

Instituto Politécnico Nacional

Escuela Superior de Cómputo

* **UNIDAD DE APRENDIZAJE:** ANÁLISIS DE ALGORITMOS
* **PROFESOR:** EDGARDO ADRIÁN FRANCO MARTÍNEZ
* **ALUMNO:**

BARRERA PÉREZ CARLOS TONATIHU

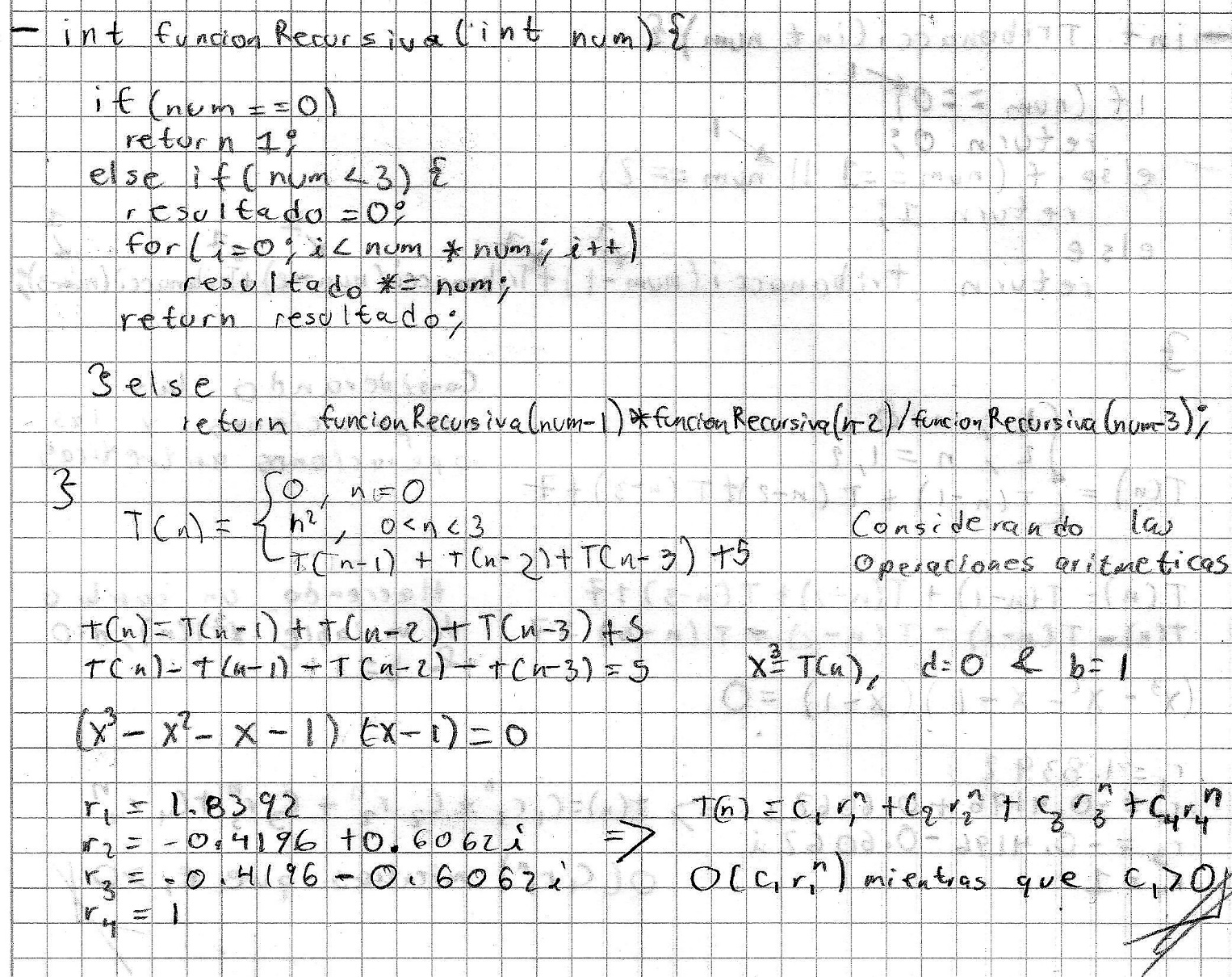


## **Ejercicio 5: Análisis de algoritmos recursivos**

## 

* **GRUPO:** 3CM3

