# Homework: Advanced Graph Algorithms - Part II

This document defines the **homework assignments** for the ["Algortihms" course @ Software University](https://softuni.bg/trainings/1194/Algorithms-September-2015).

## \* Maximum Tasks Assignment

We have **L** persons 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 persons are able to complete which tasks. Find the **maximum assignment** that assigns tasks to persons in order to complete a maximum number of tasks.

Example: we have 3 people {A, B, C} and 3 tasks {1, 2, 3}. We have the following table that 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 | ✓ | **✓** |  |

### Examples

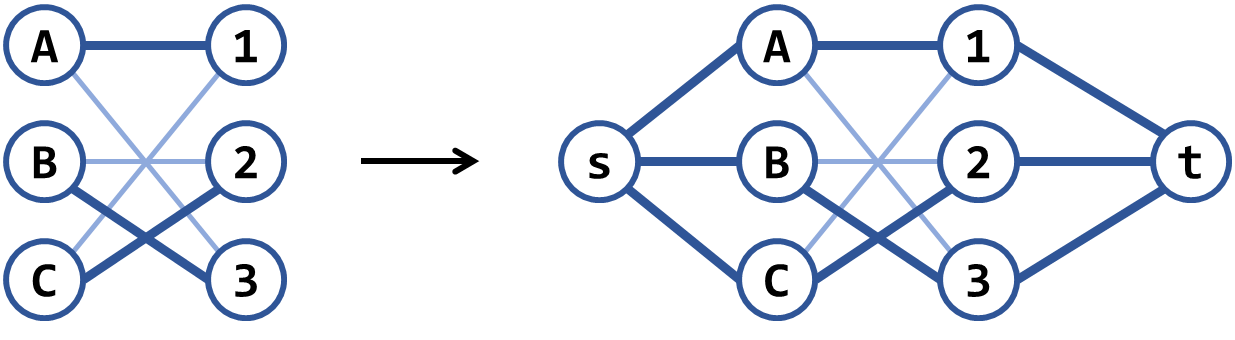
Assume persons will be marked by letters of the English alphabet and tasks by numbers starting from 1

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | **Output** | **Table** | **Comments** |
| Persons: 3  Tasks: 3  YNY  NYY  YYN | A-1  B-3  C-2 | |  |  |  |  | | --- | --- | --- | --- | |  | A | B | C | | 1 | **✓** |  | ✓ | | 2 |  | ✓ | **✓** | | 3 | ✓ | **✓** |  | | Another correct solution:  A-3  B-2  C-1 |
| Persons: 4  Tasks: 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 |

Fiddle with your implementation to find the output shown in the examples.

### Hint

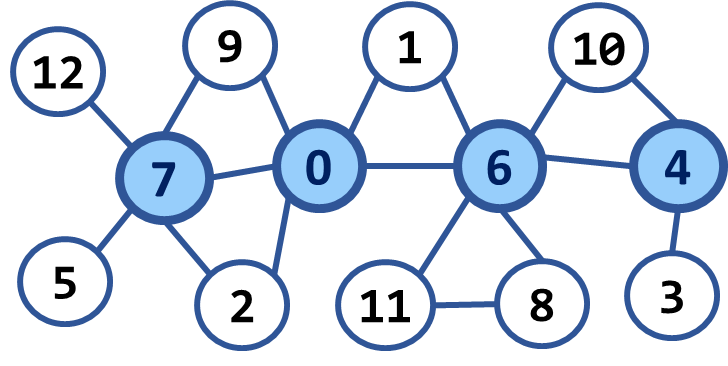
To solve the problem, we can model it as **bipartite graph** where the left nodes are the persons and the right nodes are the tasks and edges show who is able to complete each task. Then we can add **source** and **sink** and model the problem as **max-flow problem** as shown below (all edges have the 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 counter-example?

## \*\* 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 – its 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}:

