Pollard's $p-1$ factorization

Suppose you have a positive integer n and you know that it has two distinct prime factors, but we don't know what the prime factors are. Our goal is to figure out what these prime factors are. Notice that it's sufficient to just figure out one of the prime factors p, since then we can compute n/p quickly and this will be equal to the other prime factor q.

Getting started

Let us randomly choose an integer a such that $1 < a < n$. We can quickly compute $\gcd(n, a)$. This gcd must either be 1, or else it must be one of the prime factors of n. So, if it turns out that $\gcd(n, a) \neq 1$, then the gcd we've computed is a prime factor of n, and we are done.

But most integers are not multiples of one of the prime factors of n, so most likely, we will have $\gcd(a, n) = 1$, so let us suppose we are in that case. Suppose now that a wizard appears, hands you an integer L and assures you that $p - 1 \mid L$, where p is one of the prime factors of n, but the wizard vanishes before telling you what p is. You can try to use the wizard's information as follows. Compute the number $a^L - 1$. Since $p - 1 \mid L$, we know that there exists an i such that $L = i(p - 1)$, so using Fermat's little theorem we see that

$$a^L = a^{i(p-1)} = (a^{p-1})^i \equiv 1^i = 1 \bmod p.$$

In other words, $p \mid a^L - 1$. This means that $\gcd(a^L - 1, n)$ has to equal either p or else n. We can compute this gcd quickly. If it doesn't equal n, then we know that it has to equal p, so we've found a prime factor of n and we're done!

But it's also possible that it equals n. Notice that $\gcd(a^L - 1, n) = n$ means that $n \mid a^L - 1$, which means that if q is the other prime factor of n, then $q \mid a^L - 1$ also, in addition to $p \mid a^L - 1$ like we saw above. In other words, $a^L \equiv 1 \bmod q$. Suppose we divide L by $q - 1$ and write $L = j(q - 1) + k$ where j and k are integers and $0 \leq k < q - 1$. Then by Fermat's little theorem, we have

$$a^L = a^{j(q-1)+k} = (a^{q-1})^j a^k \equiv 1^j a^k = a^k \bmod q.$$

Since $a^L \equiv 1 \bmod q$, this means that $a^k \equiv 1 \bmod q$. There are two possible reasons this might happen.

(a) If $k \neq 0$, then it can happen that $a^k \equiv 1 \bmod q$.[1] We could try going back and picking a different integer a, and that will most likely solve the problem.

(b) If $k = 0$, then a^k is always congruent to 1 modulo q, as long as a is not divisible by q. Most likely we won't pick a multiple of q, and we'll keep ending up in this situation, until we accidentally happen to chance upon an a that is a multiple of q. There are not very many multiples of q, so this last situation is quite unlikely. So if it is the case that $k = 0$, we've been quite unlucky: most likely we're not going to be able to get anywhere with the wizard's information.

Of course, we don't know that q is, so we don't know that k is, so we don't know which of the above two reasons were the reason behind us seeing $\gcd(a^L - 1, n) = n$. But it's more likely that it was because $k \neq 0$, so if we happen to find that $\gcd(a^L - 1, n) = n$, we should just go back and try a different random a and hope that it works.

Wizards don't exist

Now of course, wizards don't exist, so how can we go about finding an appropriate integer L? Well, factorials of integers have lots of divisors, so we can hope that if we just attempt the above process with $L = 1!, 2!, 3!, 4!, 5!, \ldots$, then one of these will work. Here is an outline of the method we've just described. This is called Pollard's $p - 1$ factorization algorithm.

```
1. Choose a random integer $a$ such that $1 < a < n$.
```

[1]For example, note that $2^8 = 256 \equiv 1 \bmod 17$ even though 2 is relatively prime to 17 and $8 \neq 0$.
The point here is that this cannot happen if a is a primitive root modulo q. But a sizeable number of integers are primitive roots modulo q, and even if a is not a primitive root it's not guaranteed that we necessarily have to run into this situation, so choosing a different a should work. If you know enough number theory, it is a good exercise to calculate the probability that you run into this situation.

```
2. If $\gcd(a,n) > 1$, then this gcd is a prime factor of $n$, so we are done.

3. For each $r = 2, 3, \ldots$, compute $d = \gcd(a^{r!} - 1, n)$.

   a. If $d = n$, go back to step 1 and pick an integer $a$ we haven't tried yet.

   b. If $d \neq n$ but $d > 1$, then $d$ is a prime factor of $n$, so we are done.

   c. If $d = 1$, increment $r$ and repeat this loop.
```

Two are remarks about how you should go about this process. First, you might as well start with $a = 2$. This is as reasonable an initial guess as anything else.