# Calcular la complejidad para el algoritmo para el cálculo del factorial recursivo.

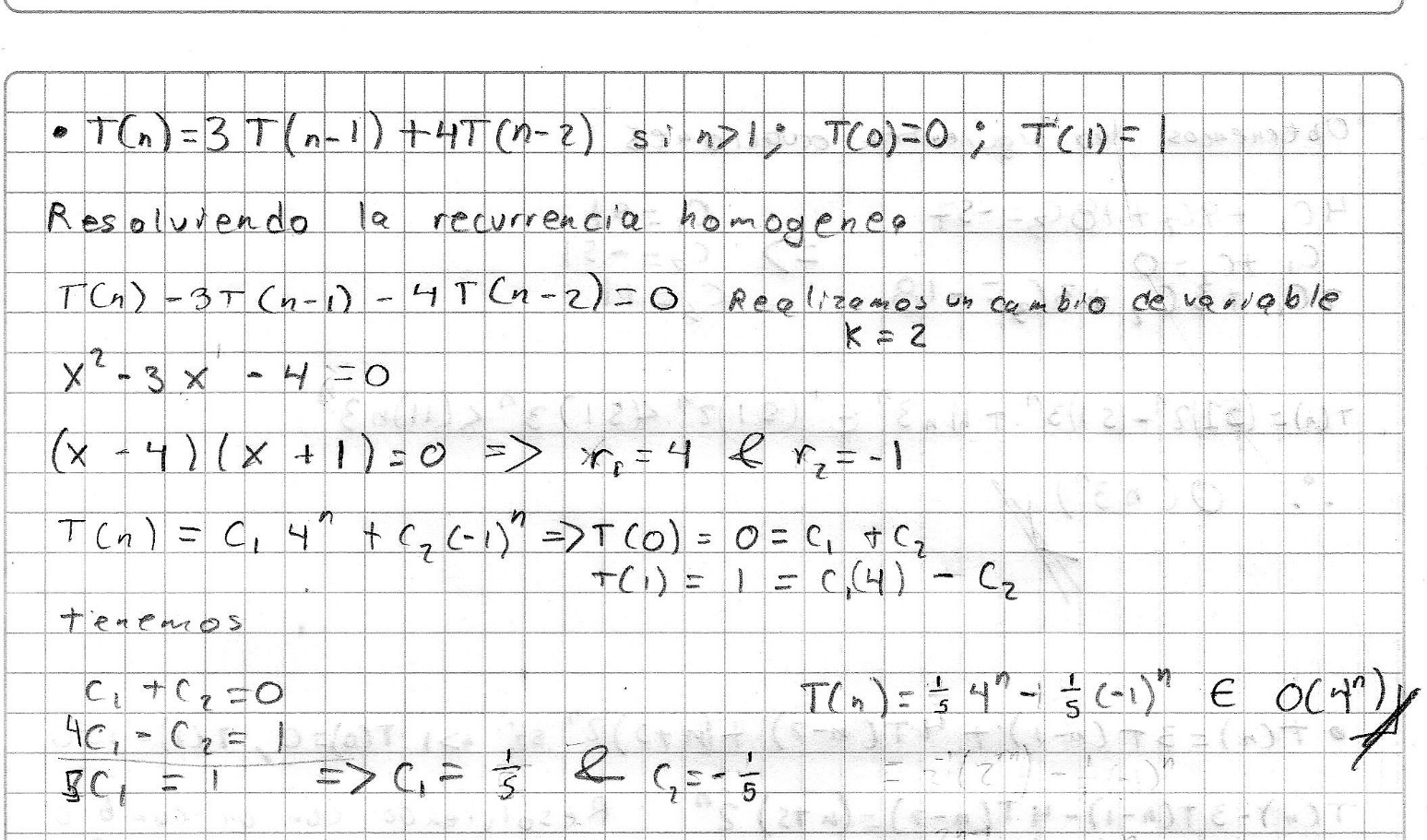


# Calcular la complejidad de la implementación recursiva del termino n de la serie de Tribonacci (0, 1, 1, 2, 4, 7, 13, 24, 44, 81, 149, 274, 504, 927, 1705, ...).

