Cannonballs

It is time to show how the Codility Certificate codenamed X (Chi) can be solved. You can still give it a try, but no certificate will be granted.

In this task, one has to simulate flying cannonballs and find places where they hit the ground. The height of the ground, extending from the cannon outwards, is stored in array A, and the levels at which the cannonballs are shot are stored in array B. For each cannonball shot at level H, we have to find the smallest index I such that $0 < I < M$ and $A[I] \geqslant H$. (If there is no such I a cannonball can be ignored.) Moreover, values in array A change, as $A[I-1]$ is increased when the cannonball falls.

We need a data structure that can quickly answer queries of the form *Where does the cannonball shot at level H hit the ground?* and that can be updated by increasing ground levels accordingly. Such a data structure can consist of two arrays: the array A representing ground levels and another array T such that $T[H]$ is the smallest value of I to satisfy $0 < I < M$ and $A[I] \geqslant H$ (or -1 if there is no such I).

Initially, we can calculate array T in a linear time. The key observation is that if $H_1 \leqslant H_2$ then $T[H_1] \leqslant T[H_2]$. In other words, the higher a cannonball flies, the greater the distance it will travel. If $T[H] = I$, then $T[H+1]$ can be calculated by checking values of array A from $A[I+1]$ on.

When a cannonball is shot, we can instantly locate the place where it hits the ground by using $T[H] = I$. However, after increasing $A[I-1]$, we have to decrease values in array T accordingly. Luckily, only one element of T can change: namely, if $T[A[I-1]] > I-1$, then it should be decreased to $I-1$. Clearly, one cannonball can be processed in constant time. Hence, the overall time complexity of this algorithm is linear.

Here is a solution in Python, resulting from the above analysis:

1: Model solution — $O(H + M + N)$

```
def cannonballs(A, B):
  M = len(A)
  N = len(B)
  H = max(B)

  T = [-1] * (H+1)
  i = 0
  for j in xrange(H+1):
    while (i < M) and (A[i] < j):
      i += 1
    T[j] = i

  for j in B:
    i = T[j]
    if 0 < i < M:
      A[i-1] += 1
      if T[A[i-1]] > i-1:
        T[A[i-1]] = i-1

  return A
```