This is a postscript for the talk, with details on a smoother, cleaner way to keep track of the arithmetic involved in finding modular inverses via the GCD, and on how to find $M^E \pmod N$.

First, inverses. We start with positive integers a and b, with $a < b$, and we want further integers u and v so that $au + bv = 1$, if that is possible, and if not, to know that it is not.

Our first step is a quotient and remainder division of a into b, yielding integers q and r so that $0 \leq r < a$ and $qa + r = b$. Here, we must introduce some notation. A *matrix* is a rectangular array of numbers. For now, they'll be two by two arrays.

Given two such arrays, there is a *matrix multiplication* operation.

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} p & q \\ r & s \end{pmatrix} := \begin{pmatrix} ap + br & aq + bs \\ cp + dr & cq + ds \end{pmatrix}.$$

One can also multiply a two by two matrix with a single column, thus,

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} p \\ r \end{pmatrix} := \begin{pmatrix} ap + br \\ cp + dr \end{pmatrix}.$$

So what matrix multiplication does is it organizes and compiles a number of multiplication and addition operations, in one handy package. It turns out that matrix multiplication is associative and distributive but not commutative, so we have to keep in mind the order of multiplication.

Now the point of this here is that the equation $r = b - qa$ is the key ingredient in this matrix equation:

$$\begin{pmatrix} -q & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} r \\ a \end{pmatrix}.$$

When the GCD algorithm has several steps, we have several q's and r's, and we number them. And then the upshot of our several steps is that we have matrices

$$Q_j = \begin{pmatrix} -q_j & 1 \\ 1 & 0 \end{pmatrix}$$

and the results of our algorithm are all wrapped up in this statement:

$$Q_n Q_{n-1} \cdots Q_1 \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} 0 \\ d \end{pmatrix}$$

where d is the GCD of the original a and b. Now from one perspective this is just a matter of old wine in new bottles. But because matrix multiplication is associative, we can carry out the multiplication not by parenthesizing it like $Q_3(Q_2(Q_1 X))$ but instead like $(Q_3 Q_2 Q_1)X = AX$, say. The final two by two matrix $A = Q_n \cdots Q_1$, say

$$A = \begin{pmatrix} s & t \\ u & v \end{pmatrix}$$

has the property that $as + bt = 0$ and $au + bv = d$. And there, all wrapped up and tied in a bow, are the desired a and b.

Example: Starting with $a = 17$ and $b = 41$, we have $q_1 = 2$ because $41 = 2 * 17 + 7$, we have $q_2 = 2$ because $17 = 2 * 7 + 3$, we have $q_3 = 2$ because $7 = 2 * 3 + 1$, and we have $q_4 = 3$ because $3 = 3 * 1 + 0$.

This translates into $X = \binom{17}{41}$ and

$$AX = Q_4Q_3Q_2Q_1X = \begin{pmatrix} -3 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} -2 & 1 \\ 1 & 0 \end{pmatrix}^3 \begin{pmatrix} 17 \\ 41 \end{pmatrix} = \begin{pmatrix} 41 & -17 \\ -12 & 5 \end{pmatrix} \begin{pmatrix} 17 \\ 41 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

so in particular, $-12*17+5*41 = 1$. This makes -12 the modular inverse of 17 mod 41, or if you want a number in the range $[0, 40]$, $41 - 12 = 29$: $29 * 17 \equiv 1 \pmod{41}$.

Now we turn to the other claim, that it is easy to compute powers in modular arithmetic. Fermats little theorem says that $a^{p-1} \equiv 1 \pmod{p}$. That means that $17^{40} \equiv 1 \pmod{41}$. But 17^{40} is a big number. And even if we modded out as we went along, so that the numbers to be multiplied were always between 1 and 40, we would have to execute 16 multiplications to complete the check. But there's a better way.

We need only calculate some of the intermediate powers. We can work our way from 40 down to 1 by subtractions and halvings, thus, $40 \to 20 \to 10 \to 5 \to 4 \to 2 \to 1$.

We can work our way back up, then, via additions of 1, and doubling. This means that if we start with $a_1 = 17$, then $a_2 = 17^2 \pmod{41} = 2$, then $a_3 = a_2^2 = 4$, (fourth power) then $a_4 = 17 * a_3 \pmod{40} = 27$, (fifth power) then $a_5 = a_4^2 \pmod{41} = 32$, (tenth power) $a_6 = a_5^2 \pmod{41} = 40$, (that's the 20th power), and finally, $a_7 = 40^2 \pmod{41} = 1$ (40th power, as claimed). When the power to be computed is in the trillions instead of dozens, the relative savings are enormous. What would have been completely infeasible is not that big a deal, because the number of multiplications required is on the order of the number of binary digits in the power to be computed.

Here's a closing question. Now that we have a way to solve $xy \equiv 1 \pmod{p}$, maybe it would be interesting to look at the graph of y as a function of x. What might we see?

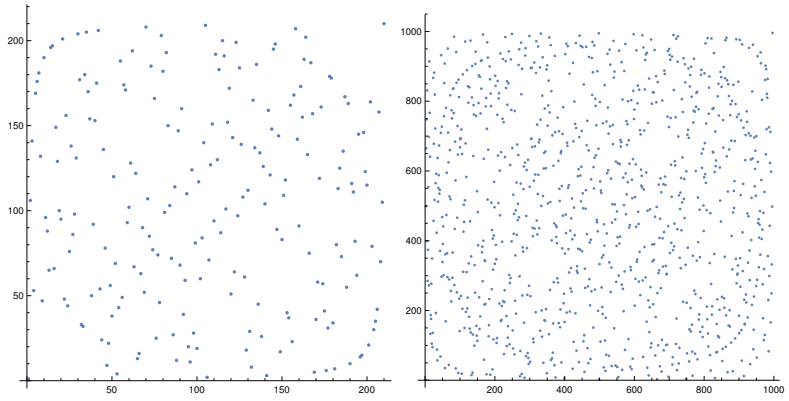

Figure 1: Graphs of y=1/x mod 211 and 997