Daily Coding Problem #251

Problem

This problem was asked by Amazon.

Given an array of a million integers between zero and a billion, out of order, how can you efficiently sort it? Assume that you cannot store an array of a billion elements in memory.

Solution

Sorting by an algorithm like quicksort or merge sort would give us an average time complexity of `O(N log N)`. But we can take advantage of the fact that our input is bounded and only consists of integers to do even better. One algorithm that can perform particularly well in these cases is called radix sort.

To see how this works, suppose we have a list of non-negative numbers, such as `[4, 100, 54, 537, 2, 89]`, and we know ahead of time that no number has more than three digits. Then we can (stably) sort our list using three passes, corresponding to each digit:

- First, we order by the ones' place, giving us `[100, 2, 4, 54, 537, 89]`.
- Next, we order by the tens' place, giving us `[100, 2, 4, 537, 54, 89]`.
- Finally, we order by the hundreds' place, giving us `[2, 4, 54, 89, 100, 537]`.

Note that if a given number doesn't have a tens' or hundreds' place, we assign that place value zero.

Each of these sorts is performed using counting sort, which gets around the efficiency limits of comparison sorts like quicksort. In counting sort, we bucket each number into an array of size equal to our base, in this case `10`. Then, we read off the elements in each bucket, in order, to get the new sorted array.

We can implement this as follows:

```
def counting_sort(array, digit, base=10):
    counts = [[] for _ in range(base)]

    for num in array:

        d = (num // base ** digit) % base
        counts[d].append(num)

    result = []
    for bucket in counts:
        result.extend(bucket)

    return result

def radix_sort(array, digits=10):
    for digit in range(digits):
        array = counting_sort(array, digit)

    return array
```

Counting sort takes `O(N + M)`, where `M` is the maximum bucket size. For our problem, we can consider this to be `O(N)`. We must perform this sort k times, where k is the number of digits in the maximum number, in this case `10`. The time complexity of this algorithm is therefore `O(10 * N)`, or if we treat k as constant, `O(N)`.