

Samuel Peña 202028273

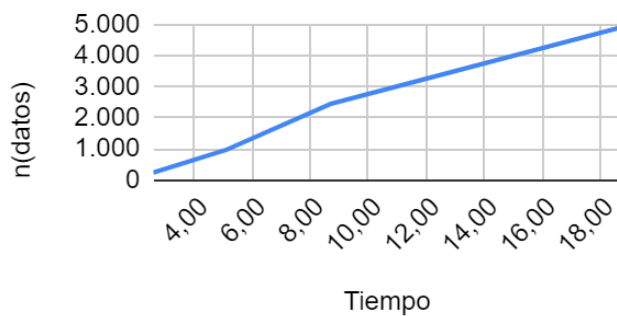
Tomas Diaz 202220658

Alejandro Narváez 202123110

REQ 1:

MÁQUINA Alejandro		
REQ 1		
%	n(datos)	tiempo
5%	245	2,58
20%	980	5,11
50%	2451	8,71
100%	4903	18,72

tiempo frente a n(datos)



Req 1:

Tamaño : 1 asignación

Años : Crea un diccionario por cada año  $O(n)$

busca : 1 asignación

Mayer : 1 asignación

for :  $O(n)$

Repeticiones : 1 asignación

Respuesta :  $O(N)$  ordena los años. Debe recorrer toda la lista

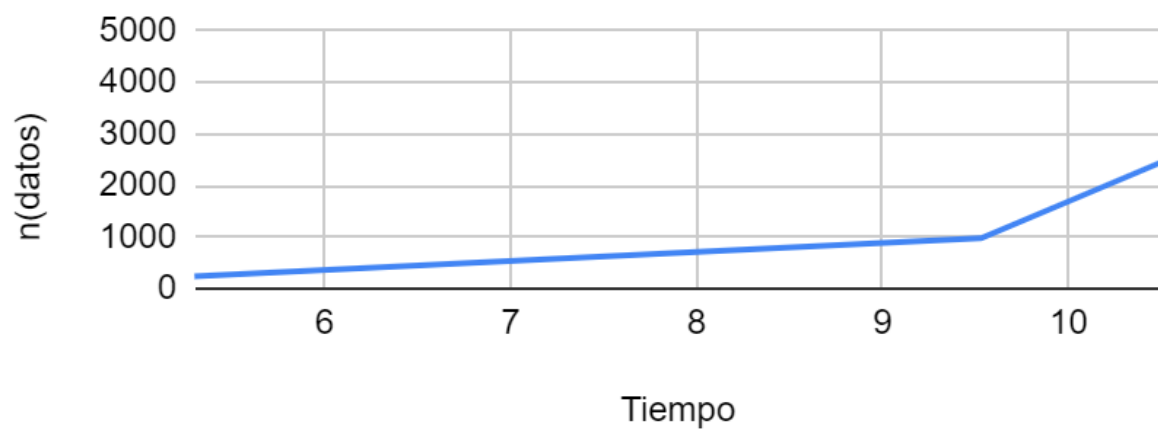
