after(a)}, used to calculate adjustment ratios;
totalstretch(a) = (Σy)_{after(a)}, used to calculate adjustment ratios;
totalshrink(a) = (Σz)_{after(a)}, used to calculate adjustment ratios;
totaldemerits(a) = minimum total demerits up to this breakpoint;
previous(a) = pointer to the best node for the preceding breakpoint;
link(a) = pointer to the next node in the list.
create an active node representing the beginning of the paragraph =
begin A := new node (position = 0, line = 0, fitness = 1,
totalwidth = 0, totalstretch = 0, totalshrink = 0,
totaldemerits = 0, previous = Λ, link = Λ);

P := Λ;
end.

operation ‘for b := 1 to m do <if b is a legal breakpoint> then <main loop>’

\[ L := \Sigma W - \text{totalwidth}(a); \]
if \( t_b = \text{‘penalty’} \) then \( L := L + w_b; \)
\( j := \text{line}(a) + 1; \)
if \( L < l_j \) then
begin \( Y := \Sigma Y - \text{totalstretch}(a); \)
if \( Y > 0 \) then \( r := (l_j - L)/Y \) else \( r := \infty; \)
end
else if \( L > l_j \) then
begin \( Z := \Sigma Z - \text{totalshrink}(a); \)
if \( Z > 0 \) then \( r := (l_j - L)/Z \) else \( r := \infty; \)
end
else \( r := 0. \)

\[ \Sigma W := \Sigma Y := \Sigma Z := 0; \]
for \( b := 1 \) to \( m \) do
\begin{align*}
\text{if } t_b & = \text{‘box’} \text{ then } \Sigma W := \Sigma W + w_b \\
\text{else if } t_b & = \text{‘glue’} \text{ then } \\
\begin{align*}
\text{begin if } t_{b-1} & = \text{‘box’} \text{ then } \langle \text{main loop} \rangle; \\
\Sigma W & := \Sigma W + w_b; \Sigma Y := \Sigma Y + y_b; \Sigma Z := \Sigma Z + z_b; \\
\end{align*}
\text{end}
\text{else if } p_b & \neq +\infty \text{ then } \langle \text{main loop} \rangle.
\end{align*}

operation ‘compute the adjustment ratio \( Y \) from \( a \) to \( b \)’
Complete Algorithm

\[ \text{create an active node representing the beginning of the paragraph} \]

\[ \text{begin } A := \text{new node(position = 0, line = 0, fitness = 1),} \]
\[ \text{totalwidth = 0, totalstretch = 0, totalshrink = 0,} \]
\[ \text{totaldemerits = 0, previous = } \Lambda, \text{ link = } \Lambda; \]
\[ P := \Lambda; \]
\[ \text{end.} \]

\[ \text{begin } a := A; \text{ preva := } \Lambda; \]
\[ \text{loop: } D_0 := D_1 := D_2 := D_3 := D := +\infty; \]
\[ \text{loop: } \text{nexta} := \text{link(a)}; \]
\[ \text{compute the adjustment ratio } r \text{ from } a \text{ to } b; \]
\[ \text{if } r < -1 \text{ or } p_h = -\infty \text{ then } \langle \text{deactivate node } a \rangle \text{ else } \text{preva} := a; \]
\[ \text{if } -1 \leq r \leq p \text{ then } \]
\[ \text{begin } \text{compute demerits } d \text{ and fitness class } c; \]
\[ \text{if } d < D, \text{ then } \]
\[ \text{begin } D_c := d; \text{ } A_c := a; \text{ if } d < D \text{ then } D := d; \]
\[ \text{end; } \]
\[ \text{end; } \]
\[ a := \text{nexta}; \text{ if } a = \Lambda \text{ then exit loop; } \]
\[ \text{if line(a)} \geq j \text{ and } j < j_0 \text{ then exit loop; } \]
\[ \text{repeat; } \]
\[ \text{if } D < \infty \text{ then } \langle \text{insert new active nodes for breaks from } A_c \text{ to } b \rangle; \]
\[ \text{if } a = \Lambda \text{ then exit loop; } \]
\[ \text{repeat; } \]
\[ \text{if } A = \Lambda \text{ then } \langle \text{do something drastic since there is no feasible solution} \rangle; \]
\[ \text{end.} \]

