An introduction to binary exponentiation

How would you calculate a modular power efficiently? In the days before scientific calculators, you knew one method instinctively: when you have a calculator that only does four functions plus squaring, you can still compute arbitrary powers by repeated squaring and multiplying.

For example, let's compute $12^{23} \mod 101$. Here's a sequence of steps that would accomplish the task:

(1) We notice that $23 = 16 + 4 + 2 + 1$ (we've computed a binary expansion of the exponent).

(2) We compute the successive squares of 12:

$$\begin{aligned} 12^1 &\equiv 12 \mod 101 \\ 12^2 &\equiv 43 \mod 101 \\ 12^4 &\equiv (12^2)^2 \equiv 43^2 \equiv 31 \mod 101 \\ 12^8 &\equiv (12^4)^2 \equiv 31^2 \equiv 52 \mod 101 \\ 12^{16} &\equiv (12^8)^2 \equiv 52^2 \equiv 78 \mod 101. \end{aligned}$$

(3) We put them together:

$$\begin{aligned} 12^{23} &\equiv 12^{16+4+2+1} \\ &\equiv 12^{16} \cdot 12^4 \cdot 12^2 \cdot 12^1 \\ &\equiv 78 \cdot 31 \cdot 43 \cdot 12 \\ &\equiv 95 \cdot 43 \cdot 12 \\ &\equiv 45 \cdot 12 \\ &\equiv 35 \mod 101 \end{aligned}$$

Here is a way to turn this into an algorithm.

Let n be a positive integer and let $x = e_1 e_2 \ldots e_r$ be an integer written in binary – for example, when $x = 23, e_1 = 1, e_2 = 0, e_3 = 1, e_4 = 1, e_5 = 1$. Here's how to compute $y^x \mod n$.

1. **INITIALIZE:** $k = 1, s_1 = 1$
2. If $e_k = 1$, let $r_k \equiv y s_k \mod n$. If $e_k = 0$, let $r_k = s_k$.
3. Let $s_{k+1} \equiv r_k^2 \mod n$.
4. If $k < r$, add 1 to k and go to Step 2.
5. If $k = r$, **RETURN** y.

Practice Exercise 1. Execute the above algorithm for $y = 12, x = 23$, and $n = 101$. Show all the steps.

Practice Exercise 2. Execute the above algorithm for $y = 3, x = 83$, and $n = 457$. Show all the steps.