Control 9

Tramos constantes longitud máxima (¡otra vez!)

El objetivo de este control es profundizar en el diseño y en el análisis de la eficiencia de algoritmos recursivos basados en el esquema "divide y vencerás".

El problema

Dado un vector `int a[n]` no vacío (es decir, n > 0) y **ordenado**, diseñar un algoritmo "divide y vencerás" que determine la longitud del tramo constante de longitud máxima.

Sugerencia: para resolver este problema te será útil plantear una generalización que, aparte de devolver la longitud del tramo constante de longitud máxima de un segmento del vector, devuelva también las longitudes del primer y del último tramo constante de dicho segmento.

Trabajo a realizar

Debe diseñarse el algoritmo DV pedido, completando los apartados indicados entre comentarios en el archivo `control9.cpp` que se proporciona como apoyo. Debe implementarse, además, el algoritmo. El punto de entrada al mismo será la función `lon_tramo_cte_max`. Si se considera necesario, deberá definirse e implementarse una generalización adecuada, y definir el algoritmo pedido como una inmersión de dicha generalización. Aparte de llevar a cabo el diseño del algoritmo, e implementar el mismo, deberás determinar justificadamente su complejidad, definiendo y resolviendo las recurrencias necesarias (la resolución se llevará a cabo utilizando los patrones de recurrencias genéricas discutidos en clase). Deberás también justificar qué ventajas y desventajas tiene tu algoritmo frente a uno iterativo con coste lineal.

Programa de prueba

Se proporcionan dos versiones alternativas (puede elegirse una u otra comentando o descomentando la definición `#define DOM_JUDGE` que se encuentra al comienzo del archivo):

- Si DOM_JUDGE está definido, el programa lee por la entrada estándar casos de prueba, los resuelve invocando a `lon_tramo_cte_max`, e imprime los resultados. Cada caso de prueba consta de 2 líneas:
  - La primera contiene el tamaño del vector.
  - La segunda contiene, en orden, los valores del vector.

  La lista de casos de prueba termina con un 0. A continuación se muestra un ejemplo de E/S:

Entrada	Salida
10 1 1 2 2 2 2 3 3 3 4 6 5 5 6 6 6 7 0	4 3

- Si DOM_JUDGE no está definido, el programa genera aleatoriamente 1000 casos de prueba y compara el resultado del algoritmo con una implementación iterativa del mismo. Este modo de ejecución es útil para someter al algoritmo a una prueba de estrés. Si se genera un caso para el cuál la implementación DV y la implementación iterativa discrepan, el programa termina y muestra el vector para el que el algoritmo falla. Para activar este modo basta comentar la línea `#define DOM_JUDGE`

El archivo completo debe entregarse a través del juez en línea de la asignatura.

Importante:

- Únicamente se evaluarán aquellas entregas que superen satisfactoriamente los casos de prueba del juez.
- No modificar el código proporcionado.
- **No se corregirá ninguna entrega en la que no se hayan incluido los nombres de los miembros del grupo que han realizado el trabajo en el comentario que se incluye al comienzo del archivo.**