Complejidad =  $O(N)$

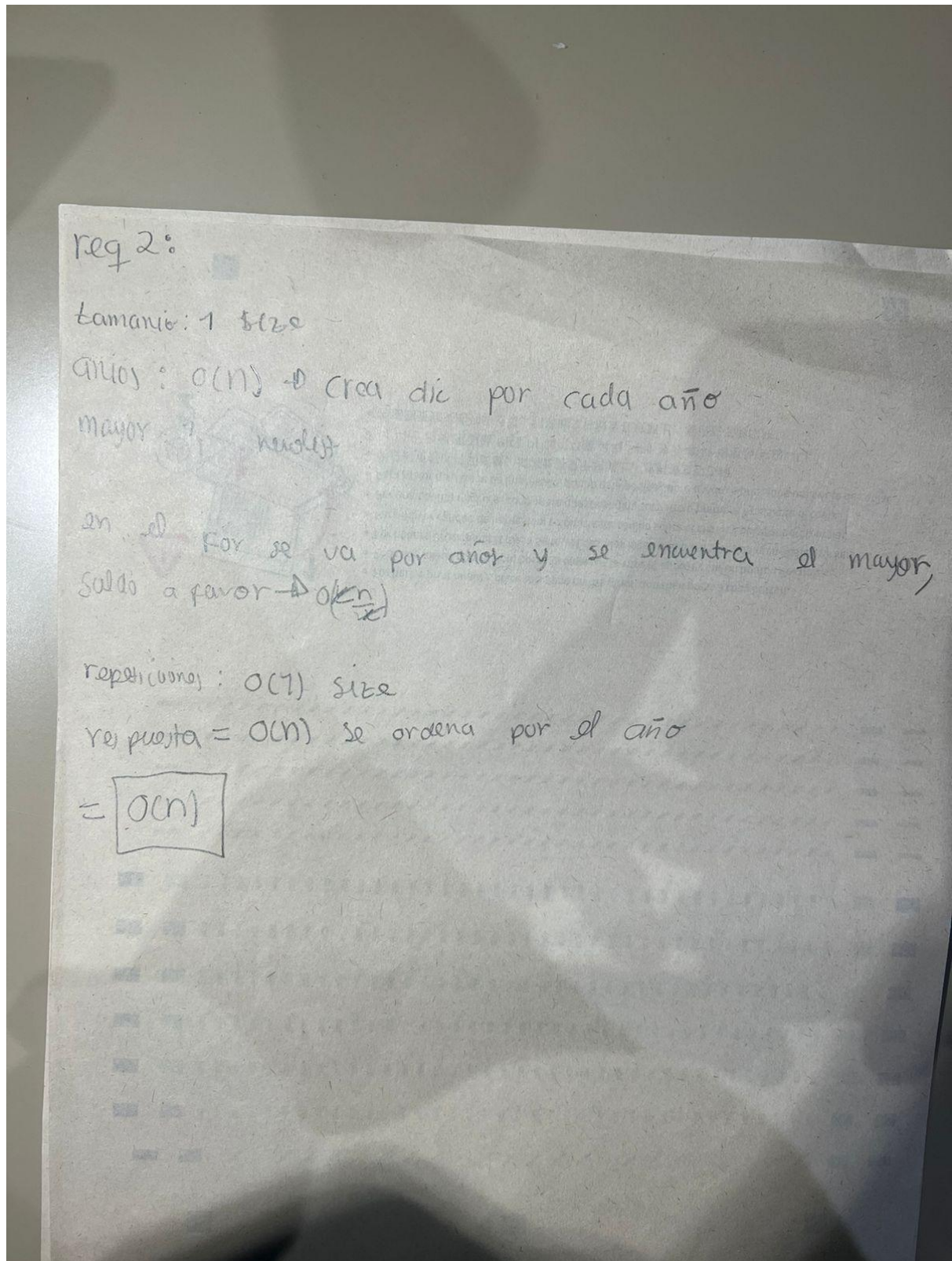
En este caso la complejidad es  $O(n)$  en el peor caso ya que es necesario que la función recorra todos los datos compuestos en la lista por medio de ciclos.

REQ 2:

%	n(datos)	tiempo
5%	245	5,3
20%	980	9,53
50%	2451	10,5
100%	4903	19.3

req 2





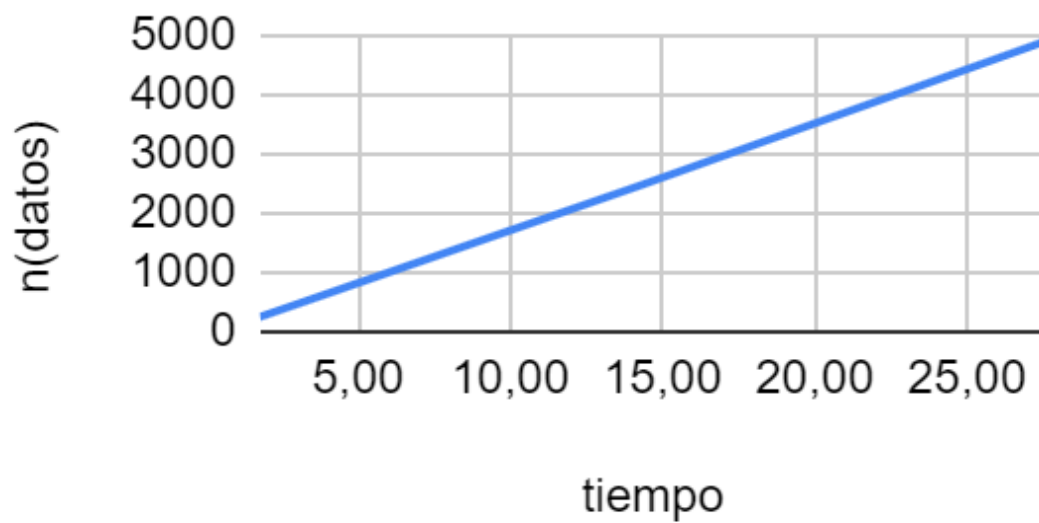
En este caso el peor y el caso promedio es  $O(N)$  porque para la iteración de datos siempre debe hacer una búsqueda uno a uno de datos.

