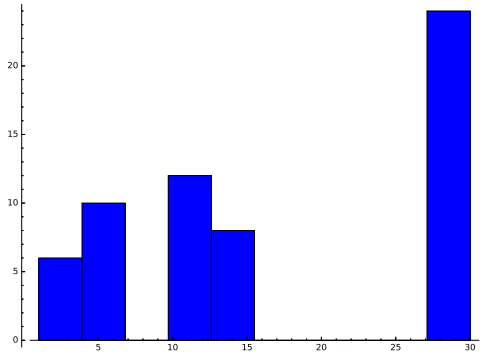
Example 105. The following illustrates how to use Sage to play with multiplicative orders.

```
Sage] r = mod(3,77)
Sage] r^10
   67
Sage] r.multiplicative_order()
   30
Sage] euler_phi(77)
   60
Sage] r.order()
   77
Sage] [mod(a,7).multiplicative_order() for a in [1..7] if gcd(a,7)==1]
   [1, 3, 6, 3, 6, 2]
Sage] histogram([mod(a,77).multiplicative_order() for a in [1..76] if gcd(a,77)==1])
```




```
Sage]
```

Example 106. Suppose there is a primitive root modulo n. Then, how many primitive roots are there in total?

Solution. Let x be a primitive root. It has order $\phi(n)$. All other invertible residues are of the form x^a.

Recall that x^a has order $\frac{\phi(n)}{\gcd(\phi(n),a)}$. This is $\phi(n)$ if and only if $\gcd(\phi(n),a)=1$. There are $\phi(\phi(n))$ values a among $1,2,\ldots,\phi(n)$, which are coprime to $\phi(n)$.

In conclusion, there are $\phi(\phi(n))$ many primitive roots modulo n.

Comment. Recall that, for instance, there is no primitive root modulo 15. That's why we needed the assumption that there should be a primitive root modulo n (which is the case if and only if n is of the form $1, 2, 4, p^k, 2p^k$ for some odd prime p).

Example 107. One can show that, for every prime p, primitive roots exist. By the previous example, it follows that the total number of primitive roots is $\phi(\phi(p)) = \phi(p-1)$. The following computations in Sage indicate that typically a "decent" proportion (25-50%) of all invertible residues are primitive roots. The exact proportion is, of course $\frac{\phi(p-1)}{p-1}$ but to say more about the magnitude, we need the factorization of $p-1$.

Advanced comment. However, the number of primitive roots can (though this is very rare) be an arbitrarily small proportion. In fact, a result of Kátai shows that, for any $x \in [0,1]$, there is a proportion $P(x)$ of primes with $\frac{\phi(p-1)}{p-1} \leqslant x$, and that $P(x)$ is a strictly increasing continuous function with $P(0) = 0$ and $P(1/2) = 1$.

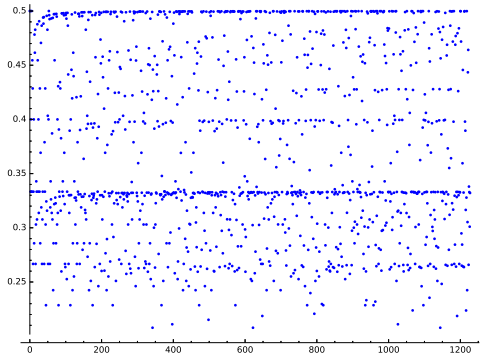
```
Sage] [p for p in prime_range(100)]
```

$$[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97]$$

```
Sage] [euler_phi(p-1)/(p-1) for p in prime_range(30)]
```

$$\left[1, \frac{1}{2}, \frac{1}{2}, \frac{1}{3}, \frac{2}{5}, \frac{1}{3}, \frac{1}{2}, \frac{1}{3}, \frac{5}{11}, \frac{3}{7}\right]$$

```
Sage] list_plot([euler_phi(p-1)/(p-1) for p in prime_range(3,10000)])
```




```
Sage]
```

Example 108. (homework) Let p be an odd prime. Show that the $\frac{\phi(p-1)}{p-1} \leqslant \frac{1}{2}$.

In other words, at most half of the invertible residues are primitive roots.

