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.

```
int funcionRecursiva(int num){
  if(num==0)
    return 1;
  else if(num<3){
    resultado=0;
    for(i=0;i<num*num;i++)
      resultado*=num;
    return resultado;
  }else
      return funcionRecursiva(num-1)*funcionRecursiva(n-2)/funcionRecursiva(num-3);
}
```

$$T(n)=\begin{cases}0, & n=0\\ n^2, & 0<n<3\\ T(n-1)+T(n-2)+T(n-3)+5\end{cases}$$

Considerando las operaciones aritmeticas

$T(n)=T(n-1)+T(n-2)+T(n-3)+5$

$T(n)-T(n-1)-T(n-2)-T(n-3)=5$ $\quad x^3=T(n)$, $d=0$ & $b=1$

$(x^3-x^2-x-1)(x-1)=0$

$r_1=1.8392$
$r_2=-0.4196+0.6062i$
$r_3=-0.4196-0.6062i$
$r_4=1$

$\Rightarrow T(n)=c_1r_1^n+c_2r_2^n+c_3r_3^n+c_4r_4^n$

$O(c_1r_1^n)$ mientras que $c_1>0$

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, ...).

```
int Tribonacci(int num){
    if (num == 0)
        return 0;
    else if (num == 1 || num == 2)
        return 1;
    else
        return Tribonacci(num-1)+Tribonacci(num-2)+Tribonacci(num-3);
}
```

$$T(n) = \begin{cases} 1, & n = 0 \\ 2, & n = 1, 2 \\ T(n-1) + T(n-2) + T(n-3) + 7 \end{cases}$$

