Programming Assignment 2: Decomposition of Graphs

Revision: June 14, 2018

Introduction

Welcome to your second programming assignment of the Graph Algorithms course! In this assignment, we focus on directed graphs and their parts.

Learning Outcomes

Upon completing this programming assignment you will be able to:

1. check consistency of Computer Science curriculum;
2. find an order of courses that is consistent with prerequisite dependencies;
3. check whether any intersection of a city is reachable from any other intersection.

Passing Criteria: 2 out of 3

Passing this programming assignment requires passing at least 2 out of 3 programming challenges from this assignment. In turn, passing a programming challenge requires implementing a solution that passes all the tests for this problem in the grader and does so under the time and memory limits specified in the problem statement.

Contents

Graph Representation in Programming Assignments

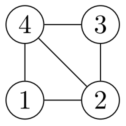
In programming assignments, graphs are given as follows. The first line contains non-negative integers n and m — the number of vertices and the number of edges respectively. The vertices are always numbered from 1 to n. Each of the following m lines defines an edge in the format `u v` where $1 \le u, v \le n$ are endpoints of the edge. If the problem deals with an undirected graph this defines an undirected edge between u and v. In case of a directed graph this defines a directed edge from u to v. If the problem deals with a weighted graph then each edge is given as `u v w` where u and v are vertices and w is a weight.

It is guaranteed that a given graph is simple. That is, it does not contain self-loops (edges going from a vertex to itself) and parallel edges.

Examples:

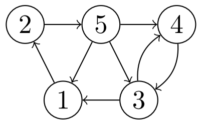
- An undirected graph with four vertices and five edges:

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



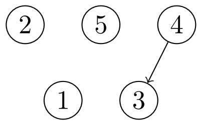
- A directed graph with five vertices and eight edges.

```
5 8
4 3
1 2
3 1
3 4
2 5
5 1
5 4
5 3
```



- A directed graph with five vertices and one edge.

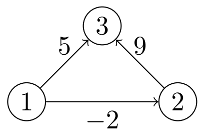
```
5 1
4 3
```



Note that the vertices 1, 2, and 5 are isolated (have no adjacent edges), but they are still present in the graph.

- A weighted directed graph with three vertices and three edges.

```
3 3
2 3 9
1 3 5
1 2 -2
```



1 Checking Consistency of CS Curriculum

Problem Introduction

A Computer Science curriculum specifies the prerequisites for each course as a list of courses that should be taken before taking this course. You would like to perform a consistency check of the curriculum, that is, to check that there are no cyclic dependencies. For this, you construct the following directed graph: vertices correspond to courses, there is a directed edge (u, v) is the course u should be taken before the course v. Then, it is enough to check whether the resulting graph contains a cycle.

Problem Description

Task. Check whether a given directed graph with n vertices and m edges contains a cycle.

Input Format. A graph is given in the standard format.

Constraints. $1 \leq n \leq 10^3$, $0 \leq m \leq 10^3$.

Output Format. Output 1 if the graph contains a cycle and 0 otherwise.

Time Limits.

language	C	C++	Java	Python	Haskell	JavaScript	Scala
time (sec)	1	1	1.5	5	2	5	3

Sample 1.

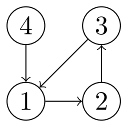
Input:

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

Output:

```
1
```

Explanation:



This graph contains a cycle: $3 \rightarrow 1 \rightarrow 2 \rightarrow 3$.

Sample 2.

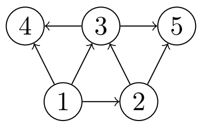
Input:

```
5 7
1 2
2 3
1 3
3 4
1 4
2 5
3 5
```

Output:

```
0
```

Explanation:



There is no cycle in this graph. This can be seen, for example, by noting that all edges in this graph go from a vertex with a smaller number to a vertex with a larger number.

Need Help?

Ask a question or check out the questions asked by other learners at this forum thread.

2 Determining an Order of Courses

Problem Introduction

Now, when you are sure that there are no cyclic dependencies in the given CS curriculum, you would like to find an order of all courses that is consistent with all dependencies. For this, you find a topological ordering of the corresponding directed graph.

Problem Description

Task. Compute a topological ordering of a given directed acyclic graph (DAG) with n vertices and m edges.

Input Format. A graph is given in the standard format.

Constraints. $1 \le n \le 10^5$, $0 \le m \le 10^5$. The given graph is guaranteed to be acyclic.

