Programming Assignment 5: Dynamic Programming 1

Revision: October 10, 2019

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. See examples of optimization problems where a natural greedy strategy produces a non-optimal result. You will see that a natural greedy move for these problems is not safe.
3. Design and implement an efficient algorithm for the following computational problems:
   (a) 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.
   (b) Design and implement a dynamic programming algorithm for a novel computational problem.

Passing Criteria: 3 out of 5

Passing this programming assignment requires passing at least 3 out of 5 programming challenges from this assignment. In turn, passing a programming challenge 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 Money Change Again

As we already know, a natural greedy strategy for the change problem does not work correctly for any set of denominations. For example, if the available denominations are 1, 3, and 4, the greedy algorithm will change 6 cents using three coins $(4 + 1 + 1)$ while it can be changed using just two coins $(3 + 3)$. Your goal now is to apply dynamic programming for solving the Money Change Problem for denominations 1, 3, and 4.

Problem Description

Input Format. Integer *money*.

Output Format. The minimum number of coins with denominations 1, 3, 4 that changes *money*.

Constraints. $1 \leq money \leq 10^3$.

Sample 1.

Input:

```
2
```

Output:

```
2
```

$2 = 1 + 1$.

Sample 2.

Input:

```
34
```

Output:

```
9
```

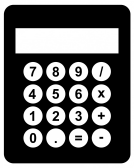
$34 = 3 + 3 + 4 + 4 + 4 + 4 + 4 + 4 + 4$.

Need Help?

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

2 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.

Sample 1.

Input:

```
1
```

Output:

```
0
1
```

Sample 2.

Input:

```
5
```

Output:

```
3
1 2 4 5
```

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
```

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*.

Need Help?

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

3 Edit Distance

Problem Introduction

The edit distance between two strings is the minimum number of operations (insertions, deletions, and substitutions of symbols) to transform one string into another. It is a measure of similarity of two strings. Edit distance has applications, for example, in computational biology, natural language processing, and spell checking. Your goal in this problem is to compute the edit distance between 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.

Sample 1.

Input:

```
ab
ab
```

Output:

```
0
```

Sample 2.

Input:

```
short
ports
```

Output:

```
3
```

An alignment of total cost 3:

s	h	o	r	t	–
–	p	o	r	t	s

Sample 3.

Input:

```
editing
distance
```

Output:

```
5
```

An alignment of total cost 5:

e	d	i	–	t	i	n	g	–
–	d	i	s	t	a	n	c	e

Need Help?

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

4 Longest Common Subsequence of Two Sequences

Problem Introduction

Compute the length of a longest common subsequence of three sequences.

Problem Description

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

Input Format. First line: n. Second line: $a_1, a_2, \ldots, a_n$. Third line: m. Fourth line: $b_1, b_2, \ldots, b_m$.

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

Output Format. Output p.

Sample 1.

Input:

```
3
2 7 5
2
2 5
```

Output:

```
2
```

A common subsequence of length 2 is $(2, 5)$.

Sample 2.

Input:

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

Output:

```
0
```

The two sequences do not share elements.

Sample 3.

Input:

```
4
2 7 8 3
4
5 2 8 7
```

Output:

```
2
```

One common subsequence is $(2, 7)$. Another one is $(2, 8)$.

Need Help?

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

5 Longest Common Subsequence of Three Sequences

Problem Introduction

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.

Sample 1.

Input:

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

Output:

```
2
```

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
```

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

Need Help?

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

6 Appendix

6.1 Compiler 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
```

Go (`golang 1.12`). File extensions: `.go`. Flags

```
go
```

Haskell (`ghc 7.8.4`). File extensions: `.hs`. Flags:

```
ghc -O2
```

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

```
javac -encoding UTF-8
java -Xmx1024m
```

JavaScript (`Node v10.15.3`). File extensions: `.js`. No flags:

```
nodejs
```

Kotlin (`Kotlin 1.2.21`). File extensions: `.kt`. Flags:

```
kotlinc
java -Xmx1024m
```

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
```

Rust (`Rust 1.28.0`). File extensions: `.rs`.

```
rustc
```

Scala (`Scala 2.11.6`). File extensions: `.scala`.

```
scalac
```

6.2 Frequently Asked Questions

Why My Submission Is Not Graded?

You need to create a submission and upload the *source file* (rather than the executable file) of your solution. Make sure that after uploading the file with your solution you press the blue "Submit" button at 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 are shown.

What Are the Possible Grading Outcomes?

There are only two outcomes: "pass" or "no pass." To pass, your program must return a correct answer on all the test cases we prepared for you, and do so under the time and memory constraints specified in the problem statement. If your solution passes, you get the corresponding feedback "Good job!" and get a point for the problem. Your solution fails if it either crashes, returns an incorrect answer, works for too long, or uses too much memory for some test case. The feedback will contain the index of the first test case on which your solution failed 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 first test to the test with the largest number.

Here are the possible outcomes:

- `Good job!` Hurrah! Your solution passed, and you get a point!
- `Wrong answer.` Your solution outputs incorrect answer for some test case. Check that you consider all the cases correctly, avoid integer overflow, output the required white spaces, 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.
- `Time limit exceeded.` Your solution worked longer than the allowed time limit for some test case. Check again the running time of your implementation. Test your program locally on the test of maximum size specified in the problem statement and check how long it works. Check that your program doesn't wait for some input from the user which makes it to wait forever.
- `Memory limit exceeded.` Your solution used more than the allowed memory limit for some test case. Estimate the amount of memory that your program is going to use in the worst case and check that it does not exceed the memory limit. Check that your data structures fit into the memory limit. 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 tests of maximum size specified in the problem statement and look at its memory consumption in the system.
- `Cannot check answer. Perhaps the output format is wrong.` This happens when you output something different than expected. For example, when you are required to output either "Yes" or "No", but instead output 1 or 0. Or your program has empty output. Or your program outputs not only the correct answer, but also some additional information (please follow the exact 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 a division by zero, accessing memory outside of the array bounds, using uninitialized variables, overly deep recursion that triggers a stack overflow, sorting with a 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 compiler and the same compiler flags as we do.
- `Internal error: exception...` Most probably, you submitted a compiled program instead of a source code.

- `Grading failed.` Something wrong happened with the system. Report this through Coursera or edX Help Center.

May I Post My Solution at the Forum?

Please do not post any solutions at 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). Our students follow the 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)."

Do I Learn by Trying to Fix My Solution?

My implementation always fails in the grader, though I already tested and stress tested it a lot. Would not it be better if you gave 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, learning from your mistakes is one of the best ways to learn.

The process of trying to invent new test cases that might fail your program is difficult but is often enlightening. Thinking about properties of your program 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, just like in real life. 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, it is important to learn how to find 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 your program on all corner cases you can imagine, constructed a set of manual test cases, applied stress testing, etc, but your program still fails, 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 by writing such a post you will realize that you missed some corner cases!), and only afterwards asking other learners to give you more ideas for tests cases.