Guion 3:

Actividad 1:

Complejidades de las clases dadas:

- ***Sustraccion1****:* O(n).
  a=1, b=1, k=0 => O(n^{k+1}).
- ***Sustraccion2****:* O(n^2).
  a=1, b=1, k=1 => O(n^{k+1}).
- ***Sustraccion3****:* O(2^n).
  a=2, b=1, k=0 => O($a^{n/b}$).
- ***Sustraccion4****: O($3^{n/2}$).*
  a=3, b=2, k=0 => O($a^{n/b}$).
- ***Division1****:* O(n).
  a=1, b=3, k=1 => O(n^k).
- ***Division2****:* O(n·logn).
  a=2, b=2, k=1 => O(n^k·logn).
- ***Division3****:* O(n).
  a=2, b=2, k=0 => O($n^{log_b a}$).
- ***Division4****:* O(n^2).
  a=4, b=2, k=0 => O($n^{log_b a}$).
- *Complejidades de* ***VectorSum1****:*

	t sum1	*t sum2*	*t sum3*
Complejidad	***O(n)***	***O(n)***	***O(n)***

- *Complejidades de* ***Fibonacci1****:*

	t fib1	*t fib2*	*t fib3*	*t fib4*
Complejidad	***O(n)***	***O(n)***	***O(n)***	***O(1.6^n)***

Medición de tiempos de las clases:

- Tiempos de los métodos de *Sustracción*:

N	*t Sustraccion1*	*t Sustraccion2*	*t Sustraccion3*	*t Sustraccion4*
1	0,00000015	0,00000261	0,000000112	0,00000015
2	0,00000235	0,00000565	0,00000670	0,00000034
4	0,00000431	0,00001316	0,00002820	0,00001478
8	0,00000818	0,00003662	0,00047700	0,00013431
16	0,00001530	0,00011803	0,12530000	0,01110000
32	0,00005000	0,00041629	8121,50000	70,2800000
64	0,00013500	0,00133500		
128	0,00025400	0,00539900		
256	0,00051900	0,01995800		
512	0,00097300	0,09010000		
1024	0,00186900	0,34720000		
2048	0,01852600	1,38040000		
4096	0,04560200	5,53600000		
8192	0,05859400	22,1661000		
16384	0,11120000	102,338000		
Complejidad	***O(n)***	***O(n^2).***	***O(2^n)***	***O($3^{n/2}$)***

Sustraccion1: Tomamos $n_1 = 8192$ y $n_2 = 16384$, con $t_1 = 0{,}058594$, para demostrar que la complejidad del algoritmo es O(n) aplicamos la siguiente formula:

$$t_2 = \frac{f(n_2)}{f(n_1)} * t_1 = \frac{n_2}{n_1} * t_1 = \frac{16384}{8192} * 0{,}058594 => \boldsymbol{t_2 = 0,117188\ s}$$

Como el tiempo resultante es aproximado al que se ha obtenido en la gráfica queda demostrado que la complejidad del algoritmo es O(n).

Sustraccion2: Tomamos $n_1 = 8192$ y $n_2 = 16384$, con $t_1 = 22{,}1661$, para demostrar que la complejidad del algoritmo es $O(n^2)$ aplicamos la siguiente formula:

$$t_2 = \frac{f(n_2)}{f(n_1)} * t_1 = \frac{{n_2}^2}{{n_1}^2} * t_1 = \frac{16384^2}{8192^2} * 22{,}1661 => \boldsymbol{t_2 = 88,6644\ s}$$

Como el tiempo resultante es aproximado al que se ha obtenido en la gráfica queda demostrado que la complejidad del algoritmo es $O(n^2)$.

