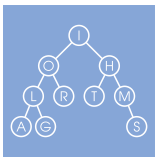


Module: Divide and Conquer (Week 4 out of 5)
Course: Algorithmic Toolbox (Course 1 out of 6)
Specialization: Data Structures and Algorithms

Programming Assignment 3: Divide-and-Conquer

Revision: August 4, 2016

Introduction

In this programming assignment, you will be practicing implementing divide-and-conquer solutions.

Learning Outcomes

Upon completing this programming assignment you will be able to:

1. Apply the divide-and-conquer technique to solve various computational problems efficiently. This will usually require you to design an algorithm that solves a problem by splitting it into several disjoint subproblems, solving them recursively, and then combining their results to get an answer for the initial problem.
2. Design and implement efficient algorithms for the following computational problems:
   (a) searching a sorted data for a key;
   (b) finding a majority element in a data;
   (c) improving the quick sort algorithm;
   (d) checking how close a data is to being sorted;
   (e) organizing a lottery;
   (f) finding the closest pair of points.

Passing Criteria: 2 out of 6

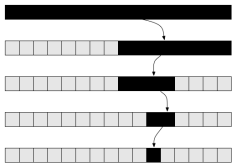
Passing this programming assignment requires passing at least 2 out of 6 code problems from this assignment. In turn, passing a code problem requires implementing a solution that passes all the tests for this problem in the grader and does so under the time and memory limits specified in the problem statement.

Contents

1 Problem: Implementing Binary Search

Problem Introduction

In this problem, you will implement the binary search algorithm that allows searching very efficiently (even huge) lists, provided that the list is sorted.



Problem Description

Task. The goal in this code problem is to implement the binary search algorithm.

Input Format. The first line of the input contains an integer n and a sequence $a_0 < a_1 < \ldots < a_{n-1}$ of n pairwise distinct positive integers in increasing order. The next line contains an integer k and k positive integers $b_0, b_1, \ldots, b_{k-1}$.

Constraints. $1 \le n, k \le 10^5$; $1 \le a_i \le 10^9$ for all $0 \le i < n$; $1 \le b_j \le 10^9$ for all $0 \le j < k$;

Output Format. For all i from 0 to $k-1$, output an index $0 \le j \le n-1$ such that $a_j = b_i$ or -1 if there is no such index.

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:

```
5 1 5 8 12 13
5 8 1 23 1 11
```

Output:

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

Explanation:
In this sample, we are given an increasing sequence $a_0 = 1$, $a_1 = 5$, $a_2 = 8$, $a_3 = 12$, $a_4 = 13$ of length five and five keys to search: $8, 1, 23, 1, 11$. We see that $a_2 = 8$ and $a_0 = 1$, but the keys 23 and 11 do not appear in the sequence a. For this reason, we output a sequence $2, 0, -1, 0, -1$.

Starter Files

The starter files contain an implementation of the linear search algorithm that just scans an array to find a given key. Try submitting a starter file to the grader to ensure that linear search indeed takes too long.

What To Do

Implement the binary search algorithm and replace the call to the linear search with a call to the binary search algorithm.

Need Help?

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

2 Problem: Finding a Majority Element

Problem Introduction

Majority rule is a decision rule that selects the alternative which has a majority, that is, more than half the votes.
Given a sequence of elements $a_1, a_2, \ldots, a_n$, you would like to check whether it contains an element that appears more than $n/2$ times. A naive way to do this is the following.

```
MajorityElement(a_1, a_2, ..., a_n):
for i from 1 to n:
  currentElement ← a_i
  count ← 0
  for j from 1 to n:
    if a_j = currentElement:
      count ← count + 1
  if count > n/2:
    return a_i
return "no majority element"
```

The running time of this algorithm is quadratic. Your goal is to use the divide-and-conquer technique to design an $O(n \log n)$ algorithm.

Problem Description

Task. The goal in this code problem is to check whether an input sequence contains a majority element.