\[ \text{compute demerits } d \text{ and fitness class } c; \]
\[ \text{begin if } p_h > 0 \text{ then } d := (1 + 100 | r |^3 + p_h)^2 \]
\[ \text{else if } p_h = -\infty \text{ then } d := (1 + 100 | r |^3 - p_h) \]
\[ \text{else } d := (1 + 100 | r |^3); \]
\[ d := d + s \cdot f_k \cdot f_{\text{position;}} \]
\[ \text{if } r < -0.5 \text{ then } c := 0 \]
\[ \text{else if } r \leq 0.5 \text{ then } c := 2 \text{ else } c := 3; \]
\[ \text{if } |c - \text{fitness(a)}| > 1 \text{ then } d := d + 7; \]
\[ d := d + \text{totaldemerits(a);} \]
\[ \text{end.} \]

\[ \text{insert new active nodes for breaks from } A_c \text{ to } b; \]
\[ \text{begin } \langle \text{compute } tw = (\Sigma x)_{\text{stretch}}, ty = (\Sigma y)_{\text{stretch}} \rangle \text{ and } tz = (\Sigma z)_{\text{stretch}}; \]
\[ \text{for } c := 0 \text{ to } 3 \text{ do if } D_c < D + 7 \text{ then } \]
\[ \text{begin } s := \text{new node(position = } b, \text{ line = line(A_c) + 1, fitness = } c, \]
\[ \text{totalwidth = } tw, \text{ totalstretch = } ty, \text{ totalshrink = } tz, \]
\[ \text{totaldemerits = } D_c, \text{ previous = } A_c, \text{ link = } a; \]
\[ \text{if } \text{preva} = \Lambda \text{ then } A = d \text{ else } \text{link(preva):=} s; \]
\[ \text{preva:}= s; \]
\[ \text{end; } \]

\[ \text{compute } tw = (\Sigma x)_{\text{stretch}}, ty = (\Sigma y)_{\text{stretch}} \rangle \text{ and } tz = (\Sigma z)_{\text{stretch}}; \]
\[ \text{begin } tw := \Sigma W; ty := \Sigma Y; tz := \Sigma Z; i := b; \]
\[ \text{loop: if } i > m \text{ then exit loop; } \]
\[ \text{if } f_i = \text{‘box’} \text{ then exit loop; } \]
\[ \text{if } f_i = \text{‘glue’} \text{ then } \]
\[ \text{begin } tw := tw + w; ty := ty + y; tz := tz + z; \]
\[ \text{end } \]
\[ \text{else if } p_i = -\infty \text{ and } i > b \text{ then exit loop; } \]
\[ i := i + 1; \]
\[ \text{repeat; } \]
\[ \text{end; } \]

\[ \langle \text{deactivate node } a \rangle = \]
\[ \text{begin if } \text{preva} = \Lambda \text{ then } A := \text{nexta else link(preva):=} \text{nexta; } \]
\[ \text{lin}k(a):= P; P := a; \]
\[ \text{end; } \]
Complete Algorithm

\[\text{choose the active node with fewest total demerits} =\]
\[
\text{begin } a := b := A; d := \text{totaldemerits}(a); \\
\text{loop: } a := \text{link}(a); \\
\text{if } a = \Lambda \text{ then exit loop;}
\]
\[
\text{if totaldemerits}(a) < d \text{ then }
\begin{align*}
\text{begin } d := & \text{totaldemerits}(a); b := a; \\
\text{end}; \\
\text{repeat;}
\end{align*}
\text{k := line(b);}
\text{end.}
\]

\[\text{choose the appropriate active node} =\]
\[
\text{begin } a := A; s := 0; \\
\text{loop: } \delta := \text{line}(a) - k; \\
\text{if } q \leq \delta < s \text{ or } s < \delta \leq q \text{ then }
\begin{align*}
\text{begin } s := & \delta; d := \text{totaldemerits}(a); b := a; \\
\text{end;}
\end{align*}
\text{else if } \delta = s \text{ and totaldemerits}(a) < d \text{ then }
\begin{align*}
\text{begin } d := & \text{totaldemerits}(a); b := a; \\
\text{end;}
\end{align*}
\text{a := link(a); if } a = \Lambda \text{ then exit loop;}
\text{repeat;}
\text{k := line(b);}
\text{end.}
\]
Computational Experience

- The algorithm can be complex or simple depending on different situations in typesetting.
- Naively implemented, this algorithm is $O(n^2)$. But optimization can improve the running time.
- Given a line candidate’s ending position, there are only a few candidate starting positions which are not terrible. We can limit our search to only the prior opportunities which are not terrible.
- If we choose an appropriate terribleness-threshold, our algorithm is $O(n)$.
- If threshold is too high, algorithm takes much longer to run (bounding on $O(n^2)$), but it may find a more globally-correct solution.
- If threshold is too low, the algorithm will find that it is impossible to find acceptable line breaks all the way to the end of the paragraph.
Thank you