Module: Dynamic Programming (Week 5 out of 5)
Course: Algorithmic Toolbox (Course 1 out of 6)
Specialization: Data Structures and Algorithms

Programming Assignment 4: Dynamic Programming

Revision: March 22, 2016

Introduction

In this programming assignment, you will be practicing implementing dynamic programming solutions. As usual, in some code problems you just need to implement an algorithm covered in the lectures, while for some others your goal will be to first design an algorithm and then implement it.

Learning Outcomes

Upon completing this programming assignment you will be able to:

1. Apply the dynamic programming technique to solve various computational problems. This will usually require you to design an algorithm that solves a problem by solving a collection of overlapping subproblems (as opposed to the divide-and-conquer technique where subproblems are usually disjoint) and combining the results.
2. Implement an efficient algorithm to optimally fill in a box with bars of gold.
3. Implement an efficient algorithm to compute the difference between two files or strings. Such algorithms are widely used in spell checking programs and version control systems.
4. Implement a more advanced algorithm for computing the maximum value of an arithmetic expression.
5. Design and implement a dynamic programming algorithm for a novel computational problem.
6. See two examples of an optimization problem where a natural greedy strategy produces a non-optimal result. You will see that a natural greedy move for these problems is not safe.

Passing Criteria: 2 out of 5

Passing this programming assignment requires passing at least 2 out of 5 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: Primitive Calculator

Problem Introduction

You are given a primitive calculator that can perform the following three operations with the current number x: multiply x by 2, multiply x by 3, or add 1 to x. Your goal is given a positive integer n, find the minimum number of operations needed to obtain the number n starting from the number 1.

Problem Description

Task. Given an integer n, compute the minimum number of operations needed to obtain the number n starting from the number 1.

Input Format. The input consists of a single integer $1 \leq n \leq 10^6$.

Output Format. In the first line, output the minimum number k of operations needed to get n from 1. In the second line output a sequence of intermediate numbers. That is, the second line should contain positive integers $a_0, a_2, \ldots, a_{k-1}$ such that $a_0 = 1$, $a_{k-1} = n$ and for all $0 \leq i < k - 1$, a_{i+1} is equal to either $a_i + 1$, $2a_i$, or $3a_i$. If there are many such sequences, output any one of them.

Time Limits. `C`: 1 sec, `C++`: 1 sec, `Java`: 1.5 sec, `Python`: 5 sec.

Memory Limit. 64Mb.

Sample 1.

Input:

```
1
```

Output:

```
0
1
```

Sample 2.

Input:

```
5
```

Output:

```
3
1 2 4 5
```

Explanation:

Here, we first multiply 1 by 2 two times and then add 1. Another possibility is to first multiply by 3 and then add 1 two times. Hence "1 3 4 5" is also a valid output in this case.

Sample 3.

Input:

```
96234
```

Output:

```
14
1 3 9 10 11 22 66 198 594 1782 5346 16038 16039 32078 96234
```

Explanation:

Again, another valid output in this case is "1 3 9 10 11 33 99 297 891 2673 8019 16038 16039 48117 96234".

Starter Files

Going from 1 to n is the same as going from n to 1, each time either dividing the current number by 2 or 3 or subtracting 1 from it. Since we would like to go from n to 1 as fast as possible it is natural to repeatedly reduce n as much as possible. That is, at each step we replace n by $\min\{n/3, n/2, n-1\}$ (the terms $n/3$ and $n/2$ are used only when n is divisible by 3 and 2, respectively). We do this until we reach 1. This gives rise to the following algorithm and it is implemented in the starter files:

```
GreedyCalculator(n):
numOperations ← 0
while n > 1:
  numOperations ← numOperations + 1
  if n mod 3 = 0:
    n ← n/3
  else if n mod 2 = 0:
    n ← n/2
  else:
    n ← n − 1
return numOperations
```

This seemingly correct algorithm is in fact incorrect. You may want to submit one of the starter files to ensure this. Hence in this case moving from n to $\min\{n/3, n/2, n-1\}$ is not *safe*.

