E. Prefix Enlightenment

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

There are n lamps on a line, numbered from 1 to n. Each one has an initial state off (0) or on (1).

You're given k subsets $A_1, \ldots, A_k$ of $\{1, 2, \ldots, n\}$, such that the intersection of any three subsets is empty. In other words, for all $1 \le i_1 < i_2 < i_3 \le k$, $A_{i_1} \cap A_{i_2} \cap A_{i_3} = \varnothing$.

In one operation, you can choose one of these k subsets and switch the state of all lamps in it. It is guaranteed that, with the given subsets, it's possible to make all lamps be simultaneously on using this type of operation.

Let m_i be the minimum number of operations you have to do in order to make the i first lamps be simultaneously on. Note that there is no condition upon the state of other lamps (between $i+1$ and n), they can be either off or on.

You have to compute m_i for all $1 \le i \le n$.

Input

The first line contains two integers n and k $(1 \le n, k \le 3 \cdot 10^5)$.

The second line contains a binary string of length n, representing the initial state of each lamp (the lamp i is off if $s_i = 0$, on if $s_i = 1$).

The description of each one of the k subsets follows, in the following format:

The first line of the description contains a single integer c $(1 \le c \le n)$ — the number of elements in the subset.

The second line of the description contains c distinct integers $x_1, \ldots, x_c$ $(1 \le x_i \le n)$ — the elements of the subset.

It is guaranteed that:

- The intersection of any three subsets is empty;
- It's possible to make all lamps be simultaneously on using some operations.

Output

You must output n lines. The i-th line should contain a single integer m_i — the minimum number of operations required to make the lamps 1 to i be simultaneously on.

Examples

input

```
7 3
0011100
3
1 4 6
3
3 4 7
2
2 3
```

output

```
1
2
3
3
3
3
3
```

input

```
8 6
00110011
3
1 3 8
5
1 2 5 6 7
2
6 8
2
3 5
```

```
2
4 7
1
2
```

output

```
1
1
1
1
1
1
4
4
```

input

```
5 3
00011
3
1 2 3
1
4
3
3 4 5
```

output

```
1
1
1
1
1
```

input

```
19 5
1001001001100000110
2
2 3
2
5 6
2
8 9
5
12 13 14 15 16
1
19
```

```
output
```

```
0
1
1
1
2
2
2
3
3
3
3
4
4
4
4
4
4
4
5
```

Note

In the first example:

- For $i = 1$, we can just apply one operation on A_1, the final states will be 1010110;
- For $i = 2$, we can apply operations on A_1 and A_3, the final states will be 1100110;
- For $i \geq 3$, we can apply operations on A_1, A_2 and A_3, the final states will be 1111111.

In the second example:

- For $i \leq 6$, we can just apply one operation on A_2, the final states will be 11111101;
- For $i \geq 7$, we can apply operations on A_1, A_3, A_4, A_6, the final states will be 11111111.

ITMO UNIVERSITY