26-2-2024

ALBERTO RAMOS LOPEZ

Ejercicios Unidad 1 TPA

# Ejercicio 1

Tabla

Descripción generada automáticamente

## A:

sum = 0;

for (int i = 0; i < n; i++)

for (int j = 0; j < n \* n; j++)

sum++;

* Este fragmento contiene dos bucles anidados. El bucle exterior se ejecuta n veces, y el bucle interior se ejecuta n^2 veces.
* El tiempo de ejecución es **O(n^3)**.

## B:

sum = 0;

for (int i = 0; i < n; i++)

for (int j = 0; j < i; j++)

sum++;

* Aquí, el bucle interior se ejecuta solo hasta i, lo que reduce la cantidad de iteraciones.
* El tiempo de ejecución es **O(n^2)**.

## C:

sum = 0;

for (int i = 0; i < n; i++)

for (int j = 0; j < i \* i; j++)

for (int k = 0; k < j; k++)

sum++;

* El bucle externo se ejecuta *n* veces, el segundo bucle se ejecuta *i^*2 veces para cada valor de *i*, y el tercer bucle se ejecuta *j* veces para cada valor de *j.*
* Tiene sentido que al introducir un valor de 2 de 0 ya que este se va reduciendo n-1 veces hasta quedar en 0 y al introducir 3 este se queda en 1.
* esta sería la ecuación que la corresponde.
* El tiempo de ejecución es **O(n^3)**.

## E:

int i = 1;

int x = 0;

while (i <= n) {

x++;

i += 2;

}

* Este fragmento utiliza un bucle while que incrementa i en 2 en cada iteración.
* El tiempo de ejecución es **O(n/2)**, que se simplifica a **O(n)**.

# Ejercicio 2

Captura de pantalla de computadora

Descripción generada automáticamente

En la línea 7 es posible cambiar el ***Math.sqrt*** por ***i \* i*** ya que también valida la propiedad que un numero multiplicado por sí mismo sea indivisible por otro que no sea si mismo.

# Ejercicio 3:

Texto

Descripción generada automáticamente con confianza media

* Si asumimos que esPrimo(n) tiene una complejidad de **O()**, entonces la complejidad total de funcion\_A sería **O()**.

(Creo que debería de ser así pero no me queda muy claro)

# Ejercicio 4:

Texto

Descripción generada automáticamente

* El bucle **for** se ejecuta hasta la raíz cuadrada de **numero** (**i <= Math.sqrt(numero)**), lo que reduce el número de iteraciones y mejora la eficiencia.
* En el bucle, verificamos si **i** es un divisor de **número**. Si es así, actualizamos la suma de divisores y también consideramos el divisor complementario (**numero / i**).
* La complejidad de esta función es O(√n), donde n es el número dado. Esto se debe a que la verificación de los divisores se realiza hasta la raíz cuadrada de **número**.

# Ejercicio 5:

Texto

Descripción generada automáticamente

* La función utiliza dos bucles anidados para recorrer todos los elementos de las matrices. Ambos bucles tienen una complejidad de O(N^2), donde N es el orden de las matrices.
* La función realiza un número constante de operaciones dentro de los bucles, como comparaciones de elementos.
* Por lo tanto, la complejidad total de la función es O(N^2), donde N es el orden de las matrices.