Sustraccion3: Tomamos $n_1 = 16$ y $n_2 = 32$, con $t_1 = 0{,}1253$, para demostrar que la complejidad del algoritmo es $O(2^n)$ aplicamos la siguiente formula:

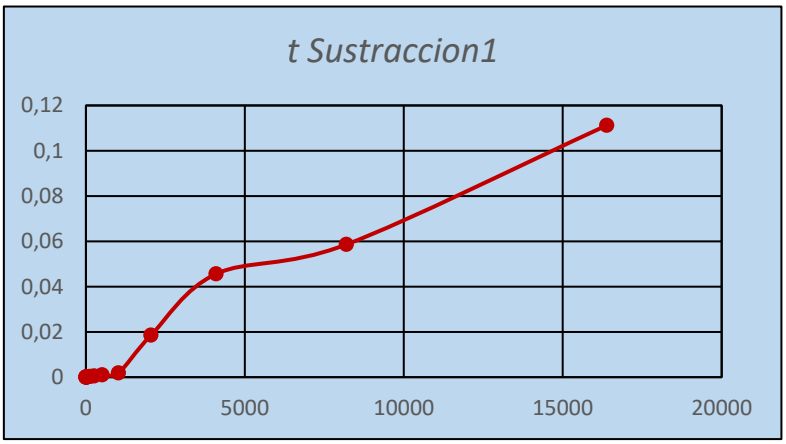
$$t_2 = \frac{f(n_2)}{f(n_1)} * t_1 = \frac{2^{n_2}}{2^{n_1}} * t_1 = \frac{2^{32}}{2^{16}} * 0{,}1253 => \boldsymbol{t_2 = 8211,6608\ s}$$

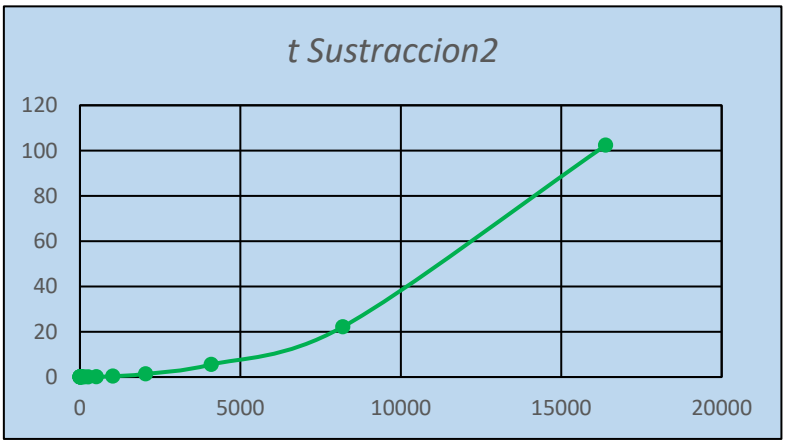
Como el tiempo resultante es aproximado al que se ha obtenido en la gráfica queda demostrado que la complejidad del algoritmo es $O(2^n)$.

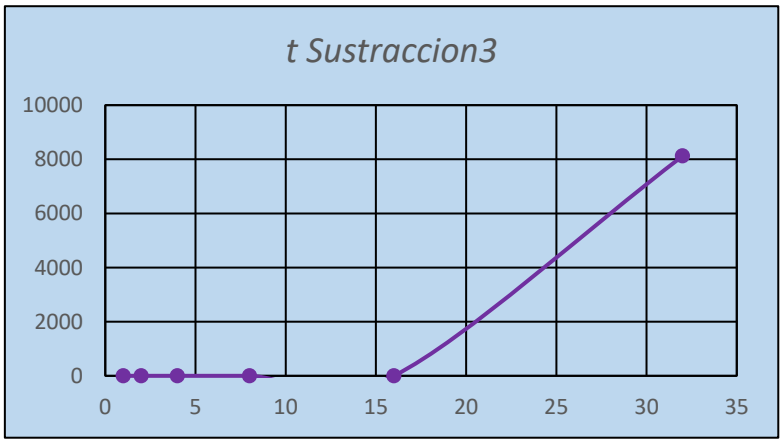
Sustraccion4: Tomamos $n_1 = 16$ y $n_2 = 32$, con $t_1 = 0{,}0111$, para demostrar que la complejidad del algoritmo es $O(3^{n/2})$ aplicamos la siguiente formula:

