The key observation to make in this problem is that if the segmentation $y_1 y_2 \ldots y_n$ is an optimal one for the string y, then the segmentation $y_1 y_2 \ldots y_{n-1}$ would be an optimal segmentation for the prefix of y that excludes y_n (because otherwise we could substitute the optimal solution for the prefix in the original problem and get a better solution).

Given this observation, we design the subproblems as follows. Let $Opt(i)$ be the score of the best segmentation of the prefix consisting of the first i characters of y. We claim that the recurrence

$$Opt(i) = min_{j \leq i}\{Opt(j-1) + Quality(j \ldots n)\}$$

would give us the correct optimal segmentation (where $Quality(\alpha \ldots \beta)$ means the quality of the word that is formed by the characters starting from position α and ending in position β). Notice that the desired solution is $Opt(n)$.

We prove the correctness of the above formula by induction on the index i. The base case is trivial, since there is only one word with one letter.

For the inductive step, assume that we know that the Opt function as written above finds the optimum solution for the indices less than i, and we want to show that the value $Opt(i)$ is the optimum cost of any segmentation for the prefix of y up to the i-th character. We consider the last word in the optimal segmentation of this prefix. Let's assume it starts at index $j \leq i$. Then according to our key observation above, the prefix containing only the first $j-1$ characters must also be optimal. But according to our induction hypothesis, $Opt(j)$ will yield us the value of the aforementioned optimal segmentation. Therefore the optimal cost $Opt(i)$ would be equal to $Opt(j)$ plus the cost of the last word.

But notice that our above recurrence exactly does this calculation for each possibility of the last word. Therefore our recurrence will correctly find the cost of the optimal segmentation.

As for the running time, a simple implementation (direct evaluation of the above formula starting at index 1 until n, where n is the number of characters in the input string) will yield a quadratic algorithm.

[1] ex931.924.160