TEMA 2

Secuencias: Ventanas

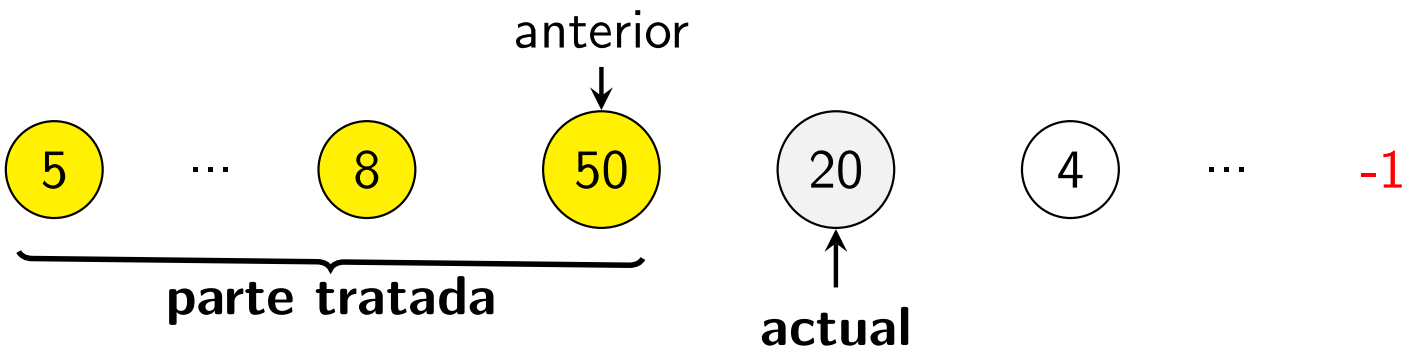
Emma Rollón
erollon@cs.upc.edu

Departamento de Ciencias de la Computación

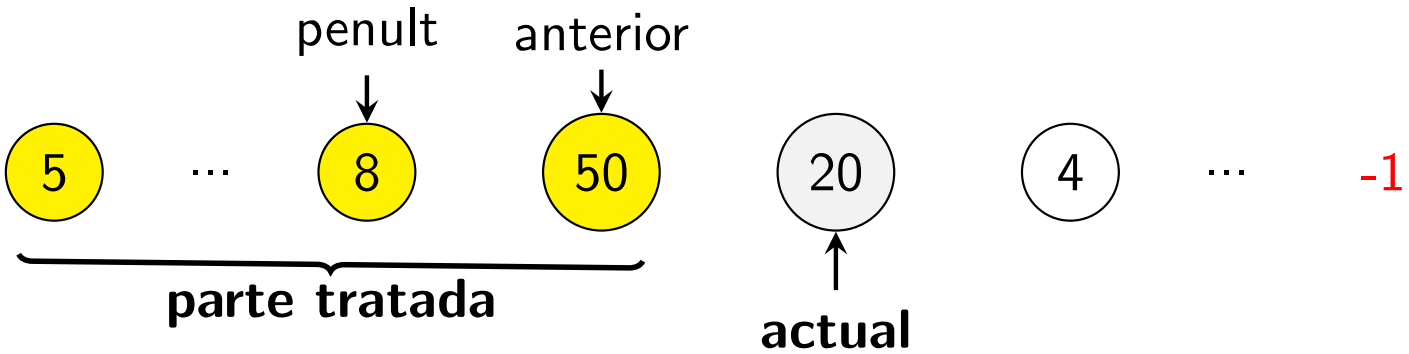
¿Qué es una ventana?

Se denomina **ventana** al conjunto de elementos justo a la izquierda del elemento actual que tenemos que mantener durante el tratamiento de la secuencia.

Ventana de 1 elemento:



