advent_of_code_day 13

December 20, 2021

1 Advent of Code

[ ]:
```
# set up the environment
import numpy as np
```

1.1 Day 13: Transparent Origami

You reach another volcanically active part of the cave. It would be nice if you could do some kind of thermal imaging so you could tell ahead of time which caves are too hot to safely enter.

Fortunately, the submarine seems to be equipped with a thermal camera! When you activate it, you are greeted with:

Congratulations on your purchase! To activate this infrared thermal imaging camera system, please enter the code found on page 1 of the manual.

Apparently, the Elves have never used this feature. To your surprise, you manage to find the manual; as you go to open it, page 1 falls out. It's a large sheet of transparent paper! The (transparent paper)[https://en.wikipedia.org/wiki/Transparency_(projection)] is marked with random dots and includes instructions on how to fold it up (your puzzle input). For example:

```
6,10
0,14
9,10
0,3
10,4
4,11
6,0
6,12
4,1
0,13
10,12
3,4
3,0
8,4
1,10
2,14
8,10
9,0
```

```
fold along y=7
fold along x=5
```

The first section is a list of dots on the transparent paper. 0,0 represents the top-left coordinate. The first value, x, increases to the right. The second value, y, increases downward. So, the coordinate 3,0 is to the right of 0,0, and the coordinate 0,7 is below 0,0. The coordinates in this example form the following pattern, where # is a dot on the paper and . is an empty, unmarked position:

```
...#..#..#.
....#......
..........
#.........
...#....#.#
..........
..........
..........
..........
..........
.#....#.##.
....#......
......#...#
#.........
#.#.......
```

Then, there is a list of **fold instructions**. Each instruction indicates a line on the transparent paper and wants you to fold the paper **up** (for horizontal y=… lines) or **left** (for vertical x=… lines). In this example, the first fold instruction is fold along y=7, which designates the line formed by all of the positions where y is 7 (marked here with -):

```
...#..#..#.
....#......
..........
#.........
...#....#.#
..........
..........
-----------
..........
..........
.#....#.##.
....#......
......#...#
#.........
#.#.......
```

Because this is a horizontal line, fold the bottom half up. Some of the dots might end up overlapping after the fold is complete, but dots will never appear exactly on a fold line. The result of doing this fold looks like this:

```
#.##..#..#.
#...#......
```

```
......#...#
#...#......
.#.#..#.###
...........
...........
```

Now, only 17 dots are visible.

Notice, for example, the two dots in the bottom left corner before the transparent paper is folded; after the fold is complete, those dots appear in the top left corner (at 0,0 and 0,1). Because the paper is transparent, the dot just below them in the result (at 0,3) remains visible, as it can be seen through the transparent paper.

Also notice that some dots can end up **overlapping**; in this case, the dots merge together and become a single dot.

The second fold instruction is fold along x=5, which indicates this line:

```
#.##.|#..#.
#...#|.....
.....|#...#
#...#|.....
.#.#.|#.###
.....|.....
.....|.....
```

Because this is a vertical line, fold left:

```
#####
#...#
#...#
#...#
#####
.....
.....
```

The instructions made a square!

The transparent paper is pretty big, so for now, focus on just completing the first fold. After the first fold in the example above, **17** dots are visible - dots that end up overlapping after the fold is completed count as a single dot.

How many dots are visible after completing just the first fold instruction on your transparent paper?

1.2 Answer

numpy co-ordinate system is not the same as the paper co-ordinate system used in this problem. All the work will be done in the numpy system with the axis being swapped for the final results.

[ ]:
```
paper_instructions = []
with open("data/transparent_origami.dat") as file:
    temp = []
```

```
    for line in file:
        line = line.strip()
        if line == "":
            continue
        if "fold" in line:
            x, y = line.replace("fold along ", "").split("=")
            paper_instructions.append([x, int(y)])
        else:
            x, y = line.split(",")
            temp.append([int(x), int(y)])

    paper_locations= np.asarray(temp)

test_locations = np.asarray(
    [
        [6, 10],
        [0, 14],
        [9, 10],
        [0, 3],
        [10, 4],
        [4, 11],
        [6, 0],
        [6, 12],
        [4, 1],
        [0, 13],
        [10, 12],
        [3, 4],
        [3, 0],
        [8, 4],
        [1, 10],
        [2, 14],
        [8, 10],
        [9, 0],
    ]
)

test_instructions = [['y', 7], ['x', 5]]
```

[ ]:
```
program_map = paper_locations.copy()
program_instructions = paper_instructions.copy()

# determine size of paper
x_max = program_map.max(axis=0)[0]
y_max = program_map.max(axis=0)[1]

program_paper = np.zeros((x_max+1, y_max+1), dtype=bool)
for location in program_map:
```

```
    program_paper[(location[0], location[1])] = True

def fold_paper(paper: np.ndarray, f_axis: str, f_line: int) -> np.ndarray:
    if fold_axis == 'y':
        remaining_paper = paper[:,:f_line].copy()
        covering_paper = np.flip(paper[:,f_line+1:].copy(), 1)
        rp_x, rp_y = remaining_paper.shape
        cp_x, cp_y = covering_paper.shape
        start_col = rp_y - f_line
        for col in range(start_col, rp_y):
            for row in  range(cp_x):
                remaining_paper[row, col] |= covering_paper[row, col]
    else:
        remaining_paper = paper[:f_line,:].copy()
        covering_paper = np.flip(paper[f_line +1:,:].copy(), 0)
        rp_x, rp_y = remaining_paper.shape
        cp_x, cp_y = covering_paper.shape
        start_row = rp_x - fold_location
        for row in range(start_row, rp_x):
            for col in  range(cp_y):
                remaining_paper[row, col] |= covering_paper[row, col]

    return remaining_paper

fold_axis = program_instructions[0][0]
fold_location = program_instructions[0][1]

folded_paper = fold_paper(program_paper, fold_axis, fold_location ).copy()
remaining_dots = np.count_nonzero(folded_paper)
print(f'After first fold there are {remaining_dots} dots showing.\n')

folded_paper = program_paper.copy()
for fold_instruction in program_instructions:
    fold_axis = fold_instruction[0]
    fold_location = fold_instruction[1]
    folded_paper = fold_paper(folded_paper, fold_axis, fold_location).copy()

map_code = ''
for letter_matrix in np.split(folded_paper,8):
    letter = np.swapaxes(letter_matrix, 0, 1) # turn the paper the correct way!!
 ↪!
    for char_line in letter:
        for char in char_line:
            if char == True:
                map_code += '*'
            else:
                map_code += ' '
```

```
        map_code += '\n'
    map_code += '\n\n'

print(map_code)
```

```
After first fold there are 664 dots showing.

****
*
***
*
*
****


****
*
***
*
*
*


  **
   *
   *
   *
*  *
 **


*  *
* *
**
* *
* *
*  *


****
   *
  *
 *
*
****


*
```

```
*
*
*
*
****


***
*  *
***
*  *
*  *
***


*
*
*
*
*
****
```