Input Format. The first line contains an integer n, the next one contains a sequence of n non-negative integers $a_0, a_1, \ldots, a_{n-1}$.

Constraints. $1 \le n \le 10^5$; $0 \le a_i \le 10^9$ for all $0 \le i < n$.

Output Format. Output 1 if the sequence contains an element that appears strictly more than $n/2$ times, and 0 otherwise.

Time Limits.

language	C	C++	Java	Python	C#	Haskell	JavaScript	Ruby	Scala
time in seconds	1	1	1.5	5	1.5	2	5	5	3

Memory Limit. 512Mb.

Sample 1.

Input:

```
5
2 3 9 2 2
```

Output:

```
1
```

Explanation:
2 is the majority element.

Sample 2.

Input:

```
4
1 2 3 4
```

Output:

```
0
```

Explanation:
There is no majority element in this sequence.

Sample 3.

Input:

```
4
1 2 3 1
```

Output:

```
0
```

Explanation:
This sequence also does not have a majority element (note that the element 1 appears twice and hence is not a majority element).

Starter Files

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

What To Do

As you might have already guessed, this problem can be solved by the divide-and-conquer algorithm in time $O(n \log n)$. Indeed, if a sequence of length n contains a majority element, then the same element is also a majority element for one of its halves. Thus, to solve this problem you first split a given sequence into halves and make two recursive calls. Do you see how to combine the results of two recursive calls?

It is interesting to note that this problem can also be solved in $O(n)$ time by a more advanced (non-divide and conquer) algorithm that just scans the given sequence twice.

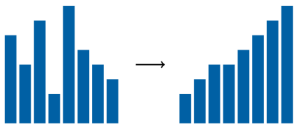
Need Help?

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

3 Problem: Improving Quick Sort

Problem Introduction

The goal in this problem is to redesign a given implementation of the randomized quick sort algorithm so that it works fast even on sequences containing many equal elements.



Problem Description

Task. To force the given implementation of the quick sort algorithm to efficiently process sequences with few unique elements, your goal is replace a 2-way partition with a 3-way partition. That is, your new partition procedure should partition the array into three parts: $< x$ part, $= x$ part, and $> x$ part.

Input Format. The first line of the input contains an integer n. The next line contains a sequence of n integers $a_0, a_1, \ldots, a_{n-1}$.

Constraints. $1 \le n \le 10^5$; $1 \le a_i \le 10^9$ for all $0 \le i < n$.

Output Format. Output this sequence sorted in non-decreasing order.

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:

```
5
2 3 9 2 2
```

Output:

```
2 2 2 3 9
```

Starter Files

In the starter files, you are given an implementation of the randomized quick sort algorithm using a 2-way partition procedure. This procedure partitions the given array into two parts with respect to a pivot x: $\le x$ part and $> x$ part. As discussed in the video lectures, such an implementation has $\Theta(n^2)$ running time on sequences containing a single unique element. Indeed, the partition procedure in this case splits the array into two parts, one of which is empty and the other one contains $n - 1$ elements. It spends cn time on this. The overall running time is then

$$cn + c(n-1) + c(n-2) + \ldots = \Theta(n^2)\,.$$

What To Do

Implement a 3-way partition procedure and then replace a call to the 2-way partition procedure by a call to the 3-way partition procedure.

Need Help?

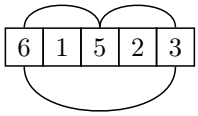
Ask a question or see the questions asked by other learners at this forum thread.

4 Advanced Problem: How Close a Data is To Being Sorted?

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

An inversion of a sequence $a_0, a_1, \ldots, a_{n-1}$ is a pair of indices $0 \le i < j < n$ such that $a_i > a_j$. The number of inversions of a sequence in some sense measures how close the sequence is to being sorted. For example, a sorted (in non-descending order) sequence contains no inversions at all, while in a sequence sorted in descending order any two elements constitute an inversion (for a total of $n(n-1)/2$ inversions).



