Lecture 18: Typesetting II

COSC 225: Algorithms and Visualization

Spring, 2023

Final Project Timeline

up to 3

- Friday, April 14: Group/Project registration
- Monday, May 1: Working prototype, critique **in class**
- Wednesday, May 10: Final submission
- Friday, May 19: Peer reviews due

Today

Typesetting II: Breaking paragraphs into lines

1. Recap: greedy line breaking
2. Quantifying “raggedness”
3. Activity: greedy line breaking
4. Finding optimal line breaks

Last Time

Breaking paragraphs into lines

Input:

- `TEXT` as a string
- `LINE_WIDTH`

Output:

- Placement of each word from `TEXT` typeset on the screen

Requirement

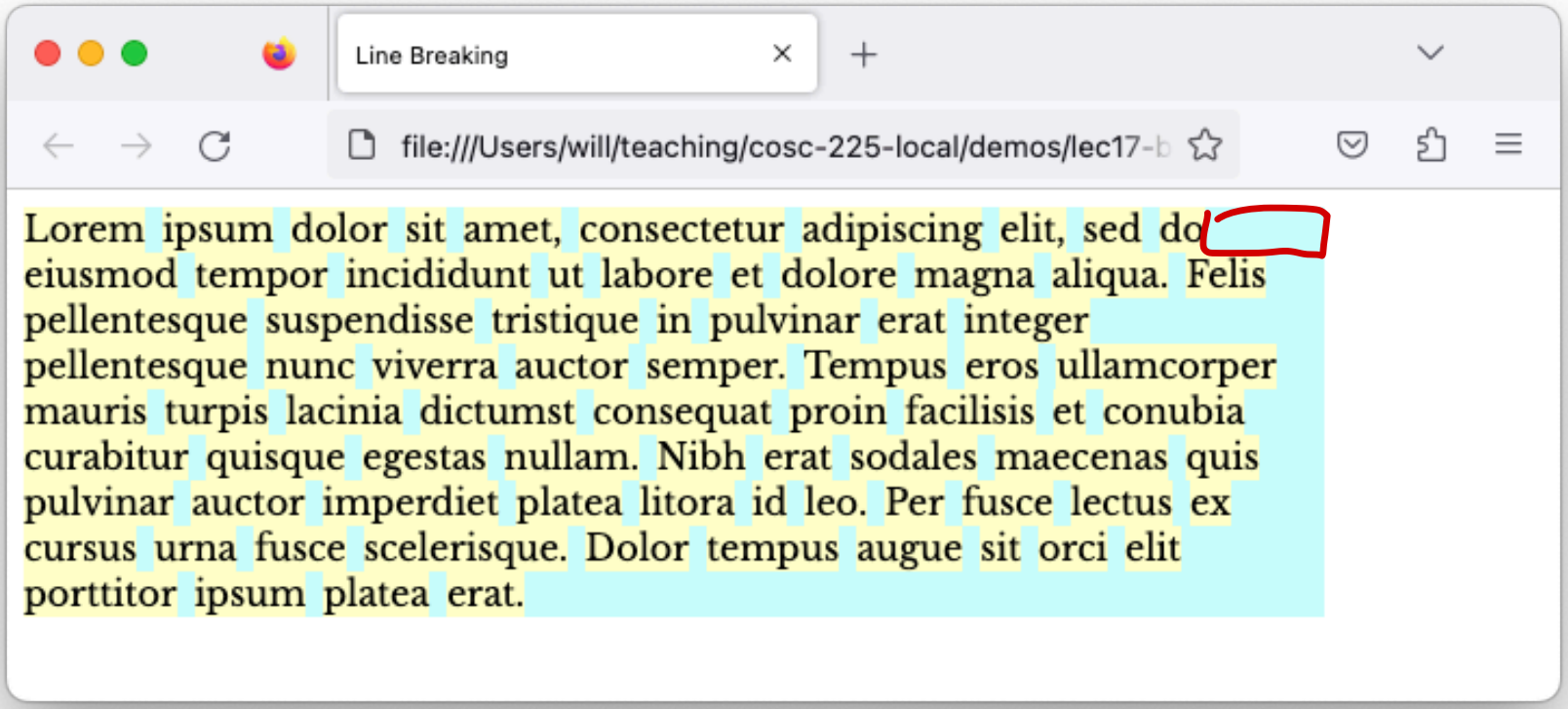
- No line of typeset text is wider than `LINE_WIDTH`