Considerando las comparaciones y las operaciones aritmeticas

$T(n) = T(n-1) + T(n-2) + T(n-3) + 7$

$T(n) - T(n-1) - T(n-2) - T(n-3) = 7$

Haciendo un cambio de variable $x^3 T(n)$, $d = 0$ & $b = 1$

$(x^3 - x^2 - x - 1)(x - 1) = 0$

$r_1 = 1.8392$

$r_2 = -0.4196 + 0.6062i$

$r_3 = -0.4196 - 0.6062i$

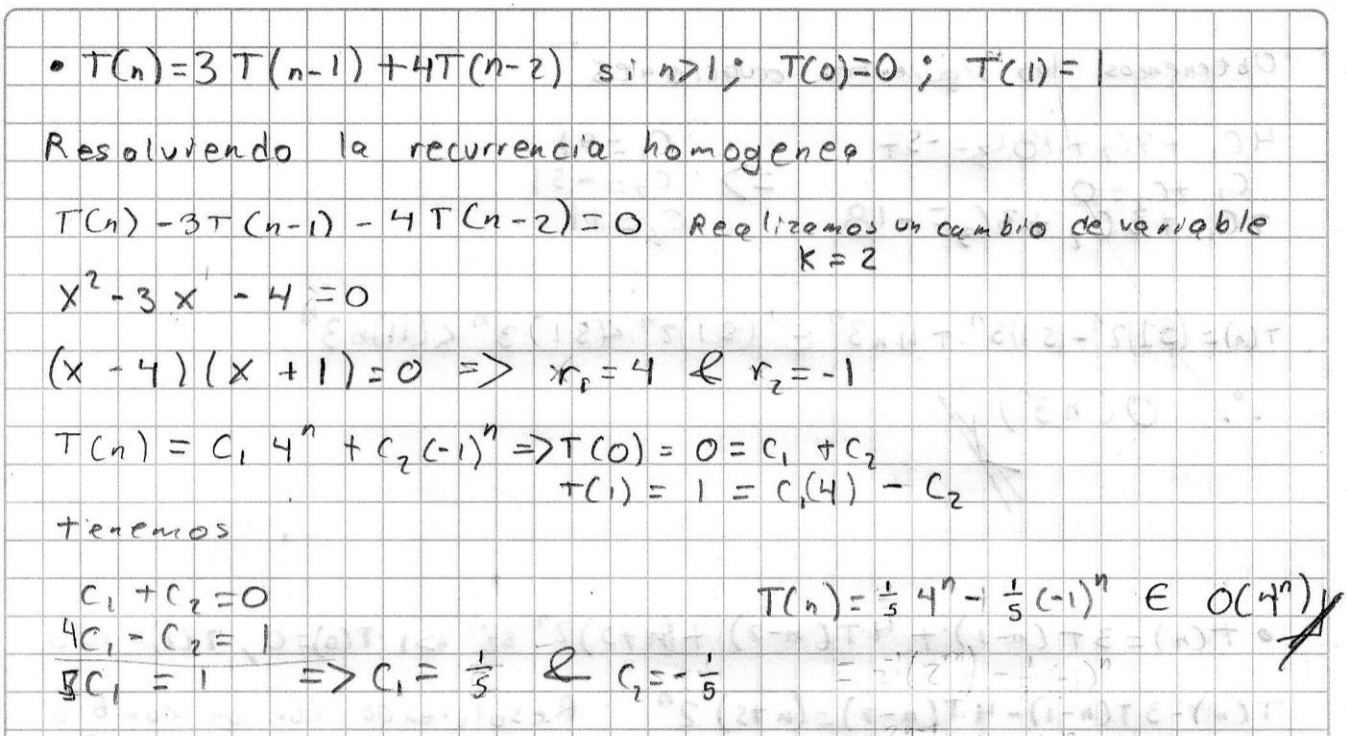
$r_4 = 1$

$\Rightarrow T(n) = C_1 r_1^n + C_2 r_2^n + C_3 r_3^n + C_4 r_4^n$

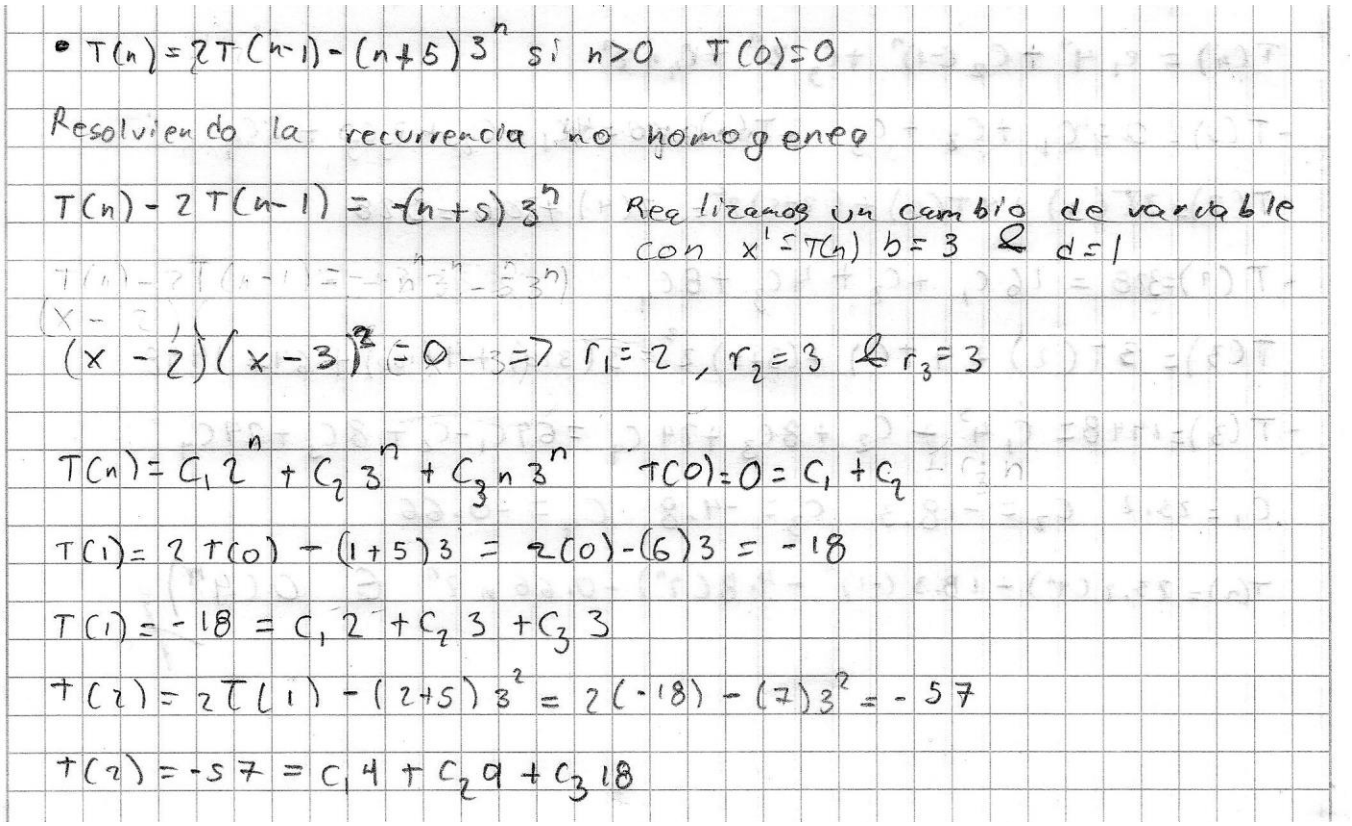
$O(C_1 r_1^n)$ mientras que $C_1 > 0$

Resolver las siguientes ecuaciones y dar su orden de complejidad.

Problema 1



Problema 2



Obtenemos las siguientes ecuaciones

$4C_1 + 9C_2 + 18C_3 = -57$
$C_1 + C_3 = 0$
$2C_1 + 3C_2 + 3C_3 = -18$

$\Rightarrow$ $C_1 = 51$, $C_2 = -51$, $C_3 = 11$

$T(n) = (51)2^n - (51)3^n + 11n3^n \in (51)2^n 4(51)3^n < (11)n3^n$

$\therefore O(n3^n)$

Problema 3

- $T(n) = 3T(n-1) + 4T(n-2) + (n+5)2^n$ si $n>1$ $T(0)=0$, $T(1)=100$

$T(n) - 3T(n-1) - 4T(n-2) = (n+5)2^n$ Resolviendo con un cambio de variable $x^2 = T(n)$, $b=2$ & $d=1$

