CS60082/CS60094 Computational Number Theory, Spring 2009

Mid-semester examination: Solutions

[*This test is open-notes. Answer all questions. Be brief and precise.*]

1 Let $a, b \in \mathbb{N}$ with $\gcd(a, b) = 1$. Assume that $a \neq 1$ and $b \neq 1$.

(a) Prove that there exist integers u, v such that $ua + vb = 1$ with $|u| < b$ and $|v| < a$. **(5)**

Solution Extended gcd calculations yield integers u, v with $ua + vb = 1$. For any $q \in \mathbb{Z}$, we then have $(u - qb)a + (v + qa)b = 1$. Euclidean division of u by b gives $u = qb + u'$ with quotient q and remainder u'. Let us denote $v' = v + qa$ for this particular value of q. But then $u'a + v'b = 1$. We already have $|u'| < b$ (since it is the remainder of Euclidean division by b), so the equation $v' = (1 - u'a)/b$ implies that $|v'| \leqslant \frac{1}{b}(1 + |u'|a) \leqslant \frac{1}{b}(1 + (b-1)a) = a - \frac{a-1}{b} < a$, since $a \neq 1$.

(b) Prove that any integer $n \geqslant ab$ can be expressed as $n = sa + tb$ with integers $s, t \geqslant 0$. **(5)**

Solution We may proceed by induction on n. For $n = ab$, we have $n = b \times a + 0 \times b$. Now, take any $n \geqslant ab$, and assume that $n = sa + tb$ for some integers $s, t \geqslant 0$. By Bézout's theorem, we have $1 = ua + vb$ for some integers u, v. Summing up gives $n + 1 = (s + u)a + (t + v)b = (s + u - qb)a + (t + v + qa)b$ for any $q \in \mathbb{Z}$. Take $q = (s + u) \operatorname{quot} b$, $s' = (s + u) \operatorname{rem} b = (s + u) - qb$, and $t' = t + v + qa$. We then have $n + 1 = s'a + t'b$ with $0 \leqslant s' < b$. If $t' < 0$, we have $n + 1 < s'a < ab$, a contradiction. Thus, $t' \geqslant 0$ too.

(c) Devise a polynomial-time (in $\log n$) algorithm to compute s and t of Part (b). **(5)**

Solution There is another way of proving Part (b), namely, start with $1 = ua + vb$ with $|u| < b$ and $|v| < a$. But then $n = (nu)a + (nv)b$. Replace nu by $nu - qb$, where $s = nu - qb$ is the remainder of Euclidean division of nu by b. Taking $t = nv + qa$ gives $n = sa + tb$ with $s, t \geqslant 0$. This leads to the following algorithm.

Use the extended gcd algorithm on a, b to compute u, v with $|u| < b$.
Take $s = (nu) \operatorname{rem} b$.
Take $t = (n - sa)/b$.

(d) Determine the running time of your algorithm. **(5)**

Solution First, note that $a < n$ and $b < n$, that is, the sizes of a and b are bounded by the size of n. The computation of u, v by the extended gcd algorithm takes a running time of $O(\log^3 n)$ (Euclidean gcd) or $O(\log^2 n)$ (binary gcd). Since $|u| < b < n$, the product nu and the subsequent reduction modulo b can be done in $O(\log^2 n)$ time. Finally, the computation of t involves multiplication, subtraction and division on operands of sizes $O(\log n)$ and can again be performed in $O(\log^2 n)$ time.

(**Remark:** The *Frobenius coin change problem* deals with the determination of the largest positive integer that cannot be represented as a linear non-negative integer combination of some given positive integers $a_1, a_2, \ldots, a_k$ with $\gcd(a_1, a_2, \ldots, a_k) = 1$. For $k = 2$, this integer is $a_1a_2 - a_1 - a_2$.)