Example Input

```
const TEXT = "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod
tempor incididunt ut labore et dolore magna aliqua. Felis pellentesque suspendisse
tristique in pulvinar erat integer pellentesque nunc viverra auctor semper. Tempus eros
ullamcorper mauris turpis lacinia dictumst consequat proin facilisis et conubia curabitur
quisque egestas nullam. Nibh erat sodales maecenas quis pulvinar auctor imperdiet platea
litora id leo. Per fusce lectus ex cursus urna fusce scelerisque. Dolor tempus augue sit
orci elit porttitor ipsum platea erat.";

const TEXT_WIDTH = 600;     // width of text block in px
const WORD_SEP = 0.5;       // minimum separation between words in em
const PARAGRAPH_INDENT = 2; // paragraph indentation in em
```

Example Output



Greedy Breaking Procedure

Idea:

- scan through words of text sequentially
- add words to the current line until adding next word would exceed `LINE_WIDTH`
- start a new line with next word

Details

- dealing with first word of paragraph (paragraph indent)
- dealing with first word of other lines (no indent)
- dealing with last line

Greedy Breaking Code

```
let curLine = createLine();
for (let s of spans) {
  let width = s.getBoundingClientRect().width;
  if (!firstWord) { width += WORD_SEP * em; }
  if (curWidth + width <= TEXT_WIDTH) {
    curLine.appendChild(s);
    curWidth += width; firstWord = false;
  } else {
    parent.appendChild(curLine);
    curLine = createLine();
    curLine.appendChild(s);
    curWidth = width - WORD_SEP * em;}}
```

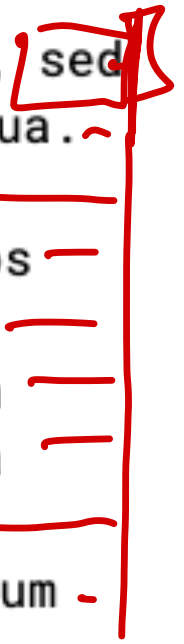
each line a div
elts for each word
add cur line to q

Some Questions

1. Is there a sensible alternative method for choosing line breaks?
2. What “better” outcomes might we want?

Critique This Paragraph

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Felis pellentesque suspendisse tristique in pulvinar erat integer pellentesque nunc viverra auctor semper. Tempus eros ullamcorper mauris turpis lacinia dictumst consequat proin facilisis et conubia curabitur quisque egestas nullam. Nibh erat sodales maecenas quis pulvinar auctor imperdiet platea litora id leo. Per fusce lectus ex cursus urna fusce scelerisque. Dolor tempus augue sit orci elit porttitor ipsum platea erat.



Is This Better?

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Felis pellentesque suspendisse tristique in pulvinar erat integer pellentesque nunc viverra auctor semper. Tempus eros ullamcorper mauris turpis lacinia dictumst consequat proin facilisis et conubia curabitur quisque egestas nullam. Nibh erat sodales maecenas quis pulvinar auctor imperdiet platea litora id leo. Per fusce lectus ex cursus urna fusce scelerisque. Dolor tempus augue sit orci elit porttitor ipsum platea erat.

Quantifying Raggedness

Aesthetic Goal. Minimize the *raggedness* of the paragraph.

→ minimize variation in whitespace trailing

difference between line and box width

→ minimize diff between shortest length & longest length

Quantifying Raggedness

Aesthetic Goal. Minimize the *raggedness* of the paragraph.

