Euler's Theorem

Reading

• Section 9.3

Practice Problems

9.3 1-12, 15-18

Notes

Euler's Theorem is an extension of Fermat's Little Theorem to the case of non-prime numbers.

Euler's Theorem

Euler's Theorem

If a is relatively prime to n, then:

$$a^{\phi(n)} \equiv 1 \bmod n$$

Since when n is prime we have $\phi(n)=n-1$, this reverts back to Fermat's theorem.

The proof is very similar to the one for Fermat's:

- Consider the set of reduced residues (those relatively prime to n). Let us denote them by $b_1, b_2, b_3, \ldots, b_{\phi(n)}$.
- ullet Because a is also a reduced residue, multiplying those reduced residues by a gives back reduced residues.
- ullet Multiplying by the (invertible) a is also 1-1, so it will simply permute them.
- We can then multiply them together: $(ab_1)(ab_2)\cdots(ab_{\phi(n)})=b_1b_2\cdots b_{\phi(n)}$ in \mathbb{Z}_n .
- Pulling the *a*'s out, we would have: $a^{\phi(n)}b_1b_2\cdots b_{\phi(n)}=b_1b_2\cdots b_{\phi(n)}$
- The *b*'s are invertible, so we can cancel them out from both sides.
- So we end up with $a^{\phi(n)} = 1$.

Repeated Squaring for fast exponentiation

It is time to discuss how to quickly compute powers of a number. The trick here is repeated squaring. But first:

If *a* is relatively prime to *n*, and $k \equiv r \mod \phi(n)$, then:

$$a^k \equiv a^r \mod n$$

So Euler's and Fermat's theorems can considerably lessen the work involved in computing $a^k \mod n$ by reducing the power k to a number less than $\phi(n)$. This might still be a very high number however. This is where repeated squaring comes in.

Let us suppose for example that we want to compute $6^{91} \mod 715$. The trick to answer that is the **binary representation** of 91: We can find that representation by looking at the powers of 2 and taking one power out at a time:

$$91 = 64 + 27 = 64 + 16 + 11 = 64 + 16 + 8 + 3 = 64 + 16 + 8 + 2 + 1$$

Based on this, we can tell that:

$$6^{91} = 6^{64+16+8+2+1} = 6^{64}6^{16}6^86^26^1$$

So if we have computed 6 raised to those powers of 2, we can put those answers together to get the answer for all other powers. Let us compute them:

$$6^{1} = 6$$

$$6^{2} = 36$$

$$6^{4} = 36^{2} = 1296 \equiv 581$$

$$6^{8} = 581^{2} = 337561 \equiv 81$$

$$6^{16} = 81^{2} = 6561 \equiv 126$$

$$6^{32} = 126^{2} = 15876 \equiv 146$$

$$6^{64} = 146^{2} = 21316 \equiv 581$$

As we compute those, we would at the same time multiply those that we need:

$$6^{1} = 6$$

$$6^{2} \cdot 6^{1} = 36 \cdot 6 = 216$$

$$6^{8} \cdot 6^{2} \cdot 6^{1} = 81 \cdot 216 = 17496 \equiv 336$$

$$6^{16} \cdot 6^{8} \cdot 6^{2} \cdot 6^{1} = 126 \cdot 336 = 42336 \equiv 151$$

$$6^{64} \cdot 6^{16} \cdot 6^{8} \cdot 6^{2} \cdot 6^{1} = 581 \cdot 151 = 87731 \equiv 501$$

So that's our final answer, $6^{91} \equiv 501 \mod 715$.

This may have been a lot of work, but let us count the operations: We had to do 7 multiplications to compute 6 raised to the powers of 2, and at most another 7 multiplications to combine those powers to get our answer. So a total of $2 \times 7 = 14$ multiplications (instead of 91).

Exponentiation via repeated squaring requires $2\log_2(k)$ multiplications.

This scales very well as the power k grows.

For completeness, let us describe an algorithm for carrying the fast exponentiation out. Those of you with programming inclinations should try implement the algorithm.

Algorithm for fast exponentiation

Inputs: n, k, aOutput: $a^k \mod n$ Local variables:

- "prod" accumulates the final product
- " \bar{b} " keeps track of the power a^{2^x} as we compute each new one

Steps:

- Initialize: prod = 1, b = a
- Repeat:
 - If a = 0 we are done. Return "prod".
 - Divide a by 2: a=2q+r. (You wouldn't need to do this step as is, can rely on using the operators for mod and div. But conceptually it happens.)
 - **-** If r = 1 we need to use this b. So $prod \leftarrow prod \cdot b \mod n$.
 - $b \leftarrow b \cdot b \mod n$.
 - $-a \leftarrow q$.