### REQ 3:

REQ 3		
-------	--	--

%	n(datos)	tiempo
5%	245	1,78
20%	980	5,91
50%	2451	14,21
		27,59
100%	4903	

## n(datos) frente a tiempo



Cálculo de complejidad:

DD
MM
AA

## Análisis de complejidad

**Reg 3:**

Complejidad (O)

crea lista: 1

Tamaño: 1

crea de iteración:  $n$

- ciclo de 10 vueltas:

1 vuelta: lista iteración actual:  $\leq n$

encuentra menor (iteración actual)  $\leq n$

carga a lista (iteración)  $\leq n$


añade elemento 1

Total ciclo  $\leq 30n$

Ordena lista que siempre tiene 0 elementos ameros:  $\leq O(n^2)$   $\rightarrow$  (constante)

- Complejidad total  $\leq 1 + 1 + n + 30n \leq kn$  donde  $k$  es una constante dada.

$\therefore$  complejidad Reg 3 =  $O(n)$



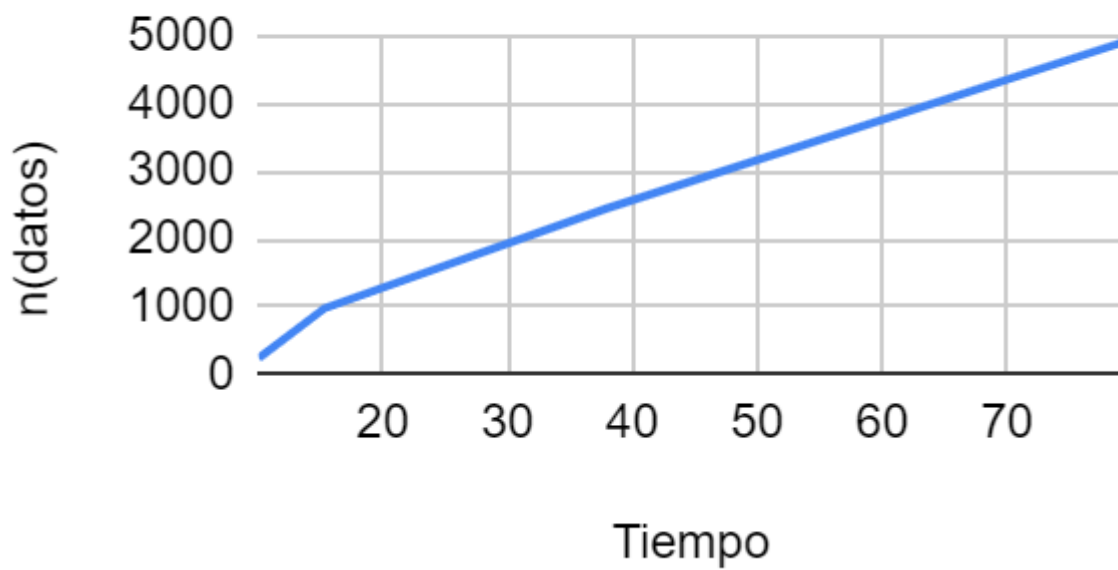
**Banco Caja Social**  
Más banco. Más amigo.

Véase que concuerda, a la tabla y gráfico, el tiempo crece de forma aparentemente lineal con respecto al  $n$  de datos.

req 5:

%	n(datos)	tiempo
5%	245	10,07
20%	980	15,35
50%	2451	37,94
100%	4903	79,32

tiempo frente a n(datos)





Req 5.