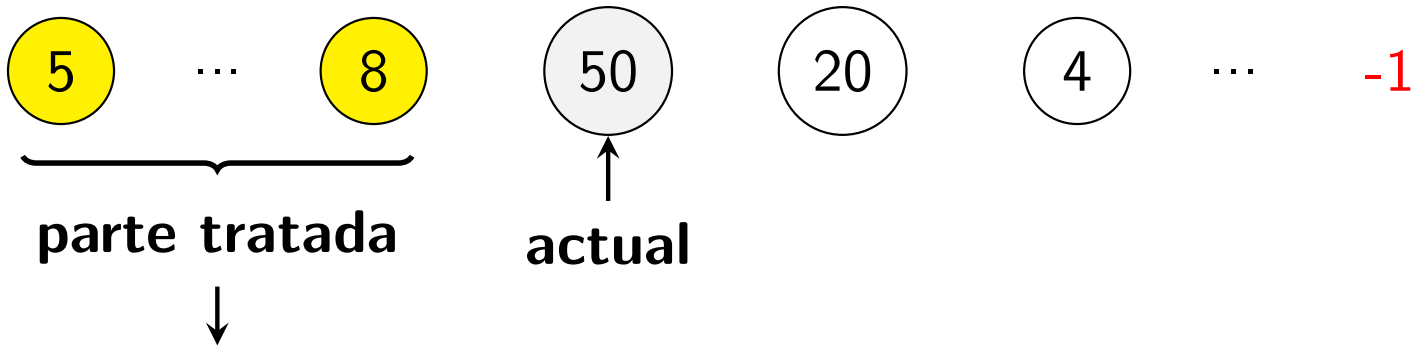
Ventana de 2 elementos:



Ejemplo 1

Dada una secuencia de naturales acabada en -1, escribir cuántas veces un número es el doble del anterior.

A. Qué información necesito mantener de la parte tratada?

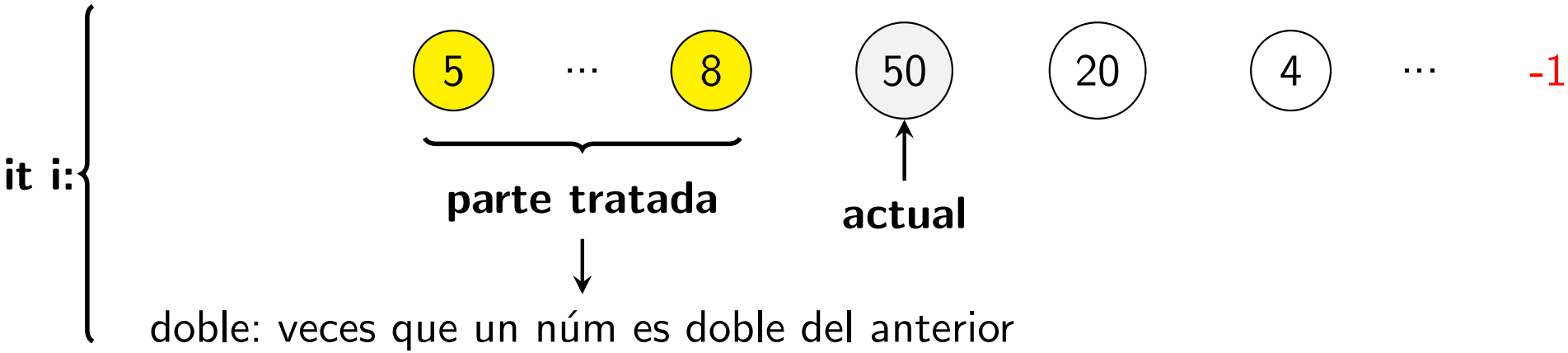


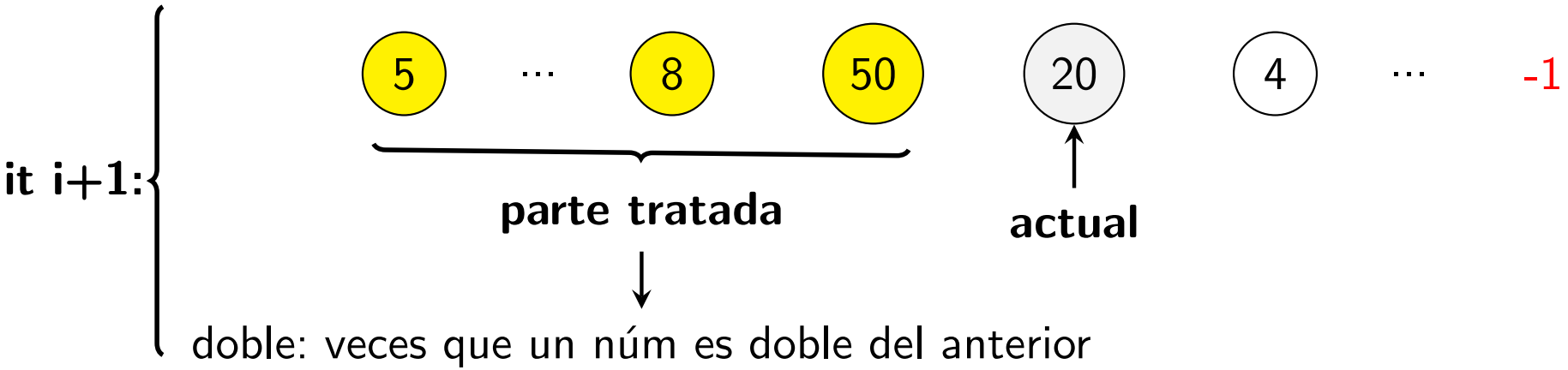
doble: veces que un núm es doble del anterior

Si soy capaz de mantener esa variable doble con el significado veces que un núm es el doble del anterior en todas y cada una de las iteraciones, cuando la parte tratada incluya toda la secuencia, ese valor será lo que tengo que escribir.

