Programming Assignment 1: Decomposition of Graphs

Revision: July 27, 2020

Introduction

Welcome to your first programming assignment of the Algorithms on Graphs class! 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 (please review the questions **??** and **??** in the FAQ section for a more detailed explanation of this behavior of the grader).

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

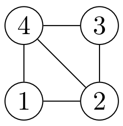
In programming assignments, graphs are given as follows. The first line contains non-negative integers n and m — the number of vertices and the number of edges respectively. The vertices are always numbered from 1 to n. Each of the following m lines defines an edge in the format `u v` where $1 \leq u,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

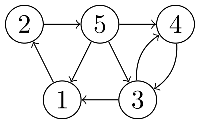
- An undirected graph with four vertices and five edges:

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



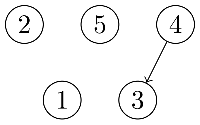
- A directed graph with five vertices and eight edges.

```
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.

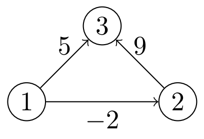
```
5 1
4 3
```



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.

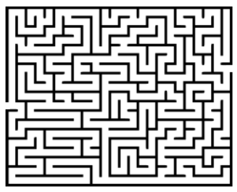
```
3 3
2 3 9
1 3 5
1 2 -2
```



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	C#	Haskell	JavaScript	Ruby	Scala
time (sec)	1	1	1.5	5	1.5	2	5	5	3

Memory Limit. 512MB.

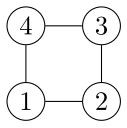
Sample 1.

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.

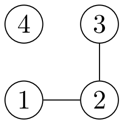
Sample 2.

Input:

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

Output:

```
0
```



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

Starter Files

The starter solutions for this problem read the input data from the standard input, pass it to a blank procedure, and then write the result to the standard output. You are supposed to implement your algorithm in this blank procedure if you are using `C++`, `Java`, or `Python3`. For other programming languages, you need to implement a solution from scratch. Filename: `reachability`

What To Do

To solve this problem, it is enough to implement carefully the corresponding algorithm covered in the lectures.

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. $1 \le n \le 10^3$, $0 \le m \le 10^3$.

Output Format. Output the number of connected components.

Time Limits.

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

Memory Limit. 512MB.

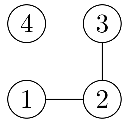
Sample 1.

Input:

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

Output:

```
2
```



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

Starter Files

The starter solutions for this problem read the input data from the standard input, pass it to a blank procedure, and then write the result to the standard output. You are supposed to implement your algorithm in this blank procedure if you are using `C++`, `Java`, or `Python3`. For other programming languages, you need to implement a solution from scratch. Filename: `connected_components`

3 Appendix

3.1 Compiler Flags

C (`gcc 7.4.0`). File extensions: `.c`. Flags:

```
gcc -pipe -O2 -std=c11 <filename> -lm
```

C++ (`g++ 7.4.0`). File extensions: `.cc`, `.cpp`. Flags:

```
g++ -pipe -O2 -std=c++14 <filename> -lm
```

If your `C/C++` compiler does not recognize `-std=c++14` flag, try replacing it with `-std=c++0x` flag or compiling without this flag at all (all starter solutions can be compiled without it). On Linux and MacOS, you most probably have the required compiler. On Windows, you may use your favorite compiler or install, e.g., `cygwin`.

C# (`mono 4.6.2`). File extensions: `.cs`. Flags:

```
mcs
```

Go (`golang 1.13.4`). File extensions: `.go`. Flags

```
go
```

Haskell (`ghc 8.0.2`). File extensions: `.hs`. Flags:

```
ghc -O2
```

Java (`OpenJDK 1.8.0_232`). File extensions: `.java`. Flags:

```
javac -encoding UTF-8
java -Xmx1024m
```

JavaScript (`NodeJS 12.14.0`). File extensions: `.js`. No flags:

```
nodejs
```

Kotlin (`Kotlin 1.3.50`). File extensions: `.kt`. Flags:

```
kotlinc
java -Xmx1024m
```

Python (`CPython 3.6.9`). File extensions: `.py`. No flags:

```
python3
```

Ruby (`Ruby 2.5.1p57`). File extensions: `.rb`.

```
ruby
```

Rust (`Rust 1.37.0`). File extensions: `.rs`.

```
rustc
```

Scala (`Scala 2.12.10`). File extensions: `.scala`.