$O(n) = (O(n))$

Orden años =  $O(n)$

1. For:  $O(n)$  → por las llaves que por valor máximo son 10

1 IP:  $O(n)$  → encuentra 3 mayores y menores  
Sub Sector =  $O(n)$  → organiza en sub sectores

2. For:  $O(n)$  va por sector a sector

3. For:  $O(n)$  se repite 4 veces

suma =  $O(n)$

$O_2 = O(n)$

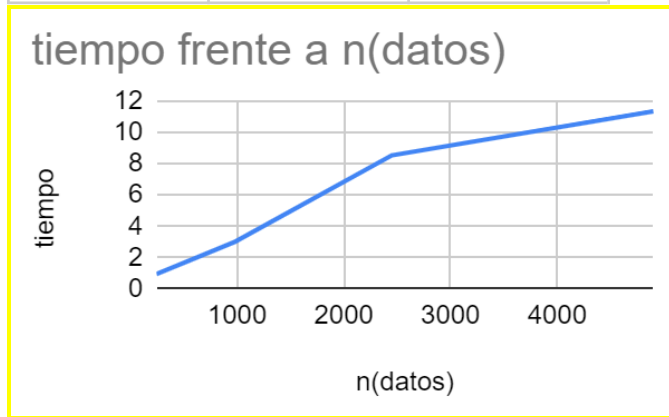
while  $O(n)$  para por todos en encontrar mayor.

= sería más de  $O(n)$  porque cuando organiza en fecha y después (sectores), hace una iteración completa en años y en sectores, hace una un poco menor, pero extensa

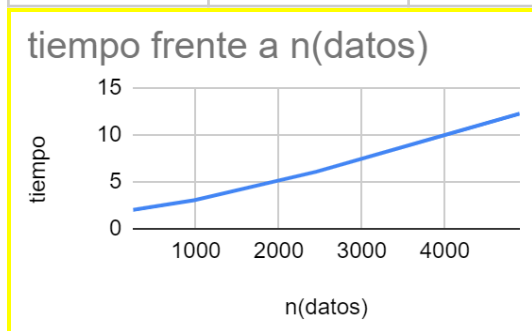
este caso es de  $O(n)$  porque a pesar de que se hacen varias reiteraciones, ninguna vuelve a basar en todo los datos, esto quiere decir que llegan situaciones donde se reitera dentro de una reiteración y aquí se hace a punto de constantes la complejidad y en ningún caso logra superar la de  $O(n)$

REQ 6:

REQ 6 (2021)		
%	n(datos)	tiempo
5%	245	0,95
20%	980	3
50%	2451	8,52
100%	4903	11,32

