

Module: [Flows in Networks \(Week 1 out of 4\)](#)
Course: [Advanced Algorithms and Complexity \(Course 5 out of 6\)](#)
Specialization: [Data Structures and Algorithms](#)

Programming Assignment 1: Flows in Networks

Revision: August 25, 2016

Introduction

Welcome to your first programming assignment of the [Advanced Algorithms and Complexity class](#)! In this programming assignment, you will be practicing implementing and applying algorithms for finding maximum flows in networks.

In this programming assignment, the grader will show you the input 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. Apply Ford-Fulkerson and/or Edmonds-Karp algorithm to solve various computational problems efficiently. This will typically require you to invent a way to reduce the problem to a maximum flow or maximum matching problem and then use an efficient algorithm to solve it.
2. Design and implement efficient algorithms for the following computational problems:
 - (a) evacuating people from the city as fast as possible;
 - (b) assigning airline crews to the aircrafts efficiently;
 - (c) visualizing stock price data compactly.

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.

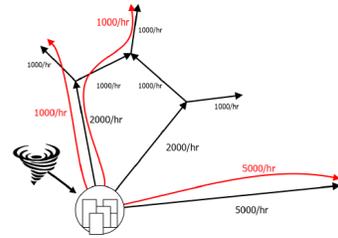
Contents

1	Problem: Evacuating People	3
2	Problem: Assigning Airline Crews to Flights	6
3	Advanced Problem: Stock Charts	8
4	General Instructions and Recommendations on Solving Algorithmic Problems	11
4.1	Reading the Problem Statement	11
4.2	Designing an Algorithm	11
4.3	Implementing Your Algorithm	11
4.4	Compiling Your Program	11
4.5	Testing Your Program	13
4.6	Submitting Your Program to the Grading System	13
4.7	Debugging and Stress Testing Your Program	13
5	Frequently Asked Questions	14
5.1	I submit the program, but nothing happens. Why?	14
5.2	I submit the solution only for one problem, but all the problems in the assignment are graded. Why?	14
5.3	What are the possible grading outcomes, and how to read them?	14
5.4	How to understand why my program fails and to fix it?	15
5.5	Why do you hide the test on which my program fails?	15
5.6	My solution does not pass the tests? May I post it in the forum and ask for a help?	16
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.	16

1 Problem: Evacuating People

Problem Introduction

In this problem, you will apply an algorithm for finding maximum flow in a network to determine how fast people can be evacuated from the given city.



Problem Description

Task. A tornado is approaching the city, and we need to evacuate the people quickly. There are several roads outgoing from the city to the nearest cities and other roads going further. The goal is to evacuate everybody from the city to the capital, as it is the only other city which is able to accommodate that many newcomers. We need to evacuate everybody as fast as possible, and your task is to find out what is the maximum number of people that can be evacuated each hour given the capacities of all the roads.

Input Format. The first line of the input contains integers n and m — the number of cities and the number of roads respectively. Each of the next m lines contains three integers u , v and c describing a particular road — start of the road, end of the road and the number of people that can be transported through this road in one hour. u and v are the 1-based indices of the corresponding cities.

The city from which people are evacuating is the city number 1, and the capital city is the city number n .

Note that all the roads are given as one-directional, that is, you cannot transport people from v to u using a road that connects u to v . Also note that there can be several roads connecting the same city u to the same city v , there can be both roads from u to v and from v to u , or there can be only roads in one direction, or there can be no roads between a pair of cities. Also note that there can be roads going from a city u to itself in the input.

When evacuating people, they cannot stop in the middle of the road or in any city other than the capital. The number of people per hour entering any city other than the evacuating city 1 and the capital city n must be equal to the number of people per hour exiting from this city. People who left a city u through some road (u, v, c) are assumed to come immediately after that to the city v . We are interested in the maximum possible number of people per hour leaving the city 1 under the above restrictions.

