Daily Coding Problem #118

Problem

This problem was asked by Google.

Given a sorted list of integers, square the elements and give the output in sorted order.

For example, given [-9, -2, 0, 2, 3], return [0, 4, 4, 9, 81].

Solution

A brute force method would be to simply square and sort the list, like so: `sorted([x ** 2 for x in lst])`. This would result in O(n log n) time.

A fater way to do this would be to notice that there are two natural sublists in `lst`: The positive numbers and negative numbers.

The positive numbers, if sorted, would still remain sorted, while negative numbers, if sorted, would be reverse sorted. So by reversing the negative numbers and then sorting it we get two sorted sections in `lst`. Then we can apply a `merge`

operation, similar to merge sort.

```
def square_sort(lst):
    negatives = [x for x in lst if x < 0]
    non_negatives = [x for x in lst if x >= 0]

    negatives_square_sorted = [x ** 2 for x in reversed(negatives)]
    non_negatives_square_sorted = [x ** 2 for x in non_negatives]

    return _merge(negatives_square_sorted, non_negatives_square_sorted)


def _merge(left_lst, right_lst):
    result = []

    i = j = 0

    while i < len(left_lst) and j < len(right_lst):
        if left_lst[i] < right_lst[j]:
            result.append(left_lst[i])
            i += 1
        elif left_lst[i] > right_lst[j]:
            result.append(right_lst[j])
            j += 1
        else:
            result.append(left_lst[i])
            result.append(right_lst[j])
            i += 1
            j += 1
```

```
    result.extend(left_lst[i:])
    result.extend(right_lst[j:])
    return result
```

This takes O(N) time.
