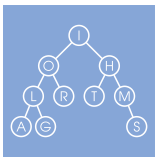

Module: Priority Queues and Disjoint Sets (Week 2 out of 4)
Course: Data Structures (Course 2 out of 6)
Specialization: Data Structures and Algorithms

Programming Assignment 2: Priority Queues and Disjoint Sets

Revision: August 26, 2016

Introduction

In this programming assignment, you will practice implementing priority queues and disjoint sets and using them to solve algorithmic problems. In some cases you will just implement an algorithm from the lectures, while in others you will need to invent an algorithm to solve the given problem using either a priority queue or a disjoint set union.

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. Apply priority queues and disjoint sets to solve the given algorithmic problems.
2. Convert an array into a heap.
3. Simulate a program which processes a list of jobs in parallel.
4. Simulate a sequence of merge operations with tables in a database.

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: Convert array into heap

Problem Introduction

In this problem you will convert an array of integers into a heap. This is the crucial step of the sorting algorithm called HeapSort. It has guaranteed worst-case running time of $O(n \log n)$ as opposed to QuickSort's average running time of $O(n \log n)$. QuickSort is usually used in practice, because typically it is faster, but HeapSort is used for external sort when you need to sort huge files that don't fit into memory of your computer.

Problem Description

Task. The first step of the HeapSort algorithm is to create a heap from the array you want to sort. By the way, did you know that algorithms based on Heaps are widely used for external sort, when you need to sort huge files that don't fit into memory of a computer?

Your task is to implement this first step and convert a given array of integers into a heap. You will do that by applying a certain number of swaps to the array. Swap is an operation which exchanges elements a_i and a_j of the array a for some i and j. You will need to convert the array into a heap using only $O(n)$ swaps, as was described in the lectures. Note that you will need to use a min-heap instead of a max-heap in this problem.

Input Format. The first line of the input contains single integer n. The next line contains n space-separated integers a_i.

Constraints. $1 \le n \le 100\,000$; $0 \le i, j \le n - 1$; $0 \le a_0, a_1, \ldots, a_{n-1} \le 10^9$. All a_i are distinct.

Output Format. The first line of the output should contain single integer m — the total number of swaps. m **must satisfy conditions** $0 \le m \le 4n$. The next m lines should contain the swap operations used to convert the array a into a heap. Each swap is described by a pair of integers i, j — the 0-based indices of the elements to be swapped. After applying all the swaps in the specified order the array must become a heap, that is, for each i where $0 \le i \le n - 1$ the following conditions must be true:

1. If $2i + 1 \le n - 1$, then $a_i < a_{2i+1}$.
2. If $2i + 2 \le n - 1$, then $a_i < a_{2i+2}$.

Note that all the elements of the input array are distinct. Note that any sequence of swaps that has length at most $4n$ and after which your initial array becomes a correct heap will be graded as correct.

Time Limits. `C`: 1 sec, `C++`: 1 sec, `Java`: 3 sec, `Python`: 3 sec. `C#`: 1.5 sec, `Haskell`: 2 sec, `JavaScript`: 3 sec, `Ruby`: 3 sec, `Scala`: 3 sec.

Memory Limit. 512Mb.

Sample 1.

Input:

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

Output:

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

Explanation:
After swapping elements 4 in position 1 and 1 in position 4 the array becomes 5 1 3 2 4.

After swapping elements 5 in position 0 and 1 in position 1 the array becomes 1 5 3 2 4.
After swapping elements 5 in position 1 and 2 in position 3 the array becomes 1 2 3 5 4, which is already a heap, because $a_0 = 1 < 2 = a_1, a_0 = 1 < 3 = a_2, a_1 = 2 < 5 = a_3, a_1 = 2 < 4 = a_4$.

Sample 2.

Input:

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

Output:

```
0
```

Explanation:
The input array is already a heap, because it is sorted in increasing order.

Starter Files

There are starter solutions only for C++, Java and Python3, and if you use other languages, you need to implement solution from scratch. Starter solutions read the array from the input, use a quadratic time algorithm to convert it to a heap and use $\Theta(n^2)$ swaps to do that, then write the output. You need to replace the $\Theta(n^2)$ implementation with an $O(n)$ implementation using no more than $4n$ swaps to convert the array into heap.

