Daily Coding Problem #314

Problem

This problem was asked by Spotify.

You are the technical director of WSPT radio, serving listeners nationwide. For simplicity's sake we can consider each listener to live along a horizontal line stretching from `0` (west) to `1000` (east).

Given a list of `N` listeners, and a list of `M` radio towers, each placed at various locations along this line, determine what the minimum broadcast range would have to be in order for each listener's home to be covered.

For example, suppose `listeners = [1, 5, 11, 20]`, and `towers = [4, 8, 15]`. In this case the minimum range would be `5`, since that would be required for the tower at position `15` to reach the listener at position `20`.

Solution

For each listener, we must find the closest tower that could provide a radio broadcast. One way we can do this is to sort the towers and perform a series of binary searches. That is, we can find where each listener lies along this sorted list, and determine the distance of their closest tower. The distance that suffices for the worst-case listener will be the overall solution.

To make things more convenient, we can add an imaginary tower at negative infinity on the left, and at positive infinity on the right, so that there will always be two towers to choose from for each listener.

```
def search(listener, towers):
    start = 0; end = len(towers) - 1

    while start < end:
        mid = start + (end - start) // 2

        if towers[mid] < listener:
            start = mid + 1
        elif towers[mid] > listener:
            end = mid
        else:
            return mid

    return start

def find_broadcast_range(listeners, towers):
    min_range = 0
    towers = [-float('inf')] + sorted(towers) + [float('inf')]

    for listener in listeners:
        idx = search(listener, towers)

        left = listener - towers[idx - 1]
        right = towers[idx] - listener

        min_range = max(min_range, min(left, right))

    return min_range
```

To analyze the time complexity of this solution, we must take into account two parts: sorting the list of towers, and binary searching this list for each listener. Given `M` towers and `N` listeners, this will be `O((M + N) log M)`.

Alternatively, if the listeners and towers are already ordered, there is a more straightforward approach. For each listener, we increment an index corresponding to the closest tower to the left, and find the closer of the left and right towers. As in the above

solution, we update our minimum range whenever we come across a larger minimum distance.

```
def find_broadcast_range(listeners, towers):
    min_range = 0; i = 0
    towers = [-float('inf')] + towers + [float('inf')]

    for listener in listeners:
        while listener > towers[i + 1]:
            i += 1

        curr = min(listener - towers[i], towers[i + 1] - listener)
        min_range = max(min_range, curr)

    return min_range
```

This approach will take `O(M + N)` time, since we only need to make one pass through each list.