Prof. Sergio A. Alvarez
21 Campanella Way, room 569
Computer Science Department
Boston College
Chestnut Hill, MA 02467 USA

http://www.cs.bc.edu/∼alvarez/
alvarez@cs.bc.edu
voice: (617) 552-4333
fax: (617) 552-6790

# CS383, Algorithms
# Spring 2009
# HW2 Solutions

1. Here is a very simple algorithm for computing the greatest common divisor of two integers:

   **Algorithm 1:** simpleGCD
   **Input:** Two integers $x \geq y > 0$.
   **Output:** The greatest common divisor of $x$ and $y$.
   SIMPLEGCD($x$, $y$)
   (1) **foreach** $d = y \cdots 1$
   (2) **if** $x \mod d = 0$ and $y \mod d = 0$
   (3) **return** d

   (a) Notice that Algorithm 1 makes at most $y$ passes through the main loop. Also, the time required for each pass is dominated by the time spent in computing the modular remainders. Does this imply that Algorithm 1 is efficient, in the sense that its running time is bounded above by a polynomial in the input size? Explain.

   **Solution**

   NO. The given argument only implies that the running time is bounded above by a polynomial in the *magnitude* of the numbers $x$ and $y$. Note that the input *size* is measured by the length of the string representations of $x$ and $y$, that is, the number of digits in their positional representations (base 2 or 10).

   (b) Analyze the running time of Algorithm 1 in detail, explaining each step. Express your answer in asymptotic notation.

   **Solution**

   Let $d$ be the number of binary digits occupied by each of the inputs $x$ and $y$. The number of passes through the main loop is $O(y)$, which is $O(2^d)$. The time required for each pass is dominated by the computation of the modular remainders, which is $O(d^2)$ as discussed in class. Hence, the tightest overall upper bound that we can give on the running time is $O(d^2 2^d)$, which is most definitely not a polynomial in $d$.

2. Let $A$ denote an unspecified algorithm that operates on a collection of pixels as the input. We are interested in the worst-case asymptotic running time of $A$.

   (a) If $A$'s running time is known to be $O(3^n)$, where $n$ is the number of pixels in the input, is it still possible for $A$ to have a polynomial running time? Explain.

**Solution**

Definitely YES! No exponential is big $O$ of a polynomial, but the statement doesn't imply that $A$ actually takes exponential time. *Big O only provides an upper bound.* If $t(n)$ is the worst-case running time of $A$ on inputs of size $n$, we're just saying that there is some finite constant $C$ such that $t(n) \leq C * 2^n$ for all integers $n > 0$. The time might well be $t(n) = 12n$, for example, based on the statement.

(b) If $A$'s running time is known to be $\Omega(n^3)$, must its running time be $\Omega(n)$ also? Explain.

**Solution**

Again, YES! *Big $\Omega$ provides a lower bound.* To say that $f(n) = \Omega(n^3)$ means that $f(n) \geq C * n^3$ for some strictly positive (non-zero in particular) constant $C$ and all integer values of $n > 0$. Since $n^3 \geq n$ for all $n \geq 0$, this implies immediately that $f(n) \geq C * n$ for all $n > 0$, which is the same thing as saying that $f(n) = \Omega(n)$.

(c) Assume that for each positive integer $n$ there is a special "test image" $I_n$ that contains exactly $n$ pixels such that $A$'s computation on input $I_n$ requires exactly $17n^3$ basic steps. Based on this information alone, what is *the most specific* conclusion that can be reached regarding the asymptotic time complexity (running time) of $A$? Use asymptotic notation ($O, \Omega, \Theta$, as appropriate). Explain your answer carefully.

**Solution**

The stated information provides a lower bound on the running time of $A$ on inputs of size $n$. This translates directly to the asymptotic notation $t(n) = \Omega(n^3)$. We don't have an upper bound on the running time, because nothing is said about how much time is required by $A$ on other inputs of size $n$. We cannot provide big $O$ or big $\Theta$ expressions for $A$'s running time in the absence of additional information.

3. (a) Write out the full computation in tabular form for the extended Euclid gcd algorithm (as discussed in class) on input $(31, 12)$. The result of the computation should be three integers $(x, y, d)$, where $d = \gcd(31, 12)$ and $x * 31 + y * 12 = 1$. Explain.

**Solution**

As described in class:

```
a b a/b a mod b x y

31 12  2  7 -5 13
12  7  1  5  3 -5
7   5  1  2 -2  3
5   2  2  1  1 -2
2   1  2  0  0  1
1   0  1  0
```

Returned values: (x=-5, y=13, d=1)

(b) Based on the preceding subtask, what is the multiplicative inverse of 12 modulo 31? Explain.

