Billboard problem
"(x_1, x_2, \ldots, x_n), (v_1, v_2, \ldots, v_n)" mile markers

viewership \( v_i \) = \# people that view billboard @ \( x_i \).

distance parameter

cannot place ads that are closer than D miles apart

goal: is to maximize viewership for an acceptable campaign
Input is \(((x_1, \ldots, x_n)(v_1, \ldots, v_n), D)\)

\[
\text{Best}_n = \max \text{ viewership for an acceptable campaign that uses the first } n \text{ billboards}
\]

\[
\text{Best}_j = \max \left\{ \text{Best}_{j-1}, V_j + \text{Best}_{\text{closest billboard that is } D \text{ away buddy}(j)} \right\}
\]
$\text{Best}_1 = V_j = V_1$

$\text{Best}_2 = \max \left\{ \text{Best}_1, 0, V_2 + \text{Best}_{\text{buddy}}(2) = V_2 \right\}$

$\text{Best}_3 = \max \left\{ \text{Best}_2, V_3 + \text{Best}_{\text{buddy}}(3) = V_3 + \text{Best}_1 \right\}$
Billboard Problem

\[
\begin{cases}
\text{BEST}_j = \max \{ \text{BEST}_{j-1}, v_j + \text{BEST}_{cl(j)} \}
\end{cases}
\]

best[0] = 0

for i=1 to n

cl = i-1

while ( \text{dist}(x[cl], x[ci]) \leq D )

\[
\text{best}[i] = \max \{ \text{best}[i-1], v[ci] + \text{best}[cl] \}
\]

return best[n]
Billboard Problem

\[ \text{BEST}_j = \max \left\{ \text{BEST}_{j-1}, v_j + \text{BEST}_{\text{cl}(j)} \right\} \]

runtime: \( \Theta(n^2) \)

best[0] = 0

for i=1 to n \( \leq n \) iterations

\[ \text{cl} = i-1 \]

while( (x[i]-x[cl]) < D \&\& cl>0 ) \( \text{cl} = \text{cl}-1 \) \( \leq \Theta(n) \)

\[ \text{best}[i] = \max(\text{best}[i-1], v_j + \text{best}[\text{cl}]) \]

return best[n]
Pre-process to find every board’s buddy.

right = n, left = n
Pre-process to find every board’s buddy.

right = n, left = n

move left until $\text{dist}(x[\text{right}], x[\text{left}]) > D$

$b[10] = 8$
Pre-process to find every board’s buddy.

right = n, left = n

$\rightarrow$ move left until dist(x[right], x[left]) > D
buddy[right] = left
move right to right
Pre-process to find every board’s buddy.

\[
\text{right} = n, \text{left} = n
\]

while right and left are valid

move left until \(\text{dist}(x[\text{right}], x[\text{left}]) > D\)

\[
\text{buddy}[\text{right}] = \text{left}
\]

move right to right
Pre-process to find every board’s buddy.

right = n, left = n

while right and left are valid

move left until dist(x[right], x[left]) > D

buddy[right] = left

move right to right the left

handle any leftover right

b[10]=8
<Preprocess buddies> – $\Theta(n)$
best[0] = 0
for i=1 to n
    cl = i-1
    while( (x[i]-x[cl])< D && cl>0 ) cl = cl - 1
    best[i] = max(best[i-1], v[j]+best[buddy[i]])
return best[n]
It was the best of times, it was the worst of times, it was the age of wisdom, it was the age of foolishness, it was the epoch of belief, it was the epoch of incredulity, it was the season of Light, it was the season of Darkness, it was the spring of hope, it was the winter of despair, we had everything before us, we had nothing before us, we were all going direct to heaven, we were all going direct the other way - in short, the period was so far like the present period, that some of its noisiest authorities insisted on its being received, for good or for evil, in the superlative degree of comparison only.
It was the best of times, it was the worst of times, it was the age of wisdom, it was the age of foolishness, it was the epoch of belief, it was the epoch of incredulity, it was the season of Light, it was the season of Darkness, it was the spring of hope, it was the winter of despair, we had everything before us, we had nothing before us, we were all going direct to heaven, we were all going direct the other way - in short, the period was so far like the present period, that some of its noisiest authorities insisted on its being received, for good or for evil, in the superlative degree of comparison only.
First rule of typesetting

never print in the margin!

are simply not allowed
It was the best of times, it was the worst of times, it was the age of wisdom, it was the age of foolishness, it was the epoch of belief, it was the epoch of incredulity, it was the season of Light, it was the season of Darkness, it was the spring of hope, it was the winter of despair, we had everything before us, we had nothing before us, we were all going direct to heaven, we were all going direct the other way - in short, the period was so far like the present period, that some of its noisiest authorities insisted on its being received, for good or for evil, in the superlative degree of comparison only.