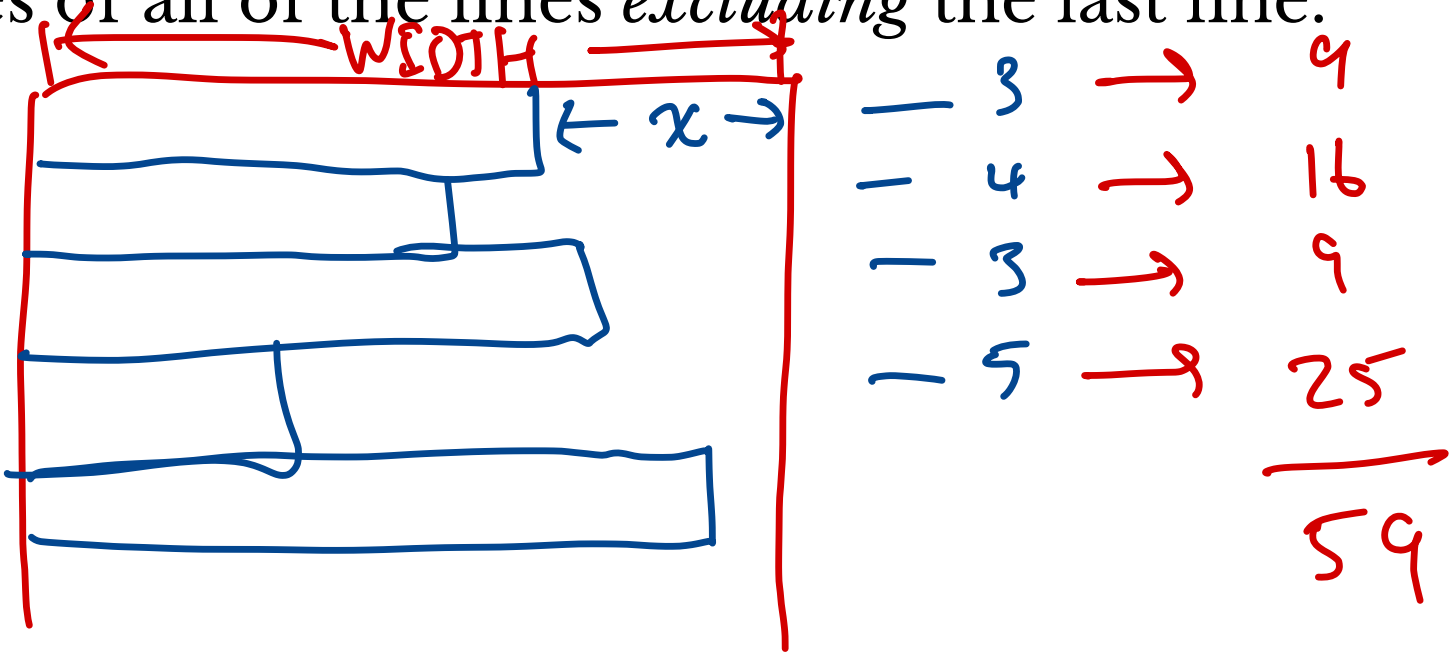
Question. How to *quantify* raggedness?

Knuth, Again

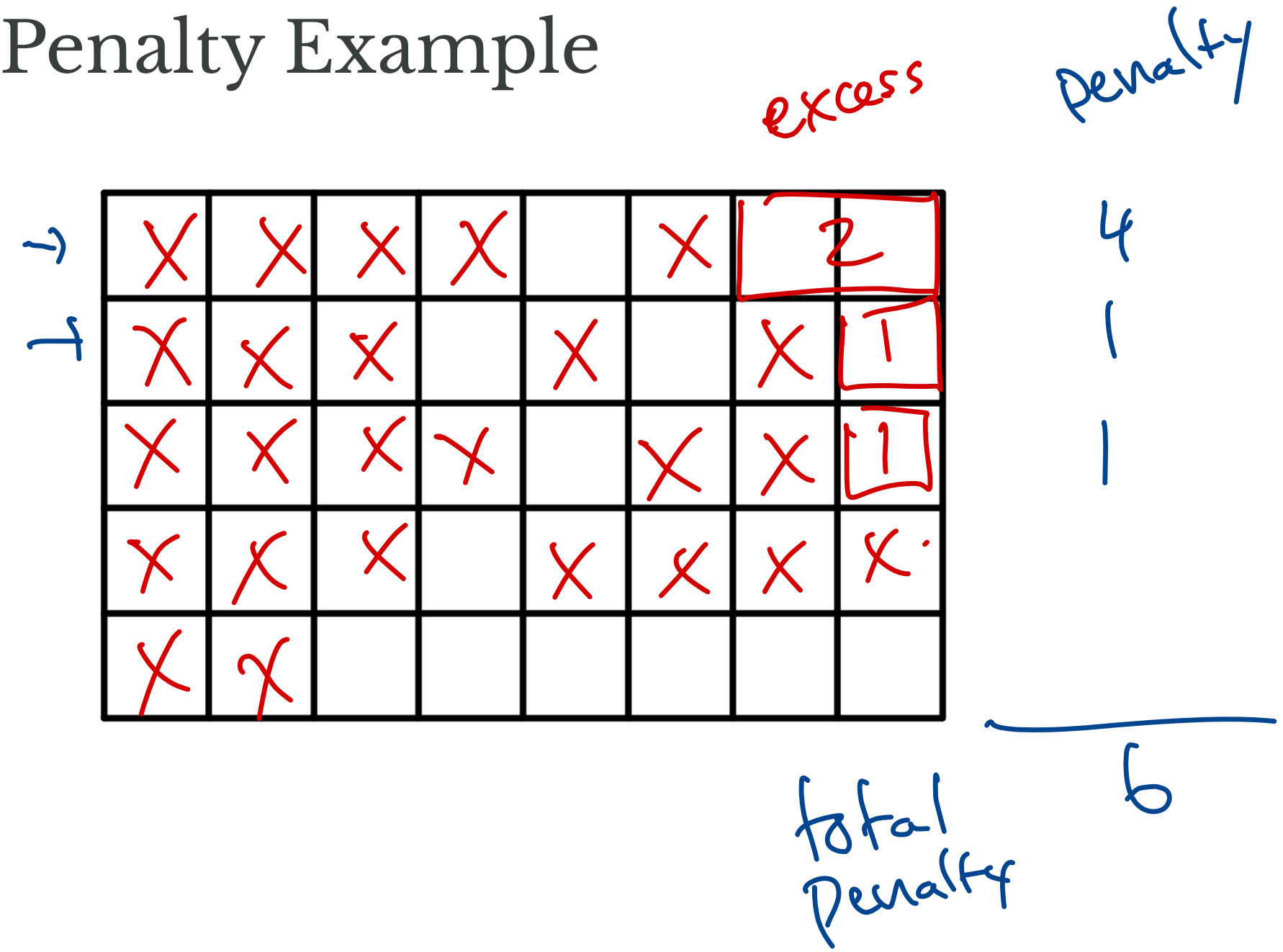
Associate a **penalty** to each line break:

