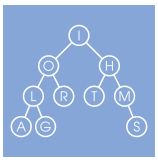


Module: Decomposition of Graphs 1 (Week 1 out of 5)
Course: Algorithms on Graphs (Course 3 out of 6)
Specialization: Data Structures and Algorithms

Programming Assignment 1: Decomposition of Graphs

Revision: August 17, 2016

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 5.4 and 5.5 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 code problems from this assignment. In turn, passing a code problem 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

1 Graph Representation in Programming Assignments

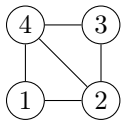
In programming assignments, graphs are given as follows. The first line contains non-negative integers n and m — the number of vertices and the number of edges respectively. The vertices are always numbered from 1 to n. Each of the following m lines defines an edge in the format `u v` where $1 \le u,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

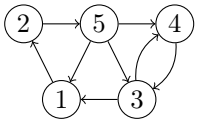
- An undirected graph with four vertices and five edges:

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



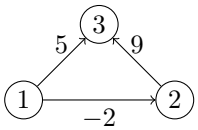
- A directed graph with five vertices and eight edges.

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



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

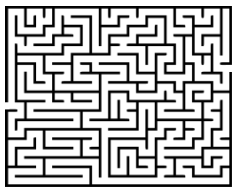
```
3 3
2 3 9
1 3 5
1 2 -2
```



2 Problem: 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. $2 \le n \le 10^3$; $1 \le m \le 10^3$; $1 \le u, v \le n$; $u \ne v$.

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 in seconds	1	1	1.5	5	1.5	2	5	5	3

Memory Limit. 512Mb.

Sample 1.

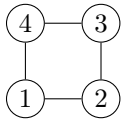
Input:

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

Output:

```
1
```

Explanation:



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

Sample 2.

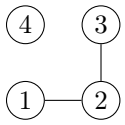
Input:

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

Output:

```
0
```

Explanation:



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.

3 Problem: 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 in seconds	1	1	1.5	5	1.5	2	5	5	3

Memory Limit. 512Mb.

Sample 1.

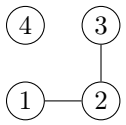
Input:

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

Output:

```
2
```

Explanation:



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`

What To Do

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

4 General Instructions and Recommendations on Solving Algorithmic Problems

Your main goal in an algorithmic problem is to implement a program that solves a given computational problem in just few seconds even on massive datasets. Your program should read a dataset from the standard input and write an answer to the standard output.

Below we provide general instructions and recommendations on solving such problems. Before reading them, go through readings and screencasts in the first module that show a step by step process of solving two algorithmic problems: link.

4.1 Reading the Problem Statement

You start by reading the problem statement that contains the description of a particular computational task as well as time and memory limits your solution should fit in, and one or two sample tests. In some problems your goal is just to implement carefully an algorithm covered in the lectures, while in some other problems you first need to come up with an algorithm yourself.

4.2 Designing an Algorithm

If your goal is to design an algorithm yourself, one of the things it is important to realize is the expected running time of your algorithm. Usually, you can guess it from the problem statement (specifically, from the subsection called constraints) as follows. Modern computers perform roughly 10^8–10^9 operations per second. So, if the maximum size of a dataset in the problem description is $n = 10^5$, then most probably an algorithm with quadratic running time is not going to fit into time limit (since for $n = 10^5$, $n^2 = 10^{10}$) while a solution with running time $O(n \log n)$ will fit. However, an $O(n^2)$ solution will fit if n is up to $10^3 = 1000$, and if n is at most 100, even $O(n^3)$ solutions will fit. In some cases, the problem is so hard that we do not know a polynomial solution. But for n up to 18, a solution with $O(2^n n^2)$ running time will probably fit into the time limit.

To design an algorithm with the expected running time, you will of course need to use the ideas covered in the lectures. Also, make sure to carefully go through sample tests in the problem description.

4.3 Implementing Your Algorithm

When you have an algorithm in mind, you start implementing it. Currently, you can use the following programming languages to implement a solution to a problem: `C`, `C++`, `C#`, `Haskell`, `Java`, `JavaScript`, `Python2`, `Python3`, `Ruby`, `Scala`. For all problems, we will be providing starter solutions for `C++`, `Java`, and `Python3`. If you are going to use one of these programming languages, use these starter files. For other programming languages, you need to implement a solution from scratch.

4.4 Compiling Your Program

For solving programming assignments, you can use any of the following programming languages: `C`, `C++`, `C#`, `Haskell`, `Java`, `JavaScript`, `Python2`, `Python3`, `Ruby`, and `Scala`. However, we will only be providing starter solution files for `C++`, `Java`, and `Python3`. The programming language of your submission is detected automatically, based on the extension of your submission.

We have reference solutions in `C++`, `Java` and `Python3` which solve the problem correctly under the given restrictions, and in most cases spend at most 1/3 of the time limit and at most 1/2 of the memory limit. You can also use other languages, and we've estimated the time limit multipliers for them, however, we have no guarantee that a correct solution for a particular problem running under the given time and memory constraints exists in any of those other languages.

Your solution will be compiled as follows. We recommend that when testing your solution locally, you use the same compiler flags for compiling. This will increase the chances that your program behaves in the

same way on your machine and on the testing machine (note that a buggy program may behave differently when compiled by different compilers, or even by the same compiler with different flags).

- `C` (`gcc 5.2.1`). File extensions: `.c`. Flags:

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

- `C++` (`g++ 5.2.1`). 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 3.2.8`). File extensions: `.cs`. Flags:

```
mcs
```

- `Haskell` (`ghc 7.8.4`). File extensions: `.hs`. Flags:

```
ghc -O
```

- `Java (Open JDK 8)`. File extensions: `.java`. Flags:

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

- `JavaScript` (`Node v6.3.0`). File extensions: `.js`. Flags:

```
nodejs
```

- `Python 2 (CPython 2.7)`. File extensions: `.py2` or `.py` (a file ending in `.py` needs to have a first line which is a comment containing "python2"). No flags:

```
python2
```

- `Python 3 (CPython 3.4)`. File extensions: `.py3` or `.py` (a file ending in `.py` needs to have a first line which is a comment containing "python3"). No flags:

```
python3
```

- `Ruby` (`Ruby 2.1.5`). File extensions: `.rb`.

```
ruby
```

- `Scala` (`Scala 2.11.6`). File extensions: `.scala`.

```
scalac
```

4.5 Testing Your Program

When your program is ready, you start testing it. It makes sense to start with small datasets — for example, sample tests provided in the problem description. Ensure that your program produces a correct result.

You then proceed to checking how long does it take your program to process a massive dataset. For this, it makes sense to implement your algorithm as a function like `solve(dataset)` and then implement an additional procedure `generate()` that produces a large dataset. For example, if an input to a problem is a sequence of integers of length $1 \le n \le 10^5$, then generate a sequence of length exactly 10^5, pass it to your `solve()` function, and ensure that the program outputs the result quickly.

Also, check the boundary values. Ensure that your program processes correctly sequences of size $n = 1, 2, 10^5$. If a sequence of integers from 0 to, say, 10^6 is given as an input, check how your program behaves when it is given a sequence $0, 0, \ldots, 0$ or a sequence $10^6, 10^6, \ldots, 10^6$. Check also on randomly generated data. For each such test check that you program produces a correct result (or at least a reasonably looking result).