\[ \text{minimize the overall penalty for } \text{slack}^{1,2} \]
Second rule of typesetting

avoid big ugly whitespaces (slack)
do not typeset in margin

don't typeset in margin

typeset every word

minimize the slack

between margin and last word on a line

one paragraph at a time
It was the best of times, it was the worst of times, it was the age of wisdom, it was the age of foolishness, it was the epoch of belief, it was the epoch of incredulity, it was the season of Light, it was the season of Darkness, it was the spring of hope, it was the winter of despair, we had everything before us, we had nothing before us, we were all going direct to heaven, we were all going direct to hell.
It was the best of times, it was the worst of times, it was the age of wisdom, it was the age of foolishness, it was the epoch of belief, it was the epoch of incredulity, it was the season of Light, it was the season of Darkness, it was the spring of hope, it was the winter of despair, we had everything before us, we had nothing before us, we were all going direct to heaven, we were all going direct
Typesetting problem

input: \[ W = \{w_1, w_2, w_3, \ldots, w_n\} \quad M \text{ margin} \]

sequence of word lengths

output: \[ L = (w_1, \ldots, w_{\ell_1}), (w_{\ell_1+1}, \ldots, w_{\ell_2}), \ldots, (w_{\ell_{n+1}}, \ldots, w_n) \]

such that all words typeset

\[ \sum_{\text{each line}} w_i \leq M \]

\[ \text{minimize } \sum_{\text{lines}} \left( M - \left( \sum_{\text{line } j} \text{ (sum of words on line } j \text{ )} \right) - \# \text{ of words on line } j \right)^2 \]
Typesetting problem

input: \[ W = \{w_1, w_2, w_3, \ldots, w_n\} \]

output: \[ L = (\widehat{w}_1, \ldots, w_{\ell_1}), (\widehat{w}_{\ell_1+1}, \ldots, w_{\ell_2}), \ldots, (\widehat{w}_{\ell_x+1}, \ldots, w_n) \]

such that
\[ c_i = \left( \sum_{j=\ell_i+1}^{\ell_{i+1}} |w_j| \right) + (\ell_{i+1} - \ell_i - 1) \]

\[ c_i \leq M \quad \forall i \]

\[ \min \sum (M - c_i)^2 \]

length of the \( i \)th line in the typesetting

spaces b/w words

\[ \ell_{i+1} \quad \ell_i + 1 \]

\[ i = \ell_i + 1 \]
how to solve

define the right variable:

\[ \text{Best}_n: \text{minimum penalty for typesetting the first } n \text{ words} \]
imagine optimal solution

very last line that the optimal solution typesets
imagine optimal solution

<table>
<thead>
<tr>
<th>First word of the last line</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sjim</td>
</tr>
</tbody>
</table>

Last word before "fwool"

Last line

Slack induced by typesetting word j as "fwool" far first n words.
some word has to be the first-word-of-last-line (fwoll)
Imagine optimal solution

FWoll is \( w_k \)

\( S_{k,n} \) slack when line starts with \( w_k \)

\( \text{last line} \)

\[
\text{Best}_n = \text{Best}_{\text{fwoll} - 1} + \frac{S_{\text{fwoll}, n}}{2}
\]

i.e. penalty for typsetting the earlier part of paragraph.
imagine optimal solution

$w_\ell$ is slack when line starts with $w_\ell$

\[ \text{BEST}_n = \text{BEST}_{\ell-1} + S_{\ell,n}^2 \]
how many candidates are there for the full?
is $w_1$ full?

there is no slack (no solution even) because words go beyond edge!

define $S_{1,n} = \infty$ if this happens
Is $w_2$ follow?

$S_{2,n} = \infty$
is \( w_i \), fwoll?
which word is fwoll?

\[
\text{BEST}_n = \min \left\{ \right. \right. 
\]
which word is fwoll?

\[
\text{BEST}_n = \min \left\{ \begin{array}{l}
\text{BEST}_0 + S_{1,n}^2 \\
\text{BEST}_1 + S_{2,n}^2 \\
\text{BEST}_2 + S_{3,n}^2 \\
\vdots \\
\text{BEST}_{\ell-1} + S_{\ell,n}^2 \\
\text{BEST}_{n-2} + S_{n-1,n}^2 \\
\text{BEST}_{n-1} + S_{n,n}^2 \\
\end{array} \right. \\
\min_{f=1} \left( \text{Best}_{f-1} + (S_{f,n})^2 \right)
\]

typesetting algorithm
typesetting algorithm

( make table for \( S_{i,j} \) slack for typesetting word \( i \) as full as \( j \) word )

for \( i=1 \) to \( n \)

\[
\text{best}[i] = \min \{ \text{best}[j] + s[j+1][i^2] \}
\]
typesetting algorithm

make table for $S_{i,j}$

\[
\begin{cases}
\text{for } i=1 \text{ to } n \\
\text{best}[i] = \min\{ \text{best}[j] + s[j+1][i]^2 \}
\end{cases}
\]

```java
// compute best_0,...,best_n
int best[] = new int[n+1];
int choice[] = new int[n+1];
best[0] = 0;
for(int i=1;i<=n;i++) {
    int min = Integer.MAX_VALUE;
    int ch = 0;
    for(int j=0;j<i;j++) {
        int t = best[j] + S[j+1][i]*S[j+1][i];
        if (t<min) { min = t; ch = j; }
    }
    best[i] = min;
    choice[i] = ch;
}
```
how to compute $S_{i,j}$