What To Do

Your goal is to design and implement a dynamic programming solution for this problem. A natural subproblem in this case is the following: $C(n)$ is the minimum number of operations required to obtain n from 1 (using the three primitive operations). How to express $C(n)$ through $C(n/3)$, $C(n/2)$, $C(n-1)$?

2 Problem: Take as Much Gold as Possible

Problem Introduction

This problem is about implementing an algorithm for the knapsack without repetitions problem.

Problem Description

Task. In this problem, you are given a set of bars of gold and your goal is to take as much gold as possible into your bag. There is just one copy of each bar and for each bar you can either take it or not (hence you cannot take a fraction of a bar).

Input Format. The first line of the input contains the capacity W of a knapsack and the number n of bars of gold. The next line contains n integers $w_0, w_1, \ldots, w_{n-1}$ defining the weights of the bars of gold.

Constraints. $1 \le W \le 10^4$; $1 \le n \le 300$; $0 \le w_0, \ldots, w_{n-1} \le 10^5$.

Output Format. Output the maximum weight of gold that fits into a knapsack of capacity W.

Time Limits. `C`: 1 sec, `C++`: 1 sec, `Java`: 1.5 sec, `Python`: 5 sec.

Memory Limit. 64Mb.

Sample 1.

Input:

```
10 3
1 4 8
```

Output:

```
9
```

Explanation:
Here, the sum of the weights of the first and the last bar is equal to 9.

Starter Files

Starter files contain an implementation of the following greedy strategy: scan the list of given bars of gold and add the current bar if it fits into the current capacity (note that, in this problem, all the items have the same value per unit of weight, for a simple reasons: they are all made of gold). As you already know from the lectures, such a greedy move is not safe. You may want to additionally submit a starter file as a solution to the grading system to ensure that this greedy algorithm indeed might produce a non-optimal result.

What To Do

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

3 Problem: Compute the Edit Distance Between Two Strings

Problem Introduction

The edit distance between two strings is the minimum number of insertions, deletions, and mismatches in an alignment of two strings.

Problem Description

Task. The goal of this problem is to implement the algorithm for computing the edit distance between two strings.

Input Format. Each of the two lines of the input contains a string consisting of lower case latin letters.

Constraints. The length of both strings is at least 1 and at most 100.

Output Format. Output the edit distance between the given two strings.

Time Limits. `C`: 1 sec, `C++`: 1 sec, `Java`: 1.5 sec, `Python`: 5 sec.

Memory Limit. 64Mb.

Sample 1.

Input:

```
ab
ab
```

Output:

```
0
```

Sample 2.

Input:

```
short
ports
```

Output:

```
3
```

Explanation:
An alignment of total cost 3:

s	h	o	r	t	–
–	p	o	r	t	s

Sample 3.

Input:

```
editing
distance
```

Output:

```
5
```

Explanation:
An alignment of total cost 5:

e	d	i	–	t	i	n	g	–
–	d	i	s	t	a	n	c	e

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 `C` and `Python2`, you need to implement a solution from scratch.

What To Do

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

4 Problem: Maximize the Value of an Arithmetic Expression

Problem Introduction

In this problem, your goal is to add parentheses to a given arithmetic expression to maximize its value.

Problem Description

Task. Find the maximum value of an arithmetic expression by specifying the order of applying its arithmetic operations using additional parentheses.

Input Format. The only line of the input contains a string s of length $2n + 1$ for some n, with symbols $s_0, s_1, \ldots, s_{2n}$. Each symbol at an even position of s is a digit (that is, an integer from 0 to 9) while each symbol at an odd position is one of three operations from {`+`,`-`,`*`}.

Constraints. $1 \leq n \leq 14$ (hence the string contains at most 29 symbols).

Output Format. Output the maximum possible value of the given arithmetic expression among different orders of applying arithmetic operations.