Constraints. $1 \leq n \leq 100$; $0 \leq m \leq 10\,000$; $1 \leq u, v \leq n$; $1 \leq c \leq 10\,000$. It is guaranteed that $m \cdot \text{EvacuatePerHour} \leq 2 \cdot 10^8$, where *EvacuatePerHour* is the maximum number of people that can be evacuated from the city each hour — the number which you need to output.

Output Format. Output a single integer — the maximum number of people that can be evacuated from the city number 1 each hour.

Time Limits.

language	C	C++	Java	Python	C#	Haskell	JavaScript	Ruby	Scala
time in seconds	1	1	5	45	1.5	2	45	45	10

Memory Limit. 512Mb.

Sample 1.

Input:

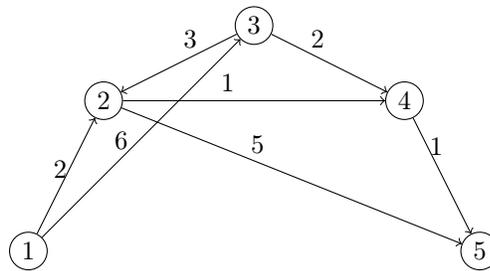
```
5 7
1 2 2
2 5 5
1 3 6
3 4 2
4 5 1
3 2 3
2 4 1
```

Output:

```
6
```

Explanation:

In this sample, the road graph with capacities looks like this:



We can evacuate 2 people through the route 1–2–5, additional 3 people through the route 1–3–2–5 and 1 more person through the route 1–3–4–5 – for a total of 6 people. It is impossible to evacuate more people each hour, as the total capacity of all roads incoming to the capital city 5 is 6 people per hour.

Sample 2.

Input:

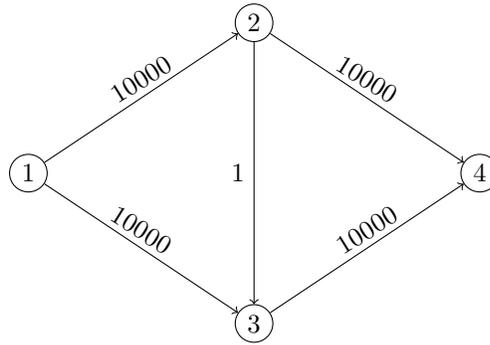
```
4 5
1 2 10000
1 3 10000
2 3 1
3 4 10000
2 4 10000
```

Output:

```
20000
```

Explanation:

In this sample, the road graph with capacities looks like this:



We can evacuate 10000 people through the route 1 – 2 – 4 and additional 10000 people through the route 1 – 3 – 4 totalling in 20000 people per hour. It is impossible to evacuate more people each hour, as the total capacity of the roads outgoing from the city number 1 is 20000 people per hour.

Pay attention to this example if you think of using a simple Ford–Fulkerson algorithm. Note how it works on such graph, and why it may be a bad idea to use this algorithm on big networks with large capacities.

Starter Files

The starter solutions for this problem read the data from the input, build a graph data structure optimized for finding maximum flow in the graph, pass it to a blank procedure for finding the maximum flow and output the result. You need to implement this procedure if you are using C++, Java, or Python3. For other programming languages, you need to implement a solution from scratch. Filename: `evacuation`

What To Do

Implement an algorithm for finding maximum flow described in the lectures, but be careful with the choice of the algorithm, see the comments for the second example from the problem statement.

Need Help?

Ask a question or see the questions asked by other learners at [this forum thread](#).

2 Problem: Assigning Airline Crews to Flights

Problem Introduction

In this problem, you will apply an algorithm for finding maximum matching in a bipartite graph to assign airline crews to flights in the most efficient way.



Problem Description

Task. The airline offers a bunch of flights and has a set of crews that can work on those flights. However, the flights are starting in different cities and at different times, so only some of the crews are able to work on a particular flight. You are given the pairs of flights and associated crews that can work on those flights. You need to assign crews to as many flights as possible and output all the assignments.

