Dibujo en blanco y negro

Descripción generada automáticamente con confianza baja**INSTITUTO POLITÉCNICO NACIONAL**

**ESCUELA SUPERIOR DE CÓMPUTO**

**--------------------ANÁLISIS DE ALGORITMOS------------------**

**ACTIVIDAD**

Diseño de soluciones DyV

**PROFESOR:**

Franco Martínez Edgardo Adrián

**ALUMNO:**

Meza Vargas Brandon David – 2020630288

**GRUPO:**

3CM13

Joven con camiseta negra

Descripción generada automáticamente

**índice**

[**Problema: Divide and Conquer 1** 3](#_Toc86771031)

[**Redacción** 3](#_Toc86771032)

[**Captura de aceptación por juez** 3](#_Toc86771033)

[**Explicación Algoritmo** 3](#_Toc86771034)

[**Análisis de complejidad en cota O()** 5](#_Toc86771035)

[**Código de solución completo** 5](#_Toc86771036)

[**Problema: INVCNT – Inversión Count** 7](#_Toc86771037)

[**Redacción** 7](#_Toc86771038)

[**Input** 7](#_Toc86771039)

[**Output** 7](#_Toc86771040)

[**Captura de aceptación por juez** 7](#_Toc86771041)

[**Explicación Algoritmo** 7](#_Toc86771042)

[**Análisis de complejidad en cota O()** 9](#_Toc86771043)

[**Código de solución completo** 10](#_Toc86771044)

[**Problema: Amigos y Regalos** 12](#_Toc86771045)

[**Redacción** 12](#_Toc86771046)

[**Captura de aceptación por juez** 12](#_Toc86771047)

[**Explicación Algoritmo** 12](#_Toc86771048)

[**Análisis de complejidad en cota O()** 12](#_Toc86771049)

[**Código de solución completo** 12](#_Toc86771050)

Antes de todo, cable hay que aclarar que en algunos problemas se usaron tipos de datos long, long long int, ya que se requieren para valores grandes y el juez lo acepte al 100%.

# **Problema: Divide and Conquer 1**

## **Redacción**

Edgardo se puso un poco intenso este semestre y puso a trabajar a sus alumnos con problemas de mayor dificultad.

La tarea es simple, dado un arreglo A de números enteros debes imprimir cual es la suma máxima en cualquier subarreglo contiguo.

Por ejemplo si el arreglo dado es {-2, -5, **6, -2, -3, 1, 5**, -6}, entonces la suma máxima en un subarreglo contiguo es **7**.

## **Captura de aceptación por juez**



Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

## **Explicación Algoritmo**

Texto

Descripción generada automáticamenteA continuación, se muestra el algoritmo empleado para resolver este problema.

Primeramente, establecemos la mitad del arreglo, para de eta forma partir el arreglo en dos y empezar a buscar las sumas de sub-arreglos por las mitades, las mismas que serán partidas hasta que no se pueda más.

Posteriormente, se hace la llamada recursiva para encontrar la suma por la parte de la derecha, que va de la mitad del arreglo hasta el primer elemento, y la suma de la parte izquierda va de un elemento después de la mitad hasta el último elemento del arreglo.

Ahora bien, expliquemos lo que sucede en la siguiente parte, solo se explica una vez ya que es lo mismo para derecha e izquierda.

Texto

Descripción generada automáticamente

Primeramente se iguala al lado izquierda el elemento que se encuentra a la mitad del arreglo, en el caso de la parte derecha se asigna el valor mitad+1. Después se hace el ciclo desde la mitad al inicio del arreglo, recorriendo así la parte izquierda. Posteriormente, se van acumulando las sumas de los elementos, si esta suma es mayor al elemento posicionado en la mitad del sub arreglo se establece como la suma mayor de este lado.

Lo mismo sucede con el lado derecho cambiando los límites.

Finalmente se suman las sumas de ambas mitades, siendo así la suma máxima el mayor elemento de la suma central, la suma de la derecha o la de la izquierda.

## **Análisis de complejidad en cota O()**

Para este problema tenemos que se parte en dos el problema cada vez que se manda a llamar a la función, esto lo hace dos veces, por la parte derecha y la izquierda, de esta manera tenemos.

T(n) = 2T(n/2)+n -🡪 n es la complejidad de la mezcla que se hace abajo.

Haciendo uso del teorema maestro tenemos:

f(n) = O(nlog22) 🡺 n = n

Por lo tanto la complejidad en cota O es:

**O(log(n)n)**

## Texto Descripción generada automáticamente**Código de solución completo**

Texto

Descripción generada automáticamente

# **Problema: INVCNT – Inversión Count**

## **Redacción**

Let A[0...n - 1] be an array of n distinct positive integers. If i < j and A[i] > A[j] then the pair (i, j) is called an inversion of A. Given n and an array A your task is to find the number of inversions of A.

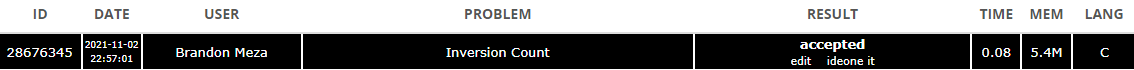
### **Input**

The first line contains t, the number of testcases followed by a blank space. Each of the t tests start with a number n (n <= 200000). Then n + 1 lines follow. In the ith line a number A[i - 1] is given (A[i - 1] <= 10^7). The (n + 1)th line is a blank space.

### **Output**

For every test output one line giving the number of inversions of A.

## **Captura de aceptación por juez**



## **Explicación Algoritmo**

A continuación se muestra el algoritmo empleado para este problema.

Texto

Descripción generada automáticamente

La función anterior, se realizará el conteo de inversiones siempre y cuando el elemento inicial sea menor al final, en otro caso el arreglo es de un elemento y no hay inversiones.

Se hacen llamadas recursivas para encontrar las inversiones por el lado de la derecha y por el lado izquierdo, posteriormente se hace la mezcla de estas.

Podemos observar que es un ordenamiento por mezcla, solo que en la función merge, que es nuestra conteoDivididas es donde contabilizaremos las inversiones.

Texto

Descripción generada automáticamente

Básicamente, en esta función se cuentan las inversiones al momento de que dos mitades del arreglo son mezcladas, se hace creando los índices del inicio que marcan los límites que tendrán al recorrerse. El índice i empieza desde la derecha y el j desde la mitad representando la izquierda.

Al momento en que el elemento i del arreglo es mayor al elemento j significa que las partes izquierdas y derechas del sub arreglo ya están ordenadas, estas serán q-i en total.

Finalmente se hace la mezcla de los elementos en el arreglo original.

## **Análisis de complejidad en cota O()**

Para este algoritmo se parte en dos el arreglo y así hasta el caso base, se hace dos veces, una para el lado izquierdo y otra para el derecho, de esta forma tenemos:

T(n) = 2T(n/2) + n -🡪 donde n es el costo de la función que hace la mezcla de las inversiones

Haciendo uso del teorema maestro tenemos:

f(n) = O(nlog22) 🡺 n = n

Por lo tanto la complejidad en cota O es:

**O(log(n)n)**

## Texto Descripción generada automáticamente**Código de solución completo**

Texto

Descripción generada automáticamente

# **Problema: Amigos y Regalos**

## **Redacción**

## **Captura de aceptación por juez**

## **Explicación Algoritmo**

## **Análisis de complejidad en cota O()**

## **Código de solución completo**