Flags

It's time to show you how the Codility challenge code-named Boron can be solved. You can still give it a try, but no certificate will be granted. The problem asks for the maximum number of flags that can be set on mountain peaks.

Fast solution $O(N \log N)$

The result can be found by bisection. If we know that x flags can be set, then we also know that $x-1, x-2, \ldots, 1$ flags can be set. Otherwise, if x flags cannot be set, then $x+1, x+2, \ldots, \sqrt{N}$ flags cannot be set either. Using bisection we can reduce the problem to checking whether x flags can be set. Notice that we can always greedily set a flag on the first peak.

Let's create an array, *peaks*, to specify whether each element i is a peak.

1: Create an array of peaks — $O(N)$.

```
def create_peaks(A):
    N = len(A)
    peaks = [False] * N
    for i in xrange(1, N - 1):
        if A[i] > max(A[i - 1], A[i + 1]):
            peaks[i] = True
    return peaks
```

The time complexity of creating an array of peaks is $O(N)$.

2: Check whether x flags can be set — $O(N)$.

```
def check(x, A):
    N = len(A)
    peaks = create_peaks(A)
    flags = x
    pos = 0
    while pos < N and flags > 0:
        if peaks[pos]:
            flags -= 1
            pos += x
        else:
            pos += 1
    return flags == 0
```

The time complexity of the function *check* is $O(N)$, so the total time complexity is $O(N \log N)$ due to the bisection time.

Golden solution $O(N)$

Firstly, we mark all the peaks. Then, by scanning the array, for every index i we can find the first peak located at an index $\geqslant i$. Let us define its position by $next[i]$. We just iterate through all the indices in reverse order and remember the earliest peak.

3: Next peak — $O(N)$.

```
def next_peak(A):
    N = len(A)
    peaks = create_peaks(A)
    next = [0] * N
    next[N - 1] = -1
    for i in xrange(N - 2, -1, -1):
        if peaks[i]:
            next[i] = i
        else:
            next[i] = next[i + 1]
    return next
```

Let us assume that we have taken i flags. Notice that if we set a flag at position pos then the next flag can only be set in positions $\geqslant pos + i$. The position can be found in a constant time (from array $next$).

4: Golden solution — $O(N)$.

```
def flags(A):
    N = len(A)
    next = next_peak(A)
    i = 1
    result = 0
    while (i - 1) * i <= N:
        pos = 0
        num = 0
        while pos < N and num < i:
            pos = next[pos]
            if pos == -1:
                break
            num += 1
            pos += i
        result = max(result, num)
        i += 1
    return result
```

Notice that for every index i we cannot take more than i flags and set more than $\frac{N}{i} + 1$ flags. We can take a maximum of $O(\sqrt{N})$ flags, and the position of each of them can be found in a constant time, so the total number of operations does not exceed $O(N + 1 + 2 + \ldots + \sqrt{N}) = O(N + \sqrt{N}^2) = O(N)$.