Useful Commands:

Range[n]: gives a list $\{1, 2, 3, 4, \ldots, n\}$

Complement[list1, list2]: gives a list of all elements of list1 that are <u>not</u> in list2

Modular Arithmetic

"clock arithmetic" — divide by a modulus and keep the remainder

examples:

$17 \pmod 5$ = remainder when you divide 17 by 5

$= 2$

$34 \pmod 3 = 1$

$174 \pmod 5 = 4$

We want to compute: $b^e \pmod m$

where b, e, m are large numbers — say, 30 digits each

(b: base, e: exponent, m: modulus)

Approach 1: Compute b^e and then take the remainder mod m

If b, e are 30 digits each, then how big is b^e?

$b^2 \approx 60$ digits

$b^3 \approx 90$ digits

$b^4 \approx 120$ digits

!

b^e wow!

$b^e = b^{\text{30-digit number}}$

b^e has about $30e$ digits

$\hookrightarrow 30e \approx 10^{31}$

b^e has about 10^{31} digits

We don't have enough memory to store b^e.

Approach 2: Repeatedly multiply $b \cdot b \cdot b \cdots b$, reduce mod m after each multiplication.

PSEUDOCODE:

result = 1

loop: repeat e times ← Too slow if e is enormous!

result = result * b (mod m) ← result is always $< m$

return result

Approach 3: repeated squaring

example: compute 3^{64} 9^{32} 81^{16}

$3^2 = 9$, $3^4 = 3^2 \cdot 3^2 = 9 \cdot 9$, $3^8 = 3^4 \cdot 3^4$

$3^{16} = 3^8 \cdot 3^8 = (3^4)^4$ $3^{32} = 3^{16} \cdot 3^{16}$ $3^{64} = 3^{32} \cdot 3^{32}$

example: $3^{25} = 3^{16+8+1} = 3^{16} \cdot 3^8 \cdot 3^1$

$25 \pmod 2 = 1$

$\lfloor \frac{25}{2} \rfloor = 12 \pmod 2 = 0$

$\lfloor \frac{12}{2} \rfloor = 6 \pmod 2 = 0$

$\lfloor \frac{6}{2} \rfloor = 3 \pmod 2 = 1$

$\lfloor \frac{3}{2} \rfloor = 1 \pmod 2 = 1$

base 2 digits of 25

base 2 representation

$1\ 1\ 0\ 0\ 1$ (two) $= 25$ (ten)

sixteens, eights, fours, twos, ones; tens, ones

IDEA: compute $3, 3^2, 3^4, 3^8, 3^{16}, \ldots$ by repeated squaring. Then multiply the appropriate powers to get 3^{25}.