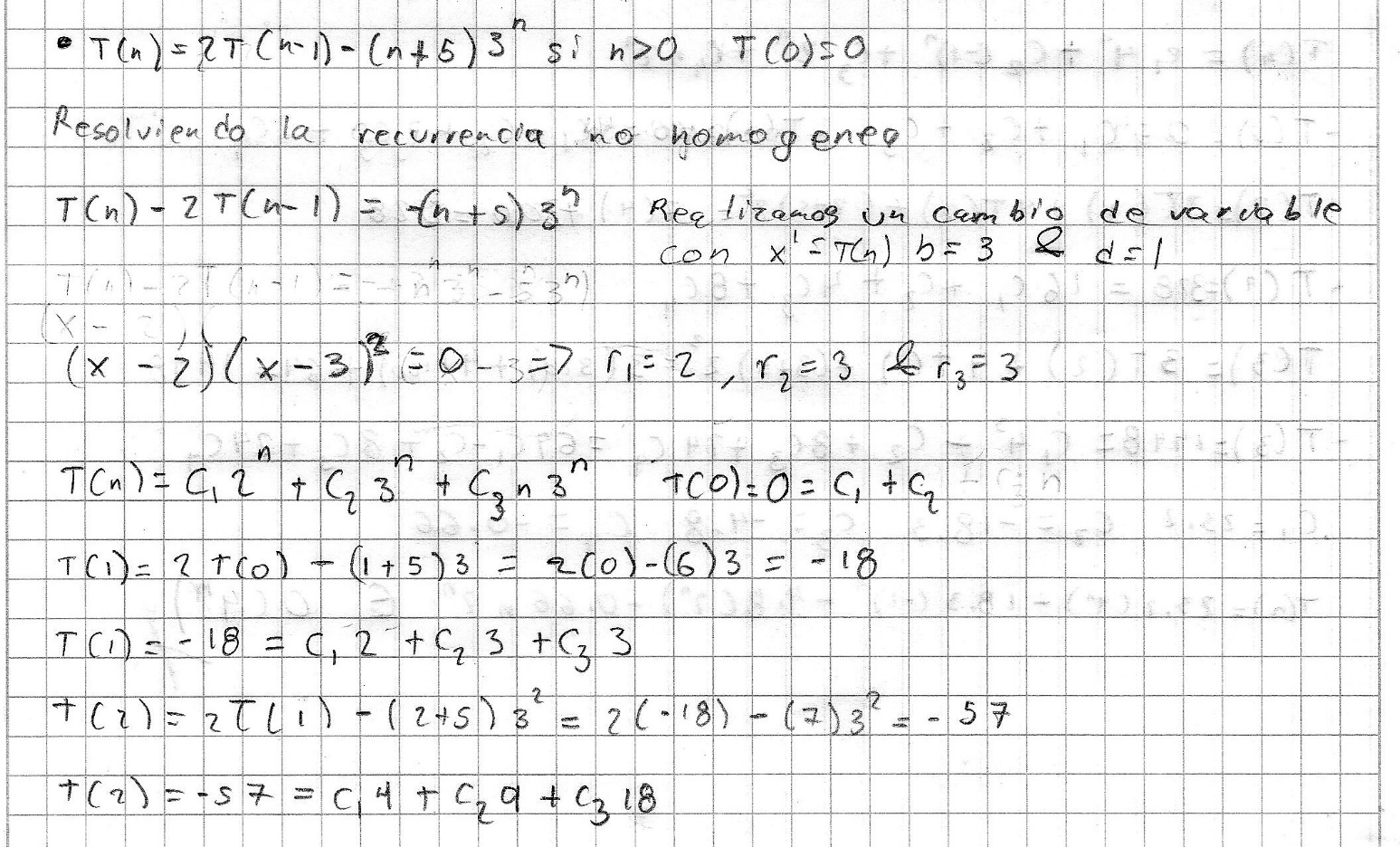


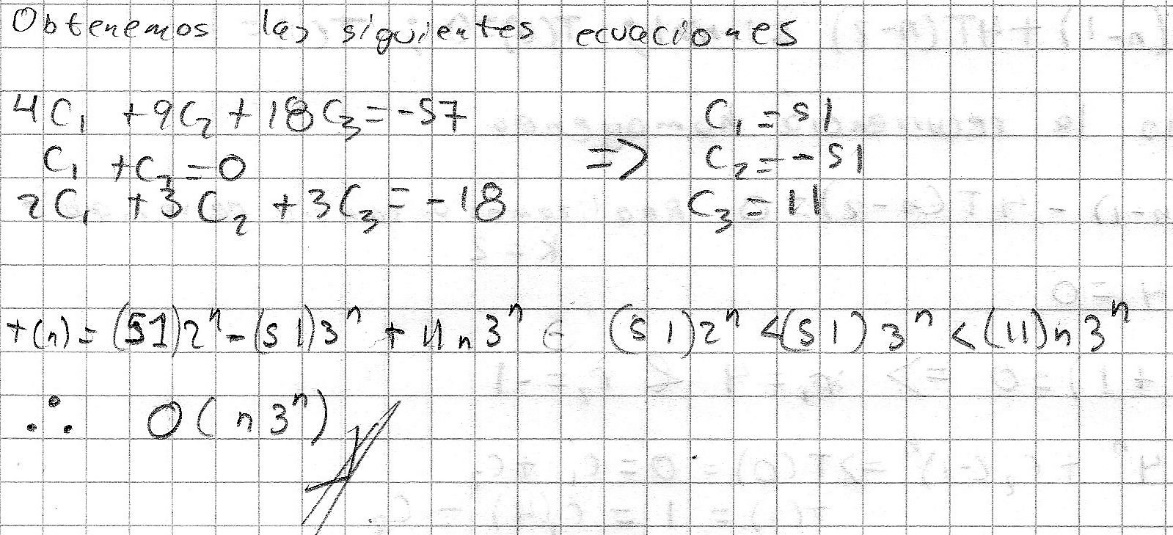
# Resolver las siguientes ecuaciones y dar su orden de complejidad.

