D. Messenger

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Each employee of the "Blake Techologies" company uses a special messaging app "Blake Messenger". All the stuff likes this app and uses it constantly. However, some important futures are missing. For example, many users want to be able to search through the message history. It was already announced that the new feature will appear in the nearest update, when developers faced some troubles that only you may help them to solve.

All the messages are represented as a strings consisting of only lowercase English letters. In order to reduce the network load strings are represented in the special compressed form. Compression algorithm works as follows: string is represented as a concatenation of n blocks, each block containing only equal characters. One block may be described as a pair (l_i, c_i), where l_i is the length of the i-th block and c_i is the corresponding letter. Thus, the string s may be written as the sequence of pairs .

Your task is to write the program, that given two compressed string t and s finds all occurrences of s in t. Developers know that there may be many such occurrences, so they only ask you to find the **number** of them. Note that p is the starting position of some occurrence of s in t if and only if $t_p t_{p+1} ... t_{p+|s|-1} = s$, where t_i is the i-th character of string t.

Note that the way to represent the string in compressed form may not be unique. For example string "`aaaa`" may be given as , , ...

Input

The first line of the input contains two integers n and m $(1 \le n, m \le 200\,000)$ — the number of blocks in the strings t and s, respectively.

The second line contains the descriptions of n parts of string t in the format "l_i-c_i" $(1 \le l_i \le 1\,000\,000)$ — the length of the i-th part and the corresponding lowercase English letter.

The second line contains the descriptions of m parts of string s in the format "l_i-c_i" $(1 \le l_i \le 1\,000\,000)$ — the length of the i-th part and the corresponding lowercase English letter.

Output

Print a single integer — the number of occurrences of s in t.

Examples

input

```
5 3
3-a 2-b 4-c 3-a 2-c
2-a 2-b 1-c
```

output

```
1
```

input

```
6 1
3-a 6-b 7-a 4-c 8-e 2-a
3-a
```

output

```
6
```

input

```
5 5
1-h 1-e 1-l 1-l 1-o
1-w 1-o 1-r 1-l 1-d
```

output

```
0
```

Note

In the first sample, t = "aaabbccccaaacc", and string s = "aabbc". The only occurrence of string s in string t starts at position $p = 2$.

In the second sample, t = "aaabbbbbbaaaaaaacccceeeeeeeeaa", and s = "aaa". The occurrences of s in t start at positions $p = 1$, $p = 10$, $p = 11$, $p = 12$, $p = 13$ and $p = 14$.