- the *excess* x is the amount of trailing whitespace
- the *penalty* of the line is x^2

The penalty of the whole paragraph is the sum of penalties of all of the lines *excluding* the last line.



Penalty Example

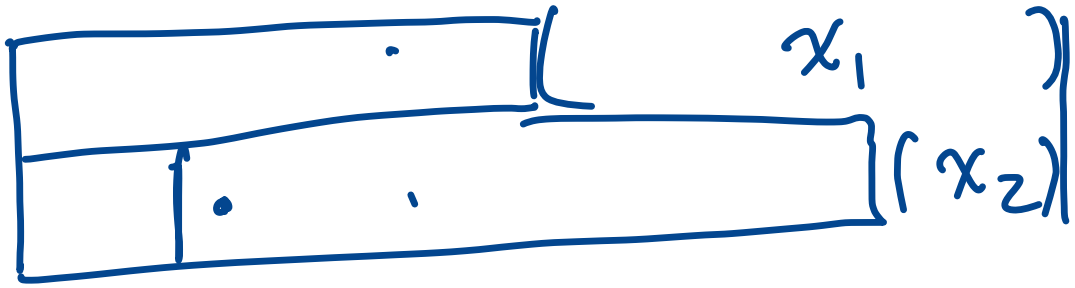


Our Goal

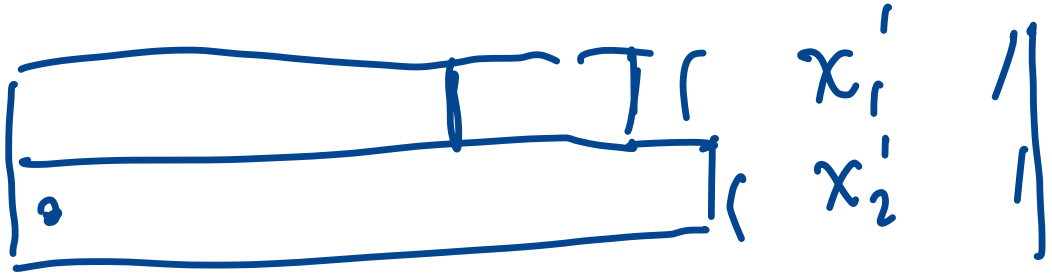
Find the line breaks for the paragraph that minimize the total penalty of the paragraph.

Why This Penalty

Question. Why is a penalty of x^2 sensible? How does penalty relate to raggedness?



