Daily Coding Problem #192

Problem

This problem was asked by Google.

You are given an array of nonnegative integers. Let's say you start at the beginning of the array and are trying to advance to the end. You can advance at most, the number of steps that you're currently on. Determine whether you can get to the end of the array.

For example, given the array `[1, 3, 1, 2, 0, 1]`, we can go from indices `0 -> 1 -> 3 -> 5`, so return `true`.

Given the array `[1, 2, 1, 0, 0]`, we can't reach the end, so return `false`.

Solution

It's tempting to use a greedy strategy and to try to immediately take the largest step we see. However, fails since it might get stuck into a local optimum. Consider `[2, 2, 0, 0]`: it's better to not take the first 2 and instead do `0 -> 1 -> 3`.

Instead, the basic idea is this: we keep track of the absolute possible furthest step we can reach, and not only compute the furthest step we can reach from the current step but all steps in between as well. This works because each step is stateless and gets "reset" whenever you land on a new index. We also can't look past the furthest step, so break if we're past it.

```
def can_reach_end(arr):
    furthest_so_far = 0
    for i in range(len(arr)):
        if i > furthest_so_far:
            break
        furthest_so_far = max(furthest_so_far, i + arr[i])
    return furthest_so_far >= len(arr) - 1
```

This takes O(n) time and constant space.
