3 Problem: Merging tables

Problem Introduction

In this problem, your goal is to simulate a sequence of merge operations with tables in a database.

Problem Description

Task. There are n tables stored in some database. The tables are numbered from 1 to n. All tables share the same set of columns. Each table contains either several rows with real data or a symbolic link to another table. Initially, all tables contain data, and i-th table has r_i rows. You need to perform m of the following operations:

1. Consider table number $destination_i$. Traverse the path of symbolic links to get to the data. That is,

   while $destination_i$ contains a symbolic link instead of real data do

   $destination_i \leftarrow \texttt{symlink}(destination_i)$

2. Consider the table number $source_i$ and traverse the path of symbolic links from it in the same manner as for $destination_i$.
3. Now, $destination_i$ and $source_i$ are the numbers of two tables with real data. If $destination_i \neq source_i$, copy all the rows from table $source_i$ to table $destination_i$, then clear the table $source_i$ and instead of real data put a symbolic link to $destination_i$ into it.
4. Print the maximum size among all n tables (recall that size is the number of rows in the table). If the table contains only a symbolic link, its size is considered to be 0.

See examples and explanations for further clarifications.

Input Format. The first line of the input contains two integers n and m — the number of tables in the database and the number of merge queries to perform, respectively.
The second line of the input contains n integers r_i — the number of rows in the i-th table.
Then follow m lines describing merge queries. Each of them contains two integers $destination_i$ and $source_i$ — the numbers of the tables to merge.

Constraints. $1 \leq n, m \leq 100\,000$; $0 \leq r_i \leq 10\,000$; $1 \leq destination_i, source_i \leq n$.

Output Format. For each query print a line containing a single integer — the maximum of the sizes of all tables (in terms of the number of rows) after the corresponding operation.

Time Limits. `C`: 2 sec, `C++`: 2 sec, `Java`: 14 sec, `Python`: 6 sec. `C#`: 3 sec, `Haskell`: 4 sec, `JavaScript`: 6 sec, `Ruby`: 6 sec, `Scala`: 14 sec.

Memory Limit. 512Mb.

Sample 1.

Input:

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

Output:

```
2
2
3
5
5
```

Explanation:
In this sample, all the tables initially have exactly 1 row of data. Consider the merging operations:

1. All the data from the table 5 is copied to table number 3. Table 5 now contains only a symbolic link to table 3, while table 3 has 2 rows. 2 becomes the new maximum size.
2. 2 and 4 are merged in the same way as 3 and 5.
3. We are trying to merge 1 and 4, but 4 has a symbolic link pointing to 2, so we actually copy all the data from the table number 2 to the table number 1, clear the table number 2 and put a symbolic link to the table number 1 in it. Table 1 now has 3 rows of data, and 3 becomes the new maximum size.
4. Traversing the path of symbolic links from 4 we have $4 \to 2 \to 1$, and the path from 5 is $5 \to 3$. So we are actually merging tables 3 and 1. We copy all the rows from the table number 1 into the table number 3, and now the table number 3 has 5 rows of data, which is the new maximum.
5. All tables now directly or indirectly point to table 3, so all other merges won't change anything.

Sample 2.

Input:

```
6 4
10 0 5 0 3 3
6 6
6 5
5 4
4 3
```

Output:

```
10
10
10
11
```

Explanation:
In this example tables have different sizes. Let us consider the operations:

1. Merging the table number 6 with itself doesn't change anything, and the maximum size is 10 (table number 1).

2. After merging the table number 5 into the table number 6, the table number 5 is cleared and has size 0, while the table number 6 has size 6. Still, the maximum size is 10.
3. By merging the table number 4 into the table number 5, we actually merge the table number 4 into the table number 6 (table 5 now contains just a symbolic link to table 6), so the table number 4 is cleared and has size 0, while the table number 6 has size 6. Still, the maximum size is 10.
4. By merging the table number 3 into the table number 4, we actually merge the table number 3 into the table number 6 (table 4 now contains just a symbolic link to table 6), so the table number 3 is cleared and has size 0, while the table number 6 has size 11, which is the new maximum size.

Starter Files

The starter solutions in C++, Java and Python3 read the description of tables and operations from the input, declare and partially implement disjoint set union, and write the output. You need to complete the implementation of disjoint set union for this problem. If you use other languages, you will have to implement the solution from scratch.

What to Do

Think how to use disjoint set union with path compression and union by rank heuristics to solve this problem. In particular, you should separate in your thinking the data structure that performs union/find operations from the merges of tables. If you're asked to merge first table into second, but the rank of the second table is smaller than the rank of the first table, you can ignore the requested order while merging in the Disjoint Set Union data structure and join the node corresponding to the second table to the node corresponding to the first table instead in you Disjoint Set Union. However, you will need to store the number of the actual second table to which you were requested to merge the first table in the parent node of the corresponding Disjoint Set, and you will need an additional field in the nodes of Disjoint Set Union to store it.

Need Help?

Ask a question or see the questions asked by other learners at this forum thread.