Carrot Rations - Alphastar CC41A MPS 2 Problem 2

Saturday, August 12, 2023 7:36 PM

Carrot Rations

Bunny is receiving his weekly rations of carrots. He notices that there is a social hierarchy amongst the rabbits. Those who are higher up the social ladder receive their rations before those who are lower.
There are N (1 <= N <= 10^5) rabbits (numbered from 1 to N). Bunny has made M (1<= M <= 50,000) observations about the rabbit’s social structure. Each observation is an ordered list of some of his family members, indicating that these rabbits should receive their carrots in the same order that they appear in this list. For example, if one of Bunny’s observations is the list 2, 5, 1, rabbit 2 gets his carrots before rabbit 5 and rabbit 5 gets his carrots before rabbit 1.
Bunny’s observations have been noticed by the rabbits giving out the carrot rations. They want to maximize the value of X for which the carrot ration order meets the conditions outlined in the first X observations. If multiple rabbits’ ration orders satisfy these first X conditions, the rabbits think that it is a longstanding tradition that rabbits with lower numbers outrank those with higher numbers. They want to give the rations out to the lowest numbered rabbits first. If multiple ration giving orders satisfy these conditions, the rabbits want to use the lexicographically smallest one. An ordering x is lexicographically smaller than an ordering y if for some j, x_i = y_i for all i < j and x_j < y_j. In other words, the two orderings are identical up to a certain point, at which x is smaller than y. Determine the best order that the rabbits will receive their carrot rations.

INPUT FORMAT

The first line contains N and M. The next M lines each describe an observation. Line i+1 describes observation i, and starts with the number of rabbits m_i listed in the observation followed by the list of m_i integers giving the ordering of rabbits in the observation. The sum of the m_i's is at most 200,000.

OUTPUT FORMAT

Output N space-separated integers, giving a permutation of 1…N containing the order in which the rabbits should receive their carrot rations.

SAMPLE INPUT

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

SAMPLE OUTPUT

```
2 1 4 3
```

From <https://lms.alphastar.academy/mod/quiz/attempt.php?attempt=452281&cmid=82254&page=1>

Immediate thought: What if we represent this as a directed graph, where higher-ranking rabbits(nodes) have directed edges to lower ranking ones (based on each input line)

Once again with the confusing alphastar wording. No clue what the start of the second paragraph is really saying about the lower numbers. Maybe it's to explain in-lore why we need to give the lexicographically smallest sol. So that seems to clear it up.

Continuing with our directed graph interpretation, if we have conflicts, we would get a cycle. At that point, we can stop adding edges since we have go just maximize the number of lines that we satisfy, in order. So when a conflict appears we're essentially free.

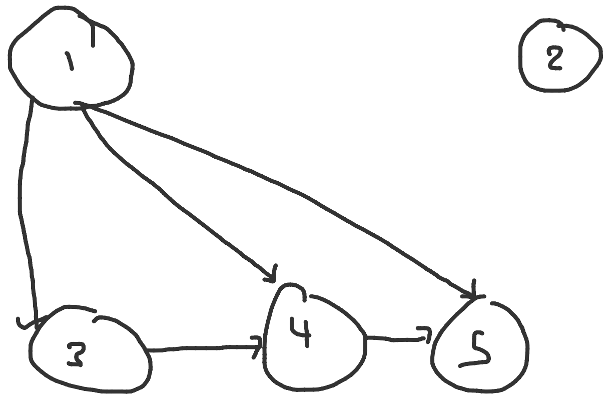
Once we have this **non-cyclical directed graph** we have a few options (they put a lesson on these things right before the practice set, I wonder why). But anyways, we can always just utilize the In-degrees method. If there is no cycle (and there won't), we'll just start there. Still thinking on how we get the lexicographically smallest representation. Going to eat, will have thoughts (thots??) soon.
^^highly unlikely, percentage chance of writer getting "thots" is exactly 0.0000%

"He did not eat lol"

Time to **DRAW IT OUT**

Let's say our test case is:

```
5 5
2 1 5
3 1 4 5
4 1 3 4 5
5 2 1 4 3 5  //this one breaks the graph
5 1 5 3 4 2
```

