Problem D. DNA Engineering

Input file:	*standard input*
Output file:	*standard output*
Time limit:	5 seconds
Memory limit:	1024 mebibytes

Tandem copy is an operation on a DNA where a consecutive sequence of one or more nucleotides is repeated, and the repetitions are directly adjacent to each other; in other words, the tandem copy operation makes a copy of a consecutive sequence of nucleotides and pastes the copy right after the copied sequence. For example, we can tandem copy `ATC` in `ATCG` and obtain `ATCATCG`. Furthermore, we can make another tandem copy operation on the resulting sequence `ATCATCG` and obtain `ATCATTCG`. The following example illustrates a series of tandem copies from `ATCG`, where the underlined sequence is copied on each step:

$$\underline{\texttt{ATC}}\texttt{G} \Rightarrow \texttt{ATCA}\underline{\texttt{T}}\texttt{CG} \Rightarrow \underline{\texttt{AT}}\texttt{CATTCG} \Rightarrow \texttt{ATATCATTCG} \Rightarrow \cdots$$

We say that `ATCG` *produces* all these sequences by tandem copy. It is easy to see that `ATCG` can produce different sequences by selecting a different portion of the sequence for the operation on each step. Furthermore, in principle, `ATCG` can produce infinitely many sequences by making different operations.

Usually, it is more expensive to tandem copy a longer portion of a sequence. For instance,

$$\texttt{ATCG} \Rightarrow \texttt{ATCATCG}$$

is a tandem copy of three nucleotides and thus is more expensive than

$$\texttt{ATCATCG} \Rightarrow \texttt{ATCATTCG}$$

which is a tandem copy of one nucleotide. In other words, the length of the copied portion on each step is crucial to determine the tandem copy cost.

Since it is easy to tandem copy a single nucleotide, it is practical for the genetic engineering lab to store sequences such that every two consecutive nucleotides in a sequence are always different; this helps the lab to reduce the storage space. For instance, since `ATTTG` can be produced by tandem copying `T` twice from `ATG`, it is better for the lab to only store the shorter sequence `ATG` instead of `ATTTG`.

Because of a recent budget cut, the lab can only perform the tandem copy on at most two nucleotides at one time. In other words, the length of the portion copied on each step is at most two. On the other hand, the lab can apply as many tandem copy operations as it desires. For example, given a sequence `ACGT`, we can apply the tandem copy operation on `C` and obtain `ACCGT`, or apply it on the sequence `CG` and obtain `ACGCGT`. But we cannot tandem copy the consecutive sequence `ACG` because its length is more than two.

Given a source string s and a target string t, your task is to count the number of valid substrings of s. The substring s' is *valid* if one can obtain a string x from s' by applying an appropriate number of tandem copy operations, and x contains t as a substring. Please note that no two consecutive nucleotides in the source string are the same, whereas two consecutive nucleotides in the target string can be the same. For example, `CCA` or `ATTGC` cannot be source strings, but they can be target strings.

Now, given $s =$ `ACATGCAT` and $t =$ `CCACATTT`, we take a substring $s' =$ `CATGC` of s and run a series of tandem copies as follows:

$$s' = \texttt{CATGC} \Rightarrow \texttt{CCATGC} \Rightarrow \texttt{CCACATGC} \Rightarrow \texttt{CCACATTGC} \Rightarrow \texttt{CCACATTTGC}$$

which contains t as its substring.

Here is another substring example. For $s' = \texttt{CAT}$:

$$s' = \texttt{CAT} \Rightarrow \texttt{CACAT} \Rightarrow \texttt{CCACAT} \Rightarrow \texttt{CCACATT} \Rightarrow \texttt{CCACATTT} = t$$

which shows that we can produce the target string from `CAT` by a series of tandem copies.

It is easy to verify that the total number of valid substrings of s is 14. Note that both the first and the second `CAT` in s are counted as different valid substrings. Thus, you need to consider all parts of string s as substrings, and count all valid substrings individually.

Here is another example.

When $s = \texttt{AC}$ and $t = \texttt{CA}$, you can take the substring `AC` and tandem copy `AC`. Then, the resulting string is `ACAC`, which contains `CA` as its substring. All other substrings of s are unable to produce `CA` as a substring, and therefore the number of valid substrings is one.

Given a source string s and a target string t, where no two consecutive characters in s are the same, write a program that outputs the number of valid substrings s' of s.

Input

The input consists of two lines. The first line is the source string s, and the second line is the target string t. Each input consists of uppercase English letters, and $1 \le |s|, |t| \le 2 \cdot 10^4$. It is guaranteed that no two consecutive characters in s are the same.

Output

Print exactly one line. The line should contain the number of valid substrings.

Examples

standard input	standard output
ATGTG TTG	9
CACTGT CCTTG	6
PQRPQR PQR	7
BCDBCD BCDBCD	1