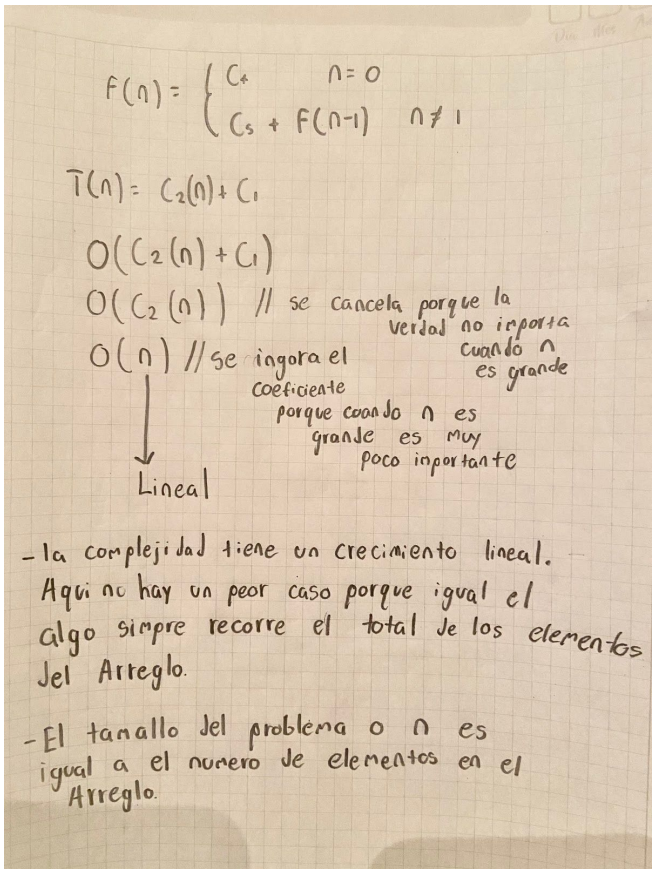
```
    Ejercicio 2:
    O(n)

public static int ArraymaxCalc(int[] nums,int index){
   int max = nums[index]; // Const
    if(index == 0){    //Const
        max = nums[0]; //Const
    }
    if(index != 0){ //Const
        int temporal = arrayMax.ArraymaxCalc(nums,index-1); //T(n-1)
        if(temporal > max){ //Const
            max = temporal; //Const
        }
    }
    return max; //Const
}
```

