Daily Coding Problem #155

Problem

This problem was asked by MongoDB.

Given a list of elements, find the majority element, which appears more than half the time (> `floor(len(lst) / 2.0)`).

You can assume that such element exists.

For example, given `[1, 2, 1, 1, 3, 4, 0]`, return 1.

Solution

One way to solve this problem is to simply use a hashmap to store the mapping of an element to its frequency. Each insertion would take O(1) for a total of O(n) time. At the end of the loop, we could query for the element which has the highest frequency:

```
def majority(elements):
    element_to_count = {}
    for element in elements:
        if element not in element_to_count:
            element_to_count[element] = 0
        element_to_count[element] += 1
    # Find the element with most count
    return max(element_to_count, key=element_to_count.get)
```

Another way is to use a voting strategy. We keep the current majority element as well as their frequency during the loop. When we see an element that's the same as the majority, we vote or increment the frequency. Otherwise, we decrement it.

```
def majority(elements):
    for i, e in enumerate(elements):
        if i == 0 or count == 0:
```

```
            majority = e
            count = 1
        elif majority == e:
            count += 1
        else:
            count -= 1
    return majority
```

We know the majority algorithm exists, so we must increment more than we decrement. Since the final count must be positive, we must have found the majority element.