slack when line starts with $w_i$ and ends $w_j$
Simplest case

\[ S_{1.1} = M - w_i \]

slack when line starts with \( w_i \) and ends \( w_i \)
Simplest case

$S_{1,1}$

$S_{1,2}$

slack when line starts with $w_1$ and ends $w_2$

$S_{12} = S_{11} - w_2 - 1$
how to compute \( S_{i,j} \)

slack when line starts with \( w_i \) and ends \( w_j \)

\[
S_{ij} = S_{ij-1} - w_{j-1}
\]

if \( S_{ij} < 0 \),
set
\[
S_{ij} = 0
\]
How to compute S

\[
\begin{array}{cccccc}
S_{1,1} & S_{1,2} & S_{1,3} & \cdots & S_{1,n} \\
S_{2,1} & S_{2,2} & S_{2,3} & \cdots & \ \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
S_{n,1} & S_{n,2} & S_{n,3} & \cdots & S_{n,n} \\
\end{array}
\]

// compute S_ij
int S[][] = new int[n+1][n+1];
for(int i=1; i<=n; i++) {
    S[i][i] = M - lens[i];
    for(int j=i+1; j<=n; j++) {
        S[i][j] = S[i][j-1] - lens[j] - 1;
        if (S[i][j]<0) {
            while(j<=n) { S[i][j++] = infty; }
        }
    }
}
It was the best of times, it was the worst of times; it was the age of wisdom, it was the age of foolishness; it was the epoch of belief, it was the epoch of incredulity; it was the season of
first step: make $S_{i,j}$

$$\begin{array}{cccccccccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 \\
\hline
1 & 40 & 36 & 32 & 27 & 24 & 17 & 14 & 10 & 6 & 0 & \infty & \infty & \infty \\
2 & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty \\
\end{array}$$

$S_{i,i} = M - |w_i|$

<table>
<thead>
<tr>
<th>Slack</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_{i,j} = S_{i,j-1} - 1 -</td>
</tr>
</tbody>
</table>

$M = 42$

- $S_{1,1} = 42 - 2 = 40$
- $S_{1,2} = 40 - 3 - 1 = 36$
- $S_{1,3} = 36 - 3 - 1 = 32$
- $S_{2,2} = 42 - 3 = 39$
first step: make $S_{i,j}$

\[
\begin{array}{cccccccccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 \\
1 & 40 & 36 & 32 & 27 & 24 & 17 & 14 & 10 & 6 & 0 & 99 & 99 & 99 \\
2 & & & & & & & & & & & & \\
\end{array}
\]

$S_{i,i} = M - |w_i|$

$S_{i,j} = S_{i,j-1} - 1 - |w_j|$
first step: make $S_{i,j}$
<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>40</td>
<td>36</td>
<td>32</td>
<td>27</td>
<td>24</td>
<td>17</td>
<td>14</td>
<td>10</td>
<td>6</td>
<td>0</td>
<td>99</td>
<td>99</td>
<td>99</td>
</tr>
<tr>
<td>2</td>
<td>39</td>
<td>35</td>
<td>30</td>
<td>27</td>
<td>20</td>
<td>17</td>
<td>13</td>
<td>9</td>
<td>3</td>
<td>0</td>
<td>99</td>
<td>99</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```
2 3 3 4 2 6 2 3 3 5 2 6 2 3 3 3 2 7 2 3 3 5 2 12 2 3 3 5 2 7 2 3 3 5 2 12 2 3 3 6 2
```
second step: compute

\[
\begin{align*}
\text{best}_0 &= 0, 1600, 1256, \ldots \\
\text{Best}_1 &= \text{best}_0 + (S_{1,1})^2 = 40^2 \\
\text{Best}_2 &= \min \left\{ \text{best}_0 + (S_{1,2})^2, \text{best}_0 + (S_{2,1})^2, 1600 + 39^2 \right\} \\
\text{Best}_3 &= \min \left\{ \text{Best}_0 + S_{1,3}^2, \text{Best}_1 + S_{2,3}^2, 1600 + 39^2 \right\} \\
\text{BEST}_i &= \min_{j=0}^{i-1} \left\{ \text{BEST}_j + S_{j+1, i}^2 \right\} \\
\end{align*}
\]
The second step is to compute

\[
\text{best } i = \min_{j=0}^{i-1} \left\{ \text{BEST}_j + S^2_{j+1,i} \right\}
\]
second step: compute

\[
\text{best}_i = \min_{j=0}^{i-1} \left\{ \text{BEST}_j + S_{j+1,i}^2 \right\}
\]
Running time

make table for \( S_{i,j} \)

for \( i=1 \) to \( n \)

\[
\text{best}[i] = \min\{ \text{best}[j] + s[j+1][i]^2 \}
\]