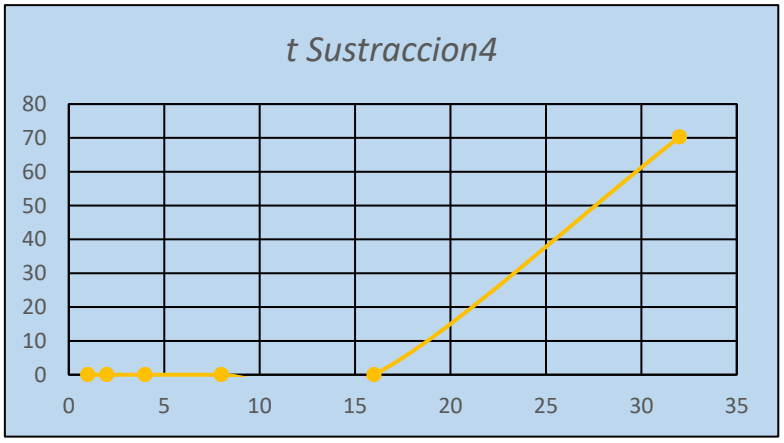
$$t_2 = \frac{f(n_2)}{f(n_1)} * t_1 = \frac{3^{\frac{n_2}{2}}}{3^{\frac{n_1}{2}}} * t_1 = \frac{3^{\frac{32}{2}}}{3^{\frac{16}{2}}} * 0{,}0111 => \boldsymbol{t_2 = 72,82\ s}$$

Como el tiempo resultante es aproximado al que se ha obtenido en la gráfica queda demostrado que la complejidad del algoritmo es $O(3^{n/2})$.


t Sustraccion1
0,12
0,1
0,08
0,06
0,04
0,02
0
0
5000
10000
15000
20000


t Sustraccion2
120
100
80
60
40
20
0
0
5000
10000
15000
20000


t Sustraccion3
10000
8000
6000
4000
2000
0
0
5
10
15
20
25
30
35


t Sustraccion4
80
70
60
50
40
30
20
10
0
0
5
10
15
20
25
30
35

- Tiempos de los métodos de *División*:

N	*t Division1*	*t Division2*	*t Division3*	*t Division4*
1	0,00000011	0,00000012	0,00000015	0,000006
2	0,00000172	0,00000805	0,00000669	0,000026
4	0,00000385	0,00001135	0,00000749	0,000030
8	0,00000621	0,00003881	0,00002877	0,000403
16	0,00001087	0,00006101	0,00003268	0,000511
32	0,00002306	0,00018776	0,00012182	0,006298
64	0,00004018	0,00031111	0,00013560	0,008114
128	0,00007836	0,00087669	0,00049357	0,100785
256	0,00015599	0,00152116	0,00055440	0,130964
512	0,00029511	0,00392000	0,00198914	1,606600
1024	0,00058426	0,00691000	0,00224720	2,139600
2048	0,00116325	0,01764000	0,00792000	28,20000
4096	0,00271000	0,03227000	0,00884000	28,58000
8192	0,00533000	0,07949000	0,03179000	447,0000
16384	0,01065000	0,14403000	0,03650000	460,0000
Complejidad	***O(n)***	***O(n·logn)***	***O(n)***	***O(n²)***

Division1: Tomamos $n_1 = 8192$ y $n_2 = 16384$, con $t_1 = 0{,}00533$, para demostrar que la complejidad del algoritmo es O(n) aplicamos la siguiente formula:

$$t_2 = \frac{f(n_2)}{f(n_1)} * t_1 = \frac{n_2}{n_1} * t_1 = \frac{16384}{8192} * 0{,}00533 => \boldsymbol{t_2 = 0{,}01066\ s}$$

Como el tiempo resultante es aproximado al que se ha obtenido en la gráfica queda demostrado que la complejidad del algoritmo es O(n).

Division2: Tomamos $n_1 = 8192$ y $n_2 = 16384$, con $t_1 = 0{,}07949$, para demostrar que la complejidad del algoritmo es O(n·logn) aplicamos la siguiente formula:

$$t_2 = \frac{f(n_2)}{f(n_1)} * t_1 = \frac{n_2 * \log n_2}{n_1 * \log n_1} * t_1 = \frac{16384 * \log 16384}{8192 * \log 8192} * 0{,}07949 => \boldsymbol{t_2 = 0{,}17121\ s}$$

Como el tiempo resultante es aproximado al que se ha obtenido en la gráfica queda demostrado que la complejidad del algoritmo es O(n·logn).