Ejemplo 1

B. Actualización?

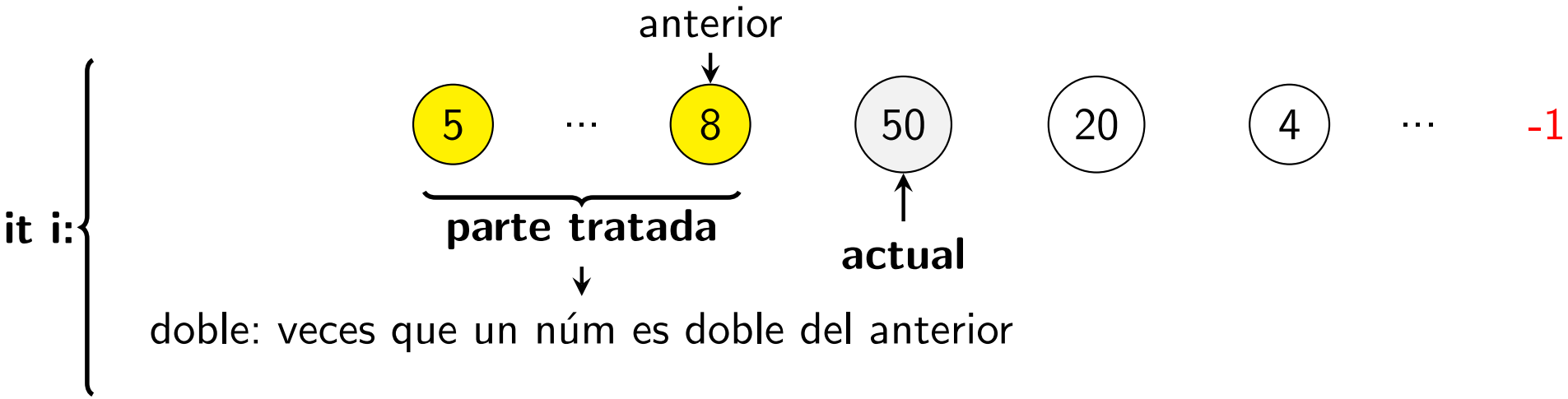


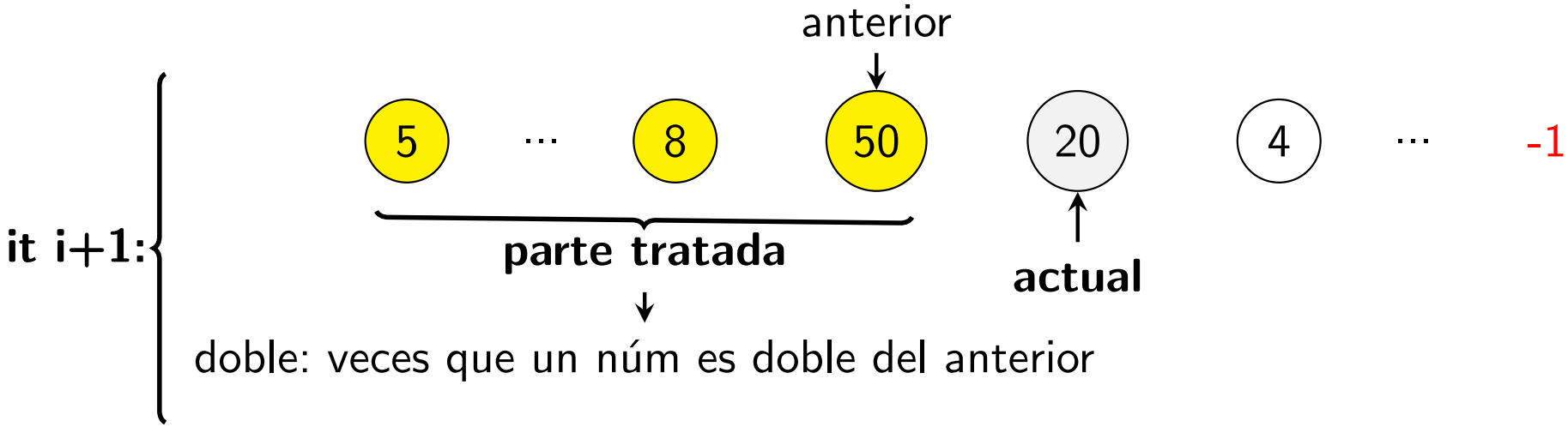


Para actualizar doble necesito saber si actual es el doble de su anterior!

