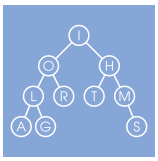


Module: Coping with NP-completeness (Week 4 out of 5)
Course: Advanced Algorithms and Complexity (Course 5 out of 6)
Specialization: Data Structures and Algorithms

Programming Assignment 4: Coping with NP-completeness

Revision: July 8, 2019

Introduction

Welcome to your last programming assignment of the Advanced Algorithms and Complexity class! In this programming assignment, you will be practicing solving NP-complete problems. Of course, currently we don't know efficient algorithms for solving these problems, but in some special cases there are efficient algorithms, and for some problems there are algorithms exponentially more efficient than the brute-force approach, although they still have exponential running time themselves.

Learning Outcomes

Upon completing this programming assignment you will be able to:

1. design a part of integrated circuit;
2. plan a totally cool party;
3. find the optimal route for a school bus;
4. reschedule the exams.

Passing Criteria: 2 out of 4

Passing this programming assignment requires passing at least 2 out of 4 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: Integrated Circuit Design

Problem Introduction

In this problem, you will determine how to connect the modules of an integrated circuit with wires so that all the wires can be routed on the same layer of the circuit.

Problem Description

Task. VLSI or Very Large-Scale Integration is a process of creating an integrated circuit by combining thousands of transistors on a single chip. You want to design a single layer of an integrated circuit. You know exactly what modules will be used in this layer, and which of them should be connected by wires. The wires will be all on the same layer, but they cannot intersect with each other. Also, each wire can only be bent once, in one of two directions — to the left or to the right. If you connect two modules with a wire, selecting the direction of bending uniquely defines the position of the wire. Of course, some positions of some pairs of wires lead to intersection of the wires, which is forbidden. You need to determine a position for each wire in such a way that no wires intersect.

This problem can be reduced to 2-SAT problem — a special case of the SAT problem in which each clause contains exactly 2 variables. For each wire i, denote by x_i a binary variable which takes value 1 if the wire is bent to the right and 0 if the wire is bent to the left. For each i, x_i must be either 0 or 1. Also, some pairs of wires intersect in some positions. For example, it could be so that if wire 1 is bent to the left and wire 2 is bent to the right, then they intersect. We want to write down a formula which is satisfied only if no wires intersect. In this case, we will add the clause $(x_1\ OR\ \overline{x_2})$ to the formula which ensures that either x_1 (the first wire is bent to the right) is true or $\overline{x_2}$ (the second wire is bent to the left) is true, and so the particular crossing when wire 1 is bent to the left AND wire 2 is bent to the right doesn't happen whenever the formula is satisfied. We will add such a clause for each pair of wires and each pair of their positions if they intersect when put in those positions. Of course, if some pair of wires intersects in any pair of possible positions, we won't be able to design a circuit. Your task is to determine whether it is possible, and if yes, determine the direction of bending for each of the wires.

Input Format. The input represents a 2-CNF formula. The first line contains two integers V and C — the number of variables and the number of clauses respectively. Each of the next C lines contains two non-zero integers i and j representing a clause in the CNF form. If $i > 0$, it represents x_i, otherwise if $i < 0$, it represents $\overline{x_{-i}}$, and the same goes for j. For example, a line "`2 3`" represents a clause $(x_2\ OR\ x_3)$, line "`1 -4`" represents $(x_1\ OR\ \overline{x_4})$, line "`-1 -3`" represents $(\overline{x_1}\ OR\ \overline{x_3})$, and line "`0 2`" cannot happen, because i and j must be non-zero.

Constraints. $1 \le V, C \le 1\,000\,000$; $-V \le i, j \le V$; $i, j \ne 0$.

Output Format. If the 2-CNF formula in the input is unsatisfiable, output just the word "UNSATISFIABLE" (without quotes, using capital letters). If the 2-CNF formula in the input is satisfiable, output the word "SATISFIABLE" (without quotes, using capital letters) on the first line and the corresponding assignment of variables on the second line. For each x_i in order from x_1 to x_V, output i if $x_i = 1$ or $-i$ if $x_i = 0$. For example, if a formula is satisfied by assignment $x_1 = 0, x_2 = 1, x_3 = 0$,

output "`-1 2 -3`" on the second line (without quotes). If there are several possible assignments satisfying the input formula, output any one of them.