Problem Description

Task. The goal in this problem is to count the number of inversions of a given sequence.

Input Format. The first line contains an integer n, the next one contains a sequence of integers $a_0, a_1, \ldots, a_{n-1}$.

Constraints. $1 \le n \le 10^5$, $1 \le a_i \le 10^9$ for all $0 \le i < n$.

Output Format. Output the number of inversions in the sequence.

Time Limits.

language	C	C++	Java	Python	C#	Haskell	JavaScript	Ruby	Scala
time in seconds	3	3	4.5	15	4.5	6	15	15	9

Memory Limit. 512Mb.

Sample 1.

Input:

```
5
2 3 9 2 9
```

Output:

```
2
```

Explanation:
The two inversions here are $(1, 3)$ $(a_1 = 3 > 2 = a_3)$ and $(2, 3)$ $(a_2 = 9 > 2 = a_3)$.

Starter Files

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

What To Do

This problem can be solved by modifying the merge sort algorithm. For this, we change both the `Merge` and `MergeSort` procedures as follows:

- `Merge`(B, C) returns the resulting sorted array and the number of pairs (b, c) such that $b \in B$, $c \in C$, and $b > c$;
- `MergeSort`(A) returns a sorted array A and the number of inversions in A.

Need Help?

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

5 Advanced Problem: Organizing a Lottery

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

You are organizing an online lottery. To participate, a person bets on a single integer. You then draw several ranges of consecutive integers at random. A participant's payoff then is proportional to the number of ranges that contain the participant's number minus the number of ranges that does not contain it. You need an efficient algorithm for computing the payoffs for all participants. A naive way to do this is to simply scan, for all participants, the list of all ranges. However, you lottery is very popular: you have thousands of participants and thousands of ranges. For this reason, you cannot afford a slow naive algorithm.

Problem Description

Task. You are given a set of points on a line and a set of segments on a line. The goal is to compute, for each point, the number of segments that contain this point.

Input Format. The first line contains two non-negative integers s and p defining the number of segments and the number of points on a line, respectively. The next s lines contain two integers a_i, b_i defining the i-th segment $[a_i, b_i]$. The next line contains p integers defining points $x_1, x_2, \ldots, x_p$.

Constraints. $1 \le s, p \le 50000$; $-10^8 \le a_i \le b_i \le 10^8$ for all $0 \le i < s$; $-10^8 \le x_j \le 10^8$ for all $0 \le j < p$.

Output Format. Output p non-negative integers $k_0, k_1, \ldots, k_{p-1}$ where k_i is the number of segments which contain x_i. More formally,

$$k_i = |\{j : a_j \le x_i \le b_j\}|.$$

Time Limits.

language	C	C++	Java	Python	C#	Haskell	JavaScript	Ruby	Scala
time in seconds	3	3	4.5	15	4.5	6	15	15	9

Memory Limit. 512Mb.

Sample 1.

Input:

```
2 3
0 5
7 10
1 6 11
```

Output:

```
1 0 0
```

Explanation:

Here, we have two segments and three points. The first point lies only in the first segment while the remaining two points are outside of all the given segments.

Sample 2.

Input:

```
1 3
-10 10
-100 100 0
```

Output:

```
0 0 1
```

Sample 3.

Input:

```
3 2
0 5
-3 2
7 10
1 6
```

Output:

```
2 0
```

Starter Files

The starter files for this problem contain an implementation of the following naive algorithm: for each point, scan the list of all segments to check how many of them contain this point. The running time of this algorithm is $O(sp)$ making it too slow to pass the given time limit.

What To Do

As you might have already guessed, you goal is to first sort the given segments somehow (so, this is a sorting problem, not a divide-and-conquer problem).

Need Help?

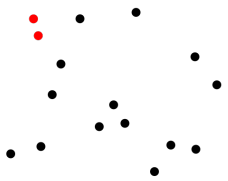
Ask a question or see the questions asked by other learners at this forum thread.

