

# Algorithm Design Laboratory with Applications

Prof. Stefano Leucci

## Problem: Book printing.

You want to print your favorite book but, due to unforeseen circumstances, you only have access to a very old printer. The printer is very rudimentary: it can only print a line  $L_i$  of text if it has a width  $|L_i|$  of at most  $W$  characters. This forces you to break each line  $L_i$  of text into several lines  $L_i^{(1)}, \dots, L_i^{(k_i)}$  such that (i) each line  $L_i^{(j)}$  contains only whole words and (ii)  $|L_i^{(j)}| \leq W$ .

After some experimentation, you realize that some of the printed lines are very short and quite unpleasant to read. In an attempt to obtain a better result, you decide to break each line  $L_i$  so that, in addition to (i) and (ii), you also have that the maximum amount of wasted space on a line, i.e., the quantity  $B = \max_{i=1, \dots, n} \max_{j=1, \dots, k_i} (W - |L_i^{(j)}|)$ , is minimized.

Design an algorithm that computes the optimal way to break lines.

**Input.** The input consists of a set of instances, or *test-cases*, of the previous problem. The first line contains the number  $T$  of test-cases. The first line of each test case contains the maximum width  $W$  supported by the printer and the number  $n$  of lines of the book. The  $i$ -th of the following  $n$  lines of the input is a string that represents the  $i$ -th line  $L_i$  of the book. More precisely,  $L_i$  has no leading or trailing spaces and only contains words of alphanumeric characters, words are separated by a single space.

**Output.** The output consists of  $T$  lines, each corresponding to a test-case. The  $i$ -th of the lines contains the integer  $B$  corresponding to the optimal way to break the input lines  $L_1, \dots, L_n$ .

## Example.

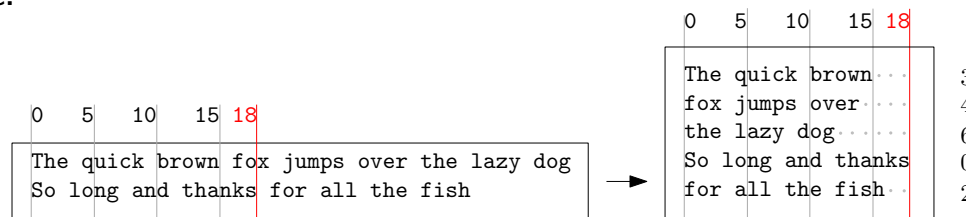


Figure 1: Left: A possible input (consisting of 2 lines) for  $W = 18$ . Right: the optimal way to break input lines. The value of  $B$  is  $\max\{3, 4, 6, 0, 2\} = 6$ .

*Input (corresponding to the above example):*

---

```
1
18 2
The quick brown fox jumps over the lazy dog
So long and thanks for all the fish
```

---

*Output:*

---

```
6
```

---

**Assumptions.**  $1 \leq T < 10$ ;  $1 \leq n < 2^{10}$ ;  $\forall i = 1, \dots, n, 1 \leq |L_i| \leq 2^{16}$ ;  $1 \leq W \leq 2^{10}$ .

The longest word in the input consists of at most  $W$  characters.

**Requirements.** Your algorithm must have an asymptotic time complexity of  $O(m \cdot W)$ , where  $m = \sum_{i=1}^n |L_i|$ .

**Notes.** A reasonable implementation should not require more than 1 second for each input file. When a line  $L_i$  is broken into one or more lines  $L_i^{(1)}, \dots, L_i^{(k_i)}$ , all leading and trailing white-spaces are removed from each  $L_i^{(j)}$ .