Time Limits.

language	`C`	`C++`	`Java`	`Python`	`C#`	`Haskell`	`JavaScript`	`Ruby`	`Scala`
time (sec)	1	1	18	16	1.5	2	16	16	36

Memory Limit. 512MB.

Sample 1.

Input:

```
3 3
1 -3
-1 2
-2 -3
```

Output:

```
SATISFIABLE
1 2 -3
```

The input formula is $(x_1\ OR\ \overline{x_3})\ AND\ (\overline{x_1}\ OR\ x_2)\ AND\ (\overline{x_2}\ OR\ \overline{x_3})$, and the assignment $x_1 = 1, x_2 = 1, x_3 = 0$ satisfies it.

Sample 2.

Input:

```
1 2
1 1
-1 -1
```

Output:

```
UNSATISFIABLE
```

Explanation:
The input formula is $(x_1\ OR\ x_1)\ AND\ (\overline{x_1}\ OR\ \overline{x_1})$, and it is unsatisfiable.

Starter Files

The starter solutions for this problem read the data from the input, pass it to a brute-force procedure that checks all possible variable assignments, and output the result. You need to implement a more efficient algorithm in this procedure if you're using `C++`, `Java`, or `Python3`. For other programming languages, you need to implement a solution from scratch. Filename: `circuit_design`

What To Do

You need to implement an algorithm described in the lectures.

Need Help?

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

2 Problem: Plan a Fun Party

Problem Introduction

In this problem, you will design and implement an efficient algorithm to plan invite the coolest people from your company to a party in such a way that everybody is relaxed, because the direct boss of any invited person is not invited.

Problem Description

Task. You're planning a company party. You'd like to invite the coolest people, and you've assigned each one of them a fun factor — the more the fun factor, the cooler is the person. You want to maximize the total fun factor (sum of the fun factors of all the invited people). However, you can't invite everyone, because if the direct boss of some invited person is also invited, it will be awkward. Find out what is the maximum possible total fun factor.

Input Format. The first line contains an integer n — the number of people in the company. The next line contains n numbers f_i — the fun factors of each of the n people in the company. Each of the next $n-1$ lines describes the subordination structure. Everyone but for the CEO of the company has exactly one direct boss. There are no cycles: nobody can be a boss of a boss of a ... of a boss of himself. So, the subordination structure is a regular tree. Each of the $n-1$ lines contains two integers u and v, and you know that either u is the boss of v or vice versa (you don't really need to know which one is the boss, but you can invite only one of them or none of them).

Constraints. $1 \le n \le 100\,000$; $1 \le f_i \le 1\,000$; $1 \le u, v \le n$; $u \ne v$.

Output Format. Output the maximum possible total fun factor of the party (the sum of fun factors of all the invited people).

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:

```
1
1000
```

Output:

```
1000
```

Explanation:
There is only one person in the company, the CEO, and the fun factor is 1000. We can just invite the CEO and get the total fun factor of 1000.

Sample 2.

Input:

```
2
1 2
1 2
```

Output:

```
2
```

Explanation:
There are two people, and one of them is the boss of another one. We can invite only one of them. If we invite the second one, the total fun factor is 2, and it is bigger than total fun factor of 1 that we get in case we invite the first one.

Sample 3.

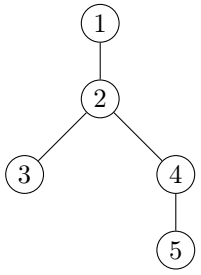
Input:

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

Output:

```
11
```

Explanation:
A possible subordination structure:



We can invite 1, 3 and 4 for a total fun factor of 11. If we invite 2, we cannot invite 1, 3 or 4, so the total fun factor will be at most 10, thus we don't invite 2 in the optimal solution. If we don't invite 4 also, we will get a fun factor of at most $1 + 3 + 5 = 9$, so we must invite 4. But then we can't invite 5, so we invite also 1 and 3 and get the total fun factor of 11.

Starter Files

The starter solutions for this problem read the data from the input, pass it to a template procedure that implements depth-first search but doesn't compute anything, and output the result. You need to augment the template procedure if you are using `C++`, `Java`, or `Python3`. For other programming languages, you need to implement a solution from scratch. Filename: `plan_party`

What To Do

You need to implement an algorithm described in the lectures.

Need Help?

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

3 Problem: School Bus

Problem Introduction

In this problem, you will determine the fastest route for a school bus to start from the depot, visit all the children's homes, get them to school and return back to depot.

Problem Description

Task. A school bus needs to start from the depot early in the morning, pick up all the children from their homes in some order, get them all to school and return to the depot. You know the time it takes to get from depot to each home, from each home to each other home, from each home to the school and from the school to the depot. You want to define the order in which to visit children's homes so as to minimize the total time spent on the route.

This is an instance of a classical NP-complete problem called Traveling Salesman Problem. Given a graph with weighted edges, you need to find the shortest cycle visiting each vertex exactly once. Vertices correspond to homes, the school and the depot. Edges weights correspond to the time to get from one vertex to another one. Some vertices may not be connected by an edge in the general case.

Input Format. The first line contains two integers n and m — the number of vertices and the number of edges in the graph. The vertices are numbered from 1 through n. Each of the next m lines contains three integers u, v and t representing an edge of the graph. This edge connects vertices u and v, and it takes time t to get from u to v. The edges are bidirectional: you can go both from u to v and from v to u in time t using this edge. No edge connects a vertex to itself. No two vertices are connected by more than one edge.

Constraints. $2 \le n \le 17$; $1 \le m \le \frac{n(n-1)}{2}$; $1 \le u, v \le n$; $u \ne v$; $1 \le t \le 10^9$.

Output Format. If it is possible to start in some vertex, visit each other vertex exactly once in some order going by edges of the graph and return to the starting vertex, output two lines. On the first line, output the minimum possible time to go through such circular route visiting all vertices exactly once (apart from the first vertex which is visited twice — in the beginning and in the end). On the second line, output the order in which you should visit the vertices to get the minimum possible time on the route. That is, output the numbers from 1 through n in the order corresponding to visiting the vertices. Don't output the starting vertex second time. However, account for the time to get from the last vertex back to the starting vertex. If there are several solutions, output any one of them. If there is no such circular route, output just -1 on a single line. Note that for $n = 2$ it is considered a correct circular route to go from one vertex to another by an edge and then return back by the same edge.

Time Limits.

language	`C`	`C++`	`Java`	`Python`	`C#`	`Haskell`	`JavaScript`	`Ruby`	`Scala`
time (sec)	1	1	1.5	45	1.5	2	45	45	3

Memory Limit. 512MB.

Sample 1.

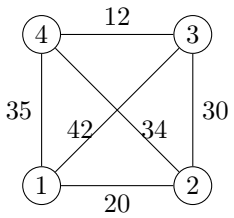
Input:

```
4 6
1 2 20
1 3 42
1 4 35
2 3 30
2 4 34
3 4 12
```

Output:

```
97
1 4 3 2
```

Explanation:



The suggested route starts in the vertex 1, goes to 4 in 35 minutes, from there to 3 in 12 minutes, from there to 2 in 30 minutes, from there back to 1 in 20 minutes, totalling in $35 + 12 + 30 + 20 = 97$ minutes. Check yourself that any other circular route visiting each vertex exactly once is longer.

Sample 2.

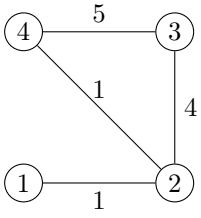
Input:

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

Output:

```
-1
```

Explanation:



There is no way to visit all the vertices exactly once, as there is only one edge from the vertex 1 (going to 2), so after leaving it you cannot return without visiting 2 twice.

Starter Files

The starter solutions for this problem read the data from the input, pass it to a brute-force procedure that tries each possible order of visit, and output the result. You need to change the brute-force procedure to implement some more efficient algorithm if you are using `C++`, `Java`, or `Python3`. For other programming languages, you need to implement a solution from scratch. Filename: `school_bus`

What To Do

You need to implement an algorithm described in the lectures.

Need Help?

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

4 Advanced Problem: Reschedule the Exams

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 problem, you will design and implement an efficient algorithm to reschedule the exams in such a way that every student can come to the exam she is assigned to, and no two friends will be passing the exam the same day.