$x_1^2 + x_2^2$



$(x_1')^2 + (x_2')^2$

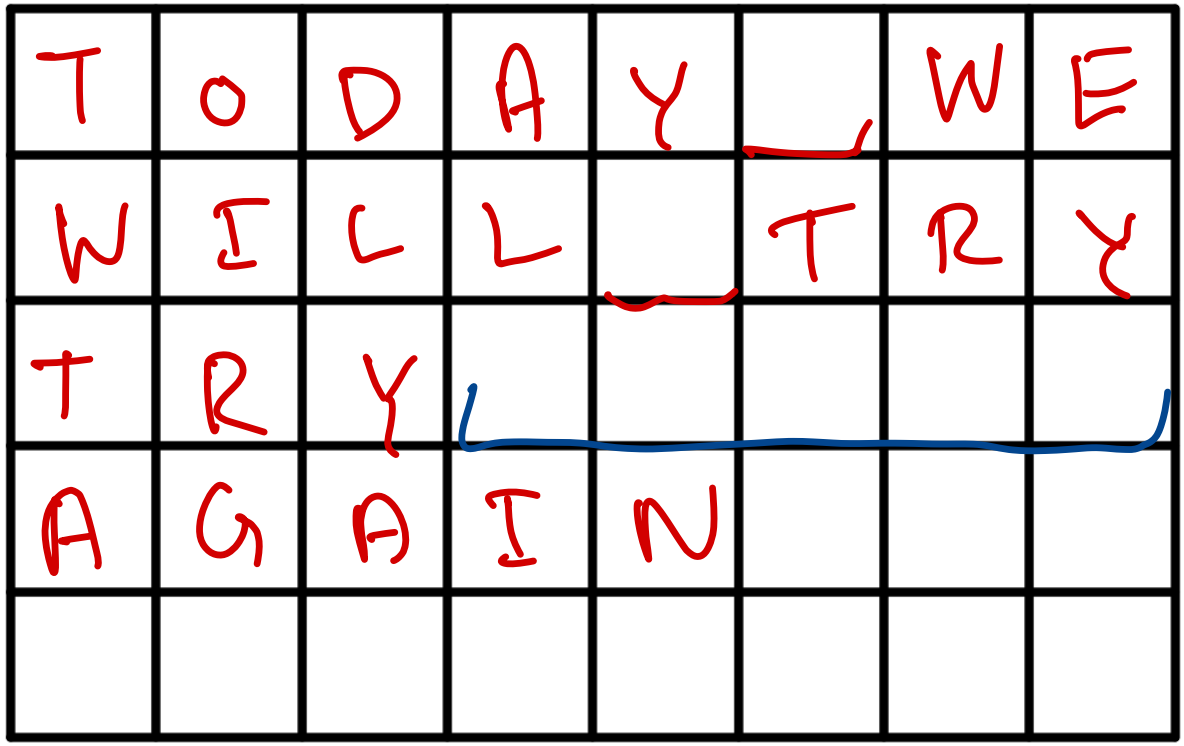
Fact. For fixed total amount of whitespace, perfectly even lines minimizes the total penalty

Activity

Typeset a short paragraph by hand!

Greedy Penalty

TODAY WE WILL TRY TRY AGAIN



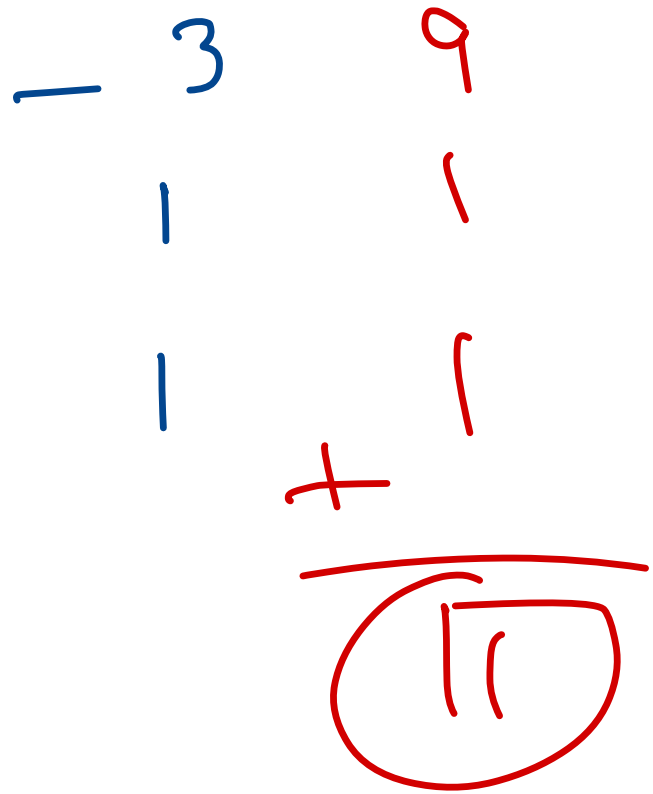
excess	penalty
0	0
0	0
5	25
—	

25

Smaller Penalty?

TODAY WE WILL TRY TRY AGAIN

T	O	D	A	Y			
W	E		W	I	C	L	
T	R	Y		T	R	Y	
A	G	A	I	N			



Greedy Penalty Example

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed	23
do eiusmod tempor incididunt ut labore et dolore magna aliqua.	368
Felis pellentesque suspendisse tristique in pulvinar erat	3893
integer pellentesque nunc viverra auctor semper. Tempus eros	1239
ullamcorper mauris turpis lacinia dictumst consequat proin	2787
facilisis et conubia curabitur quisque egestas nullam. Nibh	2007
erat sodales maecenas quis pulvinar auctor imperdiet platea	2007
litora id leo. Per fusce lectus ex cursus urna fusce	13271
scelerisque. Dolor tempus augue sit orci elit porttitor ipsum	739
platea erat.	26337