$(x^2 - 3x - 4)(x-2)^2 = 0$

$(x-4)(x+1)(x-2)^2 = 0$ $r_1 = 4$, $r_2 = -1$, $r_3 = 2$ & $r_4 = 2$

$T(n) = c_1 4^n + c_2 (-1)^n + c_3 2^n + c_4 n 2^n$

$T(0) = 0 = 4c_1 + c_2 + c_3$ $T(1) = 100 = 4c_1 - c_2 + 2c_3 + 2c_4$

$T(2) = 3T(1) + 4T(0) + (2+5)2^2 = 7(4) + 3(0) = 328$

$T(2) = 328 = 16c_1 + c_2 + 4c_3 + 8c_4$

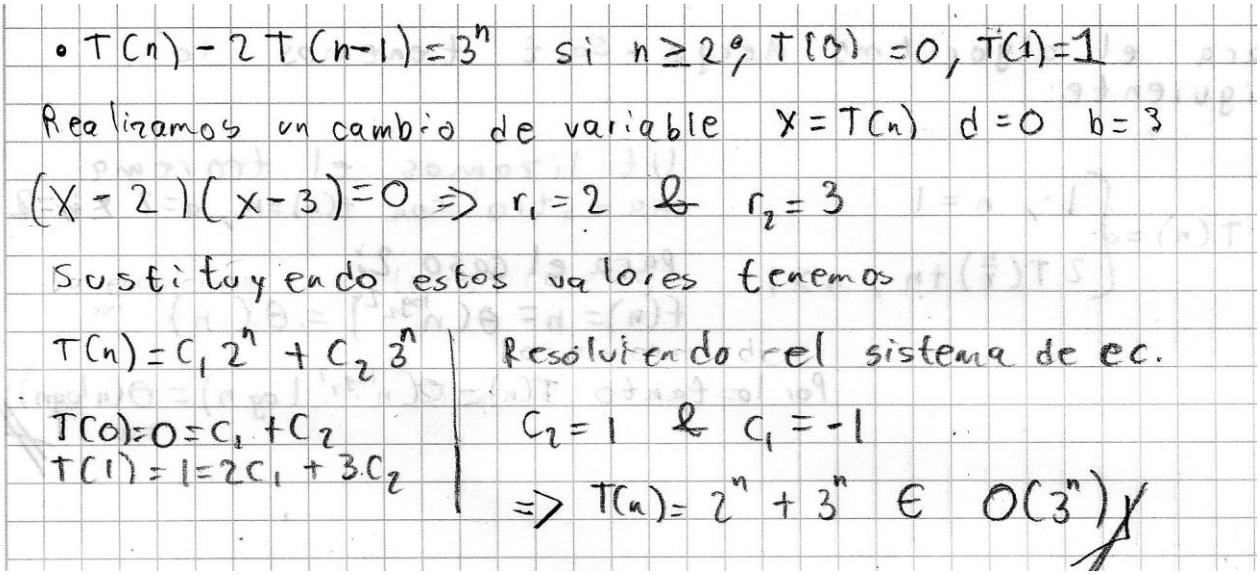
$T(3) = 3T(2) + 4T(1) + (3+5)2^3 = 3(328) + 4(100) + 64 = 1448$

$T(3) = 1448 = c_1 4^3 + c_2 + 8c_3 + 24c_4 = 64c_1 - c_2 + 8c_3 + 24c_4$

$c_1 = 23.2$ $c_2 = -18.3$ $c_3 = -4.8$ $c_4 = -0.66$

$T(n) = 23.2(4^n) + 18.3(-1)^n + 4.8(2^n) - 0.66n2^n \in O(4^n)$

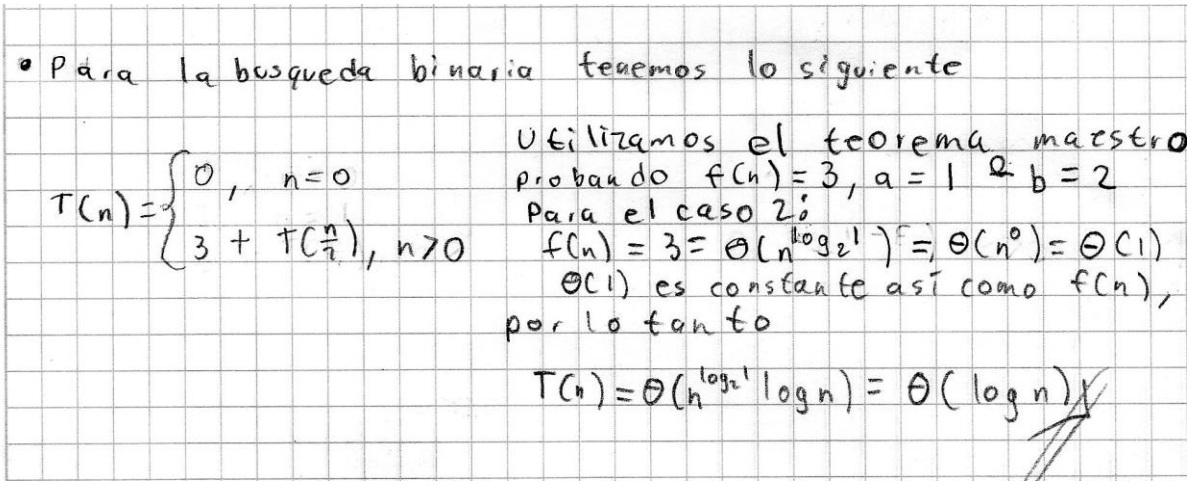
Problema 4



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.