Day 14

Part 1

One of The Historians needs to use the bathroom; fortunately, you know there's a bathroom near an unvisited location on their list, and so you're all quickly teleported directly to the lobby of Easter Bunny Headquarters.

Unfortunately, EBHQ seems to have "improved" bathroom security again after your last visit. The area outside the bathroom is swarming with robots!

To get The Historian safely to the bathroom, you'll need a way to predict where the robots will be in the future. Fortunately, they all seem to be moving on the tile floor in predictable straight lines.

You make a list (your puzzle input) of all of the robots' current positions (p) and velocities (v), one robot per line. For example:

```
p=0,4 v=3,-3
p=6,3 v=-1,-3
p=10,3 v=-1,2
p=2,0 v=2,-1
p=0,0 v=1,3
p=3,0 v=-2,-2
p=7,6 v=-1,-3
p=3,0 v=-1,-2
p=9,3 v=2,3
p=7,3 v=-1,2
p=2,4 v=2,-3
p=9,5 v=-3,-3
```

Each robot's position is given as p=x,y where x represents the number of tiles the robot is from the left wall and y represents the number of tiles from the top wall (when viewed from above). So, a position of p=0,0 means the robot is all the way in the top-left corner.

Each robot's velocity is given as v=x,y where x and y are given in tiles per second. Positive x means the robot is moving to the right, and positive y means the robot is moving down. So, a velocity of v=1,-2 means that each second, the robot moves 1 tile to the right and 2 tiles up.

The robots outside the actual bathroom are in a space which is 101 tiles wide and 103 tiles tall (when viewed from above). However, in this example, the robots are in a space which is only 11 tiles wide and 7 tiles tall.

The robots are good at navigating over/under each other (due to a combination of springs, extendable legs, and quadcopters), so they can share the same tile and don't interact with each other. Visually, the number of robots on each tile in this example looks like this:

```
1.12.......
...........
...........
......11.11
1.1........
.........1.
.......1...
```

These robots have a unique feature for maximum bathroom security: they can teleport. When a robot would run into an edge of the space they're in, they instead teleport to the other side, effectively wrapping around the edges. Here is what robot p=2,4 v=2,-3 does for the first few seconds:

Initial state:

```
...........
...........
...........
...........
..1........
...........
...........
```

After 1 second:

```
...........
....1......
...........
...........
...........
...........
...........
```

After 2 seconds:

```
...........
...........
...........
...........
...........
......1....
...........
```

After 3 seconds:

```
...........
...........
........1..
...........
...........
...........
...........
```

After 4 seconds:

```
...........
...........
...........
...........
...........
...........
..........1
```

After 5 seconds:

```
...........
...........
...........
.1.........
...........
...........
...........
```

The Historian can't wait much longer, so you don't have to simulate the robots for very long. Where will the robots be after 100 seconds?

In the above example, the number of robots on each tile after 100 seconds has elapsed looks like this:

```
......2..1.
...........
1..........
.11........
.....1.....
...12......
.1....1....
```

To determine the safest area, count the number of robots in each quadrant after 100 seconds. Robots that are exactly in the middle (horizontally or vertically) don't count as being in any quadrant, so the only relevant robots are:

```
..... 2..1.
..... .....
1.... .....

..... .....
...12 .....
.1... 1....
```

In this example, the quadrants contain 1, 3, 4, and 1 robot. Multiplying these together gives a total safety factor of 12.

Predict the motion of the robots in your list within a space which is 101 tiles wide and 103 tiles tall. What will the safety factor be after exactly 100 seconds have elapsed?

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

import numpy as np
```

In [2]:
```
test_path = Path("data/day14_test.txt")
data_path = Path("data/day14_data.txt")

def load_data(fpath: Path) -> list[dict]:
    """Load starting robot position/velocities as list of dicts."""
    robots =[]

    with fpath.open() as ifh:
        for line in [_.strip().split() for _ in ifh.readlines()]:
            pos = [int(_) for _ in line[0].split("=")[1].split(",")]
            vel = [int(_) for _ in line[1].split("=")[1].split(",")]
            robots.append({"pos": pos, "vel": vel})

    return robots
```

```
def update_robots(robots: list[dict], steps: int=1, xlim: int=11, ylim: int=7) -> lis
    """Update robot positions by the requested number of steps."""
    for robot in robots[:]:
        newpos = [(robot["pos"][0] + steps * robot["vel"][0]) % xlim,
                  (robot["pos"][1] + steps * robot["vel"][1]) % ylim]
        robot["pos"] = newpos

    return robots


def safety_factor(robots: list[dict], xlim: int, ylim: int):
    """Returns the safety factor for the current robot positions."""
    quadrants = {"UL": 0, "UR": 0, "LL": 0, "LR": 0}
    for robot in robots:
        if robot["pos"][0] < xlim//2 and robot["pos"][1] < ylim//2:
            quadrants["UL"] += 1
        elif robot["pos"][0] < xlim//2 and robot["pos"][1] > ylim//2:
            quadrants["LL"] += 1
        elif robot["pos"][0] > xlim//2 and robot["pos"][1] < ylim//2:
            quadrants["UR"] += 1
        elif robot["pos"][0] > xlim//2 and robot["pos"][1] > ylim//2:
            quadrants["LR"] += 1

    return prod(quadrants.values())


xlim, ylim = 11, 7  # Patrol area dimensions
robots = load_data(test_path)
robots = update_robots(robots, 100, xlim=xlim, ylim=ylim)
safety_factor(robots, xlim=xlim, ylim=ylim)
```

Out[2]: 12

In [3]:
```
xlim, ylim = 101, 103  # Patrol area dimensions
robots = load_data(data_path)
robots = update_robots(robots, 100, xlim=xlim, ylim=ylim)
safety_factor(robots, xlim=xlim, ylim=ylim)
```

Out[3]: 226179492

Part 2

During the bathroom break, someone notices that these robots seem awfully similar to ones built and used at the North Pole. If they're the same type of robots, they should have a hard-coded Easter egg: very rarely, most of the robots should arrange themselves into a picture of a Christmas tree.

What is the fewest number of seconds that must elapse for the robots to display the Easter egg?

In [4]:
```
def print_grid(robots: list[dict], steps: int, xlim: int, ylim: int) -> None:
    """Writes out ASCII art of final robot positions."""
    arr = np.full((xlim, ylim), ".", dtype=str)
    for robot in robots:
        arr[robot["pos"][0]][robot["pos"][1]] = "X"
    np.savetxt(Path("day14_images")/f"{steps:06d}.txt", arr.T,
               fmt="%s", delimiter="")

def all_unique_pos(robots: list[dict]) -> bool:
    """Returns True if each robot is in a unique position (no overlaps)."""
    poslist = set([tuple(robot["pos"]) for robot in robots])
```

```
    if len(poslist) == len(robots):
        return True
    return False


xlim, ylim = 101, 103
robots = load_data(data_path)

# Check first 10000 steps for whether all robots are in a unique
# position; we assume that our solution has all robots in a unique position
candidates = []  # Step count when all robots in a a unique position
for _ in range(1, 10000):
    robots = update_robots(robots, 1, xlim, ylim)
    if len(robots) == len({tuple(robot["pos"]) for robot in robots}):
        candidates.append(_)

# Render ASCII art of final positions
for steps in candidates:
    print_grid(update_robots(load_data(data_path), steps, xlim, ylim), steps, xlim, y

print(candidates)  # Part 2 solution
```

```
[7502]
```

In [ ]: