Memes

Time and Memory Limits: 1.5 seconds, 128 MB

You like Internet memes. Like that video with the people dancing! Or that picture of the animal giving silly advice. Or that movie reference that people are always using on the image boards. These are all highly specific Internet memes, which you like a lot, because – as mentioned above – you like Internet memes.

After extensive analysis of the transcripts of hundreds of thousands of videos and captioned images, you have discovered that the key to a successful Internet meme consists of two simple ingredients:

- A *critical string* (a predetermined string where **no character appears more than once**)
- Repetitions of this string as many times as possible (preferably consecutively).

For example, "beaver" is not a valid critical string, since it contains more than one "e". But the "badger" is valid!

Given the critical string "badger", we can then analyse the following *text* (larger string):

```
snakebadgerbadgertigertigertigerbadgerbadllamaduck
```

As you can see, the critical string ("badger") appears 3 times in total in the text. The longest run of consecutive critical strings is of length 2 in "badgerbadger" (indices 6 through 17 inclusive).

By changing a few characters (the ones at indices $27, 28, 29, 42, 43, 44$) we get the following new text:

```
snakebadgerbadgertigertigebadgerbadgerbadgermaduck
```

Now the critical string appears 5 times, and its longest consecutive run is 3 times in a row.

Your task is to write a program that analyses a text for the number of critical string occurences and the length of the longest run of critical strings. Furthermore, it must determine this after each change in a sequence of one-character changes to the text.

Input

- The first line of input will contain two space-separated integers N and M: the length of the critical string and the length of the text respectively.
- The second line will contain the critical string, a sequence of N distinct space-separated integers each between 1 and 1 000 000.
- The third line describes the initial text. This line consists of M space-separated integers each between 1 and 1 000 000.
- The fourth line contains a single integer Q: the number of updates made to the text.
- The following Q lines each describe an update made to the text, in the form "x_i v_i". This represents an instruction to change the character at index x_i to v_i. It is not guaranteed that v_i is different to the character already at index x_i.

Output

Your program should produce Q lines of output: one for each update made to the text.

The ith line should contain two space-separated integers α_i and β_i, where:

- α_i is the number of times the critical string appears in the text (after the first i updates have been made).
- β_i is the number of times the critical string appears in its longest consecutive run in the text (after the first i updates have been made).

Sample Input

```
6 50
98 97 100 103 101 114
115 110 97 107 101 98 97 100 103 101 114 98 97 100 103 101 114 116 105
   103 101 114 116 105 103 101 114 116 105 103 101 114 98 97 100 103 101
   114 98 97 100 108 108 97 109 97 100 117 99 107
6
27 98
28 97
29 100
42 103
43 101
44 114
```

Sample Output

```
3 2
3 2
4 2
4 2
4 2
5 3
```

Explanation

The sample input corresponds to the example described above (with characters replaced by their ASCII integer equivalents). Note however that for page space reasons, we have separated B into multiple lines, but in the input file it it will occupy only one line.

Subtasks & Constraints

For each subtask, your program will score 40% of the available marks if all α_i values are correct, and 60% of the available marks if all β_i values are correct.

- For Subtask 1 (30 points), $1 \leq N \leq M \leq 5000$ and $1 \leq Q \leq 5000$.
- For Subtask 2 (70 points), $1 \leq N \leq M \leq 100\,000$ and $1 \leq Q \leq 100\,000$.