Solution. Let $p_1, p_2, \ldots$ be the primes, in increasing order, dividing $p-1$. Since $p \neq 2$, $p-1$ is divisible by 2, so that $p_1 = 2$.

Then, $\phi(p-1) = (p-1)\underbrace{\left(1-\frac{1}{p_1}\right)}_{=1/2}\underbrace{\left(1-\frac{1}{p_2}\right)}_{\leqslant 1}\cdots \leqslant \frac{1}{2}(p-1)$.

Correspondingly, $\frac{\phi(p-1)}{p-1} \leqslant \frac{\frac{1}{2}(p-1)}{p-1} = \frac{1}{2}$, as claimed.

In fact. Note that $\left(1-\frac{1}{p_2}\right) < 1$ if there is a second prime. Our proof therefore actually shows that $\frac{\phi(p-1)}{p-1} = \frac{1}{2}$ if and only if the only $p-1$ is of the form 2^n (i.e. the only prime dividing $p-1$ is 2). Equivalently, if p is of the form 2^n+1.

Comment. Primes of the form 2^n+1 are known as **Fermat primes**. It can be shown that such a prime is, in fact, necessarily of the form $F_k = 2^{2^k}+1$. The first five numbers $F_0 = 3$, $F_1 = 5$, $F_2 = 17$, $F_3 = 257$, $F_4 = 65537$ are prime, and Fermat conjectured that F_k is prime for all $k \geqslant 0$. This was proven wrong by Euler who demonstrated that $F_5 = 2^{32}+1 = 641 \cdot 6700417$ (this was way before the time, we could ask a computer to factor not-too-large numbers). To this day, it is not known whether any further Fermat primes exist.

Example 109. The following is the number from Example 78, for which RSA Laboratories, until 2007, offered $100,000 to the first one to factorize it. Nobody has been able to do so to this day.

Has the thought crossed your mind that the challengers might be tricking everybody by chosing M to be a huge prime that cannot be factored further? Well, we'll talk more about primality testing soon. But we can actually quickly convince ourselves that M cannot be a prime. If M was prime then, by Fermat's little theorem, $2^{M-1} \equiv 1 \pmod{M}$. Below, we compute $2^{M-1} \pmod{M}$ and find that $2^{M-1} \not\equiv 1 \pmod{M}$. This proves that M is not a prime. It doesn't bring us any closer to factoring it though.

Comment. Ponder about this for a while. We can tell that a number is composite without finding its factors. Both sides to this story (first, being able to efficiently tell whether a number is prime, and second, not being able to factor large numbers) are of vital importance to modern cryptography.

```
Sage] rsa = Integer("135066410865995223349603216278805969938881475605667027524485143851\
                     526510604859533833940287150571909441798207282164471551373680419703\
                     964191743046496589274256239341020864383202110372958725762358509643\
                     110564073501508187510676594629205563685529475213500852879416377328\
                     533906109750544334999811150056977236890927563")

Sage] power_mod(2, rsa-1, rsa)

   1209390944320336158676505953529569968675400984635889512389028083675567339322020593385 3\
   348534147116662841968124107288512373904071077139405352848835710498409193003137847878 95\
   226029615123284879513798127406300472693925500331497519103479951096634123177725212482 97\
   950196643140069546889855131459759160570963857373851

Sage]
```

Let us illustrate how to actually use this number in the B-B-S PRG.

```
Sage] seed = randint(2,rsa-2)

Sage] y = seed; prg = []

Sage] for i in [1..25]:
          y = (y^2) % rsa
          prg.append(y % 2)

Sage] prg

   [0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0]

Sage]
```

If you are able, even after a gigabyte of pseudorandom bits, to predict the next bits with an accuracy better than 50% (which is just pure guessing), then you likely have a shot at factoring the big integer. You would be the first!

Of course, it is not impressive to see a few random bits in the example above. After all, the seed itself consists of 1024 random bits. The whole point is that we can, from these 1024 random bits, produce gigabytes of further pseudorandom bits. As of this day, no one would be able to distinguish these from truly random bits.

While all of this works nicely, B-B-S is considered to be too slow for most practical purposes.