Input Format. The first line of the input contains integers n and m — the number of flights and the number of crews respectively. Each of the next n lines contains m binary integers (0 or 1). If the j -th integer in the i -th line is 1, then the crew number j can work on the flight number i , and if it is 0, then it cannot.

Constraints. $1 \leq n, m \leq 100$.

Output Format. Output n integers — for each flight, output the 1-based index of the crew assigned to this flight. If no crew is assigned to a flight, output -1 as the index of the crew corresponding to it. All the positive indices in the output must be between 1 and m , and they must be pairwise different, but you can output any number of -1 's. If there are several assignments with the maximum possible number of flights having a crew assigned, output any of them.

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:

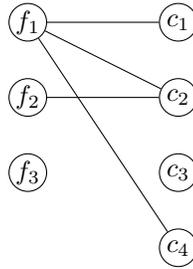
```
3 4
1 1 0 1
0 1 0 0
0 0 0 0
```

Output:

```
1 2 -1
```

Explanation:

In this sample, the bipartite graph of flights (on the left) and crews (on the right) looks like this:



We can assign first crew to the first flight and second crew to the second flight, and no crews can work on the third flight, so this is an optimal assignment.

Sample 2.

Input:

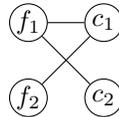
```
2 2
1 1
1 0
```

Output:

```
2 1
```

Explanation:

In this sample, the bipartite graph of flights (on the left) and crews (on the right) looks like this:



If we assign the first crew to the first flight, we won't be able to assign any crew to the second flight. It is optimal to assign the second crew to the first flight and the first crew to the second flight, because this way we have a crew assigned to each flight.

Starter Files

The starter solutions for this problem read the data from the input, pass it to a blank procedure that implements an incorrect greedy algorithm for finding the maximum matching, and output the result. You need to change this procedure to implement a correct algorithm for finding the maximum matching if you are using C++, Java, or Python3. For other programming languages, you need to implement a solution from scratch. Filename: `airline_crews`

What To Do

Implement an algorithm for finding the maximum matching in the bipartite graph that was described in the lectures.

Need Help?

Ask a question or see the questions asked by other learners at [this forum thread](#).

3 Advanced Problem: Stock Charts

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

We would like to thank the organizers of [Google Code Jam](#) — the annual worldwide programming competition held by Google — for allowing us to use this problem from one of the past rounds. Yes, that’s right: you have a problem from a world championship in programming in your first homework, but hey, this is the [fifth course](#) in the [Data Structures and Algorithms Specialization](#), and it has the words “Advanced Algorithms” in its name for a reason :) You’ve made it thus far — you can do this.



In this problem you will need to guess how to apply the algorithms from this module to find the most compact way of visualizing stock price data using charts.

Problem Description

Task. You’re in the middle of writing your newspaper’s end-of-year economics summary, and you’ve decided that you want to show a number of charts to demonstrate how different stocks have performed over the course of the last year. You’ve already decided that you want to show the price of n different stocks, all at the same k points of the year.

A simple chart of one stock’s price would draw lines between the points $(0, price_0), (1, price_1), \dots, (k - 1, price_{k-1})$, where $price_i$ is the price of the stock at the i -th point in time.

In order to save space, you have invented the concept of an overlaid chart. An overlaid chart is the combination of one or more simple charts, and shows the prices of multiple stocks (simply drawing a line for each one). In order to avoid confusion between the stocks shown in a chart, **the lines in an overlaid chart may not cross or touch.**

Given a list of n stocks’ prices at each of k time points, determine the minimum number of overlaid charts you need to show all of the stocks’ prices.

Input Format. The first line of the input contains two integers n and k — the number of stocks and the number of points in the year which are common for all of them. Each of the next n lines contains k integers. The i -th of those n lines contains the prices of the i -th stock at the corresponding k points in the year.

Constraints. $1 \leq n \leq 100; 1 \leq k \leq 25$. All the stock prices are between 0 and 1 000 000.

Output Format. Output a single integer — the minimum number of overlaid charts to visualize all the stock price data you have.

