Example 99. (review) Solve $x \equiv 2 \pmod{7}$, $x \equiv 3 \pmod{11}$.

Solution. $x \equiv 2 \cdot 11 \cdot \underbrace{11_{\bmod 7}^{-1}}_{2} + 3 \cdot 7 \cdot \underbrace{7_{\bmod 11}^{-1}}_{-3} \equiv 44 - 63 \equiv 58 \pmod{77}$

Example 100. Determine all solutions to $x^2 \equiv 4 \pmod{77}$.

Solution. By the CRT:

$$\begin{aligned} & x^2 \equiv 4 \pmod{77} \\ \iff & x^2 \equiv 4 \pmod{7} \text{ and } x^2 \equiv 4 \pmod{11} \\ \iff & x \equiv \pm 2 \pmod{7} \text{ and } x \equiv \pm 2 \pmod{11} \end{aligned}$$

Hence, there is four solutions modulo 77: $\pm 2, \pm a$. To find a, we solve $x \equiv 2 \pmod{7}$, $x \equiv -2 \pmod{11}$.

$x \equiv 2 \cdot 11 \cdot \underbrace{11_{\bmod 7}^{-1}}_{2} - 2 \cdot 7 \cdot \underbrace{7_{\bmod 11}^{-1}}_{-3} \equiv 44 + 42 \equiv 9 \pmod{77}$

Hence, the four solutions are $x \equiv \pm 2, \pm 9 \pmod{77}$.

10 Using Sage as a fancy calculator

Any serious number theory applications, such as those in cryptography, involve computations that need to be done by a machine. Let us see how to use the open-source computer algebra system **Sage** to do basic computations for us.

Sage is freely available at `sagemath.org`. Instead of installing it locally (it's huge!) we can conveniently use it in the cloud at `cocalc.com` from any browser.

Sage is built as a **Python** library, so any Python code is valid. For starters, we will use it as a fancy calculator.

Example 101. Let's start with some basics.

```
Sage] 17 % 12
```

5

```
Sage] (1 + 5) % 2  # don't forget the brackets
```

0

```
Sage] inverse_mod(17, 23)
```

19

```
Sage] xgcd(17, 23)
```

$(1, -4, 3)$

```
Sage] -4*17 + 3*23
```

1

Example 102. Can you figure out what is being computed here?

```
Sage] crt([2,-2], [7,11])
```

9

Example 103. Why is the following bad?

```
Sage] 3^1003 % 101
```

27

The reason is that this computes 3^{1003} first, and then reduces that huge number modulo 101:

```
Sage] 3^1003
```

$$3569591212598177919604229201330789788106639488430800052695284994212437212836103228760 1\backslash$$
$$01447396641767302556399781555972361067577371671671062036425358196474919874574608035466\backslash$$
$$17047063989041820507144085408031748926871104815910218235498276622866724603402112436668\backslash$$
$$09387969298949770468720050187071564942882735677962417251222021721836167242754312973216\backslash$$
$$80102291029227131545307753863985171834477895265551139587894463150442112884933077598746\backslash$$
$$04125161734774642865878855686737747603770909 40027$$

We know how to avoid computing huge intermediate numbers. Sage does the same if we instead use something like:

```
Sage] power_mod(3, 1003, 101)
```

27

Armin Straub
straub@southalabama.edu