What to Do

Change the `BuildHeap` algorithm from the lecture to account for min-heap instead of max-heap and for 0-based indexing.

Need Help?

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

2 Problem: Parallel processing

Problem Introduction

In this problem you will simulate a program that processes a list of jobs in parallel. Operating systems such as Linux, MacOS or Windows all have special programs in them called schedulers which do exactly this with the programs on your computer.

Problem Description

Task. You have a program which is parallelized and uses n independent threads to process the given list of m jobs. Threads take jobs in the order they are given in the input. If there is a free thread, it immediately takes the next job from the list. If a thread has started processing a job, it doesn't interrupt or stop until it finishes processing the job. If several threads try to take jobs from the list simultaneously, the thread with smaller index takes the job. For each job you know exactly how long will it take any thread to process this job, and this time is the same for all the threads. You need to determine for each job which thread will process it and when will it start processing.

Input Format. The first line of the input contains integers n and m.
The second line contains m integers t_i — the times in seconds it takes any thread to process i-th job. The times are given in the same order as they are in the list from which threads take jobs.
Threads are indexed starting from 0.

Constraints. $1 \le n \le 10^5$; $1 \le m \le 10^5$; $0 \le t_i \le 10^9$.

Output Format. Output exactly m lines. i-th line (0-based index is used) should contain two space-separated integers — the 0-based index of the thread which will process the i-th job and the time in seconds when it will start processing that job.

Time Limits. `C`: 1 sec, `C++`: 1 sec, `Java`: 4 sec, `Python`: 6 sec. `C#`: 1.5 sec, `Haskell`: 2 sec, `JavaScript`: 6 sec, `Ruby`: 6 sec, `Scala`: 6 sec.

Memory Limit. 512Mb.

Sample 1.

Input:

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

Output:

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

Explanation:

1. The two threads try to simultaneously take jobs from the list, so thread with index 0 actually takes the first job and starts working on it at the moment 0.
2. The thread with index 1 takes the second job and starts working on it also at the moment 0.
3. After 1 second, thread 0 is done with the first job and takes the third job from the list, and starts processing it immediately at time 1.
4. One second later, thread 1 is done with the second job and takes the fourth job from the list, and starts processing it immediately at time 2.

5. Finally, after 2 more seconds, thread 0 is done with the third job and takes the fifth job from the list, and starts processing it immediately at time 4.

Sample 2.

Input:

```
4 20
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

Output:

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

Explanation:
Jobs are taken by 4 threads in packs of 4, processed in 1 second, and then the next pack comes. This happens 5 times starting at moments 0, 1, 2, 3 and 4. After that all the $5 \times 4 = 20$ jobs are processed.

Starter Files

The starter solutions for C++, Java and Python3 in this problem read the input, apply an $\Theta(n^2)$ algorithm to solve the problem and write the output. You need to replace the $\Theta(n^2)$ algorithm with a faster one. If you use other languages, you need to implement the solution from scratch.

What to Do

Think about the sequence of events when one of the threads becomes free (at the start and later after completing some job). How to apply priority queue to simulate processing of these events in the required order? Remember to consider the case when several threads become free simultaneously.

Beware of integer overflow in this problem: use type `long long` in C++ and type `long` in Java wherever the regular type `int` can overflow given the restrictions in the problem statement.

Need Help?

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

3 Problem: Merging tables

Problem Introduction

In this problem, your goal is to simulate a sequence of merge operations with tables in a database.

Problem Description

Task. There are n tables stored in some database. The tables are numbered from 1 to n. All tables share the same set of columns. Each table contains either several rows with real data or a symbolic link to another table. Initially, all tables contain data, and i-th table has r_i rows. You need to perform m of the following operations:

1. Consider table number $destination_i$. Traverse the path of symbolic links to get to the data. That is,

   while $destination_i$ contains a symbolic link instead of real data do

   $destination_i \leftarrow \texttt{symlink}(destination_i)$

2. Consider the table number $source_i$ and traverse the path of symbolic links from it in the same manner as for $destination_i$.
3. Now, $destination_i$ and $source_i$ are the numbers of two tables with real data. If $destination_i \neq source_i$, copy all the rows from table $source_i$ to table $destination_i$, then clear the table $source_i$ and instead of real data put a symbolic link to $destination_i$ into it.
4. Print the maximum size among all n tables (recall that size is the number of rows in the table). If the table contains only a symbolic link, its size is considered to be 0.

See examples and explanations for further clarifications.

Input Format. The first line of the input contains two integers n and m — the number of tables in the database and the number of merge queries to perform, respectively.
The second line of the input contains n integers r_i — the number of rows in the i-th table.
Then follow m lines describing merge queries. Each of them contains two integers $destination_i$ and $source_i$ — the numbers of the tables to merge.

Constraints. $1 \leq n, m \leq 100\,000$; $0 \leq r_i \leq 10\,000$; $1 \leq destination_i, source_i \leq n$.

Output Format. For each query print a line containing a single integer — the maximum of the sizes of all tables (in terms of the number of rows) after the corresponding operation.

Time Limits. `C`: 2 sec, `C++`: 2 sec, `Java`: 14 sec, `Python`: 6 sec. `C#`: 3 sec, `Haskell`: 4 sec, `JavaScript`: 6 sec, `Ruby`: 6 sec, `Scala`: 14 sec.

Memory Limit. 512Mb.

Sample 1.

Input:

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

Output:

