Please follow the submission guidelines on discord.

Problem: Lecture Hall problem

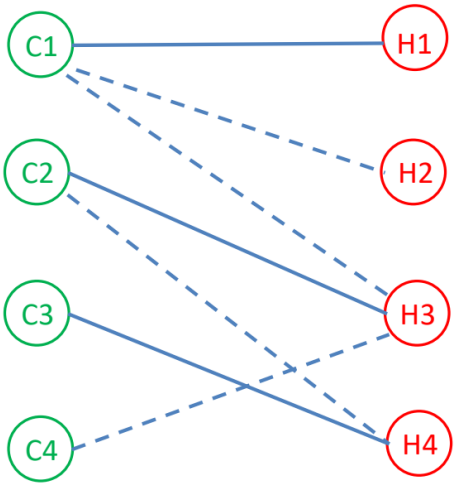
- You are the course coordinator in a university and you need to assign the courses to the lecture halls.
- You did your homework and based on the number of enrollments in each course and some other features (e.g., distance, A/V support) you know which course can potentially be taught in which lecture hall(s).
- Given these potential assignments, you want to find the maximum number of courses that could be taught at the same time.
- Write a program MA2.java that reads the database of potential assignments between courses and lecture halls (this is to be read through a file input.txt, as usual) in the format below:
  - The number of courses, $3 \leq N \leq 100$, in the first line.
  - The number of lecture halls, $3 \leq M \leq 100$, in the second line.
  - Each of the next N lines shows the course id (in increasing order from C_1 to C_N) followed by possible lecture halls (separated by space) for that course.
  - You can assume that each course's potential lecture halls will be in increasing order, and there will be at most $min(20, M)$ possible lecture halls for each course.
  - Return as output a single number: the maximum number of courses that could be run at the same time (just one number, no comments, prompts etc.). This is to be returned to the console (i.e. using System.out).

Example #1

Input:

```
4
4
C1 H1 H2 H3
C2 H3 H4
C3 H4
C4 H3
```

Output:
3



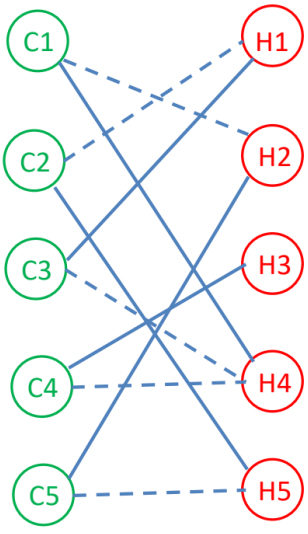
Solid lines show one example assignment with maximum course count

Example #2

Input:

```
5
5
C1 H2 H4
C2 H1 H5
C3 H1 H4
C4 H3 H4
C5 H2 H5
```

Output:
5



Solid lines show one example assignment with maximum course count

Important remarks:

- The input samples shown above don't accurately represent the formatting of the input. To make it easier to parse the input, each course/hall will be displayed as a number in the input (to avoid parsing characters "C" and "H"). Below is an example of what the input files for the two examples above actually look like.

Input:

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

Input:

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

- M does not need to be equal to N.
- Each course will have at least one potential lecture hall where it can be taught.
- You need to implement Max-Flow algorithm like it is described in the slides (see slide 23 with the pseudocode). This means you need to (1) create the residual graph, (2) find augmenting paths, (3) then update the residual graph.
- Other solutions (those which don't do steps 1-3) will get max 3 even if they work.
- Any Java libraries, classes, functions related to graphs, vertices, edges are NOT allowed (create your own).
- I recommend using adjacency matrices for graphs (it's easier), but feel free to use adjacency lists.
- Using Java queue/priority queue/linked list (and other simple data structures) is fine.

Submission through Canvas:

- Date due: Sat, March 25th, 11:59 pm.
- Submit only the Java source code file MA2.java
- Do not submit a zip file. Also no need to add your name to the submitted file's name as Canvas does it for you.
- Remember: in Java, class name should match the file name, and is case sensitive.
- Please do not create your own packages or use built-in libraries/functions/classes.