11457 Classified

Classification levels like Secret and Top Secret are well-known features of military documents. Less well known are *integrity levels*, which correspond to *how reliable* information is.

In the **Dynamic Biba integrity model**, every user and every document is assigned an integrity level. If a user writes to a document that has a higher integrity level than the user does, the document's integrity level goes down. Similarly, if a user reads a document that has a lower integrity level than the user does, the user's integrity level goes down.

Your job is to keep track of the integrity levels of a number of users and documents. Complicating the matter is the fact that integrity levels aren't simply numbers; a document might contain very trusworthy information about operations in Afganistan, while at the same time containing less trustworthy information about operations in Iraq.

Integrity levels are speficied by arbitrary labels, along with rules of the form $A \to B$, indicating that integrity level A is at most as trustworthy as integrity level B. The rules satisfy a number of conditions:

- For any label A, $A \to A$ is always true, and need not be specified.
- If A and B are two different labels, then it can never be the case that both $A \to B$ and $B \to A$ are true. However, it *can* be the case that neither $A \to B$ nor $B \to A$ are true.
- For any labels A, B, and C, if it is the case that $A \to B$ and $B \to C$ are both true, you can conclude (without it being explicitly specified) that $A \to C$ is true.
- For any two labels A and B, there is a label $G = glb(A, B)$ called the *greatest lower bound* of A and B such that $G \to A$ and $G \to B$ are both true. Further, for any label L such that $L \to A$ and $L \to B$ are both true, it is also the case that $L \to G$ is true.

You will be given as input the labels and rules defining the integrity levels, as well as the initial integrity level of a number of users and a number of documents.

Input

The first line is the number of test cases to follow. The test cases follow, one after another; the format of each test case is the following:

- The first line of the input will contain five integers separated by spaces. These integers are the number of integrity levels ℓ, the number of rules r, the number of users u, the number of documents d, and the number of actions a, in that order. None of these five integers will be greater than 10000. The integrity levels are numbered from 1 to ℓ; the users from 1 to u, and the documents from 1 to d.
- The next r lines specify the rules. Each line will consist of two integers between 1 and ℓ, separated by a space. The line 'x y' indicates that integrity level x is at most as trustworthy as integrity level y (i.e. $x \to y$)
- The next u lines specify the initial integrity levels for user number $1, 2, \ldots, u$. Each line will be an integrity level number, from 1 to ℓ.
- The next d lines specify the initial integrity levels for document number $1, 2, \ldots, d$. Each line will be an integrity level number, from 1 to ℓ.

- The next a lines specify the actions; each action will be of one of the following two forms (here, *user* is an integer between 1 and u, and *document* is an integer between 1 and d):
  - *user* `reads` *document*: The given user is reading the given document. The integrity level for the user should be changed to the *glb* of the old integrity level for the user and the integrity level of the document. Output the new integrity level for the user on a line by itself.
  - *user* `writes` *document*: The given user is writing to the given document. The integrity level for the document should be changed to the *glb* of the integrity level for the user and the old integrity level of the document. Output the new integrity level for the document on a line by itself.

Output

For each action in the input, output the new integrity level as described above.

Sample Input

```
1
9 11 2 3 5
6 9
2 6
9 4
7 5
3 8
7 2
5 3
8 4
5 1
6 1
1 8
4
2
5
9
8
1 reads 3
1 writes 2
2 reads 2
1 reads 1
2 writes 1
```

Sample Output

```
8
6
2
5
7
```