In the end, we encourage you to stress test your program to make sure it passes in the system at the first attempt. See the readings and screencasts from the first week to learn about testing and stress testing: link.

4.6 Submitting Your Program to the Grading System

When you are done with testing, you submit your program to the grading system. For this, you go the submission page, create a new submission, and upload a file with your program. The grading system then compiles your program (detecting the programming language based on your file extension, see Subsection 4.4) and runs it on a set of carefully constructed tests to check that your program always outputs a correct result and that it always fits into the given time and memory limits. The grading usually takes no more than a minute, but in rare cases when the servers are overloaded it might take longer. Please be patient. You can safely leave the page when your solution is uploaded.

As a result, you get a feedback message from the grading system. The feedback message that you will love to see is: `Good job!` This means that your program has passed all the tests. On the other hand, the three messages `Wrong answer`, `Time limit exceeded`, `Memory limit exceeded` notify you that your program failed due to one these three reasons. Note that the grader will not show you the actual test you program have failed on (though it does show you the test if your program have failed on one of the first few tests; this is done to help you to get the input/output format right).

4.7 Debugging and Stress Testing Your Program

If your program failed, you will need to debug it. Most probably, you didn't follow some of our suggestions from the section 4.5. See the readings and screencasts from the first week to learn about debugging your program: link.

You are almost guaranteed to find a bug in your program using stress testing, because the way these programming assignments and tests for them are prepared follows the same process: small manual tests, tests for edge cases, tests for large numbers and integer overflow, big tests for time limit and memory limit checking, random test generation. Also, implementation of wrong solutions which we expect to see and stress testing against them to add tests specifically against those wrong solutions.

Go ahead, and we hope you pass the assignment soon!

5 Frequently Asked Questions

5.1 I submit the program, but nothing happens. Why?

You need to create submission and upload the file with your solution in one of the programming languages `C`, `C++`, `Java`, or `Python` (see Subsections 4.3 and 4.4). Make sure that after uploading the file with your solution you press on the blue "Submit" button in 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 is shown to you.

5.2 I submit the solution only for one problem, but all the problems in the assignment are graded. Why?

Each time you submit any solution, the last uploaded solution for each problem is tested. Don't worry: this doesn't affect your score even if the submissions for the other problems are wrong. As soon as you pass the sufficient number of problems in the assignment (see in the pdf with instructions), you pass the assignment. After that, you can improve your result if you successfully pass more problems from the assignment. We recommend working on one problem at a time, checking whether your solution for any given problem passes in the system as soon as you are confident in it. However, it is better to test it first, please refer to the reading about stress testing: link.

5.3 What are the possible grading outcomes, and how to read them?

Your solution may either pass or not. To pass, it must work without crashing and return the correct answers on all the test cases we prepared for you, and do so under the time limit and memory limit constraints specified in the problem statement. If your solution passes, you get the corresponding feedback "Good job!" and get a point for the problem. If your solution fails, it can be because it crashes, returns wrong answer, works for too long or uses too much memory for some test case. The feedback will contain the number of the test case on which your solution fails 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 test number 1 to the test with the biggest number.

Here are the possible outcomes:

`Good job!` Hurrah! Your solution passed, and you get a point!

`Wrong answer.` Your solution has output incorrect answer for some test case. If it is a sample test case from the problem statement, or if you are solving Programming Assignment 1, you will also see the input data, the output of your program and the correct answer. Otherwise, you won't know the input, the output, and the correct answer. Check that you consider all the cases correctly, avoid integer overflow, output the required white space, 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. See this reading on testing: link.

`Time limit exceeded.` Your solution worked longer than the allowed time limit for some test case. If it is a sample test case from the problem statement, or if you are solving Programming Assignment 1, you will also see the input data and the correct answer. Otherwise, you won't know the input and the correct answer. Check again that your algorithm has good enough running time estimate. Test your program locally on the test of maximum size allowed by the problem statement and see how long it works. Check that your program doesn't wait for some input from the user which makes it to wait forever. See this reading on testing: link.

`Memory limit exceeded.` Your solution used more than the allowed memory limit for some test case. If it is a sample test case from the problem statement, or if you are solving Programming Assignment 1,

you will also see the input data and the correct answer. Otherwise, you won't know the input and the correct answer. Estimate the amount of memory that your program is going to use in the worst case and check that it is less than the memory limit. Check that you don't create too large arrays or data structures. 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 test of maximum size allowed by the problem statement and look at its memory consumption in the system.

`Cannot check answer. Perhaps output format is wrong.` This happens when you output something completely different than expected. For example, you are required to output word "Yes" or "No", but you output number 1 or 0, or vice versa. Or your program has empty output. Or your program outputs not only the correct answer, but also some additional information (this is not allowed, so please follow exactly the 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 division by zero, accessing memory outside of the array bounds, using uninitialized variables, too deep recursion that triggers stack overflow, sorting with 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 compilers and the same compiler options as we do. Try different testing techniques from this reading: link.

`Internal error: exception...` Most probably, you submitted a compiled program instead of a source code.

`Grading failed.` Something very wrong happened with the system. Contact Coursera for help or write in the forums to let us know.

5.4 How to understand why my program fails and to fix it?

If your program works incorrectly, it gets a feedback from the grader. For the Programming Assignment 1, when your solution fails, you will see the input data, the correct answer and the output of your program in case it didn't crash, finished under the time limit and memory limit constraints. If the program crashed, worked too long or used too much memory, the system stops it, so you won't see the output of your program or will see just part of the whole output. We show you all this information so that you get used to the algorithmic problems in general and get some experience debugging your programs while knowing exactly on which tests they fail.

However, in the following Programming Assignments throughout the Specialization you will only get so much information for the test cases from the problem statement. For the next tests you will only get the result: passed, time limit exceeded, memory limit exceeded, wrong answer, wrong output format or some form of crash. We hide the test cases, because it is crucial for you to learn to test and fix your program even without knowing exactly the test on which it fails. In the real life, often there will be no or only partial information about the failure of your program or service. You will need to find the failing test case yourself. Stress testing is one powerful technique that allows you to do that. You should apply it after using the other testing techniques covered in this reading.

5.5 Why do you hide the test on which my program fails?

Often beginner programmers think by default that their programs work. Experienced programmers know, however, that their programs almost never work initially. Everyone who wants to become a better programmer needs to go through this realization.

When you are sure that your program works by default, you just throw a few random test cases against it, and if the answers look reasonable, you consider your work done. However, mostly this is not enough. To

make one's programs work, one must test them really well. Sometimes, the programs still don't work although you tried really hard to test them, and you need to be both skilled and creative to fix your bugs. Solutions to algorithmic problems are one of the hardest to implement correctly. That's why in this Specialization you will gain this important experience which will be invaluable in the future when you write programs which you really need to get right.

It is crucial for you to learn to test and fix your programs yourself. In the real life, often there will be no or only partial information about the failure of your program or service. Still, you will have to reproduce the failure to fix it (or just guess what it is, but that's rare, and you will still need to reproduce the failure to make sure you have really fixed it). When you solve algorithmic problems, it is very frequent to make subtle mistakes. That's why you should apply the testing techniques described in this reading to find the failing test case and fix your program.