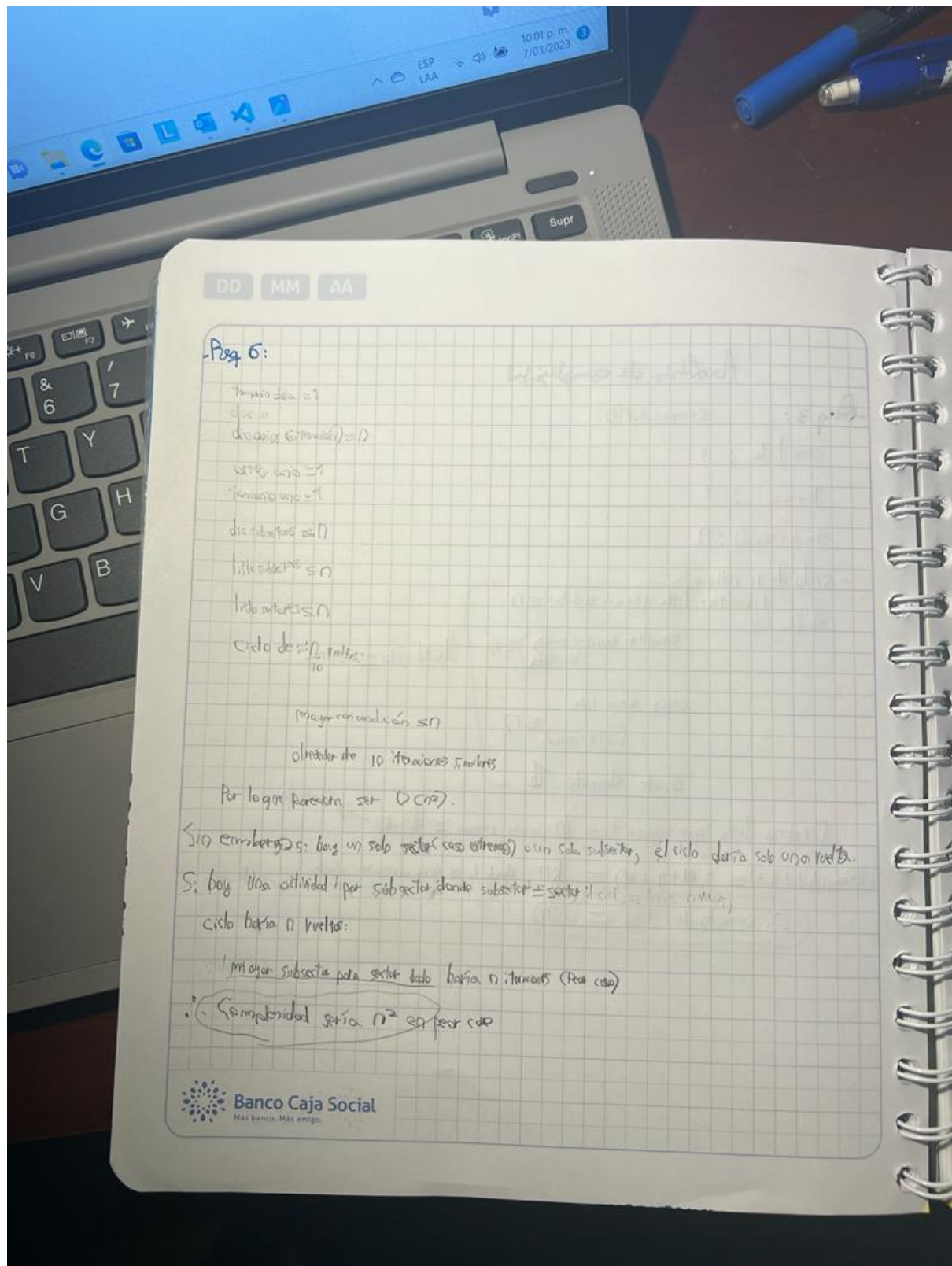


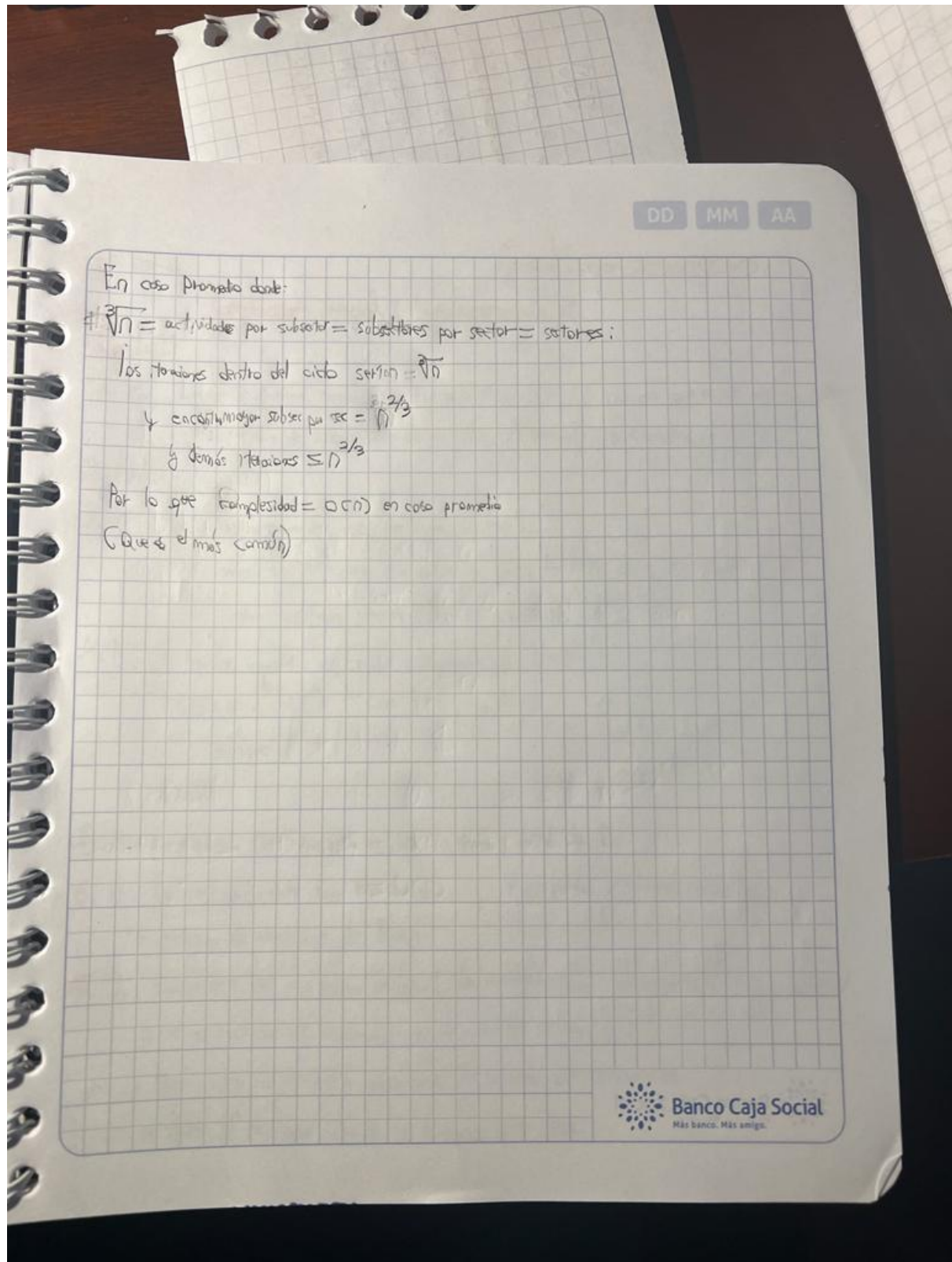
REQ 6 (2020)		
%	n(datos)	tiempo
5%	245	2
20%	980	3,03
50%	2451	6,06
100%	4903	12,24



