Daily Coding Problem #273

Problem

This problem was asked by Apple.

A fixed point in an array is an element whose value is equal to its index. Given a sorted array of distinct elements, return a fixed point, if one exists. Otherwise, return `False`.

For example, given `[-6, 0, 2, 40]`, you should return `2`. Given `[1, 5, 7, 8]`, you should return `False`.

Solution

We can trivially solve this in `O(N)` time by going through the array and checking whether each value is equal to its index.

```
def fixed_point(array):
    for i in range(len(array)):
        if array[i] == i:
            return i

    return False
```

To improve the time complexity of our solution, we can use binary search

To improve the time complexity of our solution, we can use binary search.

First, we will check the middle element of the array. If the value of this element is less than its index, we know that all the values earlier in the list will also be less than their respective indices. This is because the elements are ordered and distinct, so that values must decrease as we move backwards through the list. As a result, we know the solution will be in the right half of the array, if it exists. Similarly, if the value of the middle element is greater than its index, we can deduce that the solution must be in the left half of the array.

We continue narrowing down the solution window in this manner until we find a satisfactory element, or until there are no more elements to search, in which case we return `False`.

```
def fixed_point(array):
    low, high = 0, len(array)

    while low <= high:

        mid = (low + high) // 2
        if array[mid] == mid:
            return mid
        elif array[mid] < mid:
            low = mid + 1
        else:
            high = mid - 1

    return False
```

The complexity of this solution is the same as that of binary search, `O(log N)`.

Press