Daily Coding Problem #241

Problem

This problem was asked by Palantir.

In academia, the h-index is a metric used to calculate the impact of a researcher's papers. It is calculated as follows:

A researcher has index `h` if at least `h` of her `N` papers have `h` citations each. If there are multiple `h` satisfying this formula, the maximum is chosen.

For example, suppose `N = 5`, and the respective citations of each paper are `[4, 3, 0, 1, 5]`. Then the h-index would be 3, since the researcher has `3` papers with at least `3` citations.

Given a list of paper citations of a researcher, calculate their h-index.

Solution

The simplest way to solve this is to sort the papers in decreasing order by number of citations, and find the last index such that the value at that index is at least as great as the index.

In the example above, for example, `[5, 4, 3, 1, 0]`, we see that `citations[3] >= 3`, whereas `citations[4] < 4`.

```
def h_index(citations):
    n = len(citations)
    citations.sort(reverse=True)

    h = 0
    while h < n and citations[h] >= h + 1:
        h += 1

    return h
```

However, sorting the citations is `O(N * log N)`. A more efficient solution is to bucket sort the citations, putting all citations with count greater than `N` in their own bucket. Then, we can descend from `N` to `0`, accumulating the counts for each bucket. The accumulated total at a given index represents the number of papers with at least that many citations. So once we reach a point where `total >= index`, we have found our answer.

In the example above, `counts` would be `[1, 1, 0, 1, 1, 1]`, and the total for the third index would be `3`.

```
def h_index(citations):
    n = len(citations)
    counts = [0 for _ in range(n + 1)]

    for citation in citations:
        if citation >= n:
            counts[n] += 1
        else:
            counts[citation] += 1

    total = 0
    for i in range(n, -1, -1):
        total += counts[i]
        if total >= i:
            return i
```

The bucket sort and countdown each take `O(N)` time, so this algorithm is `O(N)`.