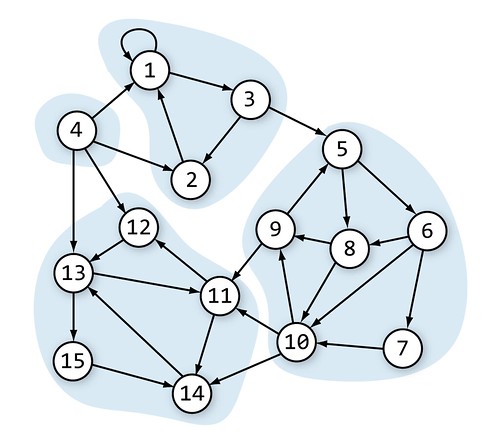
# Exercise: Graphs Strongly Connected Components Max Flow

This document defines the lab for the ["Algorithms – Advanced (C#)" course @ Software University](https://softuni.bg/trainings/3810/algorithms-advanced-with-c-sharp-september-2022). Please submit your solutions (source code) to all below-described problems in [Judge](https://judge.softuni.org/Contests/2582/Graphs-Strongly-Connected-Components-Max-Flow-Exercise).

## Electrical Substation Network

Print all connected electrical substations in a town using the existing network. The substations are represented as vertices, and the connections are represented as edges.



In this network, we have 4 connected electrical substations.

### Input

* On the first line, you will receive an integer – n – number of nodes starting from 0.
* On the second line, you will receive an integer – l – number of lines.
* On the next l lines, you will receive a node and its children in the following format: "{node}, {children1}, … {childrenN}".

### Output

* Print all connected electrical substations in **topological order** in the following format: "{node1}, {node2},… {nodeN}".
  + Nodes in each component should be printed in **topological order** too.

### Example

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Graph** |
| 3  3  0, 1  1, 2  2, 0 | 0, 2, 1 | Diagram  Description automatically generated |
| 15  15  0, 0, 2  1, 0  2, 1, 4  3, 0, 1, 11, 12  4, 5, 7  5, 7, 9, 6  6, 9  7, 8, 9  8, 4, 10  9, 8, 10, 13  10, 11, 13  11, 12  12, 10, 14  13, 12  14, 13 | 3  0, 1, 2  4, 8, 9, 6, 7, 5  10, 12, 13, 14, 11 | Same as the graph in the description above. |

## Maximum Tasks Assignment

We have **L** people and **R** tasks. **Each person can complete at most one task**. **One task can be completed by at most one person.** We have a table that shows which people can complete which tasks. Find the **maximum assignment** that assigns tasks to people to complete a maximum number of tasks.

Example: We have 3 people {A, B, C} and 3 tasks {1, 2, 3}. The following table shows whether a person can complete a certain job.

|  |  |  |  |
| --- | --- | --- | --- |
|  | A | B | C |
| 1 | ✓ |  | ✓ |
| 2 |  | ✓ | ✓ |
| 3 | ✓ | ✓ |  |

In the above table, we should make the **maximal assignment**: select from each row and each column at most one checkmark value. A sample solution is shown below:

|  |  |  |  |
| --- | --- | --- | --- |
|  | A | B | C |
| 1 | **✓** |  | ✓ |
| 2 |  | ✓ | **✓** |
| 3 | ✓ | **✓** |  |

### Input

* On the first line, you will receive an integer – people.
* On the next line, you will receive an integer – task.
* On the next people lines, you will receive a line consisting of either "Y" or "N".
  + "Y" means that this person can complete the **ith** task.

