# Day 01

## Part 1

You haven't even left yet and the group of Elvish Senior Historians has already hit a problem: their list of locations to check is currently empty. Eventually, someone decides that the best place to check first would be the Chief Historian's office.

Upon pouring into the office, everyone confirms that the Chief Historian is indeed nowhere to be found. Instead, the Elves discover an assortment of notes and lists of historically significant locations! This seems to be the planning the Chief Historian was doing before he left. Perhaps these notes can be used to determine which locations to search?

Throughout the Chief's office, the historically significant locations are listed not by name but by a unique number called the location ID. To make sure they don't miss anything, The Historians split into two groups, each searching the office and trying to create their own complete list of location IDs.

There's just one problem: by holding the two lists up side by side (your puzzle input), it quickly becomes clear that the lists aren't very similar. Maybe you can help The Historians reconcile their lists?

For example:

```
3   4
4   3
2   5
1   3
3   9
3   3
```

Maybe the lists are only off by a small amount! To find out, pair up the numbers and measure how far apart they are. Pair up the smallest number in the left list with the smallest number in the right list, then the second-smallest left number with the second-smallest right number, and so on.

Within each pair, figure out how far apart the two numbers are; you'll need to add up all of those distances. For example, if you pair up a 3 from the left list with a 7 from the right list, the distance apart is 4; if you pair up a 9 with a 3, the distance apart is 6.

In the example list above, the pairs and distances would be as follows:

- The smallest number in the left list is 1, and the smallest number in the right list is 3. The distance between them is 2.
- The second-smallest number in the left list is 2, and the second-smallest number in the right list is another 3. The distance between them is 1.
- The third-smallest number in both lists is 3, so the distance between them is 0.
- The next numbers to pair up are 3 and 4, a distance of 1.
- The fifth-smallest numbers in each list are 3 and 5, a distance of 2.
- Finally, the largest number in the left list is 4, while the largest number in the right list is 9; these are a distance 5 apart.

To find the total distance between the left list and the right list, add up the distances between all of the pairs you found. In the example above, this is `2 + 1 + 0 + 1 + 2 + 5`, a total distance of `11`!

Your actual left and right lists contain many location IDs. What is the total distance between your lists?

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

In [2]:
```
test_path = Path("data/day01_test.txt")
data_path = Path("data/day01_data.txt")

def load_data(fpath: Path) -> tuple[list]:
    """Returns left/right lists, read from file."""
    left, right = [], []  # Lists to hold left/right values

    for line in fpath.open():  # Populate left/right lists
        lval, rval = (int(_) for _ in line.split())
        left.append(lval)
        right.append(rval)

    return left, right

def distance(left: list, right: list) -> int:
    """Returns the total distance between left/right lists.

    Both left and right lists are sorted in ascending order. Then the
    absolute difference between left and right values is calculated for
    each pair, and the sum of differences is returned.
    """
    return sum([abs(l - r) for l, r in zip(sorted(left), sorted(right))])

# Test data
left, right = load_data(test_path)
distance(left, right)
```

Out[2]: 11

In [3]:
```
# Puzzle data
left, right = load_data(data_path)
distance(left, right)
```

Out[3]: 1660292

## Part 2

Your analysis only confirmed what everyone feared: the two lists of location IDs are indeed very different.

Or are they?

The Historians can't agree on which group made the mistakes or how to read most of the Chief's handwriting, but in the commotion you notice an interesting detail: a lot of location IDs appear in both lists! Maybe the other numbers aren't location IDs at all but rather misinterpreted handwriting.

This time, you'll need to figure out exactly how often each number from the left list appears in the right list. Calculate a total similarity score by adding up each number in the left list after multiplying it by the number of times that number appears in the right list.

Here are the same example lists again:

```
3   4
4   3
2   5
1   3
3   9
3   3
```

For these example lists, here is the process of finding the similarity score:

- The first number in the left list is 3. It appears in the right list three times, so the similarity score increases by `3 * 3 = 9`.
- The second number in the left list is 4. It appears in the right list once, so the similarity score increases by `4 * 1 = 4`.
- The third number in the left list is 2. It does not appear in the right list, so the similarity score does not increase `(2 * 0 = 0)`.
- The fourth number, 1, also does not appear in the right list.
- The fifth number, 3, appears in the right list three times; the similarity score increases by `9`.
- The last number, 3, appears in the right list three times; the similarity score again increases by `9`.

So, for these example lists, the similarity score at the end of this process is `31` `(9 + 4 + 0 + 0 + 9 + 9)`.

Once again consider your left and right lists. What is their similarity score?

In [4]:
```python
def sim_score(left: list, right: list) -> int:
    """Returns the similarity score between left/right lists.

    Iterates over the left list and maintains a score that is the sum
    of left list values multiplied by the number of times that value
    occurs in the right list.
    """
    score = 0  # Holds similarity score
    for val in left:  # Iterate over left list
        score += val * right.count(val)  # Update score

    return score

# Test data
left, right = load_data(test_path)
sim_score(left, right)
```

Out[4]: 31

In [5]:
```python
# Puzzle data
left, right = load_data(data_path)
sim_score(left, right)
```

Out[5]: 22776016

In [ ]:
```
```