5.6 My solution does not pass the tests? May I post it in the forum and ask for a help?

No, please do not post any solutions in 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). Recall the third item of the Coursera 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). This includes both solutions written by me, as well as any solutions provided by the course staff or others" (link).

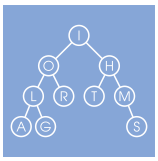
5.7 My implementation always fails in the grader, though I already tested and stress tested it a lot. Would not it be better if you give me a solution to this problem or at least the test cases that you use? I will then be able to fix my code and will learn how to avoid making mistakes. Otherwise, I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

First of all, you always learn from your mistakes.

The process of trying to invent new test cases that might fail your program and proving them wrong is often enlightening. This thinking about the invariants which you expect your loops, ifs, etc. to keep and proving them wrong (or right) 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. 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, once again, it is important to be able to locate 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 a lot (considered all corner cases that you can imagine, constructed a set of manual test cases, applied stress testing), but your program still fails and you are stuck, 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 when writing such a post you will realize that you missed some corner cases!) and only then asking other learners to give you more ideas for tests cases.



Module: Decomposition of Graphs 2 (Week 2 out of 6)
Course: Algorithms on Graphs (Course 3 out of 6)
Specialization: Data Structures and Algorithms

Programming Assignment 2: Decomposition of Graphs

Revision: November 21, 2016

Introduction

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

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 6.4 and 6.5 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. 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 code problems from this assignment. In turn, passing a code problem 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.

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

Memory Limit. 512MB.

Sample 1.

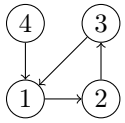
Input:

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

Output:

```
1
```

Explanation:



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

Sample 2.

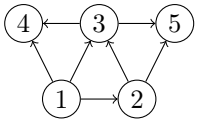
Input:

```
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.

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: `acyclicity`

What To Do

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

3 Problem: 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. $1 \leq n \leq 10^5$, $0 \leq m \leq 10^5$. 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`	`C#`	`Haskell`	`JavaScript`	`Ruby`	`Scala`
time (sec)	2	2	3	10	3	4	10	10	6

Memory Limit. 512MB.

Sample 1.

Input:

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

Output:

```
4 3 1 2
```

Explanation:

Sample 2.

Input:

```
4 1
3 1
```

Output:

```
2 3 1 4
```

Explanation:

Sample 3.

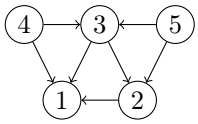
Input:

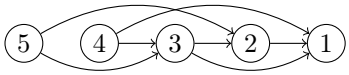
```
5 7
2 1
3 2
3 1
4 3
4 1
5 2
5 3
```

Output:

```
5 4 3 2 1
```

Explanation:





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: `toposort`

What To Do

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

4 Advanced Problem: Checking Whether Any Intersection in a City is Reachable from Any Other

We strongly recommend you start solving advanced problems only when you are done with the basic problems (for some advanced problems, algorithms are not covered in the video lectures and require additional ideas to be solved; for some other advanced problems, algorithms are covered in the lectures, but implementing them is a more challenging task than for other problems).

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. $1 \le n \le 10^4$, $0 \le m \le 10^4$.

Output Format. Output the number of strongly 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.

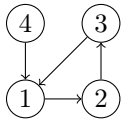
Input:

```
4 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.

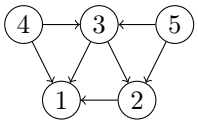
Input:

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

Output:

```
5
```

Explanation:



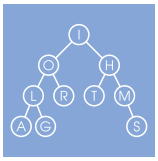
This graph has five strongly connected components: {1}, {2}, {3}, {4}, {5}.

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.

What To Do

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



Module: Paths in Graphs (Week 3 out of 5)
Course: Algorithms on Graphs (Course 3 out of 6)
Specialization: Data Structures and Algorithms

Programming Assignment 3: Paths in Graphs

Revision: July 13, 2016

Introduction

Welcome to your third programming assignment of the Algorithms on Graphs class! In this and the next programming assignments you will be practicing implementing algorithms for finding shortest paths in graphs.

Recall that starting from this programming assignment, the grader will show you only the first few tests (see the questions 5.4 and 5.5 in the FAQ section).

Learning Outcomes

Upon completing this programming assignment you will be able to:

1. compute the minimum number of flight segments to get from one city to another one;
2. check whether a given graph is bipartite.

Passing Criteria: 1 out of 2

Passing this programming assignment requires passing at least 1 out of 2 code problems from this assignment. In turn, passing a code problem 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.

2 Problem: Computing the Minimum Number of Flight Segments

Problem Introduction

You would like to compute the minimum number of flight segments to get from one city to another one. For this, you construct the following undirected graph: vertices represent cities, there is an edge between two vertices whenever there is a flight between the corresponding two cities. Then, it suffices to find a shortest path from one of the given cities to the other one.

Problem Description

Task. Given an *undirected* graph with n vertices and m edges and two vertices u and v, compute the length of a shortest path between u and v (that is, the minimum number of edges in a path from u to v).

Input Format. A graph is given in the standard format. The next line contains two vertices u and v.

Constraints. $2 \le n \le 10^5$, $0 \le m \le 10^5$, $u \ne v$, $1 \le u, v \le n$.

Output Format. Output the minimum number of edges in a path from u to v, or -1 if there is no path.

Time Limits.

language	`C`	`C++`	`Java`	`Python`	`C#`	`Haskell`	`JavaScript`	`Ruby`	`Scala`
time in seconds	2	2	3	10	3	4	10	10	6

Memory Limit. 512Mb.

Sample 1.

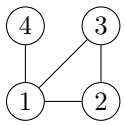
Input:

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

Output:

```
2
```

Explanation:



There is a unique shortest path between vertices 2 and 4 in this graph: $2 - 1 - 4$.

Sample 2.

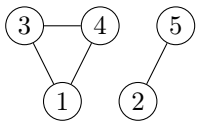
Input:

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

Output:

```
-1
```

Explanation:



There is no path between vertices 3 and 5 in this graph.

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: `bfs`

What To Do

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

3 Problem: Checking whether a Graph is Bipartite

Problem Introduction

An undirected graph is called *bipartite* if its vertices can be split into two parts such that each edge of the graph joins to vertices from different parts. Bipartite graphs arise naturally in applications where a graph is used to model connections between objects of two different types (say, boys and girls; or students and dormitories).

An alternative definition is the following: a graph is bipartite if its vertices can be colored with two colors (say, black and white) such that the endpoints of each edge have different colors.

Problem Description

Task. Given an undirected graph with n vertices and m edges, check whether it is bipartite.

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

Constraints. $1 \le n \le 10^5$, $0 \le m \le 10^5$.

Output Format. Output 1 if the graph is bipartite and 0 otherwise.

Time Limits.

language	`C`	`C++`	`Java`	`Python`	`C#`	`Haskell`	`JavaScript`	`Ruby`	`Scala`
time in seconds	2	2	3	10	3	4	10	10	6

Memory Limit. 512Mb.

Sample 1.