Second, even for $a = 2$ and reasonably small values of r like $r = 100$, $a^{r!} = 2^{100!}$ is ENORMOUS (it is larger than the number of elementary particles in the known universe!), so it is hopeless to compute $a^{r!}$ exactly. But, luckily, you don't need to. We only care about $\gcd(a^{r!} - 1, n)$, and we know that if s is the remainder of dividing $a^{r!} - 1, n$, then $\gcd(a^{r!} - 1, n) = \gcd(s, n)$. So we only need to compute $a^{r!} - 1$ modulo n, which is immediately a lot easier since we never need to work with numbers larger than n to do this. Also, note that we don't even need to compute the exponent $r!$. On the previous iteration, we computed $a^{(r-1)!}$ modulo n, so on the current iteration we can just take the number we computed last time, raise it to rth power modulo n and we get $(a^{(r-1)!})^r = a^{r!}$.

Also, as it is stated, it is not totally obvious that the algorithm terminates. But it does. First, note that for any particular a, as we keep trying different r's, at some point we will definitely get $d > 1$. For example, as soon as we get to r being equal to $p - 1$, where p is the smaller of the prime factors of n, then clearly $p - 1 \mid r!$, so like we saw above we'll be forced to have $p \mid a^{r!} - 1$, so $\gcd(a^{r!} - 1, n) = p$ or n. At this point, either we'll have to go back and try a new a, or we'll have found a prime factor of n and we'll be done. Next, we might ask: what if we keep running into the situation $\gcd(a^{r!} - 1, n) = n$? Well, we keep picking different a's, so eventually we have to pick some a that is a multiple of one of the prime factors of n. Once we pick that a, we'll have $\gcd(a, n) > 1$ and we'll be done. So Pollard's $p - 1$ factorization algorithm always works. It's just that sometimes its very very slow, and other times its quite fast.

An example

Let $n = 10001$. Let's start with $a = 2$. Then clearly $\gcd(2, 10001) = 1$, so we proceed into the loop. We first compute $a^{2!} = 2^2 = 4$. Then

$$\gcd(a^{2!} - 1, n) = \gcd(3, 10001) = 1$$

so we continue. Now $a^{3!} = (a^{2!})^3 = 4^3 = 64$, and

$$\gcd(a^{3!} - 1, n) = \gcd(63, 10001) = 1$$

so we continue. Next $a^{4!} = (a^{3!})^4 = 64^4 \equiv 5539 \bmod 10001$, and

$$\gcd(a^{4!} - 1, n) = \gcd(5538, 10001) = 1$$

so we continue. Next $a^{5!} = (a^{4!})^5 = 5539^5 \equiv 7746 \bmod 10001$, and

$$\gcd(a^{5!} - 1, n) = \gcd(7745, 10001) = 1$$

so we continue. Next we have $a^{6!} = (a^{5!})^6 = 7746^6 \equiv 1169 \bmod 10001$, and

$$\gcd(a^{6!} - 1, n) = \gcd(1168, 10001) = 73.$$

We've run into a gcd that is bigger than 1 and not equal to $n = 10001$, so jackpot! 73 must be a prime factor of n. Then we can compute quickly that $10001/73 = 137$, so

$$10001 = 73 \cdot 137$$

and we are done.

When it's fast and when it's not

In order for Pollard's $p-1$ factorization algorithm to work in a reasonable amount of time, what we need is for there to exist some reasonably small integer r such that $p-1 \mid r!$ but $q-1 \nmid r!$. In other words, we need $p-1$ to have a bunch of small prime factors. Notice in the example that $72 = 2^3 \cdot 3^2$ which divides $6!$, and that is why we found the prime factor 73 when we got to $r = 6$. Notice also that $136 = 2^3 \cdot 17$ does not divide $6!$.

This has an important security implication: if Bob is choosing prime numbers for designing an RSA public key, he should avoid prime numbers p such that $p-1$ has lots of small prime factors, because otherwise Eve will be able to factor his modulus using Pollard's $p-1$ factorization algorithm. This is interesting, because it's not obvious at all from the onset that factorizing n has anything to do with factorizing $p-1$ and $q-1$. The lesson is that even though we might build a cryptosystem on some problem that is conjectured to be difficult in general, it is still reasonable that certain special cases of that problem are not as difficult as the general case and can lead to vulnerabilities in the cryptosystem if they are not considered.

More examples to try

It should be reasonable to factor the following numbers using Pollard's $p-1$ factorization algorithm. You should try it on your own.

(a) $n = 18923$.

(b) $n = 115147$.