Programming Assignment 2: Algorithmic Warm-up

Revision: March 20, 2019

Introduction

Welcome to your second programming assignment of the Algorithmic Toolbox at Coursera! It consists of eight programming challenges. Three of them require you just to implement carefully the algorithms covered in the lectures. The remaining challenges will require you to first design an algorithm and then to implement it. For all the challenges, we provide starter solutions in `C++`, `Java`, and `Python3`. These solutions implement straightforward algorithms that usually work only for small inputs. To verify this, you may want to submit these solutions to the grader. This will usually give you a "`time limit exceeded`" message for `Python` starter files and either "`time limit exceeded`" or "`wrong answer`" message for `C++` and `Java` solutions (the reason for wrong answer being an integer overflow issue). Your goal is to replace a naive algorithm with an efficient one. In particular, you may want to use the naive implementation for stress testing your efficient implementation.

In this programming assignment, the grader will show you the input data if your solution fails on any of the tests. This is done to help you to get used to the algorithmic problems in general and get some experience debugging your programs while knowing exactly on which tests they fail. However, for all the following programming assignments, the grader will show the input data only in case your solution fails on one of the first few tests.

Learning Outcomes

Upon completing this programming assignment you will be able to:

1. See the huge difference between a slow algorithm and a fast one.
2. Play with examples where knowing something interesting about a problem helps to design an algorithm that is much faster than a naive one.
3. Implement solutions that work much more faster than straightforward solutions for the following programming challenges:
   (a) compute a small Fibonacci number;
   (b) compute the last digit of a large Fibonacci number;
   (c) compute a huge Fibonacci number modulo m;
   (d) compute the last digit of a sum of Fibonacci numbers;
   (e) compute the last digit of a partial sum of Fibonacci numbers;
   (f) compute the greatest common divisor of two integers;
   (g) compute the least common multiple of two integers.
4. Implement the algorithms covered in the lectures, design new algorithms.
5. Practice implementing, testing, and debugging your solution. In particular, you will find out how in practice, when you implement an algorithm, you bump into unexpected questions and problems not covered by the general description of the algorithm. You will also check your understanding of the algorithm itself and most probably see that there are some aspects you did not think of before you had to actually implement it. You will overcome all those complexities, implement the algorithms, test them, debug, and submit to the system.

Passing Criteria: 4 out of 8

Passing this programming assignment requires passing at least 4 out of 8 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 Fibonacci Number

Problem Introduction

Recall the definition of Fibonacci sequence: $F_0 = 0$, $F_1 = 1$, and $F_i = F_{i-1} + F_{i-2}$ for $i \geq 2$. Your goal in this problem is to implement an efficient algorithm for computing Fibonacci numbers. The starter files for this problem contain an implementation of the following naive recursive algorithm for computing Fibonacci numbers in C++, Java, and Python3:

```
FIBONACCI($n$):
if $n \leq 1$:
  return $n$
return FIBONACCI($n - 1$) + FIBONACCI($n - 2$)
```

Try compiling and running a starter solution on your machine. You will see that computing, say, F_{40} already takes noticeable time.
Another way to appreciate the dramatic difference between an exponential time algorithm and a polynomial time algorithm is to use the following visualization by David Galles: http://www.cs.usfca.edu/~galles/visualization/DPFib.html. Try computing F_{20} by a recursive algorithm by entering "20" and pressing the "Fibonacci Recursive" button. You will see an endless number of recursive calls. Now, press "Skip Forward" to stop the current algorithm and call the iterative algorithm by pressing "Fibonacci Table". This will compute F_{20} very quickly. (Note that the visualization uses a slightly different definition of Fibonacci numbers: $F_0 = 1$ instead of $F_0 = 0$. This of course has almost no influence on the running time.)

Problem Description

Task. Given an integer n, find the nth Fibonacci number F_n.

Input Format. The input consists of a single integer n.

Constraints. $0 \leq n \leq 45$.

Output Format. Output F_n.

Sample 1.

Input:
```
10
```
Output:
```
55
```

$F_{10} = 55$.

Need Help?

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

2 Last Digit of a Large Fibonacci Number

Problem Introduction

Your goal in this problem is to find the last digit of n-th Fibonacci number. Recall that Fibonacci numbers grow exponentially fast. For example,

$$F_{200} = 280\,571\,172\,992\,510\,140\,037\,611\,932\,413\,038\,677\,189\,525\,.$$

Therefore, a solution like

```
$F[0] \leftarrow 0$
$F[1] \leftarrow 1$
for $i$ from 2 to $n$:
    $F[i] \leftarrow F[i-1] + F[i-2]$
print($F[n]$ mod 10)
```

will turn out to be too slow, because as i grows the ith iteration of the loop computes the sum of longer and longer numbers. Also, for example, F_{1000} does not fit into the standard C++ `int` type. To overcome this difficulty, you may want to store in $F[i]$ not the ith Fibonacci number itself, but just its last digit (that is, F_i mod 10). Computing the last digit of F_i is easy: it is just the last digit of the sum of the last digits of F_{i-1} and F_{i-2}:

```
$F[i] \leftarrow (F[i-1] + F[i-2])$ mod 10
```

This way, all $F[i]$'s are just digits, so they fit perfectly into any standard integer type, and computing a sum of $F[i-1]$ and $F[i-2]$ is performed very quickly.

Problem Description

Task. Given an integer n, find the last digit of the nth Fibonacci number F_n (that is, F_n mod 10).

Input Format. The input consists of a single integer n.

Constraints. $0 \leq n \leq 10^7$.

Output Format. Output the last digit of F_n.

Sample 1.

Input:

```
3
```

Output:

```
2
```

$F_3 = 2$.

Sample 2.

Input:

```
331
```

Output:

```
9
```

$F_{331} = 668\,996\,615\,388\,005\,031\,531\,000\,081\,241\,745\,415\,306\,766\,517\,246\,774\,551\,964\,595\,292\,186\,469$.

Sample 3.

Input:

```
327305
```

Output:

```
5
```

F_{327305} does not fit into one line of this pdf, but its last digit is equal to 5.

Need Help?

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

3 Greatest Common Divisor

Problem Introduction

The greatest common divisor $\text{GCD}(a, b)$ of two non-negative integers a and b (which are not both equal to 0) is the greatest integer d that divides both a and b. Your goal in this problem is to implement the Euclidean algorithm for computing the greatest common divisor.
Efficient algorithm for computing the greatest common divisor is an important basic primitive of commonly used cryptographic algorithms like RSA.

$$\begin{aligned} &\text{GCD}(1344, 217)\\ =&\text{GCD}(217, 42)\\ =&\text{GCD}(42, 7)\\ =&\text{GCD}(7, 0)\\ =&7 \end{aligned}$$

Problem Description

Task. Given two integers a and b, find their greatest common divisor.

Input Format. The two integers a, b are given in the same line separated by space.

Constraints. $1 \leq a, b \leq 2 \cdot 10^9$.

Output Format. Output $\text{GCD}(a, b)$.

Sample 1.

Input:

```
18 35
```

Output:

```
1
```

18 and 35 do not have common non-trivial divisors.

Sample 2.

Input:

```
28851538 1183019
```

Output:

```
17657
```

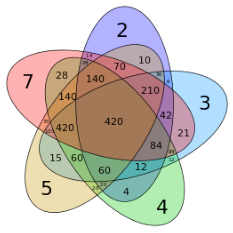
$28851538 = 17657 \cdot 1634$, $1183019 = 17657 \cdot 67$.

Need Help?

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

4 Least Common Multiple

Problem Introduction



The least common multiple of two positive integers a and b is the least positive integer m that is divisible by both a and b.

Problem Description

Task. Given two integers a and b, find their least common multiple.

Input Format. The two integers a and b are given in the same line separated by space.

Constraints. $1 \leq a, b \leq 2 \cdot 10^9$.

Output Format. Output the least common multiple of a and b.

Sample 1.

Input:
```
6 8
```
Output:
```
24
```

Among all the positive integers that are divisible by both 6 and 8 (e.g., 48, 480, 24), 24 is the smallest one.

Sample 2.

Input:
```
28851538 1183019
```
Output:
```
1933053046
```

1933053046 is the smallest positive integer divisible by both 28851538 and 1183019.

Need Help?

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

5 Fibonacci Number Again

Problem Introduction

In this problem, your goal is to compute F_n modulo m, where n may be really huge: up to 10^{18}. For such values of n, an algorithm looping for n iterations will not fit into one second for sure. Therefore we need to avoid such a loop.

To get an idea how to solve this problem without going through all F_i for i from 0 to n, let's see what happens when m is small — say, $m = 2$ or $m = 3$.

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
F_i	0	1	1	2	3	5	8	13	21	34	55	89	144	233	377	610
F_i mod 2	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0
F_i mod 3	0	1	1	2	0	2	2	1	0	1	1	2	0	2	2	1

Take a detailed look at this table. Do you see? Both these sequences are periodic! For $m = 2$, the period is 011 and has length 3, while for $m = 3$ the period is 01120221 and has length 8. Therefore, to compute, say, F_{2015} mod 3 we just need to find the remainder of 2015 when divided by 8. Since $2015 = 251 \cdot 8 + 7$, we conclude that $F_{2015} \bmod 3 = F_7 \bmod 3 = 1$.

