advent_of_code_day 03 part 1

December 8, 2021

1 Advent of Code

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

1.1 Day 3: Binary Diagnostic

1.1.1 Part 1

The submarine has been making some odd creaking noises, so you ask it to produce a diagnostic report just in case.

The diagnostic report (your puzzle input) consists of a list of binary numbers which, when decoded properly, can tell you many useful things about the conditions of the submarine. The first parameter to check is the power consumption.

You need to use the binary numbers in the diagnostic report to generate two new binary numbers (called the gamma rate and the epsilon rate). The power consumption can then be found by multiplying the gamma rate by the epsilon rate.

Each bit in the gamma rate can be determined by finding the most common bit in the corresponding position of all numbers in the diagnostic report. For example, given the following diagnostic report:

```
00100
11110
10110
10111
10101
01111
00111
11100
10000
11001
00010
01010
```

Considering only the first bit of each number, there are five 0 bits and seven 1 bits. Since the most common bit is 1, the first bit of the gamma rate is 1.

The most common second bit of the numbers in the diagnostic report is 0, so the second bit of the gamma rate is 0.

The most common value of the third, fourth, and fifth bits are 1, 1, and 0, respectively, and so the final three bits of the gamma rate are 110.

So, the gamma rate is the binary number 10110, or 22 in decimal.

The epsilon rate is calculated in a similar way; rather than use the most common bit, the least common bit from each position is used. So, the epsilon rate is 01001, or 9 in decimal. Multiplying the gamma rate (22) by the epsilon rate (9) produces the power consumption, 198.

Use the binary numbers in your diagnostic report to calculate the gamma rate and epsilon rate, then multiply them together. What is the power consumption of the submarine? (Be sure to represent your answer in decimal, not binary.)

[ ]:
```
# Import the data into a text array - easier for getting positional data
diagnostics = np.genfromtxt("data/diagnostic.dat", dtype="U")
print(diagnostics[:10])
print(len(diagnostics))
```

```
['111010101100' '100001001100' '000111101100' '100100000000'
 '001001001110' '100110101011' '001001100101' '010000010110'
 '011011001001' '001001000101']
1000
```

[ ]:
```
# get the number of bits in the data
bits = len(diagnostics[0])
print(bits)
```

```
12
```

[ ]:
```
epsilon_str = ""
gamma_str = ""

for x in range(len(diagnostics[0])):
    gamma_val = 0
    for y in diagnostics:
        if y[x] == '1':
            gamma_val += 1
        else:
            gamma_val -= 1

# take advantage of the fact that the epsilon value == not(gamma value)
# (could be done later on the whole string but just as easy to build it now)
    if gamma_val > 0:
        gamma_str += '1'
        epsilon_str +=  '0'
    else:
        gamma_str += '0'
        epsilon_str += '1'

# convert the base 2 strings to integers
```

```python
gamma_rate = int(gamma_str, 2)
epsilon_rate = int(epsilon_str, 2)

power_consumptions = gamma_rate * epsilon_rate

print("Gama rate: ", gamma_rate)
print("Epsilon rate: ", epsilon_rate)
print("Power Consumption: ", power_consumptions)
```

```
Gama rate:  3903
Epsilon rate:  192
Power Consumption:  749376
```