Cálculo complejidad:







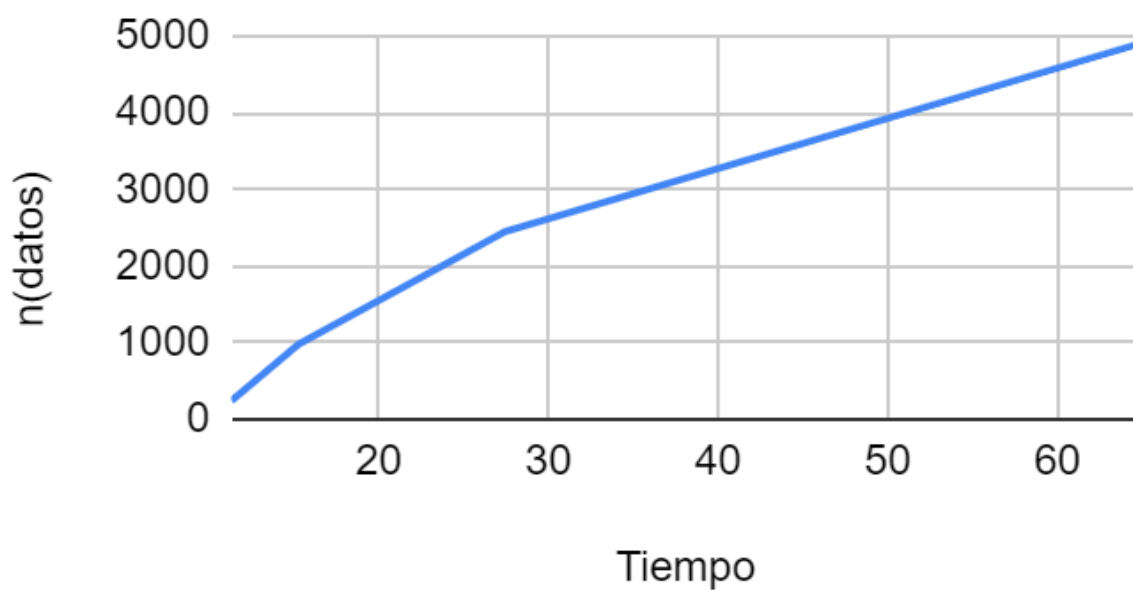
Como puede verse, en el peor caso la complejidad sería de  $n$  cuadrado y en el promedio  $n$ . Asumiendo distribución normal de los datos el promedio es más probable, y los datos tienden a ese valor, como se evidencia en la gráfica.

