College of Computer & Information Science
Northeastern University
CS5800: Algorithms

Spring 2011
Handout 2
26 January 2011

Quiz 1

Problem 1. (7 points) Modular Double Exponentiation

In class and text, we have seen algorithms for several arithmetic operations – addition, multiplication, division, exponentiation, modular arithmetic. The following table lists the running times you may assume for all the operations listed over any three n-bit numbers a, b, and c.

Operation	Running time
ADD(a, b, c): $a + b \bmod c$	$O(n^2)$
MULTIPLY(a, b, c): $a \cdot b \bmod c$	$O(n^2)$
POWER(a, b, c): $a^b \bmod c$	$O(n^3)$

In this exercise, we study the double exponentiation procedure in modular arithmetic. Give an algorithm DOUBLE-EXPONENT that takes as input four n-bit positive integers x, y, z, and p, where p is *known to be prime*, and returns $x^{y^z} \bmod p$. State its running time, in terms of n, using the big-Oh notation.

- Give your algorithm in pseudo-code. You may use any of the operations listed in the table above in your algorithm. You may also use any other operation or material we have covered in class.
- Make the running time of your algorithm as small as possible. Most important, however, is to give a correct procedure and state the correct running time for your procedure, even if it may not be the optimal achievable.
- You *do not need* to prove the correctness of your algorithm or formally establish its running time.

Answer: Here are two solutions for DOUBLE-EXPONENT(x,y,z,p)

1. POWER$(x,$ SIMPLE-POWER$(y, z), p)$.
2. POWER$(x,$ POWER$(y, z, p-1), p)$.

Let us consider Solution 1. In the above, SIMPLE-POWER(a, b) simply computes a^b without taking the modulus with respect to any number. As shown in text, exponentiation y^z can be done with $O(\log z) = O(n)$ multiplications. These multiplications, however, are of large numbers: if y and z both have n bits, then y^z can be as large as $(2^n)^{2^n}$, which can have as many as $n2^n$ bits; so the cost of some of the multiplications could be as high as $O((n2^n)^2) = O(n^2 2^{2n})$. Therefore, the time taken by SIMPLE-POWER can be bounded by $O(n^3 2^{2n})$. And the time taken by POWER is

$O(n^3 2^{3n})$ since the exponent could have as many as $n2^n$ bits. So the total time is $O(n^3 2^{3n})$. This is a correct solution, but too inefficient.

Solution 2 is much more efficient and makes use of the fact that p is prime. By Fermat's Little Theorem $x^{p-1} \bmod p = 1$ for $1 \leq x < p$. So we only need to compute the power y^z modulo $(p-1)$. This ensures that all calculations use numbers at most n bits. The running time of Solution 2 is $O(n^3) + O(n^3) = O(n^3)$.

Problem 2. (3 points) Prove or disprove: For any positive integers a and b with $a \geq b$,

$$\gcd(a+b, a-b) \geq \gcd(a, b).$$

(If you are proving the statement, a 2-3 sentence proof in English should suffice. If you are disproving the statement, you need to give values for a and b for which the above claim is not true.)

Answer: The claim is true. If $a = b$, the claim holds since $\gcd(a+b, a-b) = \gcd(2a, 0) = 2a > \gcd(a, a)$.

If $a > b$, suppose we denote $\gcd(a, b)$ by t. So we have

$$a = x \cdot t, \quad b = y \cdot t$$

in which x and y are all integers. Then we have

$$a + b = (x + y)t, \quad a - b = (x - y)t$$

So t is a common divisor of $(a + b)$ and $(a - b)$, which implies

$$\gcd(a+b, a-b) \geq t = \gcd(a, b).$$