Time Limits. `C`: 1 sec, `C++`: 1 sec, `Java`: 1.5 sec, `Python`: 5 sec.

Memory Limit. 64Mb.

Sample 1.

Input:

```
1+5
```

Output:

```
6
```

Sample 2.

Input:

```
5-8+7*4-8+9
```

Output:

```
200
```

Explanation:
$200 = (5 - ((8 + 7) \times (4 - (8 + 9))))$

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 `C` and `Python2`, you need to implement a solution from scratch.

What To Do

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

5 Advanced Problem: Longest Common Subsequence of Three Sequences

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

Problem Introduction

In this problem, your goal is to compute the length of a longest common subsequence of three sequences.

Problem Description

Task. Given three sequences $A = (a_1, a_2, \ldots, a_n)$, $B = (b_1, b_2, \ldots, b_m)$, and $C = (c_1, c_2, \ldots, c_l)$, find the length of their longest common subsequence, i.e., the largest non-negative integer p such that there exist indices $1 \le i_1 < i_2 < \cdots < i_p \le n$, $1 \le j_1 < j_2 < \cdots < j_p \le m$, $1 \le k_1 < k_2 < \cdots < k_p \le l$ such that $a_{i_1} = b_{j_1} = c_{k_1}, \ldots, a_{i_p} = b_{j_p} = c_{k_p}$

Input Format. First line: n. Second line: $a_1, a_2, \ldots, a_n$. Third line: m. Fourth line: $b_1, b_2, \ldots, b_m$. Fifth line: l. Sixth line: $c_1, c_2, \ldots, c_l$.

Constraints. $1 \le n, m, l \le 100$; $-10^9 < a_i, b_i, c_i < 10^9$.

Output Format. Output p.

Time Limits. `C`: 2 sec, `C++`: 2 sec, `Java`: 3 sec, `Python`: 10 sec.

Memory Limit. 128Mb.

Sample 1.

Input:

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

Output:

```
2
```

Explanation:
A common subsequence of length 2 is $(1, 3)$.

Sample 2.

Input:

```
5
8 3 2 1 7
7
8 2 1 3 8 10 7
6
6 8 3 1 4 7
```

Output:

```
3
```

Explanation:
One common subsequence of length 3 in this case is $(8, 3, 7)$. Another one is $(8, 1, 7)$.

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 `C` and `Python2`, you need to implement a solution from scratch.

What To Do

It might be easier to first design an algorithm finding a longest common subsequence of two (but not three) sequences. For this, review the algorithm for computing edit distance from the lectures.

6 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.

6.1 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.

6.2 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.

6.3 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++`, `Java`, `Python2`, `Python3`. 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. If you are going to use `C` or `Python2`, you need to implement a solution from scratch.

6.4 Compiling Your Program

For solving programming assignments, you can use any of the following programming languages: `C`, `C++`, `Java`, `Python2`, or `Python3`. 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.

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
```

- `C++` (`g++ 5.2.1`). File extensions: `.cc`, `.cpp`. Flags:

```
g++ -pipe -O2 -std=c++11
```

  If your `C/C++` compiler does not recognize `-std=c++11` 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`.

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

```
javac -encoding UTF-8
```

- `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
```

6.5 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.

6.6 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 6.4) 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).

6.7 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 6.5. 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!

7 Frequently Asked Questions

7.1 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 6.3 and 6.4). 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.

7.2 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.

7.3 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, the output of your program and the correct answer. Otherwise, you won't know the input, the output 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, the output of your program and the correct answer. Otherwise, you won't know the input, the output 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.

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

7.4 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.

7.5 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.

7.6 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).

7.7 Are you going to support my favorite language in programming assignments?

Currently, we are going to support `C`, `C++`, `Java`, and `Python` only, but we may add other programming languages later if there appears a huge need. To express your interest in a particular programming language, please post its name in this thread or upvote the corresponding option if it is already there.

7.8 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, it is just not true that you do not learn by trying to fix your implementation.

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.