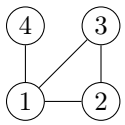
Input:

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

Output:

```
0
```

Explanation:



This graph is not bipartite. To see this assume that the vertex 1 is colored white. Then the vertices 2 and 3 should be colored black since the graph contains the edges $\{1, 2\}$ and $\{1, 3\}$. But then the edge $\{2, 3\}$ has both endpoints of the same color.

Sample 2.

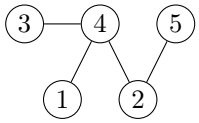
Input:

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

Output:

```
1
```

Explanation:



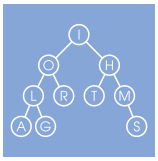
This graph is bipartite: assign the vertices 4 and 5 the white color, assign all the remaining vertices the black color.

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: `bipartite`

What To Do

Adapt the breadth-first search to solve this problem.



Module: Paths in Graphs (Week 4 out of 5)
Course: Algorithms on Graphs (Course 3 out of 6)
Specialization: Data Structures and Algorithms

Programming Assignment 4: Paths in Graphs

Revision: June 2, 2016

Introduction

Welcome to your fourth programming assignment of the Algorithms on Graphs class! In this assignments we focus on shortest paths in weighted graphs.

Learning Outcomes

Upon completing this programming assignment you will be able to:

1. compute the minimum cost of a flight from one city to another one;
2. detect anomalies in currency exchange rates;
3. compute optimal way of exchanging the given currency into all other currencies.

Passing Criteria: 2 out of 3

Passing this programming assignment requires passing at least 2 out of 3 code problems from this assignment. In turn, passing a code problem 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.

2 Problem: Computing the Minimum Cost of a Flight

Problem Introduction

Now, you are interested in minimizing not the number of segments, but the total cost of a flight. For this you construct a weighted graph: the weight of an edge from one city to another one is the cost of the corresponding flight.

Problem Description

Task. Given an *directed* graph with positive edge weights and with n vertices and m edges as well as two vertices u and v, compute the weight of a shortest path between u and v (that is, the minimum total weight of a path from u to v).

Input Format. A graph is given in the standard format. The next line contains two vertices u and v.

Constraints. $1 \le n \le 10^3$, $0 \le m \le 10^5$, $u \ne v$, $1 \le u, v \le n$, edge weights are non-negative integers not exceeding 10^3.

Output Format. Output the minimum weight of a path from u to v, or -1 if there is no path.

Time Limits. `C`: 2 sec, `C++`: 2 sec, `Java`: 3 sec, `Python`: 10 sec. `C#`: 3 sec, `Haskell`: 4 sec, `JavaScript`: 10 sec, `Ruby`: 10 sec, `Scala`: 6 sec.

Memory Limit. 512Mb.

Sample 1.

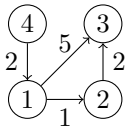
Input:

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

Output:

```
3
```

Explanation:



There is a unique shortest path from vertex 1 to vertex 3 in this graph ($1 \to 2 \to 3$), and it has weight 3.

Sample 2.

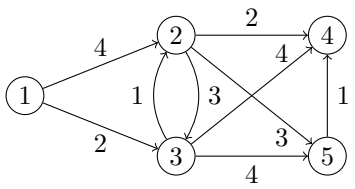
Input:

```
5 9
1 2 4
1 3 2
2 3 2
3 2 1
2 4 2
3 5 4
5 4 1
2 5 3
3 4 4
1 5
```

Output:

```
6
```

Explanation:



There are two paths from 1 to 5 of total weight 6: $1 \to 3 \to 5$ and $1 \to 3 \to 2 \to 5$.

Sample 3.

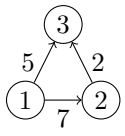
Input:

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

Output:

```
-1
```

Explanation:



There is no path from 3 to 2.

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: `dijkstra`

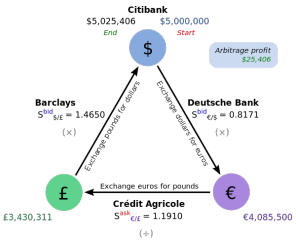
What To Do

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

3 Problem: Detecting Anomalies in Currency Exchange Rates

Problem Introduction

You are given a list of currencies $c_1, c_2, \ldots, c_n$ together with a list of exchange rates: r_{ij} is the number of units of currency c_j that one gets for one unit of c_i. You would like to check whether it is possible to start with one unit of some currency, perform a sequence of exchanges, and get more than one unit of the same currency. In other words, you would like to find currencies $c_{i_1}, c_{i_2}, \ldots, c_{i_k}$ such that $r_{i_1,i_2} \cdot r_{i_2,i_3} \cdot r_{i_{k-1},i_k}, r_{i_k,i_1} > 1$. For this, you construct the following graph: vertices are currencies $c_1, c_2, \ldots, c_n$, the weight of an edge from c_i to c_j is equal to $-\log r_{ij}$. There it suffices to check whether is a negative cycle in this graph. Indeed, assume that a cycle $c_i \to c_j \to c_k \to c_i$ has negative weight. This means that $-(\log c_{ij} + \log c_{jk} + \log c_{ki}) < 0$ and hence $\log c_{ij} + \log c_{jk} + \log c_{ki} > 0$. This, in turn, means that



$$r_{ij} r_{jk} r_{ki} = 2^{\log c_{ij}} 2^{\log c_{jk}} 2^{\log c_{ki}} = 2^{\log c_{ij} + \log c_{jk} + \log c_{ki}} > 1 .$$

Problem Description

Task. Given an directed graph with possibly negative edge weights and with n vertices and m edges, check whether it contains a cycle of negative weight.

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

Constraints. $1 \le n \le 10^3$, $0 \le m \le 10^4$, edge weights are integers of absolute value at most 10^3.

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

Time Limits. `C`: 2 sec, `C++`: 2 sec, `Java`: 3 sec, `Python`: 10 sec. `C#`: 3 sec, `Haskell`: 4 sec, `JavaScript`: 10 sec, `Ruby`: 10 sec, `Scala`: 6 sec.

Memory Limit. 512Mb.

Sample 1.

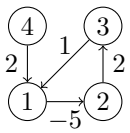
Input:

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

Output:

```
1
```

Explanation:



The weight of the cycle $1 \to 2 \to 3$ is equal to -2, that is, negative.

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: `negative_cycle`

What To Do

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

4 Advanced Problem: Exchanging Money Optimally

(Recall that advanced problems are not covered in the video lectures and require additional ideas to be solved. We therefore strongly recommend you start solving these problems only when you are done with the basic problems.)

Problem Introduction

Now, you would like to compute an optimal way of exchanging the given currency c_i into all other currencies. For this, you find shortest paths from the vertex c_i to all the other vertices.

Problem Description

Task. Given an directed graph with possibly negative edge weights and with n vertices and m edges as well as its vertex s, compute the length of shortest paths from s to all other vertices of the graph.

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

Constraints. $1 \le n \le 10^3$, $0 \le m \le 10^4$, $1 \le s \le n$, edge weights are integers of absolute value at most 10^9.

Output Format. For all vertices i from 1 to n output the following on a separate line:

- "*", if there is no path from s to u;
- "-", if there is a path from s to u, but there is no shortest path from s to u (that is, the distance from s to u is $-\infty$);
- the length of a shortest path otherwise.

Time Limits. `C`: 2 sec, `C++`: 2 sec, `Java`: 3 sec, `Python`: 10 sec. `C#`: 3 sec, `Haskell`: 4 sec, `JavaScript`: 10 sec, `Ruby`: 10 sec, `Scala`: 6 sec.

Memory Limit. 512Mb.

Sample 1.

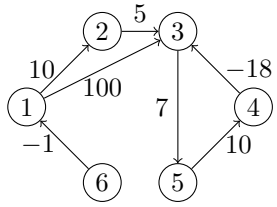
Input:

```
6 7
1 2 10
2 3 5
1 3 100
3 5 7
5 4 10
4 3 -18
6 1 -1
1
```

Output:

```
0
10
-
-
-
*
```

Explanation:



The first line of the output states that the distance from 1 to 1 is equal to 0. The second one shows that the distance from 1 to 2 is 10 (the corresponding path is $1 \to 2$). The next three lines indicate that the distance from 1 to vertices 3, 4, and 5 is equal to $-\infty$: indeed, one first reaches the vertex 3 through edges $1 \to 2 \to 3$ and then makes the length of a path arbitrary small by making sufficiently many walks through the cycle $3 \to 5 \to 4$ of negative weight. The last line of the output shows that there is no path from 1 to 6 in this graph.

Sample 2.

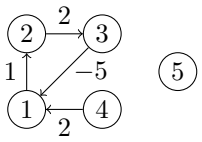
Input:

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

Output:

```
-
-
-
0
*
```

Explanation:



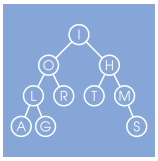
In this case, the distance from 4 to vertices 1, 2, and 3 is $-\infty$ since there is a negative cycle $1 \to 2 \to 3$ that is reachable from 4. The distance from 4 to 4 is zero. There is no path from 4 to 5.

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: `shortest_paths`

What To Do

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



Module: Minimum Spanning Trees (Week 5 out of 5)
Course: Algorithms on Graphs (Course 3 out of 6)
Specialization: Data Structures and Algorithms

Programming Assignment 5: Minimum Spanning Trees

Revision: July 4, 2016

Introduction

Welcome to the third (and the last one) programming assignment of the Algorithms on Graphs class! In this programming assignment you will be practicing implementing algorithms computing minimum spanning trees.

Learning Outcomes

Upon completing this programming assignment you will be able to:

1. connect the given cities by roads of minimum total length such that there is a path between any two cities;
2. compute an optimal clustering of the given set of objects.

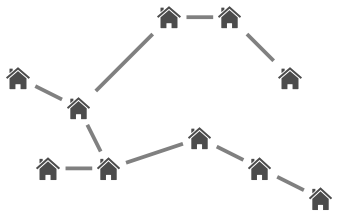
Passing Criteria: 1 out of 2

Passing this programming assignment requires passing at least 1 out of 2 code problems from this assignment. In turn, passing a code problem 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.

2 Problem: Building Roads to Connect Cities

Problem Introduction

In this problem, the goal is to build roads between some pairs of the given cities such that there is a path between any two cities and the total length of the roads is minimized.



Problem Description

Task. Given n points on a plane, connect them with segments of minimum total length such that there is a path between any two points. Recall that the length of a segment with endpoints (x_1, y_1) and (x_2, y_2) is equal to $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.

Input Format. The first line contains the number n of points. Each of the following n lines defines a point (x_i, y_i).

Constraints. $1 \leq n \leq 200$; $-10^3 \leq x_i, y_i \leq 10^3$ are integers. All points are pairwise different, no three points lie on the same line.

Output Format. Output the minimum total length of segments. The absolute value of the difference between the answer of your program and the optimal value should be at most 10^{-6}. To ensure this, output your answer with at least seven digits after the decimal point (otherwise your answer, while being computed correctly, can turn out to be wrong because of rounding issues).

Time Limits.

language	C	C++	Java	Python	C#	Haskell	JavaScript	Ruby	Scala
time in seconds	2	2	3	10	3	4	10	10	6

Memory Limit. 512Mb.

Sample 1.

Input:

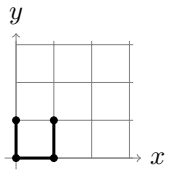
```
4
0 0
0 1
1 0
1 1
```

Output:

```
3.000000000
```

Explanation:

An optimal way to connect these four points is shown below. Note that there exists other ways of connecting these points by segments of total weight 3.



Sample 2.

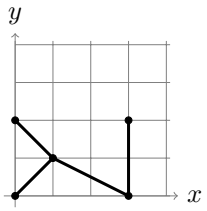
Input:

```
5
0 0
0 2
1 1
3 0
3 2
```

Output:

```
7.064495102
```

Explanation:
An optimal way to connect these four points is shown below.



The total length here is equal to $2\sqrt{2} + \sqrt{5} + 2$.

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.

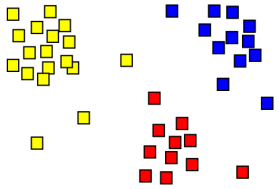
What To Do

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

3 Problem: Clustering

Problem Introduction

Clustering is a fundamental problem in data mining. The goal is to partition a given set of objects into subsets (or clusters) in such a way that any two objects from the same subset are close (or similar) to each other, while any two objects from different subsets are far apart.



Problem Description

Task. Given n points on a plane and an integer k, compute the largest possible value of d such that the given points can be partitioned into k non-empty subsets in such a way that the distance between any two points from different subsets is at least d.

Input Format. The first line contains the number n of points. Each of the following n lines defines a point (x_i, y_i). The last line contains the number k of clusters.

Constraints. $2 \le k \le n \le 200$; $-10^3 \le x_i, y_i \le 10^3$ are integers. All points are pairwise different.

Output Format. Output the largest value of d. The absolute value of the difference between the answer of your program and the optimal value should be at most 10^{-6}. To ensure this, output your answer with at least seven digits after the decimal point (otherwise your answer, while being computed correctly, can turn out to be wrong because of rounding issues).

Time Limits.

language	`C`	`C++`	`Java`	`Python`	`C#`	`Haskell`	`JavaScript`	`Ruby`	`Scala`
time in seconds	2	2	3	10	3	4	10	10	6

Memory Limit. 512Mb.

Sample 1.

Input:

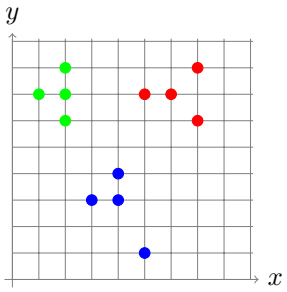
```
12
7 6
4 3
5 1
1 7
2 7
5 7
3 3
7 8
2 8
4 4
6 7
2 6
3
```

Output:

```
2.828427124746
```

Explanation:

The answer is $\sqrt{8}$. The corresponding partition of the set of points into three clusters is shown below.



Sample 2.