Optimal Penalty Example

Lorem ipsum dolor sit amet, consectetur adipiscing elit,	1730
sed do eiusmod tempor incididunt ut labore et dolore magna	3317
aliqua. Felis pellentesque suspendisse tristique in pulvinar	1128
erat integer pellentesque nunc viverra auctor semper. Tempus	1239
eros ullamcorper mauris turpis lacinia dictumst consequat	3893
proin facilisis et conubia curabitur quisque egestas nullam.	1239
Nibh erat sodales maecenas quis pulvinar auctor imperdiet	4096
platea litora id leo. Per fusce lectus ex cursus urna fusce	2460
scelerisque. Dolor tempus augue sit orci elit porttitor ipsum	739
platea erat.	19845

An Algorithmic Challenge

Input.

- array of word lengths
- whitespace parameters
- text width

Output.

- locations of line breaks that will minimize the total penalty of the paragraph

Question

How can we find the optimal line breaks given the input parameters?

Recursion?

Options: break @ cur word or not?

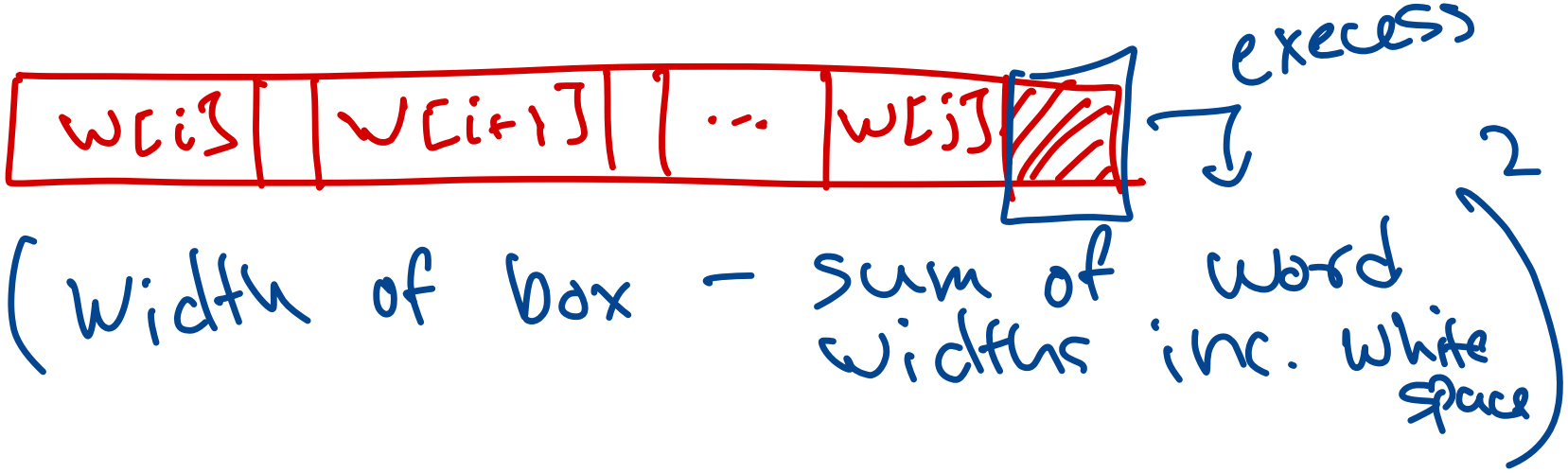
Top down along lines?

w array of words

A Basic Task

Given indices `i` and `j` with `i < j`, what is the penalty of a line containing words `w[i], w[i+1],..., w[j]`

- call this `penalty(i, j)`

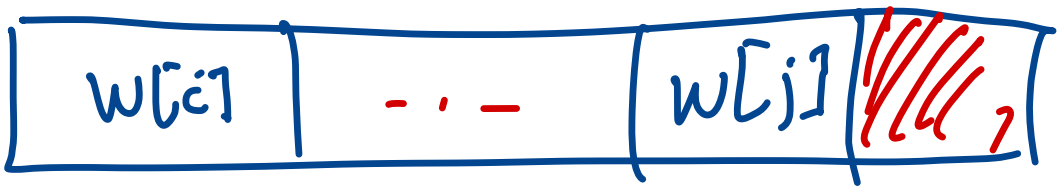


A Basic Task

Given indices `i` and `j` with `i < j`, what is the penalty of a line containing words `w[i], w[i+1],..., w[j]`

