Problema A

Valerie Parra Cortés 201619703
Cristina Isabel Gonzalez Osorio 201520019

December 11, 2018

1 Algoritmo de solución

El algoritmo utilizado se ejecuta en un solo método que es el main de la clase. Se tiene dos variables, en una se guarda el tamaño del arreglo casi ascendente más grande hasta i y en la otra el tamaño del tamaño del arreglo casi ascendente que termina en i. Si en algún momento el la longitud del arreglo casi ascendente que termina en i es mayor a la del mayor, se actualiza su valor. Adicionalmente se guarda el tamaño del arreglo que inicia en la posición del primer par de números que no son ascendentes, es decir si $\exists_k | 0 \leq k < n \wedge a[k] < a[k+1]$ y este k+1 representa la posición después la primera descendencia del arreglo, se llevará en una variable el valor entre la resta i-k, y cuando se encuentre la segunda descendencia del arreglo, el nuevo arreglo más largo será de la última descendencia, es decir de la posición i-k a la inésima.
Por ejemplo, en la figura 1 los números en azul representan las posiciones del arreglo, los rosados los valores y las franjas morada y naranja los posibles subarreglos casi ascendentes. Una vez i llega a 7, el tamaño del arreglo más grande hasta i se actualiza a 4, que es lo longitud del subarreglo naranja hasta 7.

Este algoritmo también se hubiera podido implementar recorriendo el arreglo cada vez buscando el menor o de devolviéndose cada vez que se encontraba un dos pares de números que no fueran ascendentes, sin embargo estas dos últimas formas son $\theta(n^2)$ en tiempo, por lo que se prefirió la que se encuentra implementada.

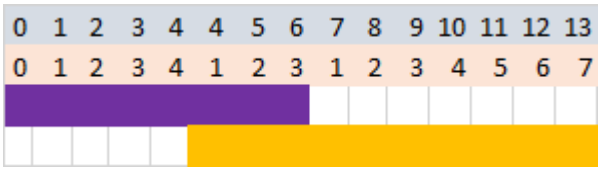

Figure 1: Ejemplo

Entrada	Descripción
n: nat	Tamaño del arreglo
a: int[0,n)	Arreglo con los números

Salida	Descripción
l	Tamaño del arreglo casi creciente más grande

$Pre = (true)$
$Post = (l = (\exists_{b:nat[0,m)}|(b \subseteq a) \wedge (\sim \exists_{l,m}|(0 \leq l < m < n) \wedge a[l] < a[l+1] \wedge a[m] < a[m+1]) : |b|)$

1.1 Complejidad espacial

El problema tendría complejidad espacial $\theta(1)$, ya que siempre se requerirían las mismas variables independiente del tamaño de la entrada

1.2 Complejidad temporal

La complejidad temporal del problema es $\theta(n)$, ya que el algoritmo recorre el arreglo 1 vez.

1.3 Comentarios finales

El algoritmo tienen su principio en las técnicas de desarrollo de algoritmos vistas en el curso.