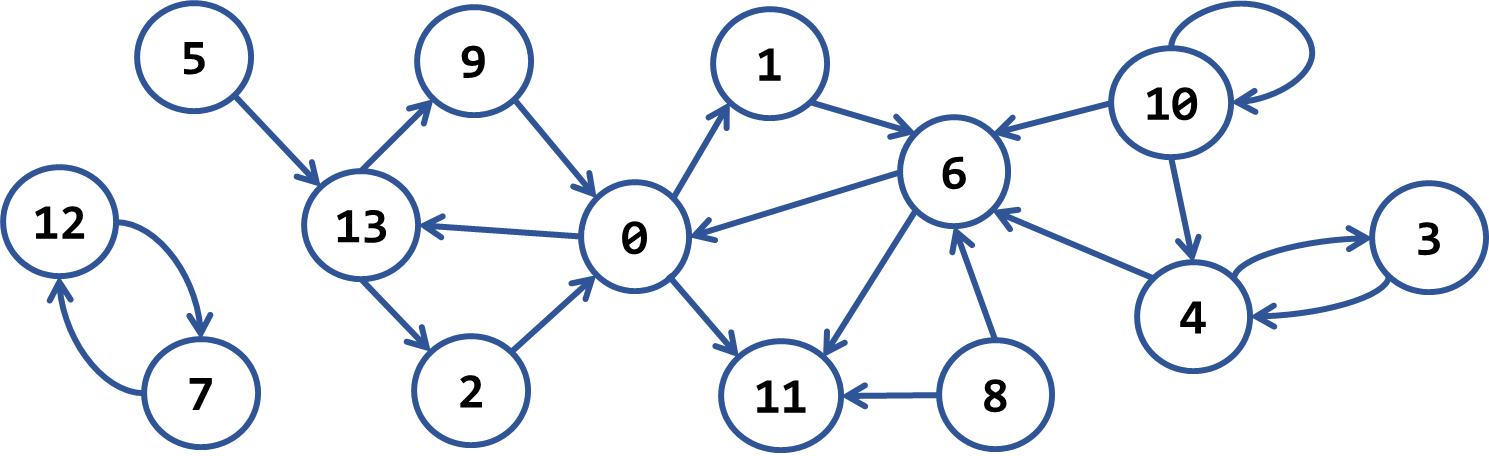


|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | **Output** | **Picture** | **Comments** |
| Nodes: 13  Edges: 17  0 1  0 2  0 6  0 7  0 9  1 6  2 7  3 4  4 6  4 10  5 7  6 8  6 10  6 11  7 9  7 12  8 11 | Number of bi-connected components: 7 |  | 5 7  12 7  0 2 7 9  1 0 6  6 8 11  4 6 10  3 4 |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

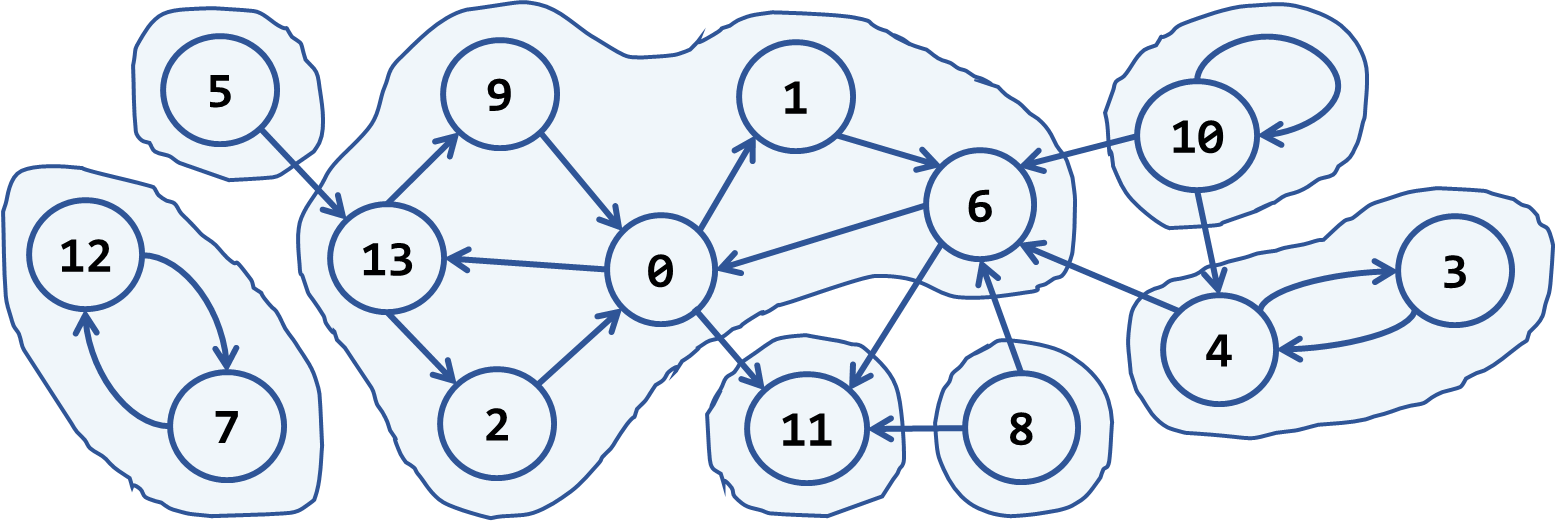
**Hint**: <http://www.cs.umd.edu/class/fall2005/cmsc451/biconcomps.pdf>

