Advent of Code 2024, day 13

Part 1 can be solved trivially with brute-force. This is not possible for part 2 due to the much larger prize targets. The problem can be described as a system of linear equations:

$$\begin{aligned} x_1 b_1 + x_2 b_2 &= p_x \\ y_1 b_1 + y_2 b_2 &= p_y \end{aligned} \tag{1}$$

This system has two unknowns b_1 and b_2, which must be integers. It can have exactly one solution, no solution, or infinite solutions. For a unique solution, the determinant must be non-zero:

$$\det \begin{pmatrix} x_1 & x_2 \\ y_1 & y_2 \end{pmatrix} = x_1 y_2 - x_2 y_1 \neq 0 \tag{2}$$

This would be easy to solve, if the unknowns were rationals. But since they are integers, it is a system of linear Diophantine equations. In general it gets complicated, but analyzing the solutions from part 1 shows that there are either none or one solution, so first let's try to solve as usual, with Cramer's rule, which looks like this for our equation system, for calculating b_1:

$$b_1 = \frac{\det \begin{pmatrix} p_x & x_2 \\ p_y & y_2 \end{pmatrix}}{d} = \frac{p_x y_2 - x_2 p_y}{x_1 y_2 - x_2 y_1} \tag{3}$$

With b_1, we can calculate b_2 a bit faster, by solving our first linear equation for b_2:

$$b_2 = \frac{p_x - x_1 b_1}{x_2} \tag{4}$$

Now to check if it is an integer solution, we just have to test if the numerators are divisible by the denominators, taking care to handle negative determinants properly. In Rust we can use the modulo operator, looks like this, including the cost function:

```
let det: i128 = (x1 * y2) as i128 - (x2 * y1) as i128;
if det != 0 {
    let num: i128 = (px * y2) as i128 - (x2 * py) as i128;
    if num.abs() % det.abs() == 0 {
        let b1 = num / det;
        let num: i128 = px as i128 - x1 as i128 * b1;
        if num % x2 as i128 == 0 {
            let b2 = num / x2 as i128;
            let cost = 3 * b1 + b2;
            solution2 += cost as u128;
        }
    }
}
```