- call this `penalty(i, j)`

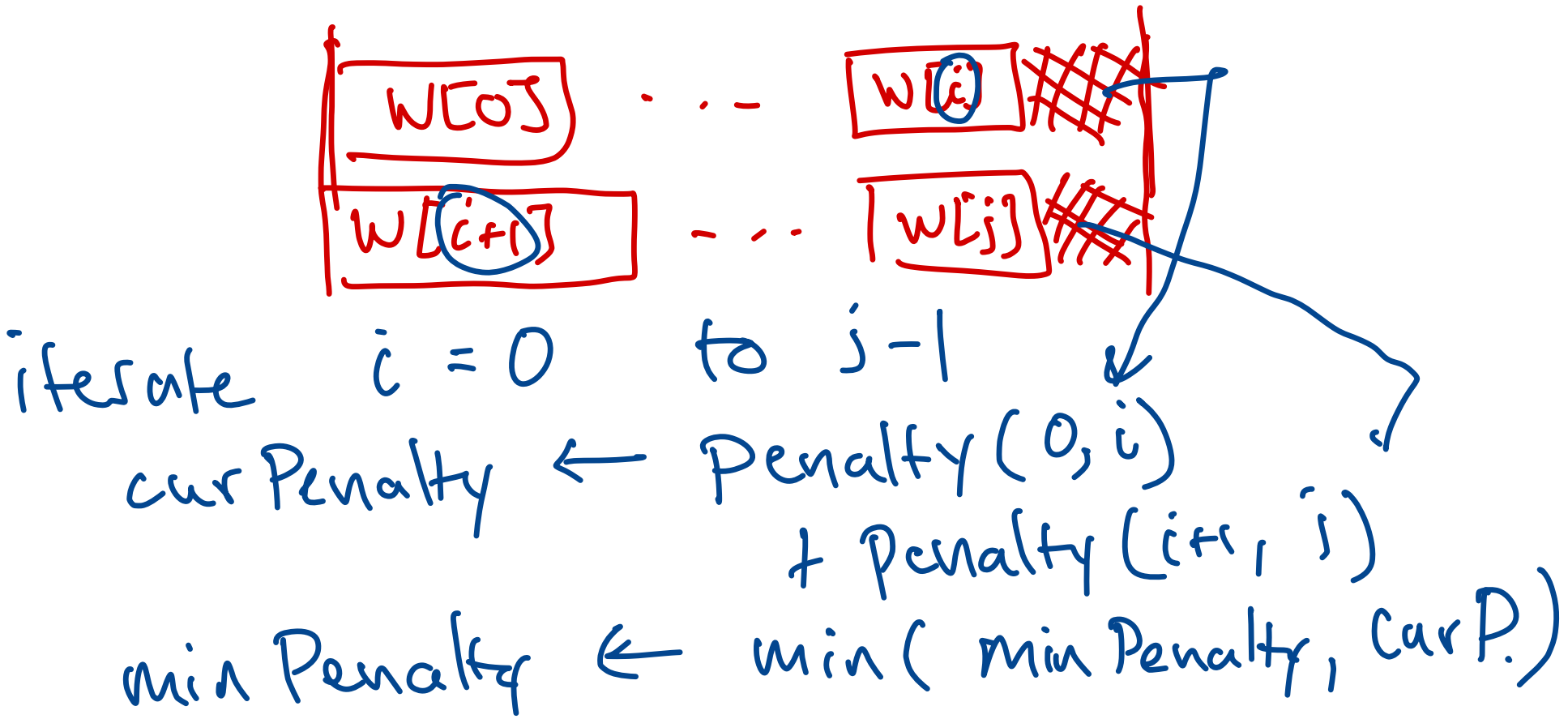
Question. Must we compute `penalty(i, j)` for all `i` and `j`?



Only need to consider i, j where total width of words is < text block (i.e. fit on a line)

Two Line Penalty

Question. Suppose we know `penalty[0, i]` for all `i` up to `j`. How could we find the minimum penalty line break for the first *two* lines setting words `0..j`?

