advent_of_code_day 11

December 18, 2021

1 Advent of Code

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

Day 11: Dumbo Octopus —

You enter a large cavern full of rare bioluminescent dumbo octopuses! They seem to not like the Christmas lights on your submarine, so you turn them off for now.

There are 100 octopuses arranged neatly in a 10 by 10 grid. Each octopus slowly gains **energy** over time and **flashes** brightly for a moment when its energy is full. Although your lights are off, maybe you could navigate through the cave without disturbing the octopuses if you could predict when the flashes of light will happen.

Each octopus has an **energy level** - your submarine can remotely measure the energy level of each octopus (your puzzle input). For example:

5483143223
2745854711
5264556173
6141336146
6357385478
4167524645
2176841721
6882881134
4846848554
5283751526

The energy level of each octopus is a value between 0 and 9. Here, the top-left octopus has an energy level of 5, the bottom-right one has an energy level of 6, and so on.

You can model the energy levels and flashes of light in steps. During a single step, the following occurs:

- First, the energy level of each octopus increases by 1.
- Then, any octopus with an energy level greater than 9 flashes. This increases the energy level of all adjacent octopuses by 1, including octopuses that are diagonally adjacent. If this causes an octopus to have an energy level greater than 9, it also flashes. This process continues as long as new octopuses keep having their energy level increased beyond 9. (An octopus can only flash at most once per step.)

- Finally, any octopus that flashed during this step has its energy level set to 0, as it used all of its energy to flash.

Adjacent flashes can cause an octopus to flash on a step even if it begins that step with very little energy. Consider the middle octopus with 1 energy in this situation:

Before any steps:
11111
19991
19191
19991
11111

After step 1:
34543
4**000**4
5***000***5
4**000**4
34543

After step 2:
45654
51115
61116
51115
45654

An octopus is **highlighted** when it flashed during the given step.

Here is how the larger example above progresses:

Before any steps:
5483143223
2745854711
5264556173
6141336146
6357385478
4167524645
2176841721
6882881134
4846848554
5283751526

After step 1:
6594254334
3856965822
6375667284
7252447257
7468496589
5278635756
3287952832
7993992245

5957959665
6394862637

After step 2:
88**0**7476555
5**0**89**0**87**0**54
8597889**6****0**8
8485769**6****00**
87**00**9**0**88**00**
66**000**88989
68**0000**5943
0000007456
9**000000**876
87**0000**6848

After step 3:
005**0**9**00**866
85**00**8**00**575
99**00000**39
97**00000**41
9935**0**8**00**63
7712**300000**
7911**25000**9
2211**30000**
0421125**000**
0021119**000**

After step 4:
2263**0**31977
0923**0**31697
0032221**5****0**
0041111163
0076191174
0053411122
0042361**12****0**
5532241122
1532247211
1132**23****0**211

After step 5:
4484144**000**
2**0**44144**000**
2253333493
1152333274
1187**3****0**3285
1164633233
1153472231
6643352233
2643358322
2243341322

After step 6:
5595255111
3155255222
3364444605
2263444496
2298414396
2275744344
2264583342
7754463344
3754469433
3354452433

After step 7:
6707366222
4377366333
4475555827
3496655709
3500625609
3509955566
3486694453
8865585555
4865580644
4465574644

After step 8:
7818477333
5488477444
5697666949
4608766830
4734946730
4740097688
6900007564
0000009666
8000004755
6800007755

After step 9:
9060000644
7800000976
6900000080
5840000082
5858000093
6962400000
8021250009
2221130009
9111128097
7911119976

After step 10:
0481112976

0031112**00**9
0041112**5**04
0081111**4**06
0099111**3**06
0093511233
0442361**13****0**
5532252**35****0**
0532**5**0**6****00**
00322**4****0000**

After step 10, there have been a total of 204 flashes. Fast forwarding, here is the same configuration every 10 steps:

After step 20:
3936556452
568655**6****8**06
449655569**0**
444865558**0**
445686557**0**
568**00**86577
7**000000**9896
0000000344
6**000000**364
46**0000**9543

After step 30:
0643334118
4253334611
3374333458
2225333337
2229333338
2276733333
2754574565
5544458511
9444447111
7944446119

After step 40:
6211111981
0421111119
0042111115
0003111115
0003111116
0065611111
0532351111
3322234597
2222222976
2222222762

After step 50:

9655556447
4865556805
4486555690
4458655580
4574865570
5700086566
6000009887
8000000533
6800000633
5680000538

After step 60:
2533334200
2743334640
2264333458
2225333337
2225333338
2287833333
3854573455
1854458611
1175447111
1115446111

After step 70:
8211111164
0421111166
0042111114
0004211115
0000211116
0065611111
0532351111
7322235117
5722223475
4572222754

After step 80:
1755555697
5965555609
4486555680
4458655580
4570865570
5700086566
7000008666
0000000990
0000000800
0000000000

After step 90:
7433333522
2643333522

2264333458
2226433337
2222433338
2287833333
2854573333
4854458333
3387779333
3333333333

After step 100:
0397666866
0749766918
0053976933
0004297822
0004229892
0053222877
0532222966
9322228966
7922286866
6789998766

After 100 steps, there have been a total of **1656** flashes.