Problema 1

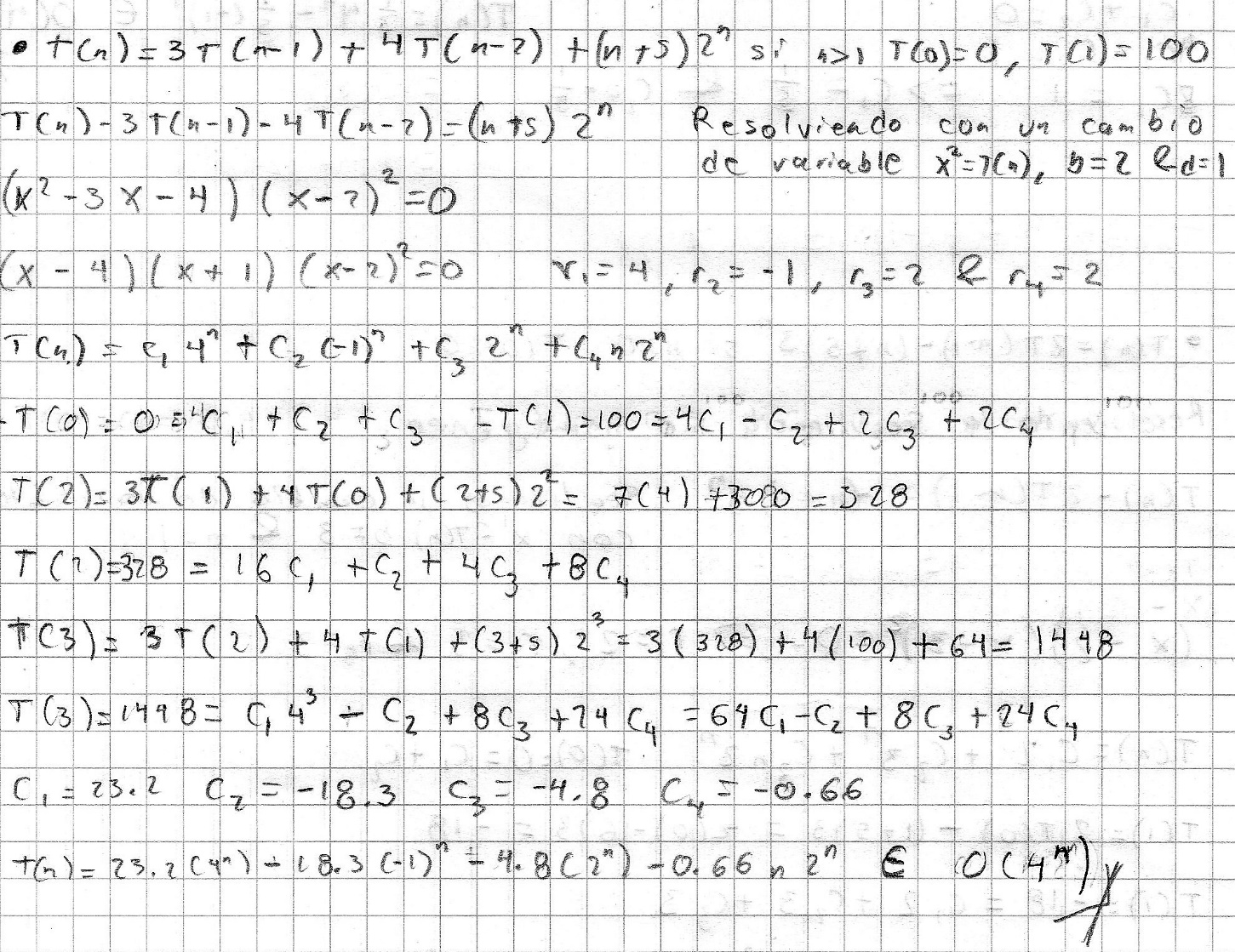


Problema 2

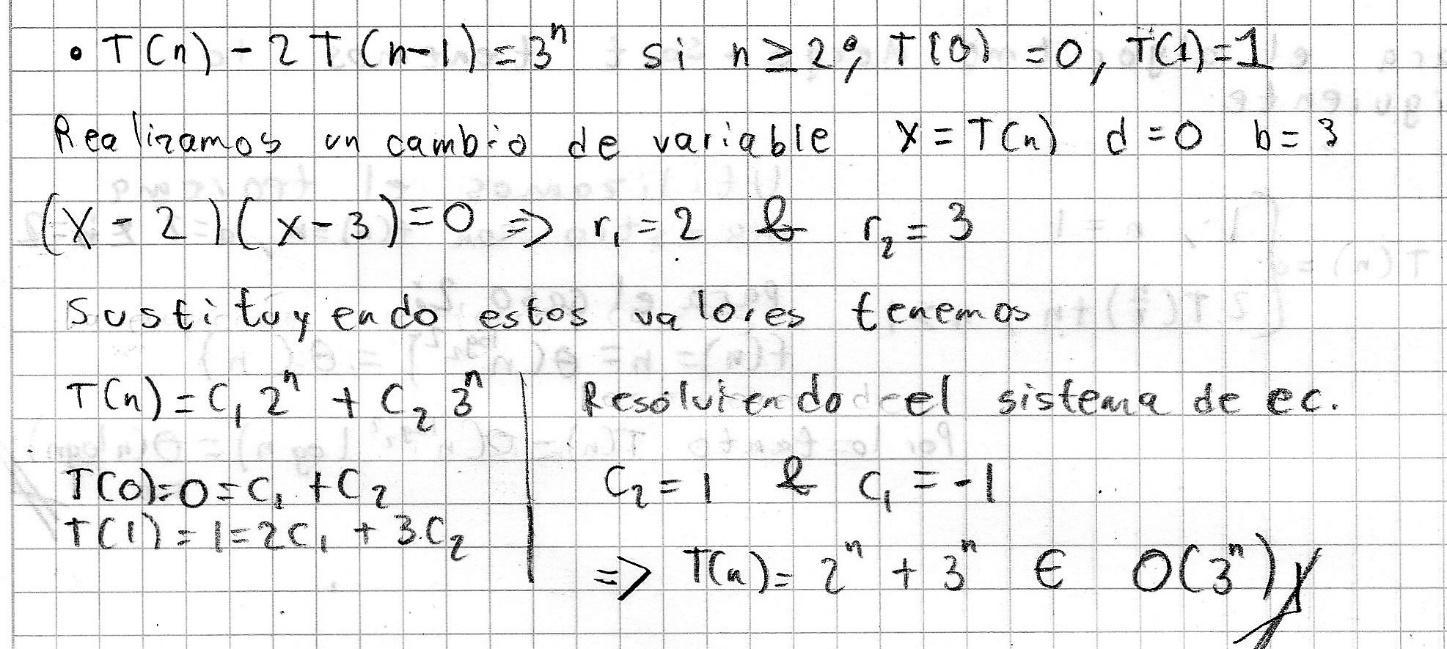




Problema 3



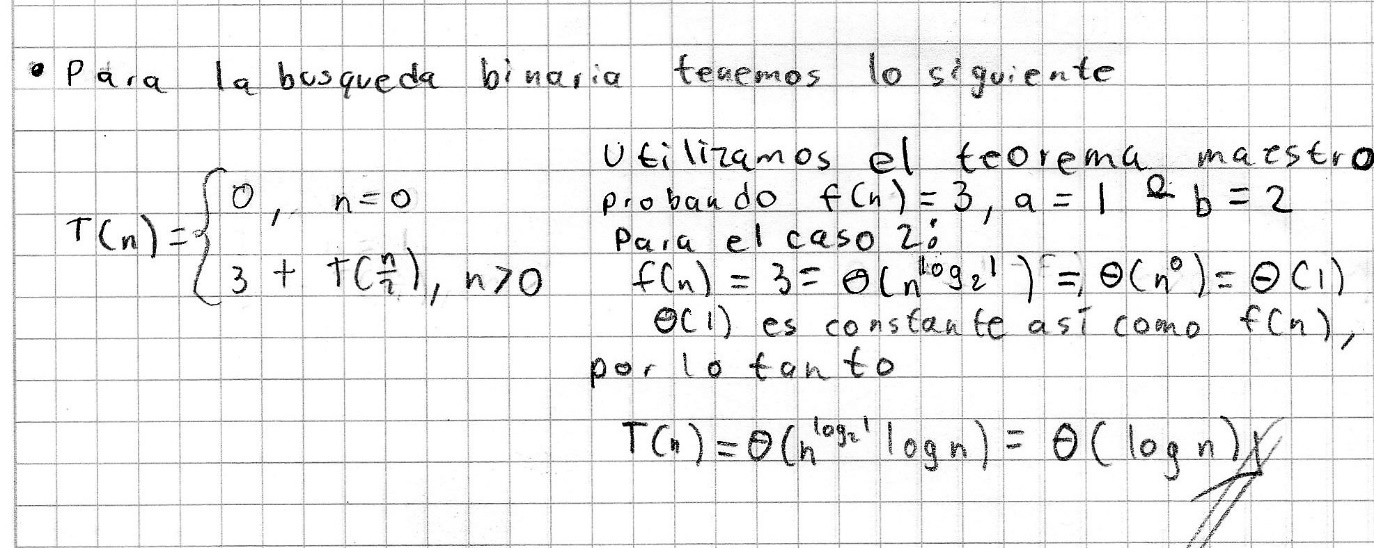
Problema 4



# Calcular la cota de complejidad del algoritmo de búsqueda binaria recursiva (Ejemplo 04).

|  |
| --- |
| int BusquedaBinaria(int num\_buscado, int numeros[], int inicio, int centro, int final) {  if (inicio>final)  return -1;  else if (num\_buscado == numeros[centro])  return centro;  else if (num\_buscado < numeros[centro])  return BusquedaBinaria(num\_buscado, numeros, inicio, (int)((inicio+centro-1)/2), centro-1);  else  return BusquedaBinaria(num\_buscado, numeros, centro+1, (int)((final+centro+1)/2), final);  } |

El análisis de este algoritmo es el siguiente.



Calcular la cota de complejidad del algoritmo Merge–Sort recursivo (Ejemplo 05).

En este caso a es un arreglo de n elementos y p & r son los índices del rango a ordenar.

|  |
| --- |
| void MergeSort(a, p, r) {  if (p < r) {  q = parteEntera((q+r)/2);  MergeSort(a, p, q);  MergeSort(a, q+1, r);  Merge(a, p, q, r);  }  } |

El análisis para este problema es el siguiente.

