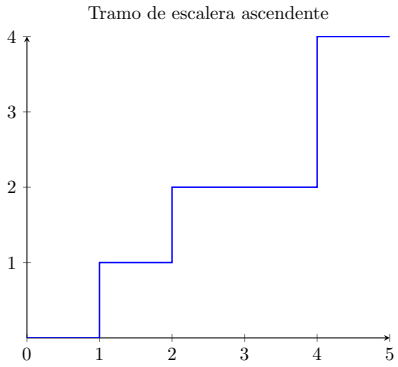
Fundamentos de Algoritmia

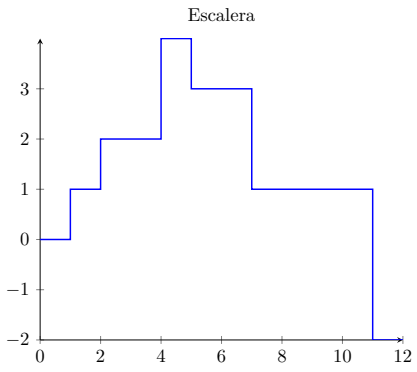
Grados en Ingeniería Informática e Ingeniería del Software. Grupos E,F,I

Examen final, 14 de Enero de 2020.

Una secuencia no vacía de números enteros, ordenados de menor a mayor forman un *tramo de escalera ascendente*. El *ancho de un escalón* viene determinado por el número de valores iguales, y la *altura del escalón* por el valor. De forma análoga, una secuencia no vacía de números enteros ordenados de mayor a menor forman un *tramo de escalera descendente*. Una secuencia de números tiene forma de *escalera* si está formada por un tramo de escalera ascendente seguido de un tramo de escalera descendente. Observad que el escalón superior es común a ambos tramos.

Por ejemplo la secuencia: $v = \{0, 1, 2, 2, 4, 3, 3, 1, 1, 1, 1, -2\}$ tiene un tramo inicial ascendente de longitud 5 que comprende los valores del 0 hasta el 4 incluido. El ancho del primer escalón es 1, y el ancho del tercer escalón es 2. La secuencia tiene un tramo final descendente de longitud 8 que comprende los valores desde el 4 incluido hasta el -2. El tercer escalón tiene ancho 4 y altura 1. La secuencia completa es una escalera.





Ejercicio 1 (3.5 puntos)

- Especifica, diseña e implementa una función que dado un vector de números enteros, calcule el número de elementos de su parte izquierda que forman un *tramo de escalera ascendente* y la longitud del escalón más ancho del tramo.
- Especifica, diseña e implementa una función que dado un vector de números enteros, calcule el número de elementos de su parte derecha que forman un *tramo de escalera descendente* y la longitud del escalón más ancho del tramo.
- Utilizando las dos funciones anteriores implementa una función que dado un vector de números enteros decida si tiene forma de escalera y calcule la longitud del escalón más ancho de toda la escalera.
- Para cada función debe indicarse el invariante de los bucles que se utilicen y el coste justificado de la implementación realizada.

Entrada. La entrada consta de varios casos de prueba. Cada caso de prueba consta de dos líneas. La primera línea indica el número de valores de la secuencia ($0 < n \leq 1000$), la segunda línea contiene los valores de la secuencia. Estos son número enteros.

El último caso tiene una única línea con el valor -1 y no debe procesarse.

Salida. Para cada caso de prueba se escribe en una línea `SI` seguido de la longitud del escalón más ancho si la secuencia es una escalera, o `NO` seguido del número de elementos de la parte izquierda que forman un tramo ascendente y del número de elementos de la parte derecha que forman un tramo descendente si la secuencia no es una escalera.