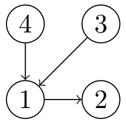
Output Format. Output *any* topological ordering of its vertices. (Many DAGs have more than just one topological ordering. You may output any of them.)

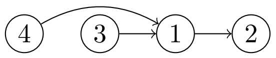
Time Limits.

language	`C`	`C++`	`Java`	`Python`	`Haskell`	`JavaScript`	`Scala`
time (sec)	1	1	1.5	5	2	5	3

Sample 1.

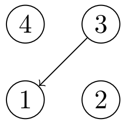
Input:
```
4 3
1 2
4 1
3 1
```
Output:
```
4 3 1 2
```

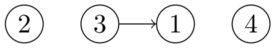




Sample 2.

Input:
```
4 1
3 1
```
Output:
```
2 3 1 4
```





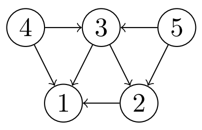
Sample 3.

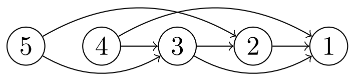
Input:

```
5 7
2 1
3 2
3 1
4 3
4 1
5 2
5 3
```

Output:

```
5 4 3 2 1
```





Need Help?

Ask a question or check out the questions asked by other learners at this forum thread.

3 Checking Whether Any Intersection in a City is Reachable from Any Other

Problem Introduction

The police department of a city has made all streets one-way. You would like to check whether it is still possible to drive legally from any intersection to any other intersection. For this, you construct a directed graph: vertices are intersections, there is an edge (u, v) whenever there is a (one-way) street from u to v in the city. Then, it suffices to check whether all the vertices in the graph lie in the same strongly connected component.

Problem Description

Task. Compute the number of strongly connected components of a given directed graph with n vertices and m edges.

Input Format. A graph is given in the standard format.

Constraints. $1 \leq n \leq 10^4$, $0 \leq m \leq 10^4$.

Output Format. Output the number of strongly connected components.

Time Limits.

language	`C`	`C++`	`Java`	`Python`	`Haskell`	`JavaScript`	`Scala`
time (sec)	1	1	1.5	5	2	5	3

Sample 1.

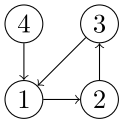
Input:

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

Output:

```
2
```

Explanation:



This graph has two strongly connected components: $\{1, 3, 2\}$, $\{4\}$.

Sample 2.

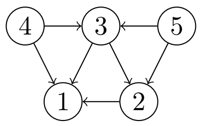
Input:

```
5 7
2 1
3 2
3 1
4 3
4 1
5 2
5 3
```

Output:

```
5
```

Explanation:



This graph has five strongly connected components: {1}, {2}, {3}, {4}, {5}.

4 Determining an Order of Courses

Problem Introduction

Now, when you are sure that there are no cyclic dependencies in the given CS curriculum, you would like to find an order of all courses that is consistent with all dependencies. For this, you find a topological ordering of the corresponding directed graph.

Problem Description

Task. Compute a topological ordering of a given directed acyclic graph (DAG) with n vertices and m edges.

Input Format. A graph is given in the standard format.

Constraints. $1 \le n \le 10^5$, $0 \le m \le 10^5$. The given graph is guaranteed to be acyclic.

Output Format. Output *any* topological ordering of its vertices. (Many DAGs have more than just one topological ordering. You may output any of them.)

Time Limits.

language	`C`	`C++`	`Java`	`Python`	`Haskell`	`JavaScript`	`Scala`
time (sec)	1	1	1.5	5	2	5	3

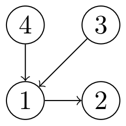
Sample 1.

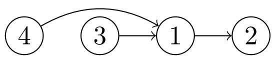
Input:

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

Output:

```
4 3 1 2
```





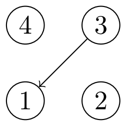
Sample 2.

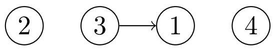
Input:

```
4 1
3 1
```

Output:

```
2 3 1 4
```





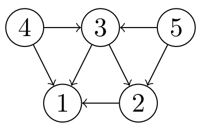
Sample 3.

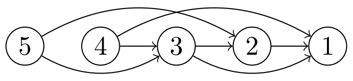
Input:

```
5 7
2 1
3 2
3 1
4 3
4 1
5 2
5 3
```

Output:

```
5 4 3 2 1
```





Need Help?

Ask a question or check out the questions asked by other learners at this forum thread.