Problem D. Ancient Berland Roads

Time limit	1000 ms
Code length Limit	50000 B
OS	Linux

Read problems statements in Mandarin Chinese , Russian and Vietnamese as well.

In Ancient Berland, there were **N** towns, along with **M** bidirectional roads connecting them. With time, some roads became unusable, and nobody repaired them.

As a person who is fond of Ancient Berland history, you now want to undertake a small research study. For this purpose, you want to write a program capable of processing the following kinds of queries:

- `D` **K** : meaning that the road numbered **K** in the input became unusable. The road numbers are 1-indexed.
- `P` **A x** : meaning that the population of the $\mathbf{A}^{th}$ town became **x**.

Let's call a subset of towns a **region** if it is possible to get from each town in the subset to every other town in the subset by the usable (those, which haven't already been destroyed) roads, possibly, via some intermediary cities of this subset. The **population** of the region is, then, the sum of populations of all the towns in the region.

You are given the initial road system, the initial population in each town and **Q** queries, each being one of two types above. Your task is to maintain the size of the most populated region after each query.

Input

The first line of each test case contains three space-separated integers — **N**, **M**, and **Q** — denoting the number of cities, the number of roads, and the number of queries, respectively.

The following line contains **N** space-separated integers, the $\mathbf{i}^{th}$ of which denotes the initial population of the $\mathbf{i}^{th}$ city.

The $\mathbf{j}^{th}$ of the following **M** lines contains a pair of space-separated integers — $\mathbf{X_j}$, $\mathbf{Y_j}$ — denoting that there is a bidirectional road connecting the cities numbered $\mathbf{X_j}$ and $\mathbf{Y_j}$.

Each of the following **Q** lines describes a query in one of the forms described earlier.

Output

Output **Q** lines. On the $\mathbf{i}^{th}$ line, output the size of the most populated region after performing **i** queries.

Constraints

- $\mathbf{1 \le X_j, Y_j \le N}$
- Roads' numbers are 1-indexed.
- There is no road that gets removed twice or more.
- $\mathbf{1 \le P_i \le 10^5}$
- Subtask 1 (30 points) : $\mathbf{1 \le N, M, Q \le 10^3}$
- Subtask 2 (70 points) : $\mathbf{1 \le N, M, Q \le 5 \times 10^5}$

Example

Input:

```
3 3 6
1 2 3
1 2
2 3
3 1
P 1 3
D 1
P 2 3
D 2
P 3 10
D 3
```

Output:

```
8
8
9
6
13
10
```

Explanation

- After the first query, the populations are **(3, 2, 3)** and the most populated region is **{1, 2, 3}**.
- After the second query the populations and the regions remain the same.

- After the third query the populations are **(3, 3, 3)** and the most populated region is again **{1, 2, 3}**.
- After the fourth query the populations remain the same, but we have two regions: **{1, 3}** and **{2}**. The most populated region is **{1, 3}**.
- After the fifth query the populations become equal to **(3, 3, 10)** respectively, and the most populated region is again **{1, 3}**.
- After the last query we have populations the same, but now every city forms it own separate region, and the most populated region is region **{3}**.