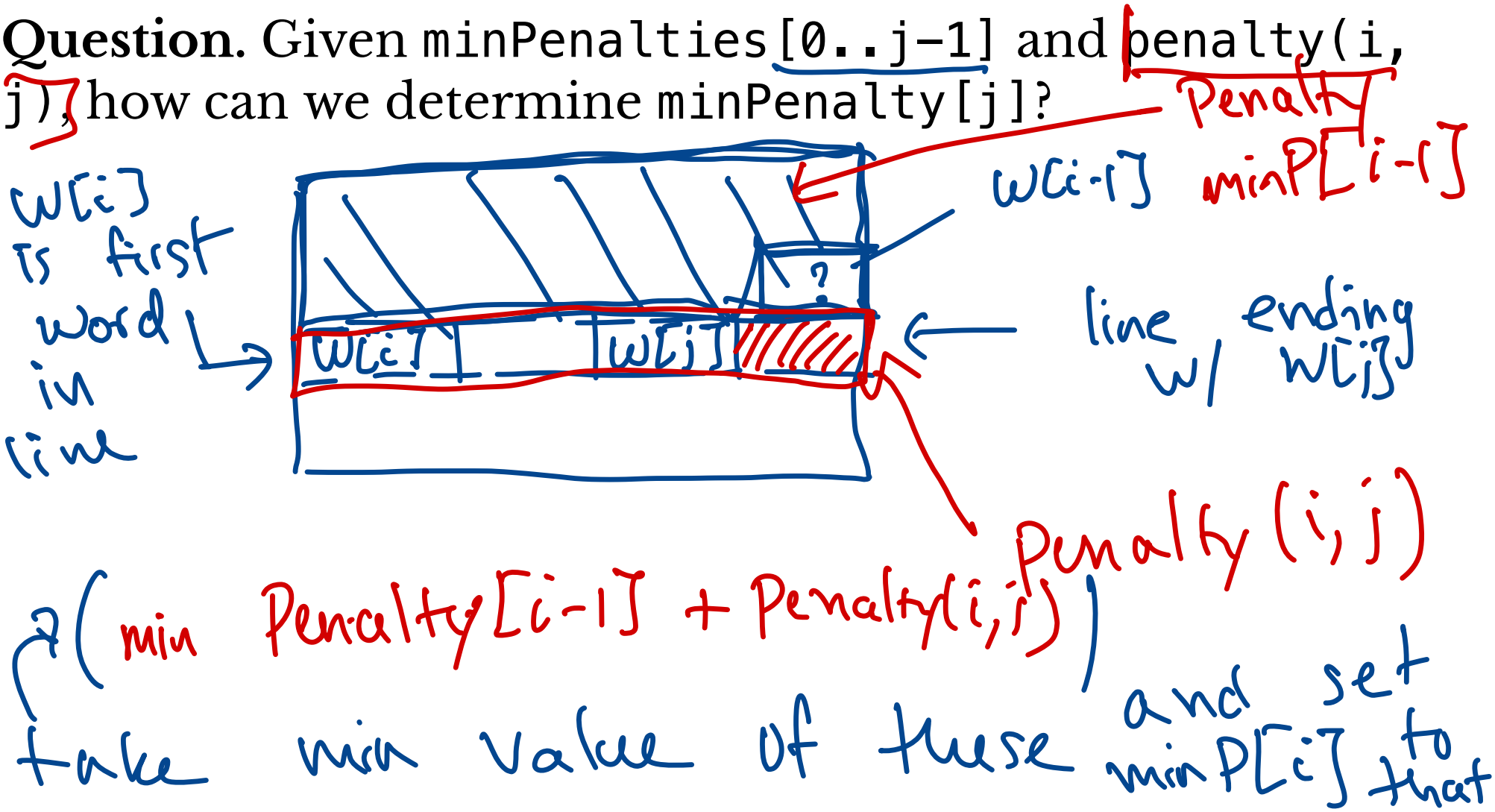


More Generally

Compute: array `minPenalties`

- `minPenalties[j]` stores the minimum total penalty of line breaks ending with a line break at `j`.

Question. Given `minPenalties[0..j-1]` and `penalty(i, j)`, how can we determine `minPenalty[j]`?



Bootstrapping

Question. Given `minPenalties[0..j-1]` and `penalty(i, j)`, how can we determine `minPenalty[j]`?

Observe. The minimal penalty of breaking at `i` and `j` is:

- `minPenalty[i-1]` + `penalty(i, j)`

prev lines

cur line ending w/ j

Bootstrapping

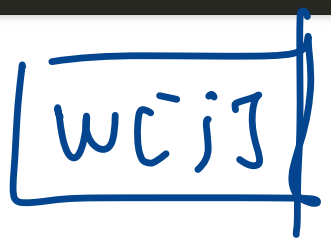
Question. Given `minPenalties[0..j-1]` and `penalty(i, j)`, how can we determine `minPenalty[j]`?

Observe. The minimal penalty of breaking at `i` and `j` is:

- `minPenalty[i-1] + penalty(i, j)`

So. The minimum possible penalty of breaking at `j` is the minimum of:

```
minPenalties[j-1] + penalty(j, j)
minPenalties[j-2] + penalty(j-1, j)
minPenalties[j-3] + penalty(j-2, j)
minPenalties[j-4] + penalty(j-3, j)
...
```



A Name

This technique for finding the optimal solution is called **dynamic programming**

Exercise

Find the minimum penalty line breaking for typesetting `TODAY WE WILL TRY TRY AGAIN` in a paragraph of line width 8.

Line Breaking Demo