Example 76. (review)

- The number of invertible residues modulo n is $\phi(n)$.
- The number of invertible quadratic residues modulo p (odd prime) is $\frac{\phi(p)}{2} = \frac{p-1}{2}$.
- The number of invertible quadratic residues modulo pq is $\frac{\phi(pq)}{4} = \frac{p-1}{2}\frac{q-1}{2}$.

  Here, p, q are distinct odd primes. See Example 75.

Example 77.

(a) List all invertible quadratic residues modulo 21. Compute the square of all these residues.

(b) Repeat the first part modulo 33 and modulo 35. When computing the squares of these, do you notice a difference modulo 35?

[Note that $35 = 5 \cdot 7$ with $5 \equiv 1 \pmod 4$. This case will be excluded in the B-B-S PRG below.]

Solution. (final answers only)

(a) Among the $\phi(21) = 12$ many invertible elements, the squares are $1, 4, 16$ (exactly a quarter). Computing the squares: $1^2 \equiv 1$, $4^2 \equiv 16$, $16^2 \equiv 4 \pmod{21}$. Note that the squares are all different!

(b) Modulo 33: among the $\phi(33) = 20$ many invertible elements, the squares are $1, 4, 16, 25, 31$ (exactly a quarter). Computing the squares: $1^2 \equiv 1$, $4^2 \equiv 16$, $16^2 \equiv 25$, $25^2 \equiv 31$, $31^2 \equiv 4 \pmod{33}$. Again, all the squares are different!

Modulo 35: among the $\phi(35) = 24$ many invertible elements, the squares are $1, 4, 9, 11, 16, 29$ (exactly a quarter). Computing the squares: $1^2 \equiv 1$, $4^2 \equiv 16$, $9^2 \equiv 11$, $11^2 \equiv 16$, $16^2 \equiv 11$, $29^2 \equiv 1 \pmod{35}$. Observe that these are not all different: for instance, $9^2 \equiv 16^2 \pmod{35}$.

The thing that is special about the case $p \equiv 1 \pmod 4$ is that, in that case, -1 is a quadratic residue. We will return to that.

The reason this makes a difference is the following: recall that an invertible quadratic residue x^2 modulo $M = pq$ has exactly four squareroots $\pm x$, $\pm y$. In order for the mapping $y \mapsto y^2 \pmod M$ to be 1-1 when restricted to invertible quadratic residues, we need that exactly one of $\pm x, \pm y$ is itself a quadratic residue. However, this is not possible if -1 is a quadratic residue, because then x and $-x$ (and, likewise, y and $-y$) are either both quadratic residues or not.

The Blum-Blum-Shup PRG (believed to be unpredictable)

The following is an example of a PRG, which is believed to be unpredictable.

More precisely, it has been shown that the ability to predict its values is equivalent to being able to efficiently solve the quadratic residuosity problem (which is believed to be hard). Currently, the best way to "solve" the quadratic residuosity problem mod M relies on factoring M. However, factoring large numbers is considered to be hard, see Example 80 (and lots of crypto relies on that).

Quadratic residuosity problem. Given big $M = pq$ and a residue x modulo M, decide whether x is a quadratic residue. (About $M/4$ are quadratic residues (see above); $M/2$ are easily determined to be nonsquare using the Jacobi symbol [don't worry if you haven't heard about that].)

(Blum-Blum-Shub PRG) Let $M = pq$ where p, q are large primes $\equiv 3 \pmod 4$.
From the seed y_0, we generate $y_{n+1} = y_n^2 \pmod M$.
The random bits we produce are $x_n = \text{least bit of}(y_n)$.

BBS is very slow, and mostly of theoretical value. However, as mentioned above, it is interesting because it is indeed unpredictable (to anyone not knowing the factorization of M) if an important number theory problem is "hard" (this can be made precise), as is believed to be the case.

Why the conditions on p and q? Recall from the CRT that an invertible quadratic residue x^2 modulo $M = pq$ has exactly four squareroots $\pm x$, $\pm y$. The condition $3 \pmod 4$ guarantees that, of these four, exactly one is itself a quadratic residue. As a consequence, the mapping $y \mapsto y^2 \pmod M$ is 1-1 when restricted to invertible quadratic residues (see Example 77).

Comment. For obvious reasons, the seed $y_0 \equiv \pm 1 \pmod M$ should be excluded. Also, for the above considerations, the seed needs to be coprime to M. However, we don't need to worry about that: running into a factor of M by accident is close to impossible (recall that nobody should be able to factor M even on purposes and with lots of time and resources).

Comment. To increase speed, at the expense of some security, we can also take several, say k, bits of y_n (as long as k is small, say, $k \leqslant \log_2 \log_2 M$).

Example 78. Generate random bits using the B-B-S PRG with $M = 77$ and seed 3.

Solution. With $y_0 = 3$, we have $y_1 = y_0^2 = 9$, followed by $y_2 = y_1^2 \equiv 4 \pmod{77}$, $y_3 \equiv 16$, $y_4 \equiv 25$, $y_5 \equiv 9$, so that the values y_n now start repeating.

These numbers are, however, not the output of the PRG. We only output the least bit of the numbers y_n, i.e. the value of $y_n \pmod 2$. For $y_1 \equiv 9$ we output 1, for $y_2 \equiv 4$ we output 0, for $y_3 \equiv 16$ we output 0, for $y_4 \equiv 25$ we output 1, and so on.

In other words, the seed 3 produces the sequence $1, 0, 0, 1, 1, 0, 0, 1, 1, 0, \ldots$ of period 4.

Comment. Note that it was completely to be expected that the numbers repeat. In fact, we immediately see that the number of possible y_n is at most the number of invertible quadratic residues, of which [by the previous example] there are only $\phi(77)/4 = 15$. See Example 106 for a better prediction.

Example 79. (extra) Generate random bits using the B-B-S PRG with $M = 209$ and seed 10. What is the period of the generated sequence? (Then repeat with seed 25.)

Solution. (final answer only) The seed 10 produces the sequence $0, 1, 0, 1, 1, 1, \ldots$ of period 6.
The seed 25 generates the sequence $1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, \ldots$ of period 12.
[By the way, it is a great idea, to let Sage assist you.]

Example 80. We mentioned that the unpredictability of the B-B-S PRG relies on the difficulty of factoring large numbers. Here's an indication how difficult it seems to be. In 1991, RSA Laboratories challenged everyone to factor several numbers including:

```
135066410865995223349603216278805969938881475605667027524485143851\
106048595338339402871505719094417982072821644715513736804197039641917 4\
304649658927425623934102086438320211037295872576235850964311056407 3501\
508187510676594629205563685529475213500852879416377328533906109750544 3\
34999811150056977236890927563
```

Since then, nobody has been able to factor this 1024 bit number (309 decimal digits). Until 2007, cash prizes were offered up to 200,000 USD, with 100,000 USD for the number above.

https://en.wikipedia.org/wiki/RSA_Factoring_Challenge

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

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

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
```

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

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.