```
2
2
3
5
5
```

Explanation:
In this sample, all the tables initially have exactly 1 row of data. Consider the merging operations:

1. All the data from the table 5 is copied to table number 3. Table 5 now contains only a symbolic link to table 3, while table 3 has 2 rows. 2 becomes the new maximum size.
2. 2 and 4 are merged in the same way as 3 and 5.
3. We are trying to merge 1 and 4, but 4 has a symbolic link pointing to 2, so we actually copy all the data from the table number 2 to the table number 1, clear the table number 2 and put a symbolic link to the table number 1 in it. Table 1 now has 3 rows of data, and 3 becomes the new maximum size.
4. Traversing the path of symbolic links from 4 we have $4 \to 2 \to 1$, and the path from 5 is $5 \to 3$. So we are actually merging tables 3 and 1. We copy all the rows from the table number 1 into the table number 3, and now the table number 3 has 5 rows of data, which is the new maximum.
5. All tables now directly or indirectly point to table 3, so all other merges won't change anything.

Sample 2.

Input:

```
6 4
10 0 5 0 3 3
6 6
6 5
5 4
4 3
```

Output:

```
10
10
10
11
```

Explanation:
In this example tables have different sizes. Let us consider the operations:

1. Merging the table number 6 with itself doesn't change anything, and the maximum size is 10 (table number 1).

2. After merging the table number 5 into the table number 6, the table number 5 is cleared and has size 0, while the table number 6 has size 6. Still, the maximum size is 10.
3. By merging the table number 4 into the table number 5, we actually merge the table number 4 into the table number 6 (table 5 now contains just a symbolic link to table 6), so the table number 4 is cleared and has size 0, while the table number 6 has size 6. Still, the maximum size is 10.
4. By merging the table number 3 into the table number 4, we actually merge the table number 3 into the table number 6 (table 4 now contains just a symbolic link to table 6), so the table number 3 is cleared and has size 0, while the table number 6 has size 11, which is the new maximum size.

Starter Files

The starter solutions in C++, Java and Python3 read the description of tables and operations from the input, declare and partially implement disjoint set union, and write the output. You need to complete the implementation of disjoint set union for this problem. If you use other languages, you will have to implement the solution from scratch.

What to Do

Think how to use disjoint set union with path compression and union by rank heuristics to solve this problem. In particular, you should separate in your thinking the data structure that performs union/find operations from the merges of tables. If you're asked to merge first table into second, but the rank of the second table is smaller than the rank of the first table, you can ignore the requested order while merging in the Disjoint Set Union data structure and join the node corresponding to the second table to the node corresponding to the first table instead in you Disjoint Set Union. However, you will need to store the number of the actual second table to which you were requested to merge the first table in the parent node of the corresponding Disjoint Set, and you will need an additional field in the nodes of Disjoint Set Union to store it.

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 $1 \le n \le 10^5$, 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.