

Almost Union-Find

I hope you know the beautiful Union-Find structure. In this problem, you're to implement something similar, but not identical.

The data structure you need to write is also a collection of disjoint sets, supporting 3 operations:

1 p q	Union the sets containing p and q . If p and q are already in the same set, ignore this command.
2 p q	Move p to the set containing q . If p and q are already in the same set, ignore this command.
3 p	Return the number of elements and the sum of elements in the set containing p .

Initially, the collection contains n sets: $\{1\}, \{2\}, \{3\}, \dots, \{n\}$.

Input

There are several test cases. Each test case begins with a line containing two integers n and m ($1 \leq n, m \leq 100,000$), the number of integers, and the number of commands. Each of the next m lines contains a command. For every operation, $1 \leq p, q \leq n$. The input is terminated by end-of-file (EOF).

Output

For each type-3 command, output 2 integers: the number of elements and the sum of elements.

Explanation

Initially: $\{1\}, \{2\}, \{3\}, \{4\}, \{5\}$

Collection after operation 1 1 2: $\{1,2\}, \{3\}, \{4\}, \{5\}$

Collection after operation 2 3 4: $\{1,2\}, \{3,4\}, \{5\}$ (we omit the empty set that is produced when taking out 3 from $\{3\}$)

Collection after operation 1 3 5: $\{1,2\}, \{3,4,5\}$

Collection after operation 2 4 1: $\{1,2,4\}, \{3,5\}$

Sample Input

```
5 7
1 1 2
2 3 4
1 3 5
3 4
2 4 1
3 4
3 3
```

Sample Output

```
3 12
3 7
2 8
```