Programming Assignment 1: Decomposition of Graphs

Revision: June 10, 2018

Introduction

In this and the next programming assignments you will be practicing implementing the basic building blocks of graph algorithms: computing the number of connected components, checking whether there is a path between the given two vertices, checking whether there is a cycle, etc. Such building blocks are used practically in all applications working with graphs: for example, finding shortest paths on maps, analyzing social networks, analyzing biological data.

In this programming assignment, the grader will show you the input and output data if your solution fails on any of the tests. This is done to help you to get used to the algorithmic problems in general and get some experience debugging your programs while knowing exactly on which tests they fail. However, for all the following programming assignments, the grader will show the input data only in case your solution fails on one of the first few tests.

Learning Outcomes

Upon completing this programming assignment you will be able to:

1. find an exit from a maze;
2. find the number of exits needed for a maze;

Passing Criteria: 1 out of 2

Passing this programming assignment requires passing at least 1 out of 2 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

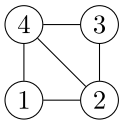
In programming assignments, graphs are given as follows. The first line contains non-negative integers n and m — the number of vertices and the number of edges respectively. The vertices are always numbered from 1 to n. Each of the following m lines defines an edge in the format `u v` where $1 \leq u, v \leq n$ 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:

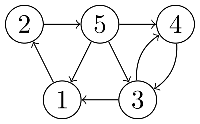
- An undirected graph with four vertices and five edges:

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



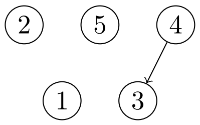
- A directed graph with five vertices and eight edges.

```
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.

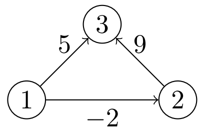
```
5 1
4 3
```



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.

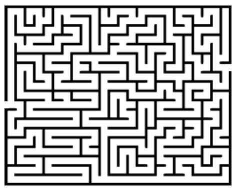
```
3 3
2 3 9
1 3 5
1 2 -2
```



1 Finding an Exit from a Maze

Problem Introduction

A maze is a rectangular grid of cells with walls between some of adjacent cells. You would like to check whether there is a path from a given cell to a given exit from a maze where an exit is also a cell that lies on the border of the maze (in the example shown to the right there are two exits: one on the left border and one on the right border). For this, you represent the maze as an undirected graph: vertices of the graph are cells of the maze, two vertices are connected by an undirected edge if they are adjacent and there is no wall between them. Then, to check whether there is a path between two given cells in the maze, it suffices to check that there is a path between the corresponding two vertices in the graph.



Problem Description

Task. Given an undirected graph and two distinct vertices u and v, check if there is a path between u and v.

Input Format. An undirected graph with n vertices and m edges. The next line contains two vertices u and v of the graph.

Constraints. $2 \leq n \leq 10^3$; $1 \leq m \leq 10^3$; $1 \leq u, v \leq n$; $u \neq v$.

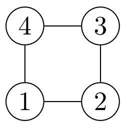
Output Format. Output 1 if there is a path between u and v and 0 otherwise.

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



In this graph, there are two paths between vertices 1 and 4: 1-4 and 1-2-3-4.

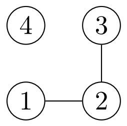
Sample 2.

Input:

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

Output:

```
0
```



In this case, there is no path from 1 to 4.

Need Help?

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

2 Adding Exits to a Maze

Problem Introduction

Now you decide to make sure that there are no dead zones in a maze, that is, that at least one exit is reachable from each cell. For this, you find connected components of the corresponding undirected graph and ensure that each component contains an exit cell.

Problem Description

Task. Given an undirected graph with n vertices and m edges, compute the number of connected components in it.

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 the number of connected components.

Time Limits.

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

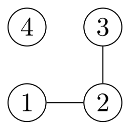
Sample 1.

Input:

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

Output:

```
2
```



There are two connected components here: $\{1, 2, 3\}$ and $\{4\}$.

Need Help?

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