6 Advanced Problem: Finding the Closest Pair of Points

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, your goal is to find the closest pair of points among the given n points. This is a basic primitive in computational geometry having applications in, for example, graphics, computer vision, traffic-control systems.



Problem Description

Task. Given n points on a plane, find the smallest distance between a pair of two (different) points. Recall that the distance between points (x_1, y_1) and (x_2, y_2) is equal to $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.

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

Constraints. $1 \le n \le 10^5$; $-10^9 \le x_i, y_i \le 10^9$ are integers.

Output Format. Output the minimum distance. The absolute value of the difference between the answer of your program and the optimal value should be at most 10^{-3}. To ensure this, output your answer with at least four digits after the decimal point (otherwise your answer, while being computed correctly, can turn out to be wrong because of rounding issues).

Time Limits.

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

Memory Limit. 512Mb.

Sample 1.

Input:

```
2
0 0
3 4
```

Output:

```
5.0
```

Explanation:

There are only two points here. The distance between them is 5.

Sample 2.

Input:

```
4
7 7
1 100
4 8
7 7
```

Output:

```
0.0
```

Explanation:
There are two coinciding points among the four given points. Thus, the minimum distance is zero.

Sample 3.

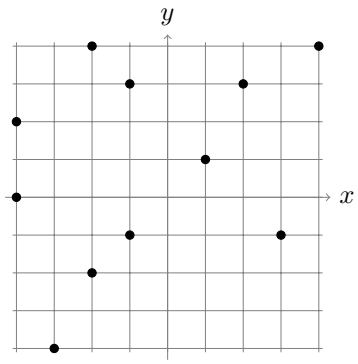
Input:

```
11
4 4
-2 -2
-3 -4
-1 3
2 3
-4 0
1 1
-1 -1
3 -1
-4 2
-2 4
```

Output:

```
1.414213
```

Explanation:
The smallest distance is $\sqrt{2}$. There are two pairs of points at this distance: $(-1, -1)$ and $(-2, -2)$; $(-2, 4)$ and $(-1, 3)$.



Starter Files

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

What To Do

A naive algorithm iterates through all pairs of points to find the minimal distance. Its running time is $\Theta(n^2)$ and it will not fit into the given time limit. Your goal therefore is to use the divide-and-conquer strategy to design a $O(n \log n)$ time algorithm. As usual, you might want to first partition the given set S of n points into two subsets S_1, S_2 of size $n/2$. You then make two recursive calls for the sets S_1 and S_2. After this, you know the minimal distances d_1 and d_2 for S_1 and S_2, respectively. But what if the minimal distance for the initial set S is realized for a pair of points p_1, p_2 such that $p_1 \in S_1$ and $p_2 \in S_2$? That is, you need to check whether there is a point $p_1 \in S_1$ and $p_2 \in S_2$ such that the distance between p_1 and p_2 is smaller than $\min\{d_1, d_2\}$. The corresponding algorithm is not easy to discover, but we still strongly encourage you to try to design it before reading a solution on the Internet or in a textbook (see, for example, [DPV08, Exercise 2.32], or [CLRS09, Section 33.4], or [KT06, Section 5.4]).

Need Help?

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

7 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.

7.1 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.

7.2 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.

7.3 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.

7.4 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
```

- `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
```

7.5 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.

7.6 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 7.4) 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).

7.7 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 7.5. 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!

8 Frequently Asked Questions

8.1 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 7.3 and 7.4). 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.

8.2 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.

8.3 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.

8.4 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.

8.5 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.

8.6 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).

8.7 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

[CLRS09] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms (3. ed.)*. MIT Press, 2009.

[DPV08] Sanjoy Dasgupta, Christos H. Papadimitriou, and Umesh V. Vazirani. *Algorithms*. McGraw-Hill, 2008.

[KT06] Jon M. Kleinberg and Éva Tardos. *Algorithm design*. Addison-Wesley, 2006.