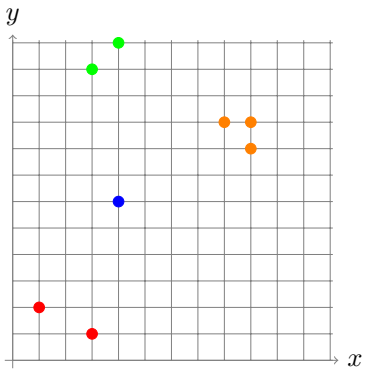
Input:

```
8
3 1
1 2
4 6
9 8
9 9
8 9
3 11
4 12
4
```

Output:

```
5.000000000
```

Explanation:
The answer is 5. The corresponding partition of the set of points into four clusters is shown below.

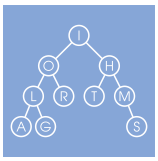


Starter Files

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.

What To Do

Think about ways of adopting the Kruskal's algorithm for solving this problem.



Module: Advanced Shortest Paths (Week 6 out of 6)
Course: Algorithms on Graphs (Course 3 out of 6)
Specialization: Data Structures and Algorithms

Programming Project: Advanced Shortest Paths

Revision: April 12, 2017

Introduction

Welcome to the Advanced Shortest Paths programming project! This project is organized as a series of coding problems in which you will implement the algorithms from the lectures, apply them to real-world road networks and social networks, and come up with your own ideas on how to speedup the algorithms even more. Due to time limitations, the test inputs in the grader range up to maps of New York and Colorado, but don't include the maps of USA and/or Europe. However, here you can find the graphs in the same format and compare the running time of your algorithms on those graphs on your own computer.

We also encourage you to create a forum thread and compete whose solution to the problem "Contraction Hierarchies Large" is the fastest in terms of both preprocessing time and query time, and also which solution is the most memory efficient. You can post the screenshots of the grader feedback in the comments, and the thread owner can keep updated the table with the top results in different nominations in the initial post itself.

The grader will show you the input data in the coding problems only if your solution fails on one of the first few tests (please review the questions 7.4 and 7.5 in the FAQ section for a more detailed explanation of this behavior of the grader). Also, in the problems with preprocessing phase (starting from the third problem, "Compute Distance with Preprocessing"), there are two separate inputs — the graph itself (for preprocessing) and the queries for computing the distance. You will be shown both inputs in case your solution fails on one of the first few tests. If your solution fails on further tests, you won't see the inputs.

Learning Outcomes

Upon completing this programming assignment you will be able to:

1. implement Bidirectional Dijkstra and use it to compute distances in social networks very quickly;
2. implement A* search algorithm and apply it to compute distances in road networks faster than the classic algorithms do;
3. implement Contraction Hierarchies algorithm and apply it to preprocess large road networks and then compute distances in them much faster;
4. implement even more heuristics speeding up Contraction Hierarchies and solve even bigger road networks;
5. use your Contraction Hierarchies implementation to go further and solve the classical logistics problem (called Travelling Salesman Problem): find the optimal path for a truck leaving the depot to visit a list of stores, deliver the goods to them and return to the depot.

1 Problem: Friend Suggestion

Problem Introduction

Social networks are live on the connections between people, so friend suggestions is one of the most important features of Facebook. One of the most important inputs of the algorithm for friend suggestion is most probably the current distance between you and the suggested person in the graph of friends connections. Your task is to implement efficient computation of this distance. The grader will test your algorithm against different real-world networks, such as a part of internet, a network of scientific citations or coauthorship, a social network of jazz musicians or even a social network of dolphins :) You need to compute the distance between two nodes in such network. We took some of the graphs from here to use in the grader, and you can play with more of them on your own computer.

Note that Python, Ruby and Javascript are too slow to solve the largest tests in time, so solutions in these languages won't be tested against some of the largest tests. Solutions in C++, Java, C#, Haskell and Scala will be tested against all the tests. Note that we only guarantee (as usual) that there exists a solution under the given time and memory constraints for C++, Java and Python3. For other languages, the solution may not exist.

Problem Description

Task. Compute the distance between several pairs of nodes in the network.

Input Format. The first line contains two integers n and m — the number of nodes and edges in the network, respectively. The nodes are numbered from 1 to n. Each of the following m lines contains three integers u, v and l describing a directed edge (u, v) of length l from the node number $u$to the node number v. (Note that some social networks are represented by directed graphs while some other correspond naturally to undirected graphs. For example, Twitter is a directed graph (with a directed edge (u, v) meaning that u follows v), while Facebook is an undirected graph (where an undirected edge $\{u, v\}$ means that u and v are friends). In this problem, we work with directed graphs only for a simple reason. It is easy to turn an undirected graph into a directed one: just replace each undirected edge $\{u, v\}$ with a pair of directed edges (u, v) and (v, u).)

The next line contains an integer q — the number of queries for computing the distance. Each of the following q lines contains two integers u and v — the numbers of the two nodes to compute the distance from u to v.

Constraints. $1 \le n \le 1\ 000\ 000$; $1 \le m \le 6\ 000\ 000$; $1 \le u, v \le n$; $1 \le l \le 1\ 000$; $1 \le q \le 1\ 000$. **For Python2, Python3, Ruby and Javascript,** $1 \le m \le 2\ 000\ 000$**.**

Output Format. For each query, output one integer on a separate line. If there is no path from u to v, output -1. Otherwise, output the distance from u to v.

Time Limits.

language	C	C++	Java	Python	C#	Haskell	JavaScript	Ruby	Scala
time (sec)	25	25	125	150	37.5	50	150	150	150

Memory Limit. 2048MB.

Sample 1.

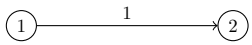
Input:

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

Output:

```
0
0
1
-1
```

Explanation:



The distance from a node to itself is always 0. The distance from 1 to 2 is 1, and there is no path from 2 to 1.

Sample 2.

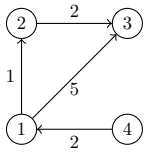
Input:

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

Output:

```
3
```

Explanation:



There is a direct edge from node 1 to node 3 of length 5, but there is a shorter path $1 \to 2 \to 3$ of length $1 + 2 = 3$.

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: `friend_suggestion`

What To Do

Implement the Bidirectional Dijkstra algorithm from the lectures.

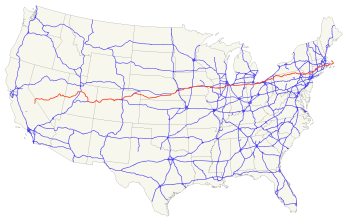
Need Help?

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

2 Problem: Compute Distance Faster Using Coordinates

Problem Introduction

In this task you will be given a description of a real-world road network with not just edges and their lengths, but also with the coordinates of the nodes. Your task is still to find the distance between some pairs of nodes, but you will need to use the additional information about coordinates to speedup your search.



Note that Python, Ruby and Javascript are too slow to solve the largest tests in time, so solutions in these languages won't be tested against some of the largest tests. Solutions in C++, Java, C#, Haskell and Scala will be tested against all the tests. Note that we only guarantee (as usual) that there exists a solution under the given time and memory constraints for C++, Java and Python3. For other languages, the solution may not exist.

Problem Description

Task. Compute the distance between several pairs of nodes in the network.

