You are given N counters, initially set to 0, and you have two possible operations on them:

* *increase(X)* − counter X is increased by 1,
* *max counter* − all counters are set to the maximum value of any counter.

A non-empty array A of M integers is given. This array represents consecutive operations:

* if A[K] = X, such that 1 ≤ X ≤ N, then operation K is increase(X),
* if A[K] = N + 1 then operation K is max counter.

For example, given integer N = 5 and array A such that:

A[0] = 3 A[1] = 4 A[2] = 4 A[3] = 6 A[4] = 1 A[5] = 4 A[6] = 4

the values of the counters after each consecutive operation will be:

(0, 0, 1, 0, 0) (0, 0, 1, 1, 0) (0, 0, 1, 2, 0) (2, 2, 2, 2, 2) (3, 2, 2, 2, 2) (3, 2, 2, 3, 2) (3, 2, 2, 4, 2)

The goal is to calculate the value of every counter after all operations.

Write a function:

def solution(N, A)

that, given an integer N and a non-empty array A consisting of M integers, returns a sequence of integers representing the values of the counters.

Result array should be returned as an array of integers.

For example, given:

A[0] = 3 A[1] = 4 A[2] = 4 A[3] = 6 A[4] = 1 A[5] = 4 A[6] = 4

the function should return [3, 2, 2, 4, 2], as explained above.

Write an **efficient** algorithm for the following assumptions:

* N and M are integers within the range [1..100,000];
* each element of array A is an integer within the range [1..N + 1].

Benim cozumum – 1 ” for loop ile “

def solution(N, A):

arr1 = [0] \* N

for K,X in enumerate(A):

if X <= N:

if arr1[X-1] ==0:

arr1[X-1] = 1

else:

arr1[X-1] += 1

elif X == ( N + 1 ) :

for i in range(len(arr1)):

arr1[i] = max(arr1)

return arr1

or

Benim cozumum – 2 ” map lambda ile “

def solution(N, A):

arr1 = [0] \* N

for K,X in enumerate(A):

if X <= N:

arr1[X-1] += 1

elif X == ( N + 1 ) :

arr1 = list(map(lambda x: max(arr1), arr1))

return arr1

X = 5

N = [3, 4, 4, 6, 1, 4, 4]

car1 = solution(X,N)

print(car1)

or

Benim cozumum – 3 ” map lambda + last1 ile (max .method YOK) “

def solution(N, A):

arr1 = [0] \* N

last1 = 0

for K,X in enumerate(A):

if X <= N:

arr1[X-1] += 1

if last1 < arr1[X-1]:

last1 = arr1[X-1]

elif X == ( N + 1 ) :

arr1 = list(map(lambda x: last1, arr1))

return arr1

# %66

X = 5

N = [3, 4, 4, 6, 1, 4, 4]

car1 = solution(X,N)

print(car1)

or

X = 5

N = [3, 4, 4, 6, 1, 4, 4]

car1 = solution(X,N)

print(car1)

def solution(N,A):

arr1 = []

current\_max = 0

last\_increase = 0

i = 0

while i <N:

arr1.append(0)

i += 1

j = 0

while j < len(A):

if A[j] > N:

last\_increase = current\_max

else:

arr1[ A [j] -1 ] = max( arr1[ A [j] -1 ] , last\_increase)

arr1[ A [j] -1 ] += 1

current\_max = max( current\_max , arr1[ A [j] -1 ])

j += 1

k = 0

while k < N:

arr1 [k] = max( arr1 [k], last\_increase)

k += 1

return arr1

OR

PYTHON – CLASS Solution

class CounterList:

max\_value = 0

base\_value = 0

values = {}

def \_\_init\_\_(self, n):

self.length = n

def increment(self, counter):

try:

self.values[counter] += 1

except KeyError:

self.values[counter] = 1

if self.values[counter] + self.base\_value > self.max\_value:

self.max\_value = self.values[counter] + self.base\_value

def set\_max(self):

self.base\_value = self.max\_value

self.values = {}

def get\_counter(self, counter):

try:

return self.values[counter] + self.base\_value

except KeyError:

return self.base\_value

def get\_list(self):

return [self.get\_counter(i+1) for i in range(self.length)]

def solution(N, A):

counter\_list = CounterList(N)

for item in A:

if item > counter\_list.length:

counter\_list.set\_max()

else:

counter\_list.increment(item)

return counter\_list.get\_list()

or

JAVA

class Solution {

public int[] solution(int N, int[] A) {

int[] counters = new int[N];

int idx = 0, max = 0, tmp = 0;

for (int i : A) {

if (i >= 1 && i <=N) {

tmp =counters[i-1];

tmp++;

max = max > tmp ? max : tmp;

counters[i-1] = tmp;

}

else if (i == (N + 1)) {

if(N >1) {

counters[0] = max;

counters[1] = max;

for (int x = 1; x < N; x +=x) {

System.arraycopy(counters, 0, counters, x, ((N - x) < x) ? (N - x) : x);

}

}

else {

counters[0]= max;

}

}

}

return counters;

}

}

Or

%67

def solution(N,A):

arr1 = [0] \* N

max\_arr1 = 0

for index,item in enumerate(A):

if item <= N:

arr1[ item - 1 ] += 1

max\_arr1 = max(arr1)

elif item > N:

arr1 = [ max\_arr1 for i in arr1 ]

return arr1