Solutions to Daily Due Sep 9 (AI generated)

Math 4440/5440, Coding and Cryptography, Fall 2026

Generated by Claude for Katherine E. Stange
(definitions and conventions follow the Fall 2024 lecture notes of Sept 20 and 23;
all computations computer-verified)

Problem 3: is $f = O(g)$? Is $g = O(f)$?

Recall the definition. For functions $f, g\colon [0,\infty) \to \mathbb{R}$ with g eventually positive, we say $f = O(g)$ if there are constants $C > 0$ and $M > 0$ such that

$$|f(x)| \le C\, g(x) \qquad \text{for all } x > M.$$

So to show $f = O(g)$ we exhibit C and M; to show $f \neq O(g)$ we show that no choice of C and M can work, usually by finding arbitrarily large x where the inequality fails.

(a) $f(x) = |\sin x|$ and $g(x) = 1/2$

$f = O(g)$**: yes.** Since $|\sin x| \le 1 = 2 \cdot \frac{1}{2}$ for every x, the definition holds with $C = 2$ and any M (say $M = 1$).

$g = O(f)$**: no.** We would need C and M with $\frac{1}{2} \le C\,|\sin x|$ for all $x > M$. But $\sin(k\pi) = 0$ for every integer k, and there are integers k with $k\pi > M$ no matter how large M is. At such an $x = k\pi$ the inequality reads $\frac{1}{2} \le 0$, which is false. So no C, M work.

(The moral: O is only an *upper* bound. A bounded function is $O(1)$, but a constant need not be O of a function that keeps returning to zero.)

(b) $f(x) = 2^x$ and $g(x) = 3^x$

$f = O(g)$**: yes.** For $x \ge 0$ we have $2^x \le 3^x$, so take $C = 1$ and $M = 1$.

$g = O(f)$**: no.** We would need $3^x \le C \cdot 2^x$ for all $x > M$, that is, $(3/2)^x \le C$ for all $x > M$. But $(3/2)^x \to \infty$ as $x \to \infty$: explicitly, for $x > \log_{3/2} C$ we have $(3/2)^x > C$. So no constant C can work, whatever M is.

(The moral: exponentials with different bases are *not* interchangeable in big-oh, unlike constant factors or lower-order terms. This is why the base matters when we say an algorithm is exponential.)

Problem 4: Baby-Step Giant-Step for $34 = 3^x \pmod{113}$

Here $p = 113$, $g = 3$ and $h = 34$; a computer confirms that 3 is a primitive root modulo 113, so x is determined modulo $p - 1 = 112$.

Step size. Take $N = \lceil \sqrt{p-1} \rceil = \lceil \sqrt{112} \rceil = 11$, since $10^2 = 100 < 112 \le 121 = 11^2$. Any x with $0 \le x < 112$ can then be written as $x = j + 11k$ with $0 \le j \le 10$ and $0 \le k \le 10$ ("base 11"), which is what the two lists below are designed to detect.

Baby steps. Multiply by $g = 3$ repeatedly, reducing modulo 113 each time. Each entry is 3 times the previous one.

j	0	1	2	3	4	5	6	7	8	9	10	11
$3^j \bmod 113$	1	3	9	27	81	17	**51**	40	7	21	63	76

For instance $3^5 = 3 \cdot 81 = 243 = 2 \cdot 113 + 17 \equiv 17$, and $3^6 = 3 \cdot 17 = 51$.

The giant-step multiplier. We need $g^{-N} = 3^{-11} \pmod{113}$. From the baby list, $3^{11} \equiv 76$, so $3^{-11} \equiv 76^{-1} \pmod{113}$. Sage gives $76^{-1} \equiv 58$ (or find it by the extended Euclidean algorithm); check: $76 \cdot 58 = 4408 = 39 \cdot 113 + 1$. So

$$g^{-N} \equiv 58 \pmod{113}.$$

Giant steps. Start at $h = 34$ and multiply by $g^{-N} = 58$ repeatedly, reducing modulo 113. The kth entry is $h \cdot g^{-Nk} = 34 \cdot 58^k$.

k	0	1	2	3	4	5	6	7	8	9	10	11
$34 \cdot 58^k \bmod 113$	34	**51**	20	30	45	11	73	53	23	91	80	7

For instance $34 \cdot 58 = 1972 = 17 \cdot 113 + 51 \equiv 51$, and $51 \cdot 58 = 2958 = 26 \cdot 113 + 20 \equiv 20$. (In practice one stops at the first collision, which here is already at $k = 1$; the full list is shown because the tool page prints it.)

The collision. The value 51 appears in both lists: as 3^6 in the baby list ($j = 6$) and as $34 \cdot 3^{-11 \cdot 1}$ in the giant list ($k = 1$). Therefore

$$3^6 \equiv 34 \cdot 3^{-11} \pmod{113} \qquad \Longrightarrow \qquad 3^{6+11} \equiv 34 \pmod{113},$$

so $x = j + Nk = 6 + 11 \cdot 1 = \mathbf{17}$.

Check. $3^{17} = 3^{11} \cdot 3^6 \equiv 76 \cdot 51 = 3876 = 34 \cdot 113 + 34 \equiv 34 \pmod{113}$. ✓

A second collision. The value 7 also appears in both lists: $3^8 \equiv 7$ ($j = 8$) and $34 \cdot 58^{11} \equiv 7$ ($k = 11$). This gives $x = 8 + 11 \cdot 11 = 129$, and indeed $129 \equiv 17 \pmod{112}$: the same answer, as it must be, since the discrete log is only defined modulo the order of g, which is 112.

Runtime. The two lists have about $N \approx \sqrt{p}$ entries each, so the algorithm takes $O(\sqrt{p})$ multiplications, compared with $O(p)$ for exhaustive search. Both are exponential in the input size $\log_2 p$, but $\sqrt{p} = 2^{\frac{1}{2}\log_2 p}$ is a big improvement in practice.