Prefix Set

by Peng Cao

codility
WE TEST CODERS

It's time to show you how the Codility Challenge codenamed (Alpha) can be solved. You can still give it a try, but no certificate will be granted. The problem asks for the shortest prefix that contains all the numbers in a given array.

Slow solution $O(n^2)$

For every prefix p we have to decide whether or not a number $A[p]$ has appeared before. If the number $A[p]$ has appeared before then we can consider smaller indices. Otherwise, it is not possible to obtain a smaller prefix than $A[0..p]$.

Using the above reasoning we can iterate through array A using decreasing prefixes and check whether the value $A[p]$ has appeared before index p. Finally, we will find a prefix that cannot be decreased, and that will be the result.

1: Slow solution — $O(n^2)$.

```
def prefixSetSlow(A):
    n = len(A)
    p = n - 1
    i = p - 1
    # keep decreasing p by 1
    while (i >= 0):
        # check if A[p] appears before in A.
        while (i > 0) and (A[i] != A[p]):
            i = i - 1
        # decrease p if possible, otherwise set i to -1
        if (A[i] == A[p]):
            p = p - 1
            i = p - 1
        else :
            i = -1
    return p
```

The time complexity of the above algorithm is $O(n^2)$, because for every prefix we can iterate through $O(n)$ elements.

Fast solution $O(n \log n)$

Notice that the value p (in above solution) can be found by bisection. Using this observation the time complexity can be reduced to $O(n \log n)$.

2: Fast solution — $O(n \log n)$.

```
def prefixSetFast(A):
    n = len(A)
    l = 0
    r = n - 1
    T = n * [False]
    # invariant: l <= p <= r
    while l < r:
        s = (l + r) / 2;
        # check if p <= s
        T = n * [False]
        for i in xrange(s + 1):
            T[A[i]] = True
        i = s + 1
        while (i < n) and T[A[i]]:
            i = i + 1
        # reduce the range l..r
        if i == n:
            r = s
        else:
            l = s + 1
    return l
```

Golden solution $O(n)$

For every index p we need to know whether the value $A[p]$ has appeared before the index p. Notice that the range of all the numbers is quite small, so we can count the elements. More precisely, we can iterate through all the elements and in the cell of index $A[p]$ we mark that the value $A[p]$ has just appeared.

3: Golden solution — $O(n)$.

```
def prefixSetGolden(A):
    n = len(A)
    occur = [False] * n
    for i in xrange(n):
        if (occur[A[i]] == False):
            occur[A[i]] = True
            result = i
    return result
```

The time complexity is $O(n)$.