

CS344: Design and Analysis of Computer Algorithms

Homework 2

Group Members: Stephen Kuo, Derek Mui

1.11) Is $4^{1536} - 9^{4824}$ divisible by 35?

Answer:

$$4^{1536} = (4^3)^{512} = 64^{512} = (35 + 29)^{512}$$

Since 35 is divisible by 35, we can continue with 29^{512}

$$29^{512} = (29^2)^{256} = 841^{256} = (840 + 1)^{256}$$

Since 840 is divisible by 35, we are left with $1^{256} \bmod 35$

$$9^{4824} = (9^2)^{2412} = 81^{2412} = (70 + 11)^{2412}$$

Since 70 is divisible by 35, we can continue with 11^{2412}

$$(11^3)^{804} = 1331^{804} = (1330 + 1)^{804}$$

Since 1330 is divisible by 35, we are left with $1^{804} \bmod 35$

$$(1^{256} \bmod 35) - (1^{804} \bmod 35) = 0$$

\therefore Yes it is divisible by 35

1.12) What is $2^{2^{2006}} \bmod 3$

Answer:

$$2^{2^{2006}} = 4^{2006} = 4^{2^{1003}} = 16^{1003} = (15 + 1)^{1003}$$

We know 15 is divisible by 3, so that leaves us with 1^{1003} .

\therefore Answer is 1

1.13) Is the difference of $5^{30,000}$ and $6^{123,456}$ a multiple of 31?

Answer:

$$5^{30000} = 5^{3^P 10000} = 125^{10000} = (124 + 1)^{10000}$$

Since 124 is divisible by 31, we are left with $1^{10000} \bmod 31$

$$6^{123456} = 6^{2^6 1728} = 36^{61728} = (31 + 5)^{61728}$$

Since 31 is divisible by 31, we are left with 5^{61728}

$$5^{61728} = 5^{3^{20576}} = 125^{20576} = (124 + 1)^{20576}$$

Since 124 is divisible by 31, that leaves us with 1^{20576}

$$(1^{10000} \bmod 31) - (1^{20576} \bmod 31) = 0$$

\therefore Yes, the difference is a multiple of 31

1.25) calculate $2^{125} \bmod 127$ using any method you choose

Answer:

$$2^{125} = (2^{119} * 2^6) = (2^{7^{17}} * 2^6) = (128^{17} * 2^6) = (127 + 1)^{17} * 2^6$$

Because 127 is divisible by 127, that leaves us with $1^{17} * 2^6$

$$1^{17} * 2^6 = 1 * 2^6 = 64$$

\therefore Answer is 64

1.33) Give an efficient algorithm to compute the least common multiple of two n -bit numbers x and y , that is, the smallest number divisible by both x and y . What is the running time of your algorithm as a function of n ?

Answer:

$(x * y)$ (include the shiftings) If x and y are both n bits, then there are n intermediate rows.
 So, $O(n) + O(n) + \dots + O(n)$ done $(n - 1)$ times yields $O(n^2)$
 Divide the same (include the shiftings) $= O(n^2)$

$\gcd(x, y) = O(n^3)$ initially n -bit integers, base case.
 Each quadratic - time division reaches within $2n$ recursive calls.
 \therefore Total $= O(n^3)$

1.39) Give a polynomial-time algorithm for computing $a^{b^c} \bmod p$, given a, b, c , and prime p .

Answer:

Fermat's little theorem: $(a^{p-1} \bmod p) \equiv 1$
 Size of input suggestion: $\log b + \log a + \log c + \log p$

$$a^{b^c} \bmod p = a^{b^{c \bmod (p-1)}} \bmod p$$

$$b \bmod (p-1)$$

$$O(\log b) * O(\log p) \Rightarrow O(\log b \log p)$$

By using repeat square to compute $(b^c \bmod (p-1))$
 $O(\log c)$ number of size almost p . Multiplication takes $O(\log^2 p)$ times

$$\therefore O(\log c \log^2 p)$$

Since $b^c \bmod (p-1) < p$, computing $(a^{b^c} \bmod p)$ with repeated squaring will get us $O(\log p \log^2 p) = O(\log^3 p)$

$$\therefore O(\log b \log p + \log c \log^2 p + \log a \log p + \log^3 p) = O(n^3)$$

Problem)
Answer: