Daily Coding Problem #191

Problem

This problem was asked by Stripe.

Given a collection of intervals, find the minimum number of intervals you need to remove to make the rest of the intervals non-overlapping.

Intervals can "touch", such as `[0, 1]` and `[1, 2]`, but they won't be considered overlapping.

For example, given the intervals `(7, 9), (2, 4), (5, 8)`, return 1 as the last interval can be removed and the first two won't overlap.

The intervals are not necessarily sorted in any order.

Solution

We can do this problem greedily. If we sort the intervals by the end time, and continuously choose the first interval that ends first, we'll minimize the number of overlapping intervals with that one. We can compute overlapping intervals quickly by just keeping track of the end of the last interval we haven't overlapped with:

```
from math import inf

def non_overlapping_intervals(intervals):
    current_end = -inf
    overlapping = 0

    for start, end in sorted(intervals, key=lambda i: i[1]):
        if start >= current_end:
            current_end = end
        else:
            overlapping += 1
    return overlapping
```

This takes O(n log n) time, since we have to sort the intervals. It also takes O(1) time.