2 Let $n \in \mathbb{N}$. Suppose that we want to compute $x^r y^s \pmod n$, where r and s are positive integers of the same bit size. By using the repeated square and multiply algorithm, one can compute $x^r \pmod n$ and $y^s \pmod n$ independently, and then multiply these two values. Alternatively, one may rewrite the square and multiply algorithm using only one loop in which the bits of both the exponents r and s are simultaneously considered. After each square operation, one multiplies by 1, x, y, or xy.

(a) Elaborate the algorithm outlined above. **(5)**

Solution

Let $r = (r_{l-1} r_{l-2} \ldots r_1 r_0)_2$ and $s = (s_{l-1} s_{l-2} \ldots s_1 s_0)_2$.
Precompute $xy \pmod n$.
Initialize $prod = 1$.
For $i = l-1, l-2, \ldots, 1, 0$ {
  Set $prod = prod^2 \pmod n$.
  If ($r_i = 1$) and ($s_i = 1$), set $prod = prod \times (xy) \pmod n$,
  else if ($r_i = 1$) and ($s_i = 0$), set $prod = prod \times x \pmod n$,
  else if ($r_i = 0$) and ($s_i = 1$), set $prod = prod \times y \pmod n$.
}

(b) What speedup is this modification expected to produce? **(5)**

Solution The modified algorithm reduces the number of square operations to half of that performed by two independent calls of the repeated square and multiply algorithm. The number of products, however, depends on the bit pattern of the exponents r and s. For random exponents, about half of the bits are one, that is, two exponentiations make about l modular multiplications. The modified algorithm skips the multiplication only when both the bits are 0—an event having a probability of $1/4$ for random exponents. Thus, the number of multiplications done by the modified algorithm is expected to be close to $0.75l$. The precomputation involves only one modular multiplication and so has negligible overhead.

(c) Generalize the concept to the computation of $x^r y^s z^t \pmod n$, and analyze the speedup. **(5)**

Solution

Let $r = (r_{l-1} r_{l-2} \ldots r_1 r_0)_2$, $s = (s_{l-1} s_{l-2} \ldots s_1 s_0)_2$, and $t = (t_{l-1} t_{l-2} \ldots t_1 t_0)_2$.
Precompute $xy, xz, yz, xyz \pmod n$.
Initialize $prod = 1$.
For $i = l-1, l-2, \ldots, 1, 0$ {
  Set $prod = prod^2 \pmod n$.
  If $((r_i, s_i, t_i) = (1,1,1))$ set $prod = prod \times (xyz) \pmod n$,
  else if $((r_i, s_i, t_i) = (1,1,0))$ set $prod = prod \times (xy) \pmod n$,
  else if $((r_i, s_i, t_i) = (1,0,1))$ set $prod = prod \times (xz) \pmod n$,
  else if $((r_i, s_i, t_i) = (0,1,1))$ set $prod = prod \times (yz) \pmod n$,
  else if $((r_i, s_i, t_i) = (1,0,0))$ set $prod = prod \times x \pmod n$,
  else if $((r_i, s_i, t_i) = (0,1,0))$ set $prod = prod \times y \pmod n$,
  else if $((r_i, s_i, t_i) = (0,0,1))$ set $prod = prod \times z \pmod n$.
}

The number of square operations performed by this modified algorithm is one-third of that made by three independent square and multiply exponentiations. For random exponents, the expected count of modular multiplication operations is $1.5l$ for three exponentiations, and $0.875l$ for our modified algorithm.

(**Remark:** Computation of elements of the form $x^r y^s \pmod n$ is quite common in cryptosystems based on the discrete logarithm problem. Making this computation faster is, therefore, useful in cryptography.)

3 (a) Prove that the polynomial $x^2 + x + 2$ is irreducible modulo 3. **(5)**

Solution Since $x^2 + x + 2$ is a quadratic polynomial, it is irreducible if and only if it does not have a root. But $0^2 + 0 + 2 \equiv 2 \not\equiv 0 \pmod 3$, $1^2 + 1 + 2 \equiv 1 \not\equiv 0 \pmod 3$ and $2^2 + 2 + 2 \equiv 2 \not\equiv 0 \pmod 3$.