Ejemplo 1

B. Actualización?





Ejemplo 1

```
int main() {
    // Inicialización
    ...;
    while (...) {
        if (actual == anterior*2) ++doble;
        anterior = actual;
        cin >> actual;
    }
    cout << doble << endl;
}
```

C. Condición del bucle?

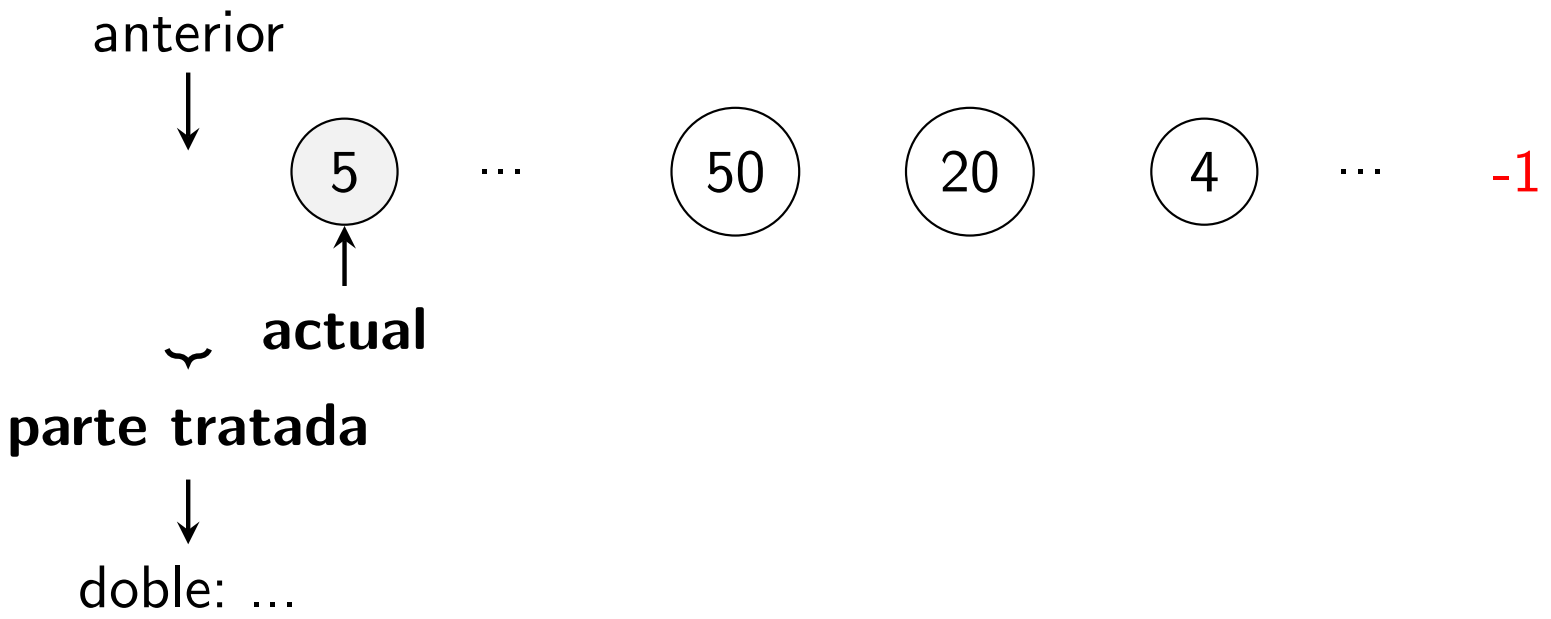
Como el final de la secuencia se indica con el centinela -1, sabré si tengo más elementos a tratar o no en función del valor de actual. Será final de secuencia cuando el actual valga -1.

```
int main() {
    // Inicialización
    ...;
    while (actual != -1) {
        if (actual == anterior*2) ++doble;
        anterior = actual;
        cin >> actual;
    }
    cout << doble << endl;
}
```

Ejemplo 1

D. Inicialización?

El 1^r elem se ha de tratar en la 1^a iteración del bucle (sec. vacía válida):



- doble: número de veces que un número es el doble de su anterior en una secuencia vacía (la parte tratada en ese momento).
- anterior: un valor tal que no actualice doble en la primera iteración.
- actual: primer valor de la entrada (elemento o centinela).