This is true in general: for any integer $m \geq 2$, the sequence F_n mod m is periodic. The period always starts with 01 and is known as Pisano period.

Problem Description

Task. Given two integers n and m, output F_n mod m (that is, the remainder of F_n when divided by m).

Input Format. The input consists of two integers n and m given on the same line (separated by a space).

Constraints. $1 \leq n \leq 10^{18}$, $2 \leq m \leq 10^3$.

Output Format. Output F_n mod m.

Sample 1.

Input:

```
239 1000
```

Output:

```
161
```

$F_{239} \bmod 1\,000 = 39\,679\,027\,332\,006\,820\,581\,608\,740\,953\,902\,289\,877\,834\,488\,152\,161 \pmod{1\,000} = 161.$

Sample 2.

Input:

```
2816213588 239
```

Output:

```
151
```

$F_{2\,816\,213\,588}$ does not fit into one page of this file, but $F_{2\,816\,213\,588} \bmod 239 = 151$.

Need Help?

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

6 Last Digit of the Sum of Fibonacci Numbers

Problem Introduction

The goal in this problem is to find the last digit of a sum of the first n Fibonacci numbers.

Problem Description

Task. Given an integer n, find the last digit of the sum $F_0 + F_1 + \cdots + F_n$.

Input Format. The input consists of a single integer n.

Constraints. $0 \le n \le 10^{18}$.

Output Format. Output the last digit of $F_0 + F_1 + \cdots + F_n$.

Sample 1.

Input:

```
3
```

Output:

```
4
```

$F_0 + F_1 + F_2 + F_3 = 0 + 1 + 1 + 2 = 4$.

Sample 2.

Input:

```
100
```

Output:

```
5
```

The sum is equal to 927 372 692 193 078 999 175, the last digit is 5.

What To Do

Instead of computing this sum in a loop, try to come up with a formula for $F_0 + F_1 + F_2 + \cdots + F_n$. For this, play with small values of n. Then, use a solution for the previous problem.

Need Help?

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

7 Last Digit of the Sum of Fibonacci Numbers Again

Problem Introduction

Now, we would like to find the last digit of a *partial* sum of Fibonacci numbers: $F_m + F_{m+1} + \cdots + F_n$.

Problem Description

Task. Given two non-negative integers m and n, where $m \leq n$, find the last digit of the sum $F_m + F_{m+1} + \cdots + F_n$.

Input Format. The input consists of two non-negative integers m and n separated by a space.

Constraints. $0 \leq m \leq n \leq 10^{18}$.

Output Format. Output the last digit of $F_m + F_{m+1} + \cdots + F_n$.

Sample 1.

Input:
```
3 7
```
Output:
```
1
```

$F_3 + F_4 + F_5 + F_6 + F_7 = 2 + 3 + 5 + 8 + 13 = 31.$

Sample 2.

Input:
```
10 10
```
Output:
```
5
```

$F_{10} = 55.$

Sample 3.

Input:
```
10 200
```
Output:
```
2
```

$F_{10} + F_{11} + \cdots + F_{200} = 734\,544\,867\,157\,818\,093\,234\,908\,902\,110\,449\,296\,423\,262$

Need Help?

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

8 Last Digit of the Sum of Squares of Fibonacci Numbers

Problem Description

Task. Compute the last digit of $F_0^2 + F_1^2 + \cdots + F_n^2$.

Input Format. Integer n.

Constraints. $0 \le n \le 10^{18}$.

Output Format. The last digit of $F_0^2 + F_1^2 + \cdots + F_n^2$.

Sample 1.

Input:

```
7
```

Output:

```
3
```

$$F_0^2 + F_1^2 + \cdots + F_7^2 = 0 + 1 + 1 + 4 + 9 + 25 + 64 + 169 = 273.$$

Sample 2.

Input:

```
73
```

Output:

```
1
```

$$F_0^2 + \cdots + F_{73}^2 = 1\,052\,478\,208\,141\,359\,608\,061\,842\,155\,201.$$

Sample 3.

Input:

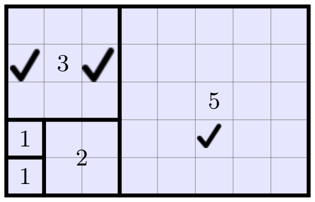
```
1234567890
```

Output:

```
0
```

What To Do

Since the brute force search algorithm for this problem is too slow (n may be as large as 10^{18}), we need to come up with a simple formula for $F_0^2 + F_1^2 + \cdots + F_n^2$. The figure below represents the sum $F_1^2 + F_2^2 + F_3^2 + F_4^2 + F_5^2$ as the area of a rectangle with vertical side $F_5 = 5$ and horizontal side $F_5 + F_4 = 3 + 5 = F_6$.



Need Help?

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