Algorithms – CMSC-27200

Repeated squaring

Instructor: László Babai

Last updated: 02-16-2015

Let a, m be integers, $m \neq 0$. Recall that $(a \bmod m)$ denotes the smallest non-negative remainder of the division of a by m. In other words, let $a = mq+r$ where $0 \le r \le |m|-1$. This r is unique and is denoted $r = (a \bmod m)$.

Problem (modular exponentiation): Calculate $(a^b \bmod m)$ where a, b, m are integers, $a, m \ge 1$, $b \ge 0$.

Solution: the method of repeated squaring.

Pseudocode A.

Initialize: $X := 1$, $B := b$, $A =: (a \bmod m)$
[X is the "accumulator" that collects the partial results]
while $B \ge 1$ **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* (verify!)

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

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 $\le 2n$ where n is the number of bits (binary digits) of b. Moreover, we never deal with integers greater than m^2. Therefore, if a, b, m each have n bits (initial zeros permitted) then every number involved has $\le 2n$ bits and the total number of bit-operations is $O(n^3)$ (using the schoolbook multiplication/division method) so this is a *polynomial-time algorithm.* (Recall that the comparison is made with the bit-length of the input, which in this case is $3n$.)

We now describe an alternative, recursive implementation. The non-recursive code is preferred.

Pseudocode B: recursive.

procedure $f(a, b, m) = (a^b \bmod m)$ $(b \ge 0, a, m \ge 1)$
    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** $f((a^2 \bmod m), b/2, m)$