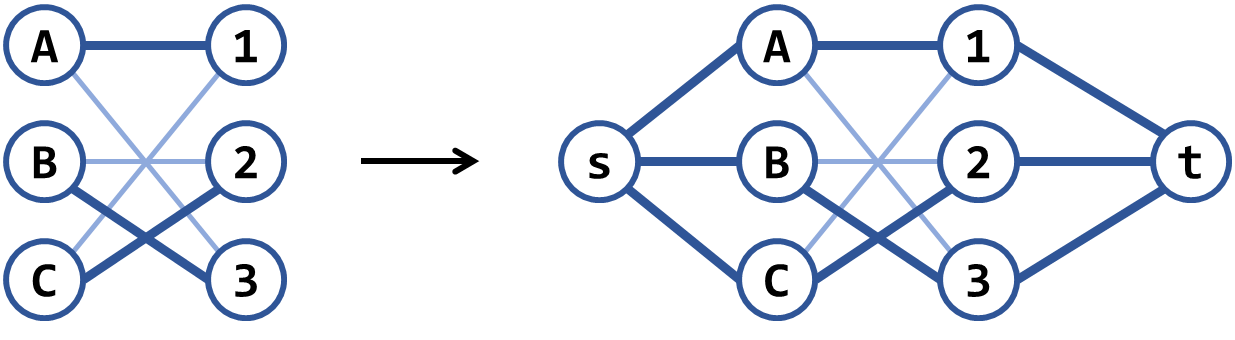
### Examples

Assume people will be marked by letters of the English alphabet and tasks by numbers starting from 1

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | **Output** | **Table** | **Comments** |
| 3  3  YNY  NYY  YYN | A-3  B-2  C-1 | |  |  |  |  | | --- | --- | --- | --- | |  | A | B | C | | 1 | **✓** |  | ✓ | | 2 |  | ✓ | **✓** | | 3 | ✓ | **✓** |  | | Another correct solution:  A-1  B-3  C-2 |
| 4  4  YNYN  NYYN  YNYY  NNNY | A-1  B-2  C-3  D-4 | |  |  |  |  |  | | --- | --- | --- | --- | --- | |  | A | B | C | D | | 1 | **✓** |  | ✓ |  | | 2 |  | **✓** | ✓ |  | | 3 | ✓ |  | **✓** | ✓ | | 4 |  |  |  | **✓** | | Another correct solution:  A-3  B-2  C-1  D-4 |

### Hint

To solve the problem, we can model it as a **bipartite graph** where the left nodes are the people, the right nodes are the tasks, and the edges show who can complete each task. Then we can add **source** and **sink** and model the problem as a **max-flow problem** as shown below (all edges have the same capacity 1):



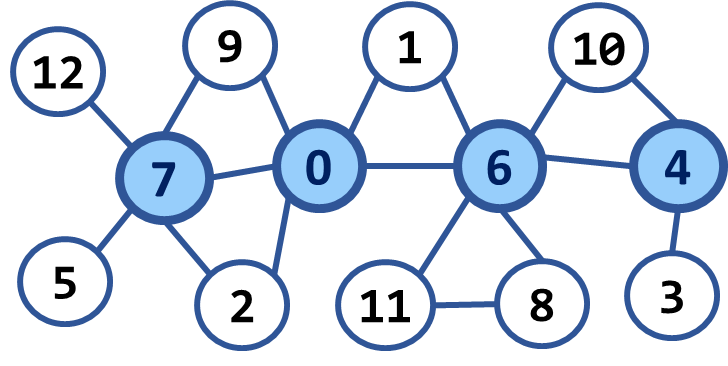
There is another, similar **greedy algorithm**: repeat while possible: connect the nodes having the smallest number of edges, then remove all other nodes connected to these edges. Note that this algorithm works in most scenarios but is wrong in some cases. Can you find a counterexample?

## Find Bi-Connected Components

Finding the **articulation points** in an undirected graph is a well-known problem in computer science. A related problem (a bit harder) is to find the **bi-connected components** in a graph – it's a set of maximal bi-connected subgraphs.

Each bi-connected component has the property that removing any of its nodes does not break the paths between all its other nodes.

Example: the below has 7 bi-connected components: {5, 7}, {12, 7}, {0, 2, 7, 9}, {1, 0, 6}, {6, 8, 11}, {4, 6, 10}, {3, 4}:



### Input

* On the first line, you will receive an integer – n – number of nodes starting from zero.
* On the second line, you will receive an integer – e – number of edges.
* On the next e lines, you will receive edges in the following format: "{first} {second}".

### Example

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | **Output** | **Picture** | **Comments** |
| 13  17  0 1  0 2  0 6  0 7  0 9  1 6  2 7  3 4  4 6  4 10  5 7  6 8  6 10  6 11  7 9  7 12  8 11 | Number of bi-connected components: 7 |  | 5 7  12 7  0 2 7 9  1 0 6  6 8 11  4 6 10  3 4 |
| 13  20  0 1  0 2  0 6  0 7  0 9  0 11  1 6  2 7  3 4  3 8  4 6  4 10  5 7  5 12  6 8  6 10  6 11  7 9  7 12  8 11 | Number of bi-connected components: 3 |  | 12 7 5  9 0 2 7  1 6 10 4 3 8 11 0 |