

The RSA algorithm requires modular exponentiation; so, it would be nice to have an efficient algorithm for doing that. Here is a small example that demonstrates the algorithm. Calculate $151^{237} \bmod 1071$. We do not want to first calculate 151^{237} and then mod by 1071. The exponentiation could result in extremely large integers, and there is no reason that that has to happen. Here is a better scheme.

Calculate the binary expansion of the exponent 237. 237 is 11101101 in binary. (The bin command on the TI-92, for example, will do this.) This means that the exponent

$$237 = 128 + 64 + 32 + 8 + 4 + 1$$

So,

$$151^{237} = 151^{128+64+32+8+4+1} = 151^{128} \times 151^{64} \times 151^{32} \times 151^8 \times 151^4 \times 151^1$$

All, mod 1071.

Next we calculate 151 raised to the 1, 4, 8, 32, 64, and 128 mod 1071. Actually, to get an easier algorithm we will calculate 151 raised to each of the powers of 2 up to 128: 1, 2, 4, 8, 16, 32, 64, and 128 mod 1071.

	Mod 1071
151^1	151
151^2	310
$151^4 = 310^2$	781
$151^8 = 781^2$	562
$151^{16} = 562^2$	970
$151^{32} = 970^2$	562
$151^{64} = 562^2$	970
$151^{128} = 970^2$	562

So,

$$151^{237} = 562 \times 970 \times 562 \times 562 \times 781 \times 151 \bmod 1071$$

Taking these factors two at a time mod 1071, we get:

$$\begin{aligned}151^{237} &= 562 \times 970 \times 562 \times 562 \times 781 \times 151 \bmod 1071 \\&= 1 \times 562 \times 562 \times 781 \times 151 \bmod 1071 \\&= 562 \times 562 \times 781 \times 151 \bmod 1071 \\&= 970 \times 781 \times 151 \bmod 1071 \\&= 373 \times 151 \bmod 1071 \\&= 631 \bmod 1071\end{aligned}$$

In *Mathematica*, this algorithm is implemented as `PowerMod`. For example, **`PowerMod[151, 237, 1071]`**