```
scalac
```

3.2 Frequently Asked Questions

Why My Submission Is Not Graded?

You need to create a submission and upload the *source file* (rather than the executable file) of your solution. Make sure that after uploading the file with your solution you press the blue “Submit” button at the bottom. After that, the grading starts, and the submission being graded is enclosed in an orange rectangle. After the testing is finished, the rectangle disappears, and the results of the testing of all problems are shown.

What Are the Possible Grading Outcomes?

There are only two outcomes: “pass” or “no pass.” To pass, your program must return a correct answer on all the test cases we prepared for you, and do so under the time and memory constraints specified in the problem statement. If your solution passes, you get the corresponding feedback "Good job!" and get a point for the problem. Your solution fails if it either crashes, returns an incorrect answer, works for too long, or uses too much memory for some test case. The feedback will contain the index of the first test case on which your solution failed and the total number of test cases in the system. The tests for the problem are numbered from 1 to the total number of test cases for the problem, and the program is always tested on all the tests in the order from the first test to the test with the largest number.

Here are the possible outcomes:

- `Good job!` Hurrah! Your solution passed, and you get a point!
- `Wrong answer.` Your solution outputs incorrect answer for some test case. Check that you consider all the cases correctly, avoid integer overflow, output the required white spaces, output the floating point numbers with the required precision, don’t output anything in addition to what you are asked to output in the output specification of the problem statement.
- `Time limit exceeded.` Your solution worked longer than the allowed time limit for some test case. Check again the running time of your implementation. Test your program locally on the test of maximum size specified in the problem statement and check how long it works. Check that your program doesn’t wait for some input from the user which makes it to wait forever.
- `Memory limit exceeded.` Your solution used more than the allowed memory limit for some test case. Estimate the amount of memory that your program is going to use in the worst case and check that it does not exceed the memory limit. Check that your data structures fit into the memory limit. Check that you don’t create large arrays or lists or vectors consisting of empty arrays or empty strings, since those in some cases still eat up memory. Test your program locally on the tests of maximum size specified in the problem statement and look at its memory consumption in the system.
- `Cannot check answer. Perhaps the output format is wrong.` This happens when you output something different than expected. For example, when you are required to output either “Yes” or “No”, but instead output 1 or 0. Or your program has empty output. Or your program outputs not only the correct answer, but also some additional information (please follow the exact output format specified in the problem statement). Maybe your program doesn’t output anything, because it crashes.
- `Unknown signal 6 (or 7, or 8, or 11, or some other).` This happens when your program crashes. It can be because of a division by zero, accessing memory outside of the array bounds, using uninitialized variables, overly deep recursion that triggers a stack overflow, sorting with a contradictory comparator, removing elements from an empty data structure, trying to allocate too much memory, and many other reasons. Look at your code and think about all those possibilities. Make sure that you use the same compiler and the same compiler flags as we do.
- `Internal error: exception...` Most probably, you submitted a compiled program instead of a source code.

- `Grading failed.` Something wrong happened with the system. Report this through Coursera or edX Help Center.

May I Post My Solution at the Forum?

Please do not post any solutions at the forum or anywhere on the web, even if a solution does not pass the tests (as in this case you are still revealing parts of a correct solution). Our students follow the Honor Code: "I will not make solutions to homework, quizzes, exams, projects, and other assignments available to anyone else (except to the extent an assignment explicitly permits sharing solutions)."

Do I Learn by Trying to Fix My Solution?

My implementation always fails in the grader, though I already tested and stress tested it a lot. Would not it be better if you gave me a solution to this problem or at least the test cases that you use? I will then be able to fix my code and will learn how to avoid making mistakes. Otherwise, I do not feel that I learn anything from solving this problem. I am just stuck.

First of all, learning from your mistakes is one of the best ways to learn.

The process of trying to invent new test cases that might fail your program is difficult but is often enlightening. Thinking about properties of your program makes you understand what happens inside your program and in the general algorithm you're studying much more.

Also, it is important to be able to find a bug in your implementation without knowing a test case and without having a reference solution, just like in real life. Assume that you designed an application and an annoyed user reports that it crashed. Most probably, the user will not tell you the exact sequence of operations that led to a crash. Moreover, there will be no reference application. Hence, it is important to learn how to find a bug in your implementation yourself, without a magic oracle giving you either a test case that your program fails or a reference solution. We encourage you to use programming assignments in this class as a way of practicing this important skill.

If you have already tested your program on all corner cases you can imagine, constructed a set of manual test cases, applied stress testing, etc, but your program still fails, try to ask for help on the forum. We encourage you to do this by first explaining what kind of corner cases you have already considered (it may happen that by writing such a post you will realize that you missed some corner cases!), and only afterwards asking other learners to give you more ideas for tests cases.