Given the starting energy levels of the dumbo octopuses in your cavern, simulate 100 steps. **How many total flashes are there after 100 steps?**

1.0.1 Note

The solution below is a bit unelegent walking the map and propogating the flashes - should be removed to a separate function and tidied up. But hey, it works, and is probably as fast as moving it to it's own function.

[ ]:
```
with open("data/octopus.dat") as file:
    problem_map = file.read().splitlines()
problem_map = np.array([[int(i) for i in x] for x in problem_map], dtype=int)

test_map = [
    "5483143223",
    "2745854711",
    "5264556173",
    "6141336146",
    "6357385478",
    "4167524645",
    "2176841721",
    "6882881134",
    "4846848554",
    "5283751526",
]
test_map = np.array([[int(i) for i in x] for x in test_map], dtype=int)
```

```
octopus_data = problem_map.copy()
max_rows, max_cols = octopus_data.shape

print(f"Before any steps \n{octopus_data}\n")

total_flashes = 0
for steps in range(100):
    octopus_data += 1
    flashed = True if np.count_nonzero(octopus_data > 9) > 0 else False
    total_flashes += np.count_nonzero(octopus_data > 9)
    octopus_data = np.where(octopus_data > 9, 0, octopus_data)
    already_flashed = np.full_like(octopus_data, False)
    print(f"After step {steps +1} \n{octopus_data}\n")
    while flashed:
        flashed = False
        flashing = (octopus_data == 0) - already_flashed
        flashing_locations = np.where(flashing == True)
        for location in range(flashing_locations[0].size):
            x = flashing_locations[0][location]
            y = flashing_locations[1][location]
            if x == 0:
                if y == 0:
                    octopus_data[x, y + 1] += 1
                    octopus_data[x + 1, y] += 1
                    octopus_data[x + 1, y + 1] += 1
                else:
                    if y == 9:
                        octopus_data[x, y - 1] += 1
                        octopus_data[x + 1, y] += 1
                        octopus_data[x + 1, y - 1] += 1
                    else:
                        octopus_data[x, y + 1] += 1
                        octopus_data[x + 1, y] += 1
                        octopus_data[x + 1, y + 1] += 1
                        octopus_data[x, y - 1] += 1
                        octopus_data[x + 1, y - 1] += 1
            else:
                if x == 9:
                    if y == 0:
                        octopus_data[x, y + 1] += 1
                        octopus_data[x - 1, y] += 1
                        octopus_data[x - 1, y + 1] += 1
                    else:
                        if y == 9:
                            octopus_data[x, y - 1] += 1
                            octopus_data[x - 1, y] += 1
```

```
                            octopus_data[x - 1, y - 1] += 1
                        else:
                            octopus_data[x, y + 1] += 1
                            octopus_data[x - 1, y] += 1
                            octopus_data[x - 1, y + 1] += 1
                            octopus_data[x, y - 1] += 1
                            octopus_data[x - 1, y - 1] += 1
                else:
                    if y > 0:
                        octopus_data[x - 1, y - 1] += 1
                        octopus_data[x, y - 1] += 1
                        octopus_data[x + 1, y - 1] += 1
                    if y < 9:
                        octopus_data[x - 1, y + 1] += 1
                        octopus_data[x, y + 1] += 1
                        octopus_data[x + 1, y + 1] += 1
                    octopus_data[x - 1, y] += 1
                    octopus_data[x + 1, y] += 1

        already_flashed = (
            already_flashed + flashing
        )  # add the current flashing locations to already_flashed
        octopus_data = np.where(
            already_flashed == True, 0, octopus_data
        )  # reset any flashing octopi that may have been increased to zero

        if np.count_nonzero(octopus_data > 9) > 0:
            flashed = True
        total_flashes += np.count_nonzero(octopus_data > 9)
        octopus_data = np.where(octopus_data > 9, 0, octopus_data)
    # print(f"Step {steps+1}\n{octopus_data}\n")
print(f"Step {steps}\n{octopus_data}\n")
print(f"total_flashes: {total_flashes}")
```

Part Two

It seems like the individual flashes aren't bright enough to navigate. However, you might have a better option: the flashes seem to be **synchronizing**!