Problem Description

Task. The new secretary at your Computer Science Department has prepared a schedule of exams for CS-101: each student was assigned to his own exam date. However, it's a disaster: not only some pairs of students known to be close friends may have been assigned the same date, but also NONE of the students can actually come to the exam at the day they were assigned (there was a misunderstanding between the secretary who asked to specify available dates and the students who understood they needed to select the date at which they cannot come). There are three different dates the professors are available for these exams, and these dates cannot be changed. The only thing that can be changed is the assignment of students to the dates of exams. You know for sure that each student can't come at the currently scheduled date, but also each student definitely can come at any of the two other possible dates. Also, you must make sure that no two known close friends are assigned to the same exam date. You need to determine whether it is possible or not, and if yes, suggest a specific assignment of the students to the dates.

This problem can be reduced to a graph problem called 3-recoloring. You are given a graph, and each vertex is colored in one of the 3 possible colors. You need to assign another color to each vertex in such a way that no two vertices connected by and edge are assigned the same color. Here, possible colors correspond to the possible exam dates, vertices correspond to students, colors of the vertices correspond to the assignment of students to the exam dates, and edges correspond to the pairs of close friends.

Input Format. The first line contains two integers n and m — the number of vertices and the number of edges of the graph. The vertices are numbered from 1 through n. The next line contains a string of length n consisting only of letters R, G and B representing the current color assignments. For each position i (1-based) in the string, if it is R, then the vertex i is colored red; if it's G, the vertex i is colored green; if it's B, the vertex i is colored blue. These are the current color assignments, and **each of them must be changed**. Each of the next m lines contains two integers u and v — vertices u and v are connected by an edge (it is possible that $u = v$).

Constraints. $1 \le n \le 1\,000$; $0 \le m \le 20\,000$; $1 \le u, v \le n$.

Output Format. If it is impossible to reassign the students to the dates of exams in such a way that no two friends are going to pass the exam the same day, and each student's assigned date has changed, output just one word "Impossible" (without quotes). Otherwise, output one string consisting of n characters R, G and B representing the new coloring of the vertices. Note that the color of each vertex must be

different from the initial color of this vertex. The vertices connected by an edge must have different colors.

Time Limits.

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

Memory Limit. 512MB.

Sample 1.

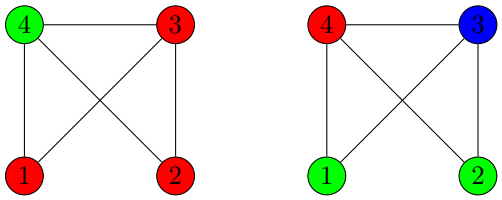
Input:

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

Output:

```
GGBR
```

Explanation:
The initial coloring and the new coloring:



Note that the vertices 1 and 2 are ok to be of the same color, as they are not connected by an edge.

Sample 2.

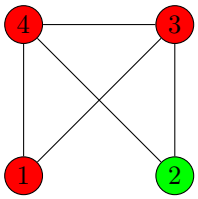
Input:

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

Output:

```
Impossible
```

Explanation:
The initial coloring:



It is impossible to recolor the vertices properly, because it means that none of the vertices 1, 3 and 4 can be red, but that leaves only two choices of colors to them — blue and green — but each pair of them is connected by an edge, so we need at least three different colors to color them properly.

Starter Files

The starter solutions for this problem read the data from the input, pass it to a procedure that just tries to set the colors of the vertices in a fixed way without looking at the edges or the current colors. You need to change the main procedure to solve the problem correctly if you are using `C++`, `Java`, or `Python3`. For other programming languages, you need to implement a solution from scratch. Filename: `reschedule_exams`

What To Do

You need to reduce this problem to another problem you already know how to solve.

Need Help?

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

5 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.

5.1 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.

5.2 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.

5.3 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.

5.4 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 -O2
```

- `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
```

5.5 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.

5.6 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 5.4) 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).

5.7 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 5.5. 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!

6 Frequently Asked Questions

6.1 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 5.3 and 5.4). 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.

6.2 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.

6.3 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.

6.4 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.

6.5 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.

6.6 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).

6.7 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