Entrada de ejemplo	**Salida de ejemplo**
5 1 2 3 2 1 8 1 2 2 2 1 -2 -2 -3 4 1 2 1 2 7 -3 -3 0 -2 1 -4 -4 6 2 2 2 2 2 2 1 4 8 1 1 2 2 3 3 4 4 8 4 4 3 3 2 2 1 1 5 1 2 3 2 2 -1	SI 1 SI 3 NO 2 1 NO 3 3 SI 6 SI 1 SI 2 SI 2 SI 2

Ejercicio 2 (3 puntos) Dada una secuencia de números enteros con forma de *escalera* y el tamaño del escalón más ancho, queremos calcular la altura de la escalera. Se entiende por la altura de la escalera, la altura del escalón más alto.

Se pide:

- Implementar una función recursiva con la técnica de divide y vencerás que resuelva el problema de forma eficiente.
- Calcular el coste de la función. Justificar el coste dando la recurrencia que lo define y haciendo el desplegado.

Entrada.

La entrada consta de varios casos de prueba. Cada caso de prueba consta de dos líneas. La primera línea indica el número de valores del vector ($0 < n \leq 1000$) y la longitud del escalón más ancho. La segunda línea contiene los valores del vector (números enteros). El último caso es el valor -1 y no debe procesarse.

Salida. Para cada caso de prueba se escribe en una línea la altura de la escalera.

Entrada de ejemplo	**Salida de ejemplo**
5 1 1 2 3 2 1 8 3 1 2 2 2 1 -2 -2 -3 6 6 2 2 2 2 2 2 1 1 4 8 2 1 1 2 2 3 3 4 4 8 2 4 4 3 3 2 2 1 1 15 6 1 1 1 1 1 1 5 4 3 2 2 2 2 2 2 15 6 2 2 2 2 2 2 3 4 5 1 1 1 1 1 1 -1	3 2 2 4 4 4 5 5

Ejercicio 3 (3.5 puntos) Queremos pintar una escalera y para ello contamos con una serie de botes de pintura de diferentes colores. Para que quede bonita, hemos pensado que cada escalón debe estar pintado de un único color, pero dos escalones consecutivos deben tener colores diferentes. Conocemos lo que nos cuesta la pintura de cada color y la longitud en centímetros que podemos pintar con lo que tenemos en cada bote, ahora queremos saber de que color nos conviene pintar cada escalón para que nos cueste lo mínimo posible. Los saltos entre los escalones no los vamos a pintar por ahora.

Se pide implementar una función con la técnica de vuelta atrás que resuelva el problema de forma eficiente.

Entrada. La entrada consta de varios casos de prueba. Cada caso de prueba consta de $c + 2$ líneas. La primera línea indica el número de escalones que tiene la escalera ($0 < n < 10$) y el número de colores diferentes que tenemos para pintar ($0 < c < 5$) . La segunda línea contiene el ancho de cada escalón medido en centímetros. Las c líneas siguientes tienen la cantidad de pintura que tenemos de cada color, medida en centímetros que podemos pintar seguida del precio por centímetro de esa pintura.

El final de casos se indica con el valor -1.

Salida. Para cada caso de prueba se escriben dos líneas. En la primera se indica el coste de pintar la escalera completa. En la segunda se indica el color que debemos utilizar para cada escalón, empezando por el de más a la izquierda. Si existen varias soluciones con el mismo coste, escribiremos aquella en que se utilicen antes los primeros colores que nos dieron en la entrada, es decir, elegiremos la solución que tenga el primer peldaño con un color de número más bajo. Si existen varias soluciones de coste mínimo con el color de número más bajo en el primer peldaño, seleccionaremos la que tenga el color de número más bajo en el segundo peldaño etc. Si no se puede pintar la escalera se escribirá `NO`.

Entrada de ejemplo	**Salida de ejemplo**
3 2 2 2 2 3 1 4 2 5 2 1 1 1 1 1 3 3 3 1 5 3 1 3 1 2 1 6 10 2 3 6 6 2 1 1 1 3 2 -1	10 1 0 1 9 1 0 1 0 1 46 0 2 1 2 1 NO