Day 4

Part 1

"Looks like the Chief's not here. Next!" One of The Historians pulls out a device and pushes the only button on it. After a brief flash, you recognize the interior of the Ceres monitoring station!

As the search for the Chief continues, a small Elf who lives on the station tugs on your shirt; she'd like to know if you could help her with her word search (your puzzle input). She only has to find one word: `XMAS`.

This word search allows words to be horizontal, vertical, diagonal, written backwards, or even overlapping other words. It's a little unusual, though, as you don't merely need to find one instance of `XMAS` - you need to find all of them. Here are a few ways `XMAS` might appear, where irrelevant characters have been replaced with .:

```
..X...
.SAMX.
.A..A.
XMAS.S
.X....
```

The actual word search will be full of letters instead. For example:

```
MMMSXXMASM
MSAMXMSMSA
AMXSXMAAMM
MSAMASMSMX
XMASAMXAMM
XXAMMXXAMA
SMSMSASXSS
SAXAMASAAA
MAMMMXMMMM
MXMXAXMASX
```

In this word search, `XMAS` occurs a total of 18 times; here's the same word search again, but where letters not involved in any `XMAS` have been replaced with .:

```
....XXMAS.
.SAMXMS...
...S..A...
..A.A.MS.X
XMASAMX.MM
X.....XA.A
S.S.S.S.SS
.A.A.A.A.A
..M.M.M.MM
.X.X.XMASX
```

Take a look at the little Elf's word search. How many times does `XMAS` appear?

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

import numpy as np
```

In [2]:
```
test_path = Path("data/day04_test.txt")
data_path = Path("data/day04_data.txt")

word = "XMAS"

def load_data(fpath: Path) -> np.array:
    """Returns grid as numpy array."""
    data = fpath.open().readlines()
    data = [list(_.strip()) for _ in data]
    return np.array(data)

def count_word(grid: np.array, word: str) -> list[tuple[int, int, int]]:
    """Returns list of counts of the word in the grid.

    Tuple has format (count, start row, start column)
    """
    nrows, ncols = grid.shape

    hits = []  # holds located words

    for row in range(nrows):
        for col in range(ncols):
            if grid[row][col] == word[0]:  # found word start
                locals = [word[0]] * 8
                for idx in range(1, len(word)):
                    # Get letters outwards in all eight directions from the start
                    check = [grid[row-idx if row-idx >= 0 else 0][col],
                             grid[row-idx if row-idx >= 0 else 0][col+idx if col+idx
                             grid[row][col+idx if col+idx < ncols else ncols-1],
                             grid[row+idx if row+idx < nrows else nrows-1][col+idx if
                             grid[row+idx if row+idx < nrows else nrows-1][col],
                             grid[row+idx if row+idx < nrows else nrows-1][col-idx if
                             grid[row][col-idx if col-idx >= 0 else 0],
                             grid[row-idx if row-idx >= 0 else 0][col-idx]]
                    # If we overran the grid, substitute the letter with a blank
                    if row - idx < 0:
                        check[0] = check[1] = check[7] = ""
                    if row + idx >= nrows:
                        check[3] = check[4] = check[5] = ""
                    if col - idx < 0:
                        check[5] = check[6] = check[7] = ""
                    if col + idx >= ncols:
                        check[1] = check[2] = check[3] = ""
                    locals = [_[0] + _[1] for _ in zip(locals, check)]
                if word in locals:  # Count how many times the assembled word is spel
                    hits.append((locals.count(word), row, col))

    return hits


grid = load_data(test_path)
hits = count_word(grid, word)
sum([_[0] for _ in hits])
```

Out[2]: 18

In [3]:
```
grid = load_data(data_path)
hits = count_word(grid, word)
sum([_[0] for _ in hits])
```

Out[3]: 2447

Part 2

The Elf looks quizzically at you. Did you misunderstand the assignment?

Looking for the instructions, you flip over the word search to find that this isn't actually an XMAS puzzle; it's an `X-MAS` puzzle in which you're supposed to find two MAS in the shape of an X. One way to achieve that is like this:

```
M.S
.A.
M.S
```

Irrelevant characters have again been replaced with . in the above diagram. Within the `X`, each `MAS` can be written forwards or backwards.

Here's the same example from before, but this time all of the `X-MAS`es have been kept instead:

```
.M.S......
..A..MSMS.
.M.S.MAA..
..A.ASMSM.
.M.S.M....
..........
S.S.S.S.S.
.A.A.A.A..
M.M.M.M.M.
..........
```

In this example, an `X-MAS` appears `9` times.

Flip the word search from the instructions back over to the word search side and try again. How many times does an `X-MAS` appear?

In [4]:
```python
def count_xmas(grid: np.array) -> list[tuple]:
    """Returns locations of X-MAS patterns.

    Tuple has format (centre row, centre column)
    """
    nrows, ncols = grid.shape

    hits = []

    for row in range(1, nrows-1):
        for col in range(1, ncols-1):
            if grid[row][col] == "A":  # found possible centre
                opt1 = grid[row-1][col-1] + grid[row][col] + grid[row+1][col+1]
                opt2 = grid[row+1][col-1] + grid[row][col] + grid[row-1][col+1]
                if opt1 in ("SAM", "MAS") and opt2 in ("SAM", "MAS"):
                    hits.append((row, col))

    return(hits)

grid = load_data(test_path)
hits = count_xmas(grid)
len(hits)
```

Out[4]: 9

In [5]:
```python
grid = load_data(data_path)
hits = count_xmas(grid)
len(hits)
```

Out[5]: 1868

In [ ]: