Algorithms – CS-27200/CS-37000

Pseudocodes for basic algorithms in Number Theory: the Euclidean algorithm and Repeated squaring

Problem 1. Calculate the g.c.d. of two positive integers, $a \geq b \geq 0$.

Solution: the Euclidean algorithm.

Pseusocode 1A.

```
Initialize: A := a, B := b
    while B ≥ 1 do
        division: A = Bq + R, 0 ≤ R ≤ B − 1
        A := B, B := R
    end(while)
return A
```

The **correctness** of the algorithm follows from the following *loop invariant:*

$$\text{g.c.d.}(A, B) \ = \text{g.c.d.}(a, b).$$

(In addition, at the end we use the fact that g.c.d.$(A, 0) = A$.)

The **efficiency** of the algorithm follows from the observation that after every two rounds, the value of B is reduced to less than half. (Prove!) This implies that the number of rounds is $\leq 2n$ where n is the number of binary digits of b. Therefore the total number of bit-operations is $O(n^3)$, so this is a *polynomial-time algorithm.* (Good job, Euclid!)

Pseusocode 1B: recursive.

```
procedure g.c.d.(a, b)    (a ≥ b ≥ 0)
    if b = 0 then return a
    else division: a = bq + r, 0 ≤ r ≤ b − 1
        return g.c.d.(b, r)
```

(This code does not require a separate analysis except to clarify that it encodes the same algorithm. Clarify!) (OVER)

Problem 2. Calculate $a^b \bmod m$ where a, b, m are integers, $a, m \geq 1$, $b \geq 0$.

Solution: the method of repeated squaring.

Pseusocode 2A.

Initialize: $X := 1$, $B := b$, $A = (a \bmod m)$
    while $B \geq 1$ **do**
        if B odd **then** $B := B - 1$, $X := (AX \bmod m)$
        else $B := B/2$, $A := (A^2 \bmod m)$
    end(**while**)
return X

The **correctness** of the algorithm follows from the following *loop invariant:*

$$XA^B \equiv a^b \bmod m.$$

The **efficiency** of the algorithm follows from the observation that after every two rounds, the value of B is reduced to less than half. (Prove!) This implies that the number of rounds is $\leq 2n$ where n is the number of binary digits of b. Moreover, we never deal with integers greater than m. Therefore the total number of bit-operations is $O(n(\log m)^2) \leq O((\log a + \log b + \log m)^3)$, so this is a *polynomial-time algorithm:* the length of the input is the total number of bits of a, b, m, which is $\approx \log a + \log b + \log m$.

Pseusocode 2B: recursive.

procedure $f(a, b, m) = (a^b \bmod m)$ $(b \geq 0, a, m \geq 1)$
    if $b = 0$ **then return** 1
    elseif b odd **then return** $a \cdot f(a, b-1, m) \bmod m$
    elseif b even **then return** $f((a^2 \bmod m), b/2, m)$

(This code does not require a separate analysis except to clarify that it encodes the same algorithm. Clarify!)

Note. For both problems, the explicit (nonrecursive) versions of the algorithms are preferable.