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:

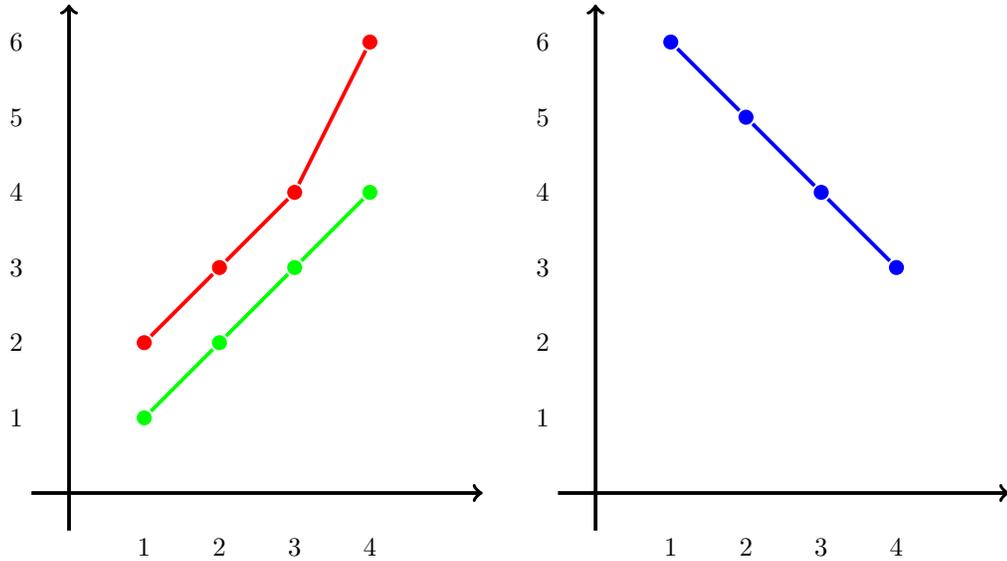
```
3 4
1 2 3 4
2 3 4 6
6 5 4 3
```

Output:

```
2
```

Explanation:

This data can be put into two following overlaid charts:



However, we cannot put all the data in one overlaid chart, as the lines corresponding to the third stock would touch the lines corresponding to the second stock, because they have the same price value at the third point.

Sample 2.

Input:

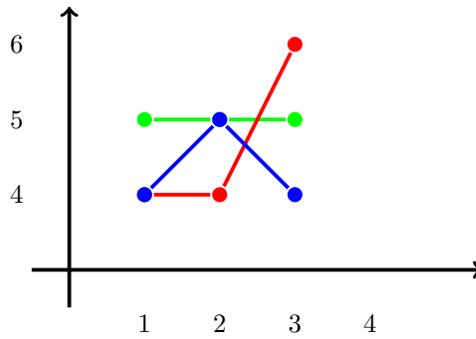
```
3 3
5 5 5
4 4 6
4 5 4
```

Output:

```
3
```

Explanation:

Each stock can be put on its own overlaid stock chart, of course. But no two stocks can be put on the same overlaid stock chart: first and second would intersect between points 2 and 3, first and third would touch in point 2, second and third would touch in point 1.



Starter Files

The starter solutions for this problem read the data from the input, pass it to a procedure and output the result. This procedure tries to pack the stock price data into the minimum possible number of overlaid charts using a greedy algorithm, but it sometimes fails. You need to implement another algorithm in this procedure if you are using C++, Java, or Python3. For other programming languages, you need to implement a solution from scratch. Filename: `stock_charts`

What To Do

Determine what are the conditions under which two stocks can be put on the same chart. Then think when more than 2 stocks can be put on the same chart.

Try to reduce the problem to the maximum matching in a bipartite graph problem, and to the first trick on the way is to create a bipartite graph of stocks with **two nodes for each stock**.

Need Help?

Ask a question or see the questions asked by other learners at [this forum thread](#).

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 $1 \leq n \leq 10^5$, 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 $0, 0, \dots, 0$ or a sequence $10^6, 10^6, \dots, 10^6$. 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.

References