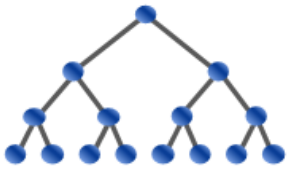

USACO 2022 US Open Contest, Gold
Problem 1. Custodial Cleanup

Return to Problem List

Time Remaining: **3 hrs, 53 min, 54 sec**

Submitted; Results below show the outcome for each judge test case

* 1 10.3mb 15ms	* 2 10.3mb 19ms	* 3 10.3mb 15ms	* 4 10.3mb 15ms	* 5 10.3mb 15ms	* 6 10.3mb 15ms	* 7 13.0mb 137ms	* 8 10.3mb 50ms	* 9 13.0mb 131ms	* 10 13.0mb 119ms	* 11 13.1mb 131ms	* 12 10.3mb 53ms
			* 13 12.9mb 123ms	* 14 13.1mb 136ms	* 15 13.1mb 130ms	* 16 10.3mb 50ms	* 17 10.3mb 61ms	* 18 13.1mb 136ms			

English (en)

Due to the disorganized structure of his mootels (much like motels but with bovine rather than human guests), Farmer John has decided to take up the role of the mootel custodian to restore order to the stalls.

Each mootel has N stalls labeled 1 through N ($1 \le N \le 10^5$) and M ($0 \le M \le 10^5$) corridors that connect pairs of stalls to each other bidirectionally. The ith stall is painted with color C_i and initially has a single key of color S_i in it. FJ will have to rearrange the keys to appease the cows and restore order to the stalls.

FJ starts out in stall 1 without holding any keys and is allowed to repeatedly do one of the following moves:

- Pick up a key in the stall he is currently in. FJ can hold multiple keys at a time.
- Place down a key he is holding into the stall he is currently in. A stall may hold multiple keys at a time.
- Enter stall 1 by moving through a corridor.
- Enter a stall other than stall 1 by moving through a corridor. He can only do this if he currently holds a key that is the same color as the stall he is entering.

Unfortunately, it seems that the keys are not in their intended locations. To restore order to FJ's mootel, the ith stall requires that a single key of color F_i is in it. It is guaranteed that S is a permutation of F.

For T different mootels ($1 \le T \le 100$), FJ starts in stall 1 and needs to place every key in its appropriate location, ending back in stall 1. For each of the T mootels, please answer if it is possible to do this.

INPUT FORMAT (input arrives from the terminal / stdin):

The first line contains T, the number of mootels (test cases).

Each test case will be preceded by a blank line. Then, the first line of each test case contains two integers N and M.

The second line of each test case contains N integers. The i-th integer on this line, C_i, means that stall i has color C_i ($1 \le C_i \le N$).

The third line of each test case contains N integers. The i-th integer on this line, S_i, means that stall i initially holds a key of color S_i ($1 \le S_i \le N$).

The fourth line of each test case contains N integers. The i-th integer on this line, F_i, means that stall i needs to have a key of color F_i in it ($1 \le F_i \le N$).

The next M lines of each test case follow. The i-th of these lines contains two distinct integers, u_i and v_i ($1 \le u_i, v_i \le N$). This represents that a corridor exists between stalls u_i and v_i. No corridors are repeated.

The sum of N over all mootels will not exceed 10^5, and the sum of M over all mootels will not exceed $2 \cdot 10^5$.

OUTPUT FORMAT (print output to the terminal / stdout):

For each mootel, output YES on a new line if there exists a way for FJ to return a key of color F_i to each stall i and end back in stall 1. Otherwise, output NO on a new line.

SAMPLE INPUT:

```
2

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

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

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

SAMPLE OUTPUT:

```
YES
NO
```

For the first test case, here is a possible sequence of moves:

```
Current stall: 1. Keys held: []. Keys in stalls: [3, 4, 3, 4, 2]
(pick up key of color 3)
Current stall: 1. Keys held: [3]. Keys in stalls: [x, 4, 3, 4, 2]
(move from stall 1 to 2, allowed since we have a key of color C_2=3)
Current stall: 2. Keys held: [3]. Keys in stalls: [x, 4, 3, 4, 2]
(pick up key of color 4)
Current stall: 2. Keys held: [3, 4]. Keys in stalls: [x, x, 3, 4, 2]
(move from stall 2 to 1 to 4 to 5, allowed since we have keys of colors C_4=4 and C_5=3)
Current stall: 5. Keys held: [3, 4]. Keys in stalls: [x, x, 3, 4, 2]
(pick up key of color 2 and place key of color 3)
Current stall: 5. Keys held: [2, 4]. Keys in stalls: [x, x, 3, 4, 3]
(move from stall 5 to 4 to 1 to 3, allowed since we have keys of colors C_4=4 and C_3=2)
Current stall: 3. Keys held: [2, 4]. Keys in stalls: [x, x, 3, 4, 3]
(pick up key of color 3 and place key of color 4)
Current stall: 3. Keys held: [2, 3]. Keys in stalls: [x, x, 4, 4, 3]
(move from stall 3 to stall 2 and place key of color 3)
Current stall: 2. Keys held: [2]. Keys in stalls: [x, 3, 4, 4, 3]
(move from stall 2 to stall 1 and place key of color 2)
Current stall: 1. Keys held: []. Keys in stalls: [2, 3, 4, 4, 3]
```

For the second test case, there exists no way for FJ to return a key of color F_i to each stall i and end back at stall 1.

SAMPLE INPUT:

```
5

2 0
1 2
2 2
2 2

2 1
1 1
2 1
2 1
1 2

2 1
1 1
2 1
1 2
1 2

2 1
1 1
1 2
2 1
1 2

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

SAMPLE OUTPUT:

```
YES
```

```
YES
NO
YES
NO
```

SCORING:

- Test cases 3-6 satisfy $N, M \leq 8$.
- Test cases 7-10 satisfy $C_i = F_i$.
- Test cases 11-18 satisfy no additional constraints.

Problem credits: Eric Yachbes

Language: C

Source File: 选择文件 未选择任何文件

Submit Solution

Previous Submissions:

Sun, Mar 26, 2023 00:20:40 EDT (C++17)