Input Format. The first line contains two integers n and m — the number of nodes and edges in the network, respectively. The nodes are numbered from 1 to n. Each of the following n lines contains the coordinates x and y of the corresponding node. Each of the following m lines contains three integers u, v and l describing a directed edge (u, v) of length l from the node number u to the node number v. It is guaranteed that $l \geq \sqrt{(x(u) - x(v))^2 + (y(u) - y(v))^2}$ where $(x(u), y(u))$ are the coordinates of u and $(x(v), y(v))$ are the coordinates of v. The next line contains an integer q — the number of queries for computing the distance. Each of the following q lines contains two integers u and v — the numbers of the two nodes to compute the distance from u to v.

Constraints. $1 \leq n \leq 110\ 000$; $1 \leq m \leq 250\ 000$; $-10^9 \leq x, y \leq 10^9$; $1 \leq u, v \leq n$; $0 \leq l \leq 100\ 000$; $1 \leq q \leq 10\ 000$. **For Python2, Python3, Ruby and Javascript,** $1 \leq n \leq 11\ 000$, $1 \leq m \leq 30\ 000$**.**

Output Format. For each query, output one integer. If there is no path from u to v, output -1. Otherwise, output the distance from u to v.

Time Limits.

language	C	C++	Java	Python	C#	Haskell	JavaScript	Ruby	Scala
time (sec)	50	50	100	50	75	100	50	50	100

Memory Limit. 2048MB.

Sample 1.

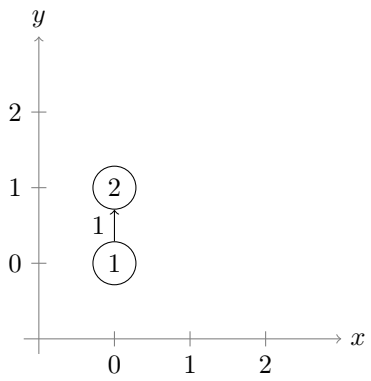
Input:

```
2 1
0 0
0 1
1 2 1
4
1 1
2 2
1 2
2 1
```

Output:

```
0
0
1
-1
```

Explanation:



The distance from a node to itself is always 0. The distance from 1 to 2 is 1, and there is no path from 2 to 1.

Sample 2.

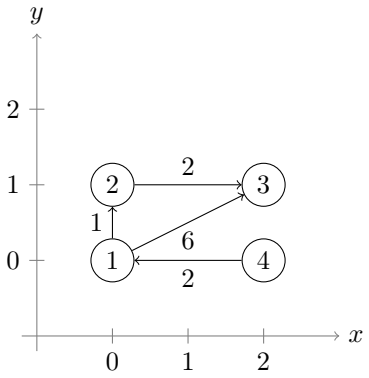
Input:

```
4 4
0 0
0 1
2 1
2 0
1 2 1
4 1 2
2 3 2
1 3 6
1
1 3
```

Output:

```
3
```

Explanation:



There is a direct edge from node 1 to node 3 of length 6, but there is a shorter path $1 \to 2 \to 3$ of length $1 + 2 = 3$.

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: `dist_with_coords`

What To Do

Implement the A* algorithm from the lectures.

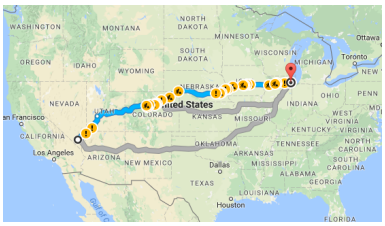
Need Help?

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

3 Problem: Compute Distance with Preprocessing

Problem Introduction

In this task you will be first given a graph of a real road network, and you can preprocess it as you wish under the preprocessing time limit. Then you will get a set of queries for computing distance, and you will need to answer all of them under the separate time limit for queries. You will have to respond to queries much faster than in the previous problem.



Note that Python, Ruby and Javascript are too slow to solve the largest tests in time, so solutions in these languages won't be tested against some of the largest tests. Solutions in C++, Java, C#, Haskell and Scala will be tested against all the tests. Note that we only guarantee (as usual) that there exists a solution under the given time and memory constraints for C++, Java and Python3. For other languages, the solution may not exist.

Problem Description

Task. Compute the distance between several pairs of nodes in the network.

Input Format. You will be given the input for this problem in two parts. The first part contains the description of a road network, the second part contains the queries. You have a separate time limit for preprocessing the graph. Under this time limit, you need to read the graph and preprocess it. After you've preprocessed the graph, you need to output the string "Ready" (without quotes) and flush the output buffer (the starter files for C++, Java and Python3 do that for you; if you use another language, you will have to find out how to do this). Only after you output the string "Ready" you will be given the queries. You will have a time limit for the querying part, and under this time limit you will need to input all the queries and output the results for each of the quires.

The first line of the road network description contains two integers n and m — the number of nodes and edges in the network, respectively. The nodes are numbered from 1 to n. Each of the following m lines contains three integers u, v and l describing a directed edge (u, v) of length l from the node number u to the node number v.

The first line of the queries description contains an integer q — the number of queries for computing the distance. Each of the following q lines contains two integers u and v — the numbers of the two nodes to compute the distance from u to v.

Constraints. $1 \le n \le 110\,000$; $1 \le m \le 250\,000$; $1 \le u, v \le n$; $1 \le l \le 200\,000$; $1 \le q \le 10\,000$. It is guaranteed that the correct distances are less than $1\,000\,000\,000$. **For Python2, Python3, Ruby and Javascript,** $1 \le n \le 11\,000$, $1 \le m \le 25\,000$, $1 \le q \le 1\,000$**.**

Output Format. After you've read the description of the road network and done your preprocessing, output one string "Ready" (without quotes) on a separate line and flush the output buffer. Then read the queries, and for each query, output one integer on a separate line. If there is no path from u to v, output -1. Otherwise, output the distance from u to v.

Time Limits.

language	C	C++	Java	Python	C#	Haskell	JavaScript	Ruby	Scala
preprocessing (sec)	10	10	45	50	15	20	50	50	90
query time (sec)	2	2	9	10	3	4	10	10	18

Memory Limit. 2048MB.

Sample 1.

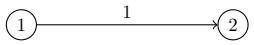
Input:

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

Output:

```
Ready
0
0
1
-1
```

Explanation:



The distance from a node to itself is always 0. The distance from 1 to 2 is 1, and there is no path from 2 to 1.

Sample 2.

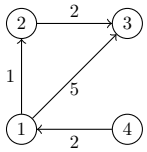
Input:

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

Output:

```
Ready
3
```

Explanation:



There is a direct edge from node 1 to node 3 of length 5, but there is a shorter path $1 \to 2 \to 3$ of length $1 + 2 = 3$.

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: `dist_preprocess_small`

What To Do

Implement the Contraction Hierarchies algorithm from the lectures. It is not necessary to implement all the heuristics to pass, only some of them.

Need Help?

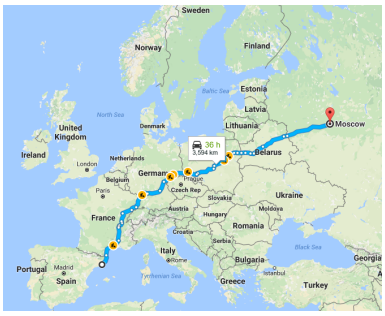
Ask a question or see the questions asked by other learners at this forum thread.

