CS 70 Discrete Mathematics and Probability Theory
Fall 2016 Seshia and Walrand

# HW 4

1. **Sundry**

   Before you start your homework, write down your team. Who else did you work with on this homework? List names and email addresses. (In case of hw party, you can also just describe the group.) How did you work on this homework? Working in groups of 3-5 will earn credit for your "Sundry" grade.

   Please copy the following statement and sign next to it:

   *I certify that all solutions are entirely in my words and that I have not looked at another student's solutions. I have credited all external sources in this write up.*

2. (32 points) **Modular Arithmetic**

   Solve the following equations for $x$ and $y$ modulo the indicated modulus, or show that no solution exists. Show your work.

   (a) (8 points) $9x \equiv 1 \pmod{11}$.

   (b) (8 points) $10x + 23 \equiv 3 \pmod{31}$.

   (c) (8 points) $3x + 15 \equiv 4 \pmod{21}$.

   (d) (8 points) The system of simultaneous equations $3x + 2y \equiv 0 \pmod{7}$ and $2x + y \equiv 4 \pmod{7}$.

3. (8 points) **Don't Try This at Home**

   A ticket in the lottery consists of six numbers chosen from $1, 2, \ldots, 48$ (repetitions allowed). After everyone has bought their tickets, the manager picks 5 winning numbers from this set at random. Your ticket wins if it contains each of these winning numbers. Order is irrelevant.

   Prove that if you buy all possible tickets for which the sum of the six entries on the ticket is divisible by 47, then you are guaranteed to have a winner.

4. (50 points) **Further extending the extended GCD algorithm**

   In class, you learned how to use Euclid's algorithm to find the GCD of 2 numbers $x$ and $y$. You also saw an extended version of the algorithm that allowed you to find 2 other numbers $a$ and $b$ such that $ax + by = \text{GCD}(x, y)$.

   In this problem, you're going to analyze an algorithm that finds the GCD of *more than* 2 numbers. That is, you're given $n$ numbers $[x_1, x_2, \ldots, x_n]$, and your job is to develop/analyze an algorithm that finds the greatest natural number (the GCD) that divides all the given numbers.

(a) (10 points) Suppose you're given $n$ non-negative numbers $[x_1, x_2, \ldots, x_n]$, at least one of which is strictly positive. Of these $n$ numbers, let $z$ be the smallest number that is strictly positive. Before going ahead, convince yourself that such a $z$ exists.

Now, suppose I take each number $x_i \neq z$ on the list above, and replace it with the remainder that I get when I divide $x_i$ by $z$. Let's say this results in a new list $[y_1, y_2, \ldots, y_n]$. Show that:

$$\text{GCD}(x_1, x_2, \ldots, x_n) = \text{GCD}(y_1, y_2, \ldots, y_n).$$

(b) (10 points) Consider the algorithm `GCDmany` below that is intended to compute the GCD of a list of numbers.

```
algorithm GCDmany (list [$x_1$, $x_2$, ..., $x_n$]):
    nz = list of all non-zero elements of [$x_1$, $x_2$, ..., $x_n$]
    if length(nz) == 1:
        return nz[0]   # the first and only element of nz
    [idx, m] = min(nz)   # position, value of smallest element of nz
    foreach $k \neq$ idx such that $0 \leq k <$ length(nz):
        nz[k] = nz[k] mod m
    return GCDmany(nz)
```

Using the result that you proved in Part 1, show that the `GCDmany` algorithm correctly returns the GCD of any list of non-negative numbers, provided that at least one of the numbers in the list is strictly positive.

(c) (10 points) Suppose I run the `GCDmany` algorithm on $n$ positive numbers, each represented by $m$ bits. Derive a bound (in terms of $m$ and $n$) for the number of recursion calls that the algorithm execution will result in.

(d) (10 points) Suppose I tell you the following:

- Listing all non-zero elements given $n$ $m$-bit numbers can be done in at most $mn$ computer operations,
- Computing the minimum of $n$ $m$-bit numbers can be done using at most $4mn$ operations,
- The remainder on dividing one $m$-bit number by another can be calculated within $3m^2$ operations,
- All other steps needed by the `GCDmany` algorithm take at most 10 computer operations, and
- Each computer operation takes $1ns$ on my computer.

Based on the above, derive an upper bound for how long it will take for my computer to find the GCD of 100 positive 64-bit numbers using the `GCDmany` algorithm above.

(e) (10 points) Suppose I'm interested in computing not just the GCD $G$ of the $n$ positive numbers $x_1$ to $x_n$, but also integer coefficients $a_1$ to $a_n$ such that:

$$\sum_{i=1}^{n} a_i x_i = G.$$

Describe how you will modify the `GCDmany` algorithm to also output the coefficients above. Write down the steps in your modified algorithm in a format similar to the `GCDmany` algorithm above.

5. (Optional) **The last digit**
   Let $a$ be a positive integer. Consider the following sequence of numbers $x$ defined by:

$$\begin{aligned} x_0 &= a \\ x_n &= x_{n-1}^2 + x_{n-1} + 1 \text{ if } n > 0 \end{aligned}$$

   (a) Show that if the last digit of $a$ is 3 or 7, then for every $n$, the last digit of $x_n$ is respectively 3 or 7.

   (b) Show that there exist $k > 0$ such that the last digit of $x_n$ for $n \geq k$ is constant. Give the smallest possible $k$, *no matter what $a$ is.*