6832 Bit String Reordering

You have to reorder a given bit string as specified. The only operation allowed is swapping adjacent bit pairs. Please write a program that calculates the minimum number of swaps required.

The initial bit string is simply represented by a sequence of bits, while the target is specified by a *run-length code*. The run-length code of a bit string is a sequence of the lengths of maximal consecutive sequences of zeros or ones in the bit string. For example, the run-length code of "011100" is "1 3 2". Note that there are two different bit strings with the same run-length code, one starting with zero and the other starting with one. The target is either of these two.

In Sample Input 1, bit string "100101" should be reordered so that its run-length code is "1 3 2", which means either "100011" or "011100". At least four swaps are required to obtain "011100". On the other hand, only one swap is required to make "100011". Thus, in this example, 1 is the answer.

Input

The input file contains several test cases, each of them as described below.

Each test case is formatted as follows.

N M
b_1 b_2 ... b_N
p_1 p_2 ... p_M

The first line contains two integers N $(1 \leq N \leq 15)$ and M $(1 \leq M \leq N)$. The second line specifies the initial bit string by N integers. Each integer b_i is either '0' or '1'. The third line contains the run-length code, consisting of M integers. Integers p_1 through p_M represent the lengths of consecutive sequences of zeros or ones in the bit string, from left to right. Here, $1 \leq p_j$ for $1 \leq j \leq M$ and $\sum_{j=1}^{M} p_j = N$ hold. It is guaranteed that the initial bit string can be reordered into a bit string with its run-length code $p_1, \ldots, p_M$.

Output

For each test case, output the minimum number of swaps required.

Sample Input

```
6 3
1 0 0 1 0 1
1 3 2
7 2
1 1 1 0 0 0 0
4 3
15 14
1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
1 1 1 1 1 1 1 1 1 1 1 1 1 2
1 1
0
1
```

Sample Output

```
1
12
7
0
```