4 Advanced Problem: Compute Distance with Preprocessing on Larger Road Networks

We strongly recommend you start solving advanced problems only when you are done with the basic problems (for some advanced problems, algorithms are not covered in the video lectures and require additional ideas to be solved; for some other advanced problems, algorithms are covered in the lectures, but implementing them is a more challenging task than for other problems).

Problem Introduction

This problem is the same as the previous one, but you will have to solve larger road networks under the same time limits.



Note that Python, Ruby and Javascript are too slow to solve the largest tests in time, so solutions in these languages won't be tested against some of the largest tests. Solutions in C++, Java, C#, Haskell and Scala will be tested against all the tests. Note that we only guarantee (as usual) that there exists a solution under the given time and memory constraints for C++, Java and Python3. For other languages, the solution may not exist.

Problem Description

Task. Compute the distance between several pairs of nodes in the network.

Input Format. See the input format for the previous problem.

Constraints. $1 \le n \le 500\,000$; $1 \le m \le 1\,100\,000$; $1 \le u, v \le n$; $1 \le l \le 200\,000$; $1 \le q \le 10\,000$. It is guaranteed that the correct distances are less than $1\,000\,000\,000$. **For Python2, Python3, Ruby and Javascript,** $1 \le n \le 11\,000$, $1 \le m \le 25\,000$, $1 \le q \le 1\,000$.

Output Format. See the output format for the previous problem.

Time Limits.

language	`C`	`C++`	`Java`	`Python`	`C#`	`Haskell`	`JavaScript`	`Ruby`	`Scala`
preprocessing (sec)	55	55	220	55	82.5	110	55	55	220
query time (sec)	5	5	20	5	7.5	10	5	5	20

Memory Limit. 2048MB.

See the examples in the previous problem.

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: `dist_preprocess_large`

What To Do

Implement the Contraction Hierarchies algorithm from the lectures. It will be necessary to implement more of the heuristics and tricks than for the previous problem. You can also come up with your own ideas and techniques for speeding up the solution. We encourage you to compete on the forums, whose solution is the fastest (both in terms of preprocessing time and query time) and uses less memory!

Need Help?

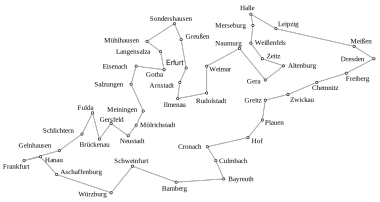
Ask a question or see the questions asked by other learners at this forum thread.

5 Advanced Problem: Travelling Salesman Problem

We strongly recommend you start solving advanced problems only when you are done with the basic problems (for some advanced problems, algorithms are not covered in the video lectures and require additional ideas to be solved; for some other advanced problems, algorithms are covered in the lectures, but implementing them is a more challenging task than for other problems).

Problem Introduction

In this task you will solve the classical logistics problem called Travelling Salesman Problem: you are given the location of a depot and the location of a list of stores on a road network, and you need to find the shortest path for a truck to start in the depot, visit each of the stores to deliver the goods there, and return back to the depot.



Note that Python, Ruby and Javascript are too slow to solve the largest tests in time, so solutions in these languages won't be tested against some of the largest tests. Solutions in C++, Java, C#, Haskell and Scala will be tested against all the tests. Note that we only guarantee (as usual) that there exists a solution under the given time and memory constraints for C++, Java and Python3. For other languages, the solution may not exist.

Problem Description

Task. Compute the length of the shortest path starting in the depot, visiting each store at least once and returning to the depot.

Input Format. You will be given the input for this problem in two parts. The first part contains the description of a road network, the second part contains the queries. You have a separate time limit for preprocessing the graph. Under this time limit, you need to read the graph and preprocess it. After you've preprocessed the graph, you need to output the string "Ready" (without quotes) and flush the output buffer (the starter files for C++, Java and Python3 do that for you; if you use another language, you will have to find out how to do this). Only after you output the string "Ready" you will be given the queries. You will have a time limit for the querying part, and under this time limit you will need to input all the queries and output the results for each of the quires.

The first line of the road network description contains two integers n and m — the number of nodes and edges in the network, respectively. The nodes are numbered from 1 to n. Each of the following m lines contains three integers u, v and l describing a directed edge (u, v) of length l from the node number u to the node number v.

The first line of the queries description contains an integer q — the number of queries for computing the distance. Each of the following q lines starts with the integer k — the number of points the truck must visit, including all the stores and the depot. There are k more integers on the same line. The first of them is the number of the node corresponding to the depot location. The next $k-1$ integers are the numbers of the nodes corresponding to the store locations.

Constraints. $1 \le n \le 110\,000$; $1 \le m \le 250\,000$; $1 \le u, v \le n$; $1 \le l \le 100\,000$; $1 \le q \le 100$; $1 \le k \le 20$. It is guaranteed that all the answers are less than $1\,000\,000\,000$. **For Python2, Python3, Ruby and Javascript,** $1 \le n \le 11\,000$**,** $1 \le m \le 25\,000$**,** $1 \le k \le 20$**.**

Output Format. After you've read the description of the road network and done your preprocessing, output one string "Ready" (without quotes) on a separate line and flush the output buffer. Then read the queries, and for each query, output one integer on a separate line. If there is no path starting in

depot, visiting each store at least once and returning to the depot, output -1. Otherwise, output the length of the shortest path starting at a depot, visiting each store at least once and returning to the depot.

Time Limits.

language	C	C++	Java	Python	C#	Haskell	JavaScript	Ruby	Scala
preprocessing (sec)	20	20	50	50	30	40	50	50	50
query time (sec)	16	16	40	40	24	32	40	40	40

Memory Limit. 2048MB.

Sample 1.

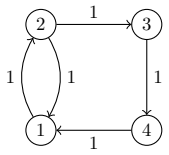
Input:

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

Output:

```
Ready
2
4
4
```

Explanation:



For the first query, we need to start in the node 1, visit node 2 and return to node 1. The shortest path for that is to get directly from 1 to 2 and then directly back from 2 to 1. The length is 2.

For the second query, we need to start in the node 1, visit node 3 and return to node 1. A shortest path for that is to get from 1 to 2, then from 2 to 3, then return from 3 to 2, then return from 2 to 1. Another shortest path would be to go from 1 to 2, then from 2 to 3, then from 3 to 4, then from 4 to 1. The length is 4.

For the third query, we need to start in the node 1, visit all the other nodes and return to node 1. The shortest path for that is to go from 1 to 2, then from 2 to 3, then from 3 to 4, then from 4 to 1. The length is 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: `travelling_salesman_problem`

What To Do

First you need to compute the pairwise distances between all the stores and between the depot and all the stores, to and from. To do that quickly, you will need to use your solution to the previous problem (preprocess the graph using Contraction Hierarchies preprocessing, then find the distance between each pair of interesting locations using Contraction Hierarchies querying). After that, you have the following problem: given a graph with $k \leq 20$ nodes and weighted edges, find the shortest path starting in some node, visiting all the nodes at least once and returning to the initial node. This is the classical Travelling Salesman Problem which can be solved for small values of k using dynamic programming. You can try to come up with this dynamic programming yourself, or watch this lecture about it in the next course of our Specialization, Advanced Algorithms and Complexity.

Need Help?

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