Math 1580 – Problem Set 3. Due Friday Sep. 30, 4pm

Problem 1. Show that the following identities hold in big O notation:

(a) Show that

$$x^2 + \sqrt{x} + \ln x = \mathcal{O}(x^2)$$

More generally, show that if $f_1(x), \ldots, f_k(x) = \mathcal{O}\big(g(x)\big)$, then

$$f_1(x) + \cdots + f_k(x) = \mathcal{O}\big(g(x)\big).$$

(b) For any integer $N \geq 0$ and any real $\epsilon > 0$,

$$(\ln x)^N = \mathcal{O}(x^\epsilon)$$

(In particular, N can be arbitrarily large and ϵ can be arbitrarily small.)

(c) Logarithms satisfy

$$\ln(x^a) = \mathcal{O}\big(\ln(x^b)\big) \text{ for any } a, b \in \mathbb{R}, \text{ and}$$
$$\log_a x = \mathcal{O}(\log_b x) \text{ for any } a, b \in \mathbb{R}.$$

(d) In constrast to logarithms, exponentials satisfy

$$e^{c_1 x} = \mathcal{O}\left(e^{c_2 x}\right) \text{ if and only if } c_1 \leq c_2, \text{ and}$$
$$a^x = \mathcal{O}(b^x) \text{ if and only if } a \leq b.$$

(e) Show that

$$x^N e^x = \mathcal{O}\big(e^{cx}\big)$$

for any integer $N \geq 0$ and $c > 1$.

Problem 2. Let $a_1, a_2, \ldots, a_k$ be integers with $\gcd(a_1, a_2, \ldots, a_k) = 1$ (the largest positive integer dividing all of $a_1, \ldots, a_k$ is 1). Prove that the equation

$$a_1 u_1 + a_2 u_2 + \cdots + a_k u_k = 1$$

has a solution in integers $u_1, \ldots, u_k$. (Hint: repeatedly apply the exteded Euclidean algorithm. You may find it easier to prove a more general statement in which $\gcd(a_1, \ldots, a_k)$ is allowed to be larger than 1.)

Problem 3. Use Shanks's babystep–giantstep method to solve the following discrete logarithm problems:

(a) $2^x = 13$ in $\mathbb{F}_{23}$. (You computed this by brute force on the last homework.)

(b) $11^x = 21$ in $\mathbb{F}_{71}$.

Problem 4. Use the Pohlig-Hellman algorithm to solve the discrete logarithm problems $g^x = a$ in $\mathbb{F}_p$, where

(a) $p = 433, \ g = 7, \ a = 166.$

(b) $p = 746497, \ g = 10, \ a = 243278.$

(You may wish to use a calculator with a `mod` operation to speed up your intermediate computations. If you do not have one, you can enter "`n` mod m" into Google and it will return $n \pmod{m}$.)