This problem is very similar in flavor to the segmented least squares problem. We observe that the last line ends with word w_n and has to start with some word w_j; breaking off words $w_j, \ldots, w_n$ we are left with a recursive sub-problem on $w_1, \ldots, w_{j-1}$.

Thus, we define $OPT[i]$ to be the value of the optimal solution on the set of words $W_i = \{w_1, \ldots, w_i\}$. For any $i \leq j$, let $S_{i,j}$ denote the slack of a line containing the words $w_i, \ldots, w_j$; as a notational device, we define $S_{i,j} = \infty$ if these words exceed total length L. For each fixed i, we can compute all $S_{i,j}$ in $O(n)$ time by considering values of j in increasing order; thus, we can compute all $S_{i,j}$ in $O(n^2)$ time.

As noted above, the optimal solution must begin the last line somewhere (at word w_j), and solve the sub-problem on the earlier lines optimally. We thus have the recurrence

$$OPT[n] = \min_{1 \leq j \leq n} S_{i,n}^2 + OPT[j-1],$$

and the line of words $w_j, \ldots, w_n$ is used in an optimum solution if and only if the minimum is obtained using index j.

Finally, we just need a loop to build up all these values:

```
Compute all values S_{i,j} as described above.
Set OPT[0] = 0
For k = 1, ..., n
  OPT[k] = min_{1≤j≤k} (S_{j,k}^2 + OPT[j-1])
Endfor
Return OPT[n].
```

As noted above, it takes $O(n^2)$ time to compute all values $S_{i,j}$. Each iteration of the loop takes time $O(n)$, and there are $O(n)$ iterations. Thus the total running time is $O(n^2)$.

By tracing back through the array OPT, we can recover the optimal sequence of line breaks that achieve the value $OPT[n]$ in $O(n)$ additional time.

[1] ex771.275.715