Division3: Tomamos $n_1 = 4096$ y $n_2 = 16384$, con $t_1 = 0{,}00884$, para demostrar que la complejidad del algoritmo es O(n) aplicamos la siguiente formula:

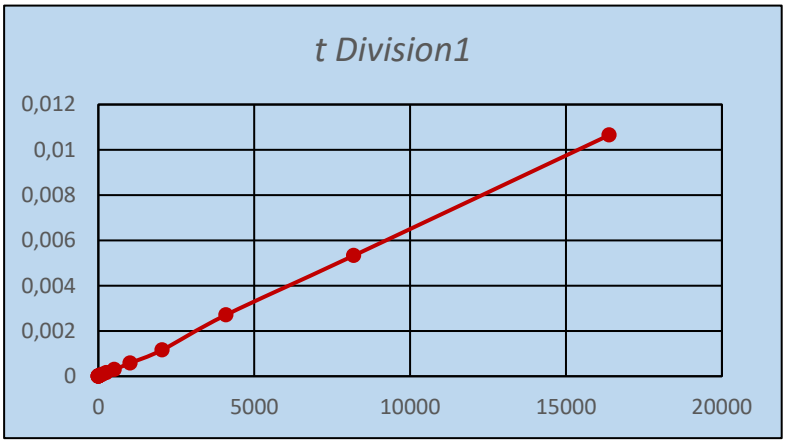
$$t_2 = \frac{f(n_2)}{f(n_1)} * t_1 = \frac{n_2}{n_1} * t_1 = \frac{16384}{4096} * 0{,}00884 => \boldsymbol{t_2 = 0{,}03536\ s}$$

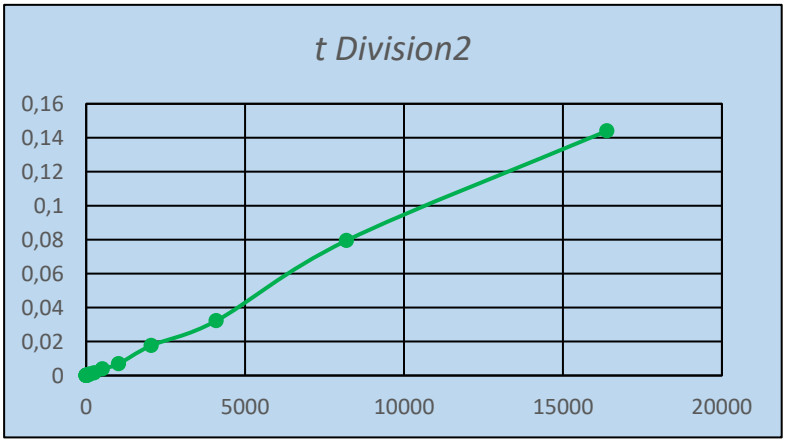
Como el tiempo resultante es aproximado al que se ha obtenido en la gráfica queda demostrado que la complejidad del algoritmo es O(n).

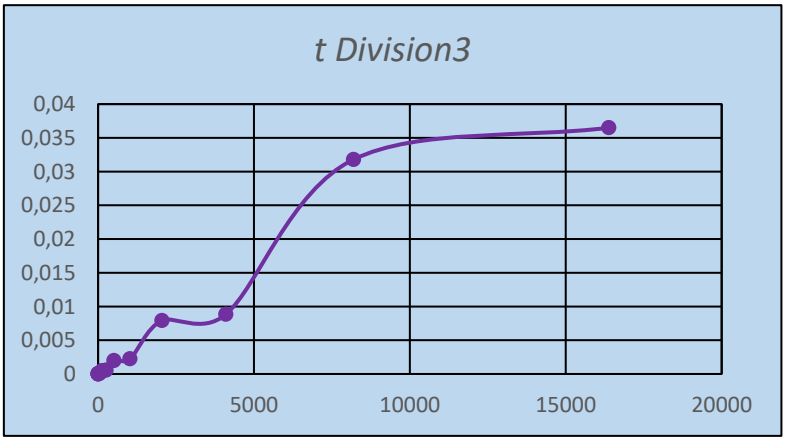
Division4: Tomamos $n_1 = 4096$ y $n_2 = 16384$, con $t_1 = 28{,}58$, para demostrar que la complejidad del algoritmo es $O(n^2)$ aplicamos la siguiente formula:

