Math 3159A Group Assignment 3 Fall 2020

Instructions

- This assignment is due on Tuesday, October 27, 2020 at 2:00 PM EDT. Late submissions will **not** be accepted.
- This assignment consists of one problem with two parts. You must submit both parts to receive full credit.
- Your solution needs to be formatted using the LaTeX template available on OWL. Note that there are different templates available for regular assignments and group assignments. You should use the one for group assignments.
- All group members are expected to be working on the solution and every member should attend all group meetings.
- The Scribe will be submitting the assignment on behalf of the group. It is assumed that every member of the group has proofread the submission.
- All solutions must be written in full sentences.
- You are not allowed to use online resources and should only discuss the solution with members of your group.
- This assignment is worth 5 points.

Part 1.

Suppose that an RSA key pair (N, e) was compromised, so that you have access to the triple (N, e, d), with e the encryption exponent and d the decryption exponent. Your goal is to factor N using this information.

To do that, you should modify the Miller–Rabin test to return a factorization of $N = pq$, taking advantage of its being a product of two primes, rather than only asserting `composite` or `probably prime`. Just as the Miller–Rabin test, your algorithm will be probabilistic.

Your solution should consist of two components:

1. An algorithm that given a triple (N, e, d) where $N = pq$ for primes p and q, and

$$de \equiv 1 \bmod (p-1)(q-1),$$

returns p and q.

2. A justification why it works.

Part 2.

1. Write a function in Python3 called `solve` that, given a triple (N, e, d), with $N = pq$, a product of two large primes, e an encryption exponent, and d a decryption exponent, as in the RSA algorithm, returns either p or q, or fails.

2. Download the file `generate_input.py`, and use it to obtain a list of 10 triples of the form (N, e, d) by importing the file

```
from generate_input import generate_input
```

   and running the function

```
generate_input("[last three digits of your student number]")
```

   (Note the quotation marks.)

3. Run your method `solve` on all these inputs.

As part of your submission, include:

1. The *Python code* implementing your solution;

2. The 10 *inputs you generated*, and the *output of your program* run on these inputs. One input and one output per line.

Examples

Here are some examples of what your function `solve` should do.

```
>>> solve(10, 7, 3)
(2, 5)
>>> solve(33, 19, 19)
(3, 11)
>>> solve(323, 173, 5)
(17, 19)
```

Notes

- It is not a problem if your program does not work for every input (N, e, d). However there are simple and efficient programs that work on all the inputs given by `generate_input.py`. You will lose points if your program does not work on all these inputs.

- Incorrect answers will be penalized more than missing answers. (It is straightforward to verify the correctness of your submission!)

- Make sure that your algorithm terminates on the inputs we provide.
- You may not use any trivial brute-force algorithms. You must implement the algorithm developed in Part 1 of the assignment.
- The file `generate_input.py` is written in Python3, and so should be your solution. Make sure you are using a 64bit version of Python3.
- Your code should not make use of any external libraries such as `numpy` or `math`. All the auxiliary functions should be implemented by you, and should be included in your submission. You should only use the most basic arithmetic operations such as `+`, `-`, `*`, `//`, `%`.
- Comments in the code are not mandatory. However in the case of an incorrect solution, the comments can provide grounds for partial credit.