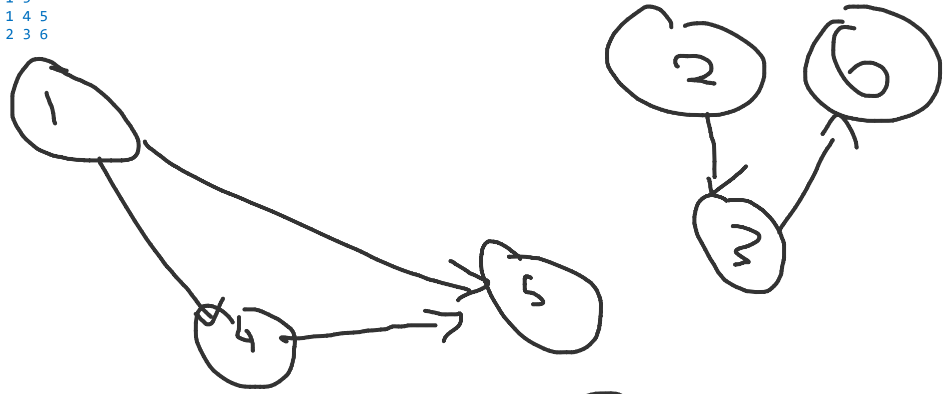


This one becomes our graph. For this one I think we'll have to insert it at the best possible time. Basically, we first create an order without the floaters:

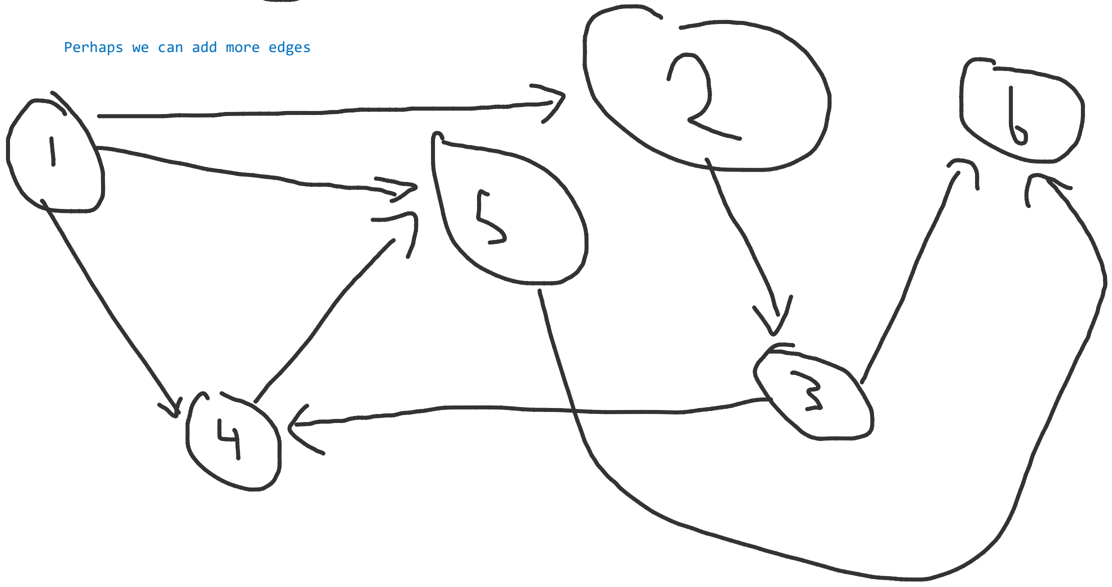
1, 3, 4, 5

Then we go through with a queue of floaters sorted least to greatest. When we find a number in the order greater than the current front of queue, we add that there instead. However this definitely doesn't hold up for graphs with multiple components (orders) such as the below:

```
6 3
2 1 5
3 1 4 5
3 2 3 6
```



Perhaps we can add more edges



9 days of degeneracy later
Yea no I think the idea of adding de edge is gud. We just go thru all nodes, see if the next node is in its connected component - if not, add a edge to it. This ensures we don't break hierarchy and also ensures the lexicographically smallest solution. Time to watch de veejio

De veejio sae:

We can simply keep track of in-degrees for every node - for those nodes with in-degree 0, we add them in the sorted order. Then, we delete all edges coming from those nodes and process the new 0-in-degree nodes. This works because when we remove those edges, we've already added the preceding nodes, so it is guarunteed our order is satisfied.

"…we would get a cycle, and that would be pretty sad"
-unknown Alphastar instructor, commenting on the "fail" condition for attempting to satisfy more requirements in this problem

Another observation made is that if no nodes have in-degree 0, we have a cycle so we can keep track of that. Failing to remove all nodes means a cycle exists.

At this point the idea is to just binary search the value of X and apply our previously discussed methods to determine if it works or not.

1600PST 08/22/2023 confirmed working.