Represent $\mathbb{F}_9$ as $\mathbb{F}_3(\theta)$, where $\theta^2 + \theta + 2 = 0$.

(b) Find the roots of $x^2 + x + 2$ in $\mathbb{F}_9$. **(5)**

Solution By construction, θ itself is a root of $x^2 + x + 2$. Its other root is $\theta^3 = -\theta(\theta + 2) = -\theta^2 - 2\theta = \theta + 2 - 2\theta = -\theta + 2 = 2\theta + 2$.

(c) Find the roots of $x^2 + x + 2$ in $\mathbb{Z}_9$. **(5)**

Solution Since $x^2 + x + 2$ is irreducible over $\mathbb{Z}_3$, it has no roots modulo 3 and so no roots modulo $3^2 = 9$ too.

(d) Prove that θ is a primitive element of $\mathbb{F}_9$. **(5)**

Solution The order of $\mathbb{F}_9^*$ is $9 - 1 = 8 = 2^3$. We have $\theta^4 = (\theta + 2)^2 = \theta^2 + \theta + 1 = -\theta - 2 + \theta + 1 = -1 = 2 \neq 1$, that is, θ is a primitive element of $\mathbb{F}_9$.

(e) Prove that the polynomial $y^2 - \theta$ is irreducible over $\mathbb{F}_9$. **(5)**

Solution Suppose not. Then, it has one root α (in fact, both the roots) in $\mathbb{F}_9$, that is, $\theta = \alpha^2$. But then $\theta^4 = \alpha^8 = 1$ (since $\alpha \in \mathbb{F}_9^*$), that is, θ is not a primitive element of $\mathbb{F}_9$, a contradiction.

Represent $\mathbb{F}_{81}$ as $\mathbb{F}_9(\psi)$, where $\psi^2 - \theta = 0$.

(f) Determine whether ψ is a primitive element of $\mathbb{F}_{81}$. **(5)**

Solution We have $\psi^{16} = (\psi^2)^8 = \theta^8 = 1$, that is, ψ is not a primitive element of $\mathbb{F}_{81}$.

(g) Find the minimal polynomial of ψ over $\mathbb{F}_3$. **(5)**

Solution The conjugates of ψ over $\mathbb{F}_3$ are ψ, $\psi^3 = \theta\psi$, $\psi^9 = \theta^3\psi^3 = \theta^4\psi = 2\psi$ and $\psi^{27} = 8\psi^3 = 2\theta\psi$. Therefore, the minimal polynomial of ψ over $\mathbb{F}_3$ is

$$\begin{aligned}
& (x - \psi)(x - \theta\psi)(x - 2\psi)(x - 2\theta\psi) \\
= \; & (x - \psi)(x + \psi)(x - \theta\psi)(x + \theta\psi) \\
= \; & (x^2 - \psi^2)(x^2 - \theta^2\psi^2) \\
= \; & (x^2 - \theta)(x^2 - \theta^3) \\
= \; & x^4 - (\theta + \theta^3)x^2 + \theta^4 \\
= \; & x^4 - (\theta + 2\theta + 2)x^2 + 2 = x^4 - 2x^2 + 2 = x^4 + x^2 + 2.
\end{aligned}$$

There are other ways of arriving at this polynomial. First, note that $\theta^2 + \theta + 2 = 0$ and $\psi^2 = \theta$. Combining these two equations gives $\psi^4 + \psi^2 + 2 = 0$, that is, ψ is a root of the polynomial $x^4 + x^2 + 2 \in \mathbb{F}_3[x]$. The degree of ψ (over $\mathbb{F}_3$) is 4, so $x^4 + x^2 + 2$ has to be irreducible modulo 3. Finally, since ψ cannot satisfy two different monic irreducible polynomials in $\mathbb{F}_3[x]$ of degree 4, the minimal polynomial of ψ over $\mathbb{F}_3$ has to be $x^4 + x^2 + 2$.