In the example above, the first time all octopuses flash simultaneously is step **195**:

After step 193:
5887777777
8877777777
7777777777
7777777777
7777777777
7777777777
7777777777

7777777777
7777777777
7777777777

After step 194:
6988888888
9988888888
8888888888
8888888888
8888888888
8888888888
8888888888
8888888888
8888888888
8888888888

After step 195:
0000000000
0000000000
0000000000
0000000000
0000000000
0000000000
0000000000
0000000000
0000000000
0000000000

If you can calculate the exact moments when the octopuses will all flash simultaneously, you should be able to navigate through the cavern. **What is the first step during which all octopuses flash?**

[ ]:
```
with open("data/octopus.dat") as file:
    problem_map = file.read().splitlines()
problem_map = np.array([[int(i) for i in x] for x in problem_map], dtype=int)

test_map = [
    "5483143223",
    "2745854711",
    "5264556173",
    "6141336146",
    "6357385478",
    "4167524645",
    "2176841721",
    "6882881134",
    "4846848554",
    "5283751526",
]
test_map = np.array([[int(i) for i in x] for x in test_map], dtype=int)
```

```
octopus_data = problem_map.copy()
max_rows, max_cols = octopus_data.shape
octopus_population = octopus_data.size

# print(f"Before any steps \n{octopus_data}\n")

total_flashes = 0
all_flashed = False
max_iterations = 400  # set a limit just in case
step = 0
while not all_flashed:
    step += 1
    octopus_data += 1
    flashed = True if np.count_nonzero(octopus_data > 9) > 0 else False
    total_flashes += np.count_nonzero(octopus_data > 9)
    octopus_data = np.where(octopus_data > 9, 0, octopus_data)
    already_flashed = np.full_like(octopus_data, False)
    while flashed:
        flashed = False
        flashing = (octopus_data == 0) - already_flashed
        flashing_locations = np.where(flashing == True)
        for location in range(flashing_locations[0].size):
            x = flashing_locations[0][location]
            y = flashing_locations[1][location]
            if x == 0:
                if y == 0:
                    octopus_data[x, y + 1] += 1
                    octopus_data[x + 1, y] += 1
                    octopus_data[x + 1, y + 1] += 1
                else:
                    if y == 9:
                        octopus_data[x, y - 1] += 1
                        octopus_data[x + 1, y] += 1
                        octopus_data[x + 1, y - 1] += 1
                    else:
                        octopus_data[x, y + 1] += 1
                        octopus_data[x + 1, y] += 1
                        octopus_data[x + 1, y + 1] += 1
                        octopus_data[x, y - 1] += 1
                        octopus_data[x + 1, y - 1] += 1
            else:
                if x == 9:
                    if y == 0:
                        octopus_data[x, y + 1] += 1
                        octopus_data[x - 1, y] += 1
                        octopus_data[x - 1, y + 1] += 1
```

```
                    else:
                        if y == 9:
                            octopus_data[x, y - 1] += 1
                            octopus_data[x - 1, y] += 1
                            octopus_data[x - 1, y - 1] += 1
                        else:
                            octopus_data[x, y + 1] += 1
                            octopus_data[x - 1, y] += 1
                            octopus_data[x - 1, y + 1] += 1
                            octopus_data[x, y - 1] += 1
                            octopus_data[x - 1, y - 1] += 1
                else:
                    if y > 0:
                        octopus_data[x - 1, y - 1] += 1
                        octopus_data[x, y - 1] += 1
                        octopus_data[x + 1, y - 1] += 1
                    if y < 9:
                        octopus_data[x - 1, y + 1] += 1
                        octopus_data[x, y + 1] += 1
                        octopus_data[x + 1, y + 1] += 1
                    octopus_data[x - 1, y] += 1
                    octopus_data[x + 1, y] += 1

        already_flashed = (
            already_flashed + flashing
        )  # add the current flashing locations to already_flashed
        octopus_data = np.where(
            already_flashed == True, 0, octopus_data
        )  # reset any flashing octopi that may have been increased to zero

        if np.count_nonzero(octopus_data > 9) > 0:
            flashed = True
        total_flashes += np.count_nonzero(octopus_data > 9)
        octopus_data = np.where(octopus_data > 9, 0, octopus_data)
    # print(f"After step {step} \n{octopus_data}\n")
    if np.count_nonzero(octopus_data == 0) == octopus_population:
        all_flashed = True

print(f"On step {step} all the octopus population flashed")
print(f"total_flashes: {total_flashes}")
```

```
On step 354 all the octopus population flashed
total_flashes: 5607
```

1.0.2 Note

This one was a real devil as it kept giving the wrong answer although I couldn't find a fault in my logic. Turned out I was checking for the octopi all flasing at the start of the run instead of at the

end so was missing the octopi that flashed due to a neighbour flashing.