Ejemplo 1

```
int main() {
    // Inicialización
    int doble = 0;
    int anterior = -1;
    int actual;
    cin >> actual;
    while (actual != -1) {
        if (actual == anterior*2) ++doble;
        anterior = actual;
        cin >> actual;
    }
    cout << doble << endl;
}
```

Observación

Fíjate que el razonamiento concuerda con la forma que ya vimos en la que se hace la lectura cuando la secuencia acaba en centinela.

Ejemplo 2

Dada una secuencia de naturales con al menos dos elementos, decir el número de picos que hay en esa secuencia. Un pico es una terna de números tal que el central es mayor estricto que sus vecinos a izquierda y derecha.

Juegos de pruebas:

E: 2 3	S: 0
E: 2 3 1	S: 1
E: 2 3 1 5	S: 2
E: 2 3 1 1 1 2	S: 1

Variaciones:

- La secuencia no es vacía $\longrightarrow$ el primer elemento da valor a anterior y a penúltimo.
- La secuencia es vacía $\longrightarrow$ tendríamos que comprobar con un condicional si tiene primer elemento y, si es así, hacer la inicialización del punto anterior.

Ejemplo 2: código (secuencia con al menos dos elementos)

```
int n_picos = 0;
int penul, ant, act;
cin >> penul >> ant;    // existen al menos dos elementos
while (cin >> act) {
    if (penul < ant and ant < act) ++n_picos;
    penul = ant;
    ant = act;
}
cout << n_picos << endl;
```

Observaciones

Como nos aseguran que existen al menos dos elementos:

$\Longrightarrow$ los utilizamos para inicializar penul y ant

$\Longrightarrow$ el bucle trata a partir del posible tercer elemento de la secuencia (se ha de comprobar si existe o no)

Ejemplo 2: código primera variación (secuencia no vacía)

```
int n_picos = 0;
int penul, act;
cin >> penul;          // existe al menos un elemento
int ant = penul;
while (cin >> act) {
    if (penul < ant and ant < act) ++n_picos;
    penul = ant;
    ant = act;
}
cout << n_picos << endl;
```

Observaciones

Que la secuencia no sea vacía quiere decir que tiene al menos 1 elemento:

$\Longrightarrow$ utilizamos ese primer elemento para inicializar penul y ant

- $\longrightarrow$ es correcto porque en la primera iteración (si se hace) a n_picos no se le sumará 1, que es lo que queremos.

$\Longrightarrow$ el bucle trata a partir del posible segundo elemento de la secuencia (se ha de comprobar si existe o no)

Ejemplo 2: código segunda variación (secuencia vacía)

```
int n_picos = 0;
int penul;
if (cin >> penul) {    // la secuencia puede ser vacía: comprobar que exista 1er elemento
                       //  (en la entrada nos dan directamente la secuencia)
    int ant = penul;
    int act;           // <- declarar aquí, no fuera del condicional
    while (cin >> act) {
        if (penul < ant and ant < act) ++n_picos;
        penul = ant;
        ant = act;
    }
}
cout << n_picos << endl;
```

Observaciones

Como la secuencia puede ser vacía:

- $\Longrightarrow$ Primero pienso qué valores inventados puedo darles a penul y ant para que en las dos primeras iteraciones (si las hay) a n_picos no se le sume 1.
- $\Longrightarrow$ Como no encuentro valores adecuados, pienso si conociendo el primer elemento de la secuencia (si existe) sí que puedo dar a penul y ant valores adecuados (sí que puedo: es el caso de la primera variación).
- $\Longrightarrow$ Como la secuencia puede ser vacía, tengo que comprobar que ese primer elemento exista (esta comprobación depende de cómo nos den en la entrada la secuencia).

Otros ejemplos

- Dado un natural n, escribir SI si sus dígitos son crecientes de izquierda a derecha o NO en caso contrario.

  Juegos de pruebas:

E: 13456	S: SI
E: 1354	S: NO
E: 5	S: SI

- Dada una secuencia de enteros no vacía, decir cuál es la longitud máxima de sus llanuras. Una llanura es una subsecuencia del mismo elemento.

  Juegos de pruebas:

E: 4 4 2 3 4	S: 2
E: 4 2 2 3	S: 2
E: 4 3 2 2 2	S: 3