Daily Coding Problem #228

Problem

This problem was asked by Twitter.

Given a list of numbers, create an algorithm that arranges them in order to form the largest possible integer. For example, given `[10, 7, 76, 415]`, you should return `77641510`.

Solution

At first glance the answer might seem simple: just sort the numbers in decreasing order, lexicographically, and then concatenate them. But we can see from our input that this does not quite work: since `76 > 7`, we would end up with `76741510`.

One way to get around this issue is to look instead at the concatenation of compared elements. In the previous example, the two options for combining these numbers are `'7' + '76'` and `'76' + '7'`. Since `'776' > '767'`, we should prefer the first one. We can therefore create a custom comparison function to plug into our sort that carries out this logic. Once we have sorted the

list accordingly, we can join the elements together to form the largest possible integer.

```
def compare(a, b):
    if str(a) + str(b) > str(b) + str(a):
        return -1
    else:
        return 1

def get_largest_value(nums):
    return ''.join([str(i) for i in sorted(nums, cmp=compare)])
```

Unfortunately, this code only works in `Python 2`, as `cmp` is deprecated in `Python 3`. While there are recommended workarounds, we can use this as an excuse to explore a different solution.

Again, we must come up with a comparison method that somehow places 7 before 76. Perhaps repeating the last digit of each integer until all integers are the same length would work: in this case, since `77 > 76`, this would return the correct answer. However, this wouldn't work for a comparison of 513 and 513444, since we would like 513 to come first.

A better solution is to repeat the entire number. In other words, we should concatenate each number with itself until reaching the maximum length (slicing the end off if necessary). For the example above, when we compare 513513 and 513444, we can see that the former is correctly placed first.

Once we have a list of these normalized numbers, we can sort our original input by decreasing order of our new list, and return the joined result.

```
def get_largest_value(nums):
    nums = [str(x) for x in nums]
    length = len(max(nums, key=len))

    normalized = []
    for i, x in enumerate(nums):
        element = x * (length // len(x) + 1)
```

```
        normalized.append(element[:length])

    ordered = sorted(zip(nums, normalized), key=lambda x: x[1], reverse=True)

    return ''.join([x[0] for x in ordered])
```

Both of these algorithm take `O(N * log N)` time to sort the input, and use `O(N)` extra space.