Hay que notar también que si bien el peor caso sería  $n$  cuadrado, la constante que divide a esta expresión sería de alrededor de 100, por lo que la convexidad de la función se empezaría a notar después.

Así también la función  $T(n)$  no es absolutamente predecible en este caso, pues hay procesos que pueden incrementarse o no en función de  $n$  dependiendo de si son o no del año seleccionado.

req 7:

req 7 i

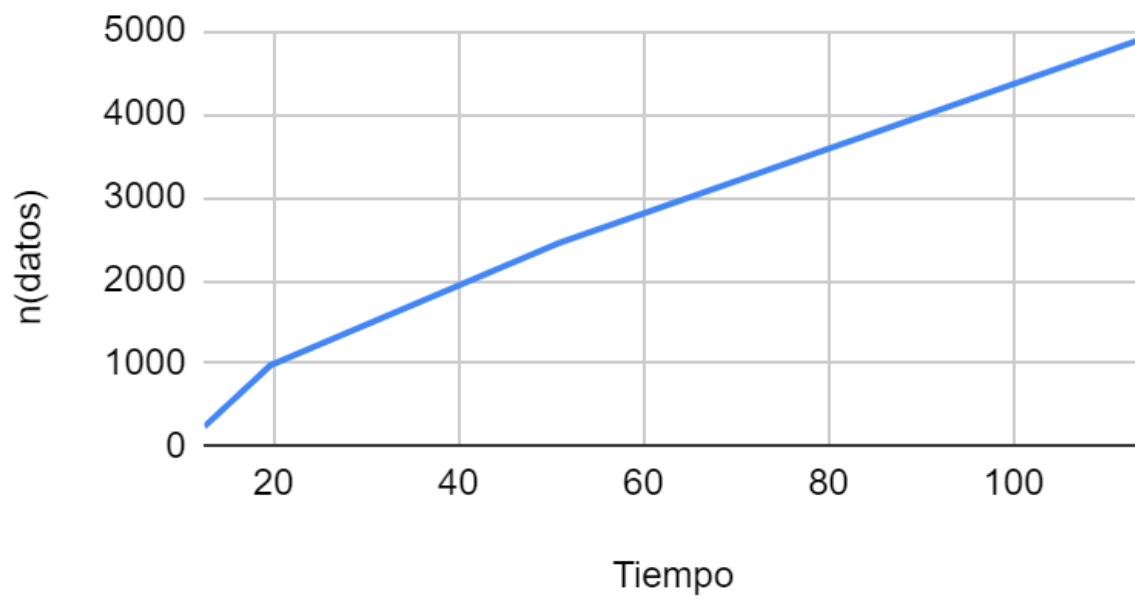


REQ 7	LAPSO 2012-2015	NUMERO 5
%	n(datos)	tiempo
5%	245	11,31
20%	980	15,25
50%	2451	27,45
100%	4903	64,86

REQ 7	LAPSO 2012-2020	NUMERO 10
%	n(datos)	tiempo
5%	245	12,54
20%	980	19,69
50%	2451	50,88

100%	4903	113,88
------	------	--------

req 8 ii





Reg 7:

Tamaño: 7

Cinco:  $O(n)$

Orden cinco:  $O(1)$

For: es de orden 7-10 porque en este caso solo es el intervalo de años y crea una lista al respecto con los años válidos.

While: Como solo va por la lista primera, que es igual de larga a la variedad de años en el lapso.

Merge:  $n \log n$

While se repite la cantidad que asigne para encontrar el top for como esta organizado crea una lista con el primero de cada uno.

encontrar menor entre los años encontrados ya ordenados

se mete a lista y se borra previamente

Complejidad  $n \log n$ , porque que mas se demora.

este caso si es un poco mas demorado porque uso la funcion de merge, en un ciclo por las veces del año, de igual manera en lo demas complejidades nunca se logra superar, porque siempre son reiteraciones mas pequeñas que se calcula sobre las constantes, que termina siendo una complejidad de  $O(n)$