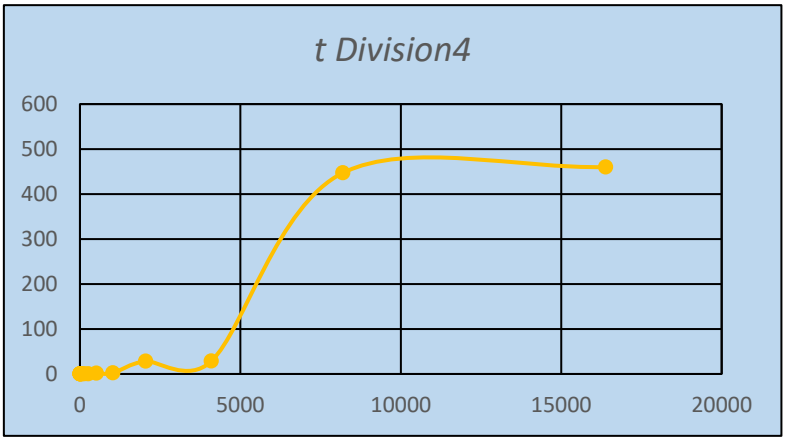
$$t_2 = \frac{f(n_2)}{f(n_1)} * t_1 = \frac{{n_2}^2}{{n_1}^2} * t_1 = \frac{16384^2}{4096^2} * 28{,}58 => \boldsymbol{t_2 = 457{,}28\ s}$$

Como el tiempo resultante es aproximado al que se ha obtenido en la gráfica queda demostrado que la complejidad del algoritmo es $O(n^2)$.


t Division1
0,012
0,01
0,008
0,006
0,004
0,002
0
0
5000
10000
15000
20000


t Division2
0,16
0,14
0,12
0,1
0,08
0,06
0,04
0,02
0
0
5000
10000
15000
20000


t Division3
0,04
0,035
0,03
0,025
0,02
0,015
0,01
0,005
0
0
5000
10000
15000
20000


t Division4
600
500
400
300
200
100
0
0
5000
10000
15000
20000

- Tiempos de los métodos de *VectorSum1*:

N	*t sum1*	*t sum2*	*t sum3*
1	0,0000018	0,0000007	0,0000002
2	0,0000018	0,0000026	0,0000021
4	0,0000022	0,0000034	0,0000079
8	0,0000024	0,0000063	0,0000114
16	0,0000045	0,0000124	0,0000321
32	0,0000083	0,0000242	0,0000526
64	0,0000164	0,0000513	0,0001392
128	0,0000347	0,0000990	0,0002155
256	0,0000660	0,0001934	0,0005653
512	0,0001260	0,0003741	0,0008779
1024	0,0002450	0,0010664	0,0022762
2048	0,0004930	0,0023850	0,0035176
4096	0,0009870	0,0044164	0,0091672
8192	0,0019600	0,0091656	0,0141645
16384	0,0039440	0,0173356	0,0367278
Complejidad	***O(n)***	***O(n)***	***O(n)***

Sum1: Tomamos $n_1 = 8192$ y $n_2 = 16384$, con $t_1 = 0{,}0019600$, para demostrar que la complejidad del algoritmo es O(n) aplicamos la siguiente formula:

$$t_2 = \frac{f(n_2)}{f(n_1)} * t_1 = \frac{n_2}{n_1} * t_1 = \frac{16384}{8192} * 0{,}0019600 => \boldsymbol{t_2 = 0{,}00392\ s}$$

Como el tiempo resultante es aproximado al que se ha obtenido en la gráfica queda demostrado que la complejidad del algoritmo es O(n).

