E. Lucky Array

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Petya loves lucky numbers. Everybody knows that lucky numbers are positive integers whose decimal representation contains only the lucky digits **4** and **7**. For example, numbers **47**, **744**, **4** are lucky and **5**, **17**, **467** are not.

Petya has an array consisting of n numbers. He wants to perform m operations of two types:

- add $l\ r\ d$ — add an integer d to all elements whose indexes belong to the interval from l to r, inclusive $(1 \le l \le r \le n, 1 \le d \le 10^4)$;
- count $l\ r$ — find and print on the screen how many lucky numbers there are among elements with indexes that belong to the interval from l to r inclusive $(1 \le l \le r \le n)$. Each lucky number should be counted as many times as it appears in the interval.

Petya has a list of all operations. The operations are such that after all additions the array won't have numbers that would exceed 10^4. Help Petya write a program that would perform these operations.

Input

The first line contains two integers n and m $(1 \le n, m \le 10^5)$ — the number of numbers in the array and the number of operations correspondingly. The second line contains n positive integers, none of which exceeds 10^4 — those are the array numbers. Next m lines contain operations, one per line. They correspond to the description given in the statement.

It is guaranteed that after all operations are fulfilled each number in the array will not exceed 10^4.

Output

For each operation of the second type print the single number on the single line — the number of lucky numbers in the corresponding interval.

Examples

input Copy

```
3 6
2 3 4
count 1 3
count 1 2
add 1 3 2
count 1 3
add 2 3 3
count 1 3
```

output Copy

```
1
0
1
1
```

input Copy

```
4 5
4 4 4 4
count 1 4
add 1 4 3
count 1 4
add 2 3 40
count 1 4
```

output Copy

```
4
4
4
```

Note

In the first sample after the first addition the array will look in the following manner:

```
4 5 6
```

After the second addition:

```
4 8 9
```

The second sample after the first addition:

```
7 7 7 7
```

After the second addition:

```
7 47 47 7
```