Day 13

Part 1

Next up: the lobby of a resort on a tropical island. The Historians take a moment to admire the hexagonal floor tiles before spreading out.

Fortunately, it looks like the resort has a new arcade! Maybe you can win some prizes from the claw machines?

The claw machines here are a little unusual. Instead of a joystick or directional buttons to control the claw, these machines have two buttons labeled A and B. Worse, you can't just put in a token and play; it costs 3 tokens to push the A button and 1 token to push the B button.

With a little experimentation, you figure out that each machine's buttons are configured to move the claw a specific amount to the right (along the X axis) and a specific amount forward (along the Y axis) each time that button is pressed.

Each machine contains one prize; to win the prize, the claw must be positioned exactly above the prize on both the X and Y axes.

You wonder: what is the smallest number of tokens you would have to spend to win as many prizes as possible? You assemble a list of every machine's button behavior and prize location (your puzzle input). For example:

Button A: X+94, Y+34 Button B: X+22, Y+67 Prize: X=8400, Y=5400

Button A: X+26, Y+66 Button B: X+67, Y+21 Prize: X=12748, Y=12176

Button A: X+17, Y+86 Button B: X+84, Y+37 Prize: X=7870, Y=6450

Button A: X+69, Y+23 Button B: X+27, Y+71 Prize: X=18641, Y=10279

This list describes the button configuration and prize location of four different claw machines.

For now, consider just the first claw machine in the list:

- Pushing the machine's A button would move the claw 94 units along the X axis and 34 units along the Y axis.
- Pushing the B button would move the claw 22 units along the X axis and 67 units along the Y axis.
- The prize is located at X=8400, Y=5400; this means that from the claw's initial position, it would need to move exactly 8400 units along the X axis and exactly 5400 units along the Y axis to be perfectly aligned with the prize in this machine.

The cheapest way to win the prize is by pushing the A button 80 times and the B button 40 times. This would line up the claw along the X axis (because 80*94 + 40*22 = 8400) and along the Y axis (because 80*34 + 40*67 = 5400). Doing this would cost 80*3 tokens for the A presses and 40*1 for the B presses, a total of 280 tokens.

For the second and fourth claw machines, there is no combination of A and B presses that will ever win a prize.

For the third claw machine, the cheapest way to win the prize is by pushing the A button 38 times and the B button 86 times. Doing this would cost a total of 200 tokens.

So, the most prizes you could possibly win is two; the minimum tokens you would have to spend to win all (two) prizes is `480`.

You estimate that each button would need to be pressed no more than 100 times to win a prize. How else would someone be expected to play?

Figure out how to win as many prizes as possible. What is the fewest tokens you would have to spend to win all possible prizes?

In [1]:
```
from pathlib import Path

import numpy as np
```

In [2]:
```
test_path = Path("data/day13_test.txt")
data_path = Path("data/day13_data.txt")


def load_data(fpath: Path, offset: int=0) -> dict[dict]:
    """Returns dict of dicts describing each arcade machine"""
    machines = {}  # One key per machine

    with fpath.open() as ifh:
        machine_id = 0
        machines[machine_id] = {"xvals": {}, "yvals": {}}  # One dict each for x- and
        for line in [_.strip().split() for _ in ifh.readlines()]:
            if len(line) == 0:  # start new machine with a newline
                machine_id += 1
                machines[machine_id] = {"xvals": {}, "yvals": {}}
            elif line[0] == "Button":
                machines[machine_id]["xvals"][line[1][0]] = int(line[2].split("+")[1]
                machines[machine_id]["yvals"][line[1][0]] = int(line[3].split("+")[1]
            elif line[0] == "Prize:":
                machines[machine_id]["xvals"]["prize"] = int(line[1].split("=")[1][:-
                machines[machine_id]["yvals"]["prize"] = int(line[2].split("=")[1]) +

    return machines


def solve_machines(machines: dict[dict]) -> list[tuple[int, np.array]]:
    """Returns list of tuples containing machine numbers and A/B solutions.

    The puzzle today is a linear algebra problem - a system of simultaneous
    equations. We _could_ code the algorithm from scratch/refer to Numerical
    Methods, but numpy exists.
    """
    solutions = []

    for machine_id, params in machines.items():
        xarr = [params["xvals"]["A"], params["xvals"]["B"]]
        yarr = [params["yvals"]["A"], params["yvals"]["B"]]
        X = np.array([xarr, yarr])
        Y = np.array([params["xvals"]["prize"], params["yvals"]["prize"]]).T
        solutions.append((machine_id, np.linalg.solve(X, Y)))

    return solutions


def cost_solutions(solutions: list[tuple[int, np.array]], tokens: dict, machines: dic
    """Returns the cost of winning all prizes on the machines"""
    results = []
```

```
    for solution, machine in zip(solutions, machines):
        params = machines[machine]
        # Solution as integer. Note we use round(). Can't tell you how long I sat
        # with the bug of using int() here.
        solA, solB = round(solution[1][0]), round(solution[1][1])
        resX = solA * params["xvals"]["A"] + solB * params["xvals"]["B"]
        resY = solA * params["yvals"]["A"] + solB * params["yvals"]["B"]

        # Some solutions are not integers, and the float representation is
        # inexact, so we double-check and populate the list of costs
        if resX == params["xvals"]["prize"] and resY == params["yvals"]["prize"]:  #
            results.append(tokens["A"] * solA +  tokens["B"] * solB)
        else:
            results.append(0)

    return results


tokens = {"A": 3, "B": 1}
machines = load_data(test_path)
solutions = solve_machines(machines)
sum(cost_solutions(solutions, tokens, machines))
```

Out[2]: 480

In [3]:
```
machines = load_data(data_path)
solutions = solve_machines(machines)
sum(cost_solutions(solutions, tokens, machines))
```

Out[3]: 30413

Part 2

As you go to win the first prize, you discover that the claw is nowhere near where you expected it would be. Due to a unit conversion error in your measurements, the position of every prize is actually 10000000000000 higher on both the X and Y axis!

Add 10000000000000 to the X and Y position of every prize. After making this change, the example above would now look like this:

```
Button A: X+94, Y+34
Button B: X+22, Y+67
Prize: X=10000000008400, Y=10000000005400

Button A: X+26, Y+66
Button B: X+67, Y+21
Prize: X=10000000012748, Y=10000000012176

Button A: X+17, Y+86
Button B: X+84, Y+37
Prize: X=10000000007870, Y=10000000006450

Button A: X+69, Y+23
Button B: X+27, Y+71
Prize: X=10000000018641, Y=10000000010279
```

Now, it is only possible to win a prize on the second and fourth claw machines. Unfortunately, it will take many more than 100 presses to do so.

Using the corrected prize coordinates, figure out how to win as many prizes as possible. What is the fewest tokens you would have to spend to win all possible prizes?

In [4]:
```
machines = load_data(test_path, offset=1e13)
solutions = solve_machines(machines)
sum(cost_solutions(solutions, tokens, machines))
```

Out[4]: 875318608908

In [5]:
```
machines = load_data(data_path, offset=1e13)
solutions = solve_machines(machines)
sum(cost_solutions(solutions, tokens, machines))
```

Out[5]: 92827349540204

In [ ]:
```

```