Estructuras de Datos y Algoritmos

Grados en Ingeniería Informática

Examen Extraordinario, 25 de enero de 2017, Parte Algoritmos

Nota : VARIACIÓN EN LA REDACCIÓN SOBRE ORIGINAL

Nombre: _____ **Grupo:** _____

Laboratorio: _____ **Puesto:** _____ **Usuario de DOMjudge:** _____

1. **(4 puntos)** Sea un vector $V[0..N)$ *ordenado* de enteros con $N >= 0$. Se puede observar que *siempre* tiene forma de *escalera*, en el sentido de que sus elementos se repiten un número e_i de veces, dando lugar a *peldaños o escalones*. Definimos el *ancho* de un escalón como el número de repeticiones ($e_i > 0$) del elemento que forma ese escalón. Así, el siguiente vector tendría forma de escalera, con 6 peldaños de anchos 4, 2, 4, 1, 2 y 1:

1	1	1	1	2	2	3	3	3	3	4	7	7	8

Decimos que una escalera tiene peldaños de *ancho creciente* si el número de elementos de cada escalón es mayor o igual que los del escalón anterior. El caso anterior NO es un ejemplo de escalera de *ancho creciente*; si lo es, en cambio el siguiente, con anchos 2,2,4,6.

1	1	2	2	3	3	3	3	4	4	4	4	4	4

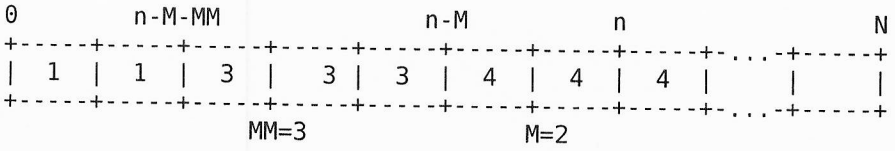
Implementa un algoritmo iterativo que, dado un vector v de enteros de longitud $0 < n \leq 1000$ que representa una escalera válida, (es decir, vector ordenado) diga si es una escalera con peldaños de ancho creciente. Además de implementar el algoritmo, deberás escribir su precondición, postcondición, invariante y función de cota de los bucles y calcular su complejidad.

Entrada	**Salida**
n v	
3 1 2 2	SI
3 1 2 3	SI
3 1 1 2	NO
3 1 1 1	SI
6 2 2 4 4 6 6	SI
6 2 3 3 4 4 5	NO
1 3	SI
2 3 3	SI
0	

```
/*
  As a decission problem, we transform it into a optimization problem

  P : N >= 0 and sorted(V)
  Q : N = min i : 0 <= i <= N and (i < N -> R(V,i)) : i
      and MM <= M

  where

  R(V,i) ::= ( i > 0 && (V[i]!=V[i-1])
                        (j = min k : 0 <= k < i and AllesEq(V,k,i) : k)
                        (q = min k : 0 <= k <= j and AllesEq(V,k,i) : k)
                        (i-j) < (j - q) )

  M = max p : 0 <= p <= N and AllEq(V,p,N) : N - p
  MM = max p : 0 <= p <= N-M and AllEq(V,p,N-M) : N-M - p
```

don't scare!

Informally, we are searching for the first pos (min i)
such that beeing different to the previous (V[i]!=V[i-1]), ends an
step whose length (i-j) is less than the previous (j-q). If such a pos
does not exists, (i=N), the actual step musts be at least as longer as
the previous (MM<=M)




```
  I  : Q[N/n] and 0 <= n <= N and

  B  : (n<N) && (MM <= M || (n==0) || V[n]==V[n-1])   **

  ** I and n==0 => MM <= M , and (n==0) can be ommited in B


  C(N,n) = N-n >= 0


  Step:
  -----
     n = n + 1 (Clearly , quota decreases)


  Init:
  -----
     n,M,MM = 0, 0, 0

  Restore:
  -------
     case n>0 && V[n]==V[n-1]
         M = M + 1
     eoc
         M, MM = 1 , M
     esac
```

So we have

Pseudocode:

```
   -----------
      n,M,MM = 0, 0, 0
      while (n<N) && (MM <= M || V[n]==V[n-1])
        case n>0 && V[n]==V[n-1]
          M = M + 1
        eoc
          M, MM = 1 , M
        esac
        n = n + 1
      done
      return n==N && MM <= M ;



*/

#include <iostream>   // cin, cout
#include <algorithm>  // swap

using namespace std;

#define MAX 1000

/* See full development and derivation below */
bool stairs (const int V[], const int N)
{
  int n,M,MM;
  for (n=M=MM=0; (n < N) && (MM<=M || (V[n]==V[n-1])) ; n++)
    if (n &&  (V[n]==V[n-1]))
      M += 1 ;
    else
      {
        MM = M;
        M = 1 ;
      }
  return ((n==N) && (MM <= M));
}

int main(int argc, char *args[])
{
  int N;
  int V[MAX];
  for (cin >> N  ; N ; cin >> N)
    {
      for(int n=0;n<N;n++)  cin >> V[n];
      cout << (stairs(V,N)?"SI":"NO") << endl;
    }

  return 0 ;
}
```