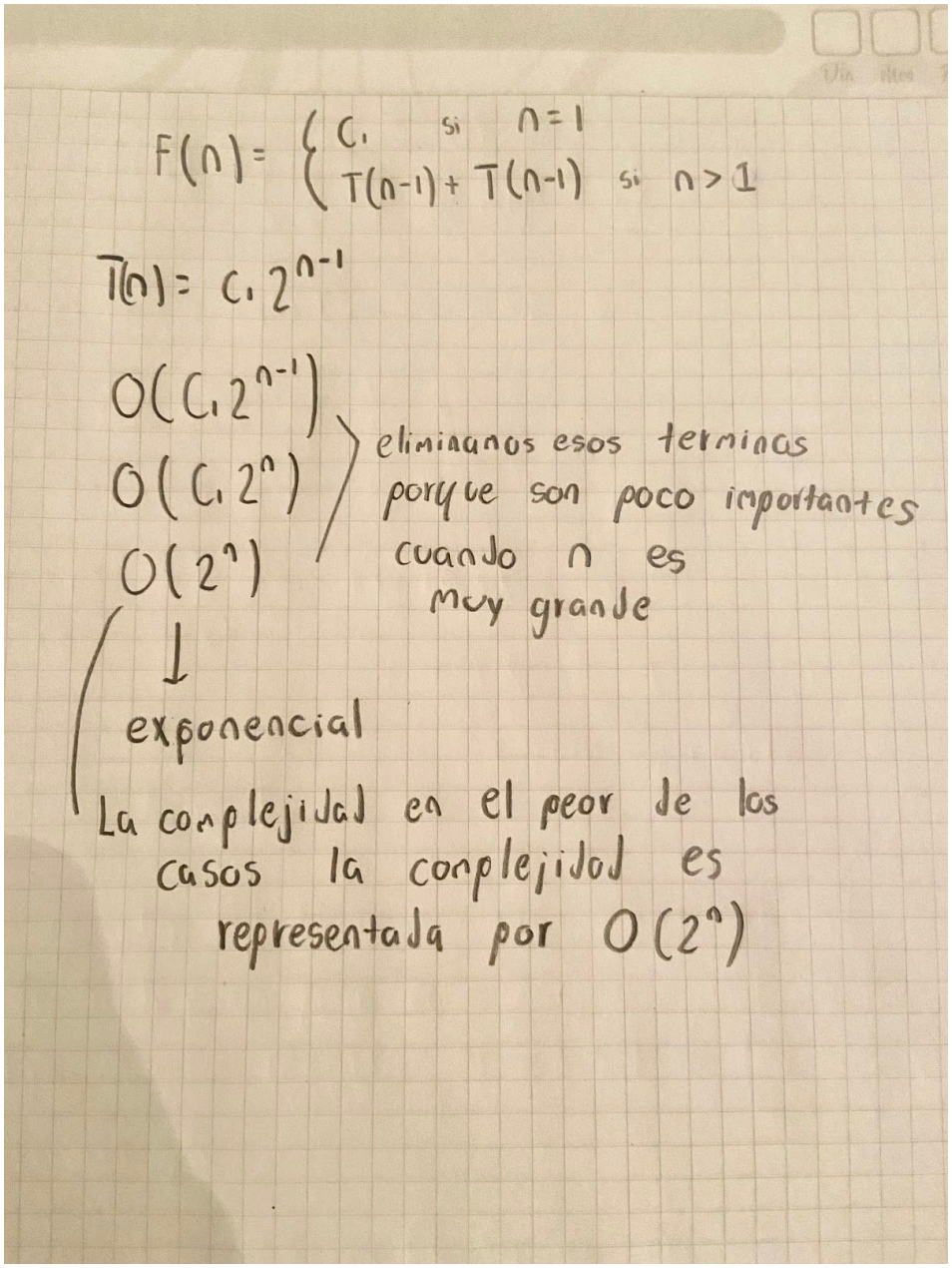
1.2 Identificar el tamaño del n:
El tamaño del arreglo es el número de elementos que me faltan por chequear no sea más grande que el anterior.



Ejercicio 2
O(2^n)

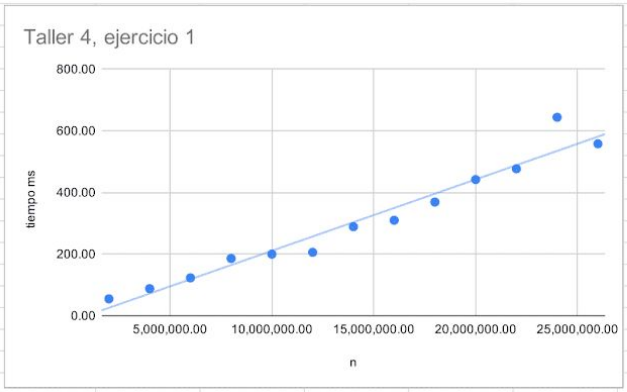
```
16      public static boolean subVolumen(int[] objetos, int total, int index) {
17          if (index >= objetos.length) { C17 = 2 Instrucciones = O(1)
18              return total == 0; C18 = 2 Instrucciones = O(1)
19          }                                                          C20 n*n instrucciones
20          return subVolumen(objetos, total - objetos[index], index + 1) || subVolumen(objetos, total, index + 1);
21      }                                          F(n) = F(n-1)*F(n-1) + 4
22  }
```

2.2 Identificar el tamaño del n:
El tamaño d n es el número de elementos en el arreglo.



Graficas

n	tiempo ms
2,000,000.00	55.00
4,000,000.00	88.00
6,000,000.00	123.00
8,000,000.00	186.00
10,000,000.00	200.00
12,000,000.00	206.00
14,000,000.00	289.00
16,000,000.00	310.00
18,000,000.00	369.00
20,000,000.00	442.00
22,000,000.00	477.00
24,000,000.00	644.00
26,000,000.00	558.00



n	ms
16	2
17	3
18	3
19	5
20	7
21	11
22	16
23	32
24	52
25	118
26	203
27	459
28	787
29	1906
30	3162
31	7421
32	12706
33	31420
34	50712
35	118352
36	221356