Sum2: Tomamos $n_1 = 8192$ y $n_2 = 16384$, con $t_1 = 0{,}0091656$, para demostrar que la complejidad del algoritmo es O(n) aplicamos la siguiente formula:

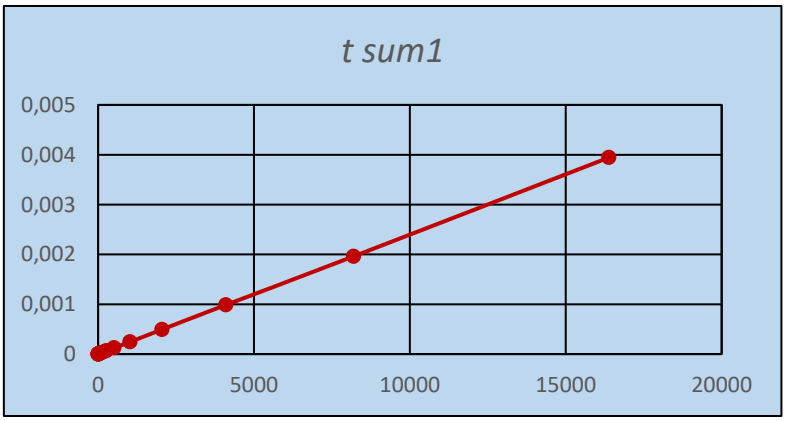
$$t_2 = \frac{f(n_2)}{f(n_1)} * t_1 = \frac{n_2}{n_1} * t_1 = \frac{16384}{8192} * 0{,}0091656 => \boldsymbol{t_2 = 0{,}0183312\ s}$$

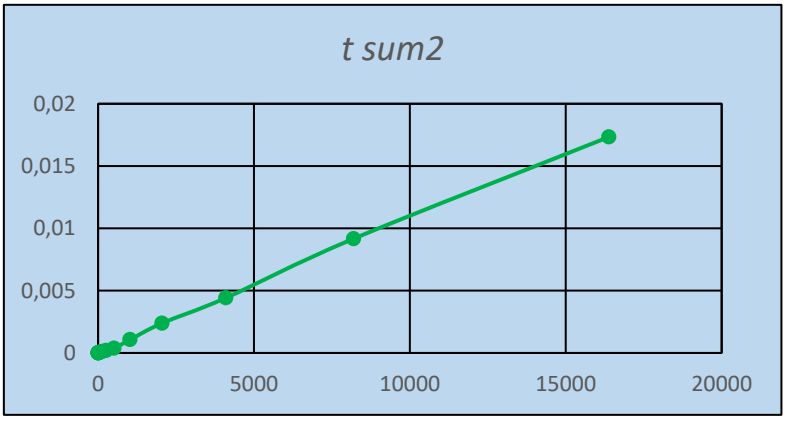
Como el tiempo resultante es aproximado al que se ha obtenido en la gráfica queda demostrado que la complejidad del algoritmo es O(n).

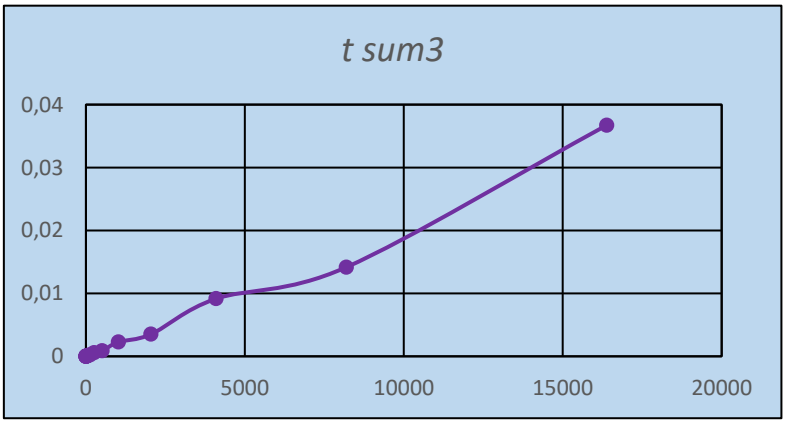
Sum3: Tomamos $n_1 = 8192$ y $n_2 = 16384$, con $t_1 = 0{,}0141645$, para demostrar que la complejidad del algoritmo es O(n) aplicamos la siguiente formula:

$$t_2 = \frac{f(n_2)}{f(n_1)} * t_1 = \frac{n_2}{n_1} * t_1 = \frac{16384}{8192} * 0{,}0141645 => \boldsymbol{t_2 = 0{,}028329\ s}$$