## \*\* Supplement Graph to Make It Strongly-Connected

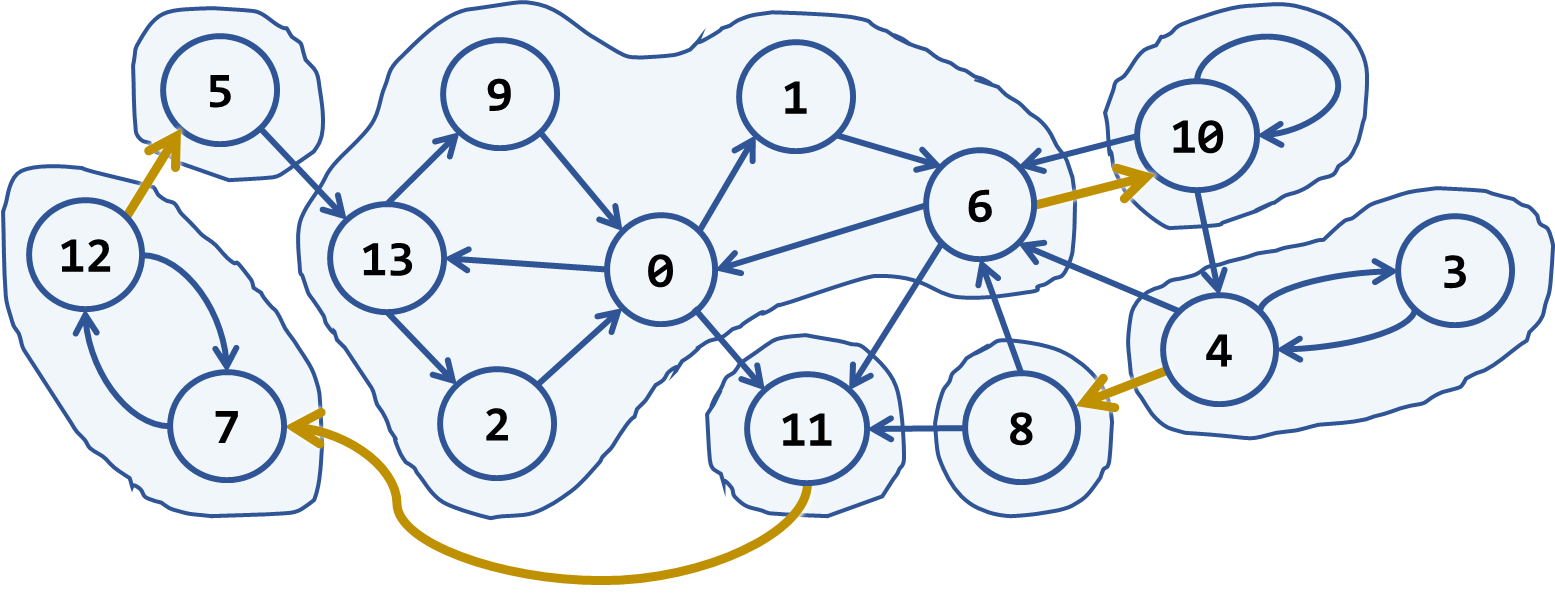
Find the minimum number of (directed) **edges to introduce into a directed graph to make it strongly connected** (from any vertex you can go to any other vertex). Also, find one configuration of edges to add that satisfies the property and reaches the minimum. For example, let’s consider the following graph:



It has the following 7 **strongly-connected components**:



To make it strongly-connected, we should connect each strongly-connected-component to some of the others. For our sample, we need to add at least **4 new edges**. A sample solution is given below:



Sample input and output:

|  |  |
| --- | --- |
| **Input** | **Output** |
| Nodes: 14  Edges: 21  12 -> 7  7 -> 12  5 -> 13  13 -> 9  13 -> 2  9 -> 0  2 -> 0  0 -> 13  0 -> 1  0 -> 11  1 -> 6  6 -> 0  6 -> 11  8 -> 11  8 -> 6  10 -> 6  10 -> 10  10 -> 4  4 -> 6  4 -> 3  3 -> 4 | New edges needed: 4  11 -> 7  12 -> 5  6 -> 10  4 -> 8 |

**Hint**: see <http://stackoverflow.com/questions/14305236/minimal-addition-to-strongly-connected-graph>.