**Solution**

The extended Euclid algorithm (see above) shows that

$$-5 * 31 + 13 * 12 = 1$$

Taking the remainder modulo 31 on both sides, we find:

$$13 * 12 = 1 \pmod{31}$$

Hence,

$$12^{-1} = 13 \pmod{31}$$

(c) Suppose that $x$ is a number such that $125 * x$ exceeds a multiple of 17 by exactly 1 (as do the numbers 18, 35, 52...). What is the remainder of $x$ modulo 17? Explain.

**Solution**

The statement implies that

$$125 * x = 1 \pmod{17}$$

or, equivalently,

$$6 * x = 1 \pmod{17}$$

since

$$125 = 6 \pmod{17}$$

To solve for $x$, we multiply both sides by the multiplicative inverse of 6 mod 17. In fact, we see that (modulo 17), $x$ is just the multiplicative inverse in question. We just need to compute the inverse, which we can do via the extended Euclid algorithm:

```
a b a/b a mod b x y

17 6  2  5 -1  3
6  5  1  1  1 -1
5  1  5  0  0  1
1  0  1  0
```

We find that the inverse is 3 (you can also check that $3 * 6 \equiv 1 \pmod{17}$), so

$$x \equiv 3 \pmod{17}$$

4. Use the substitution rule of modular arithmetic to compute each of the following values. Include a step by step explanation and an answer in each case.

(a) $55 * 973 \pmod{19}$

**Solution**

The substitution rule allows us to replace all values with their modular remainders without affecting the result. Hence, since 38 and 950 are multiples of 19, the desired value is the same as $17 * 4 \pmod{19}$, which equals 11.

(b) $251^{29} \pmod 3$

**Solution**

First determine that $251 \equiv 2 \pmod 3$ (since 249 is a multiple of 3). By the substitution rule, the target value is therefore the same as $2^{29} \pmod 3$. Now leverage the fact that $2^3 \equiv -1 \pmod 3$, by expressing $2^{29}$ as $2^{29} * 2^2 = (2^3)^9 * 4$ and then replacing 4 by its mod 3 remainder, which is 1:

$$251^{29} \equiv 2^{29} \equiv (2^3)^9 * 4 \equiv (-1)^9 * 1 \equiv -1 \ (\bmod 3)$$

Finally, convert the answer to the range $\{0 \cdots 3-1\}$ by adding 3 (which is congruent to 0 (mod 3)):

$$251^{29} \equiv 2 \ (\bmod 3)$$

(c) $30^{11} + 2^{500} \pmod{25}$

**Solution**

Deal with the two terms in the sum separately. For the power of 2 term, notice that $2^5 \equiv 7 \pmod{25}$, so that $2^10 \equiv -1 \pmod{25}$ (since $49 \equiv -1 \pmod{25}$). Therefore,

$$2^{500} \equiv (2^{10})^{50} \equiv (-1)^{50} \equiv 1 \ (\bmod 25)$$

Replace the base of the other term, 30, by its mod 25 remainder, which is 5, and notice that $5^2 \equiv 0 \pmod{25}$. This yields:

$$30^{11} \equiv 5^{11} \equiv (5^2)^5 * 5 \equiv 0 \ (\bmod 25)$$

Combining our analyses for the two terms, we conclude that

$$30^{11} + 2^{500} \equiv 1 \ (\bmod 25)$$

5. Romeo's public RSA key is:

$$N = 1238231 \quad e = 806903$$

Juliet wishes to send Romeo a secret message $s$, an uppercase text string describing one of her favorite things. She first encodes $s$ as an integer $m$ as follows: each character of $s$ is converted to its numerical position in the alphabet ($A = 1$, $B = 2$, etc.) and the resulting string of numbers is interpreted as an integer value in base 32 positional notation. For example, the string "ABC" yields the integer value

$$1 * 32^2 + 2 * 32 + 3$$

Juliet then sends the RSA-encrypted value of $m$ below, using Romeo's public key:

$$m^e \bmod N$$

Suppose you manage to intercept the encrypted value, which happens to be 510546. Can you break the code and recover Juliet's original message $s$?

## Solution

The main obstacle toward recovering Romeo's private key is to factor $N$ into the product of two primes. This task is feasible in this example because Romeo is behind the times and is using a very small value of $N$ in his public key. A brute-force approach yields:

$$N = p * q, \quad \text{where } p = 1201,\ q = 1031$$

The next step is to recover the private decryption exponent, $d$, as the multiplicative inverse of $e \bmod (p-1) * (q-1)$. This task is completed easily using the extended Euclidean gcd algorithm. Details are omitted (exercise). The result is $d = 218567$. With $d$ in hand, we proceed to decode the transmitted message:

$$m = 510546^d \ (\bmod N) = 19041$$

Be careful to use sufficiently long integers for this calculation; otherwise the results will be incorrect. Finally, convert the decoded numerical value back to text. What is Juliet's secret message?