Como el tiempo resultante es aproximado al que se ha obtenido en la gráfica queda demostrado que la complejidad del algoritmo es O(n).


t sum1
0,005
0,004
0,003
0,002
0,001
0
0
5000
10000
15000
20000


t sum2
0,02
0,015
0,01
0,005
0
0
5000
10000
15000
20000


t sum3
0,04
0,03
0,02
0,01
0
0
5000
10000
15000
20000

- Tiempos de los métodos de *Fibonacci1*:

N	*t fib1*	*t fib2*	*t fib3*	*t fib4*
1	0,0000024	0,0000006	0,00000006	0,0000004
2	0,0000034	0,0000042	0,00000031	0,0000013
4	0,0000045	0,0000052	0,00000046	0,0000066
8	0,0000076	0,0000066	0,00000091	0,0000588
16	0,0000192	0,0000136	0,00000180	0,0028747
32	0,0000316	0,0000314	0,00000314	5,9100000
64	0,0000666	0,0000907	0,00000949	
128	0,0000714	0,0002194	0,00001960	
256	0,0001249	0,0005963	0,00004113	
512	0,0002413	0,0036522	0,00008258	
1024	0,0018056	0,0074026	0,00027753	
2048	0,0019917	0,0064578	0,00149228	
4096	0,0024245	0,0080170	0,01153120	
8192	0,0036851	0,0439900	0,01653110	
16384	0,0073405	0,0671860	0,03454120	
Complejidad	***O(n)***	***O(n)***	***O(n)***	***O(1.6^n)***

Fib1: Tomamos $n_1 = 8192$ y $n_2 = 16384$, con $t_1 = 0{,}0036851$, para demostrar que la complejidad del algoritmo es O(n) aplicamos la siguiente formula:

$$t_2 = \frac{f(n_2)}{f(n_1)} * t_1 = \frac{n_2}{n_1} * t_1 = \frac{16384}{8192} * 0{,}0036851 => \boldsymbol{t_2 = 0{,}0073702\ s}$$

Como el tiempo resultante es aproximado al que se ha obtenido en la gráfica queda demostrado que la complejidad del algoritmo es O(n).

Fib2: Tomamos $n_1 = 8192$ y $n_2 = 16384$, con $t_1 = 0{,}04399$, para demostrar que la complejidad del algoritmo es O(n) aplicamos la siguiente formula:

$$t_2 = \frac{f(n_2)}{f(n_1)} * t_1 = \frac{n_2}{n_1} * t_1 = \frac{16384}{8192} * 0{,}04399 => \boldsymbol{t_2 = 0{,}08798\ s}$$

Como el tiempo resultante es aproximado al que se ha obtenido en la gráfica queda demostrado que la complejidad del algoritmo es O(n).

Fib3: Tomamos $n_1 = 8192$ y $n_2 = 16384$, con $t_1 = 0{,}01653110$, para demostrar que la complejidad del algoritmo es O(n) aplicamos la siguiente formula:

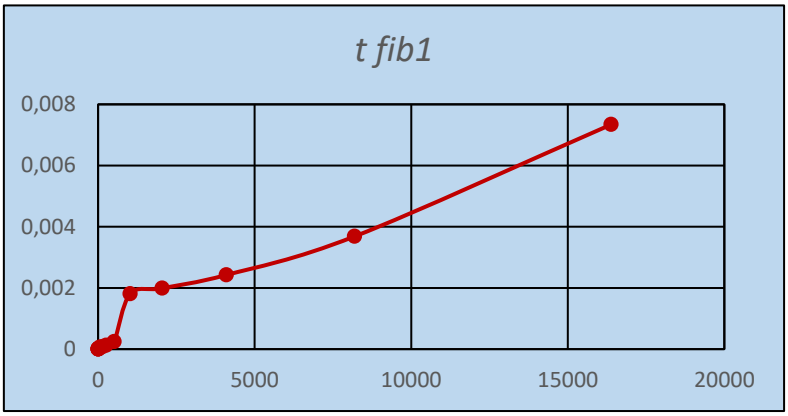
$$t_2 = \frac{f(n_2)}{f(n_1)} * t_1 = \frac{n_2}{n_1} * t_1 = \frac{16384}{8192} * 0{,}01653110 => \boldsymbol{t_2 = 0{,}0330622\ s}$$

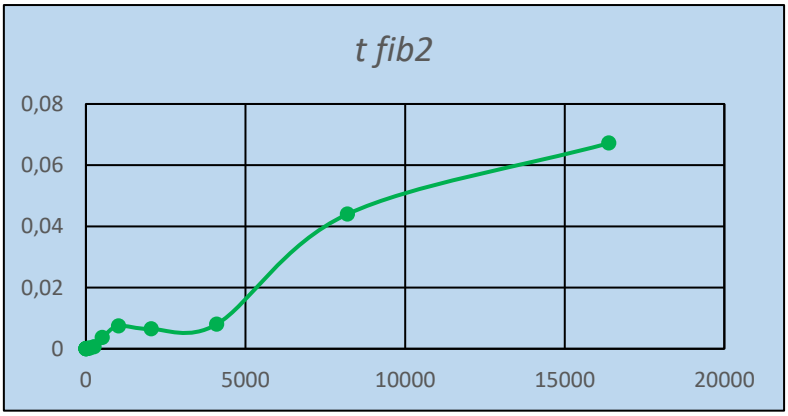
Como el tiempo resultante es aproximado al que se ha obtenido en la gráfica queda demostrado que la complejidad del algoritmo es O(n).

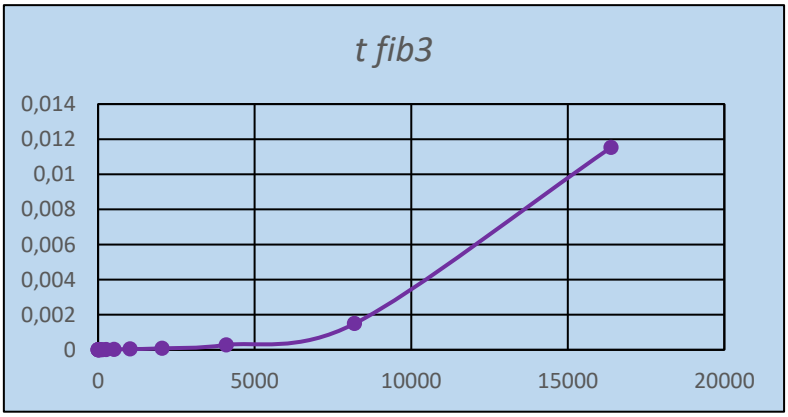
Fib4: Tomamos $n_1 = 16$ y $n_2 = 32$, con $t_1 = 0{,}0028747$, para demostrar que la complejidad del algoritmo es $O(1{,}6^n)$ aplicamos la siguiente formula:

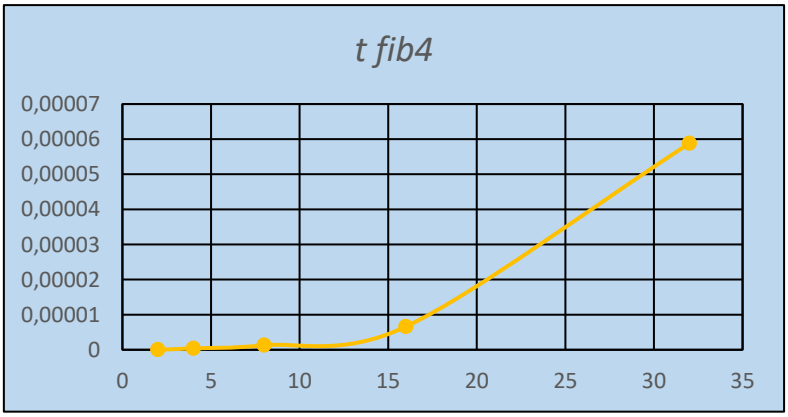
$$t_2 = \frac{f(n_2)}{f(n_1)} * t_1 = \frac{1{,}6^{n_2}}{1{,}6^{n_1}} * t_1 = \frac{1{,}6^{32}}{1{,}6^{16}} * 0{,}0028747 => \boldsymbol{t_2 = 5{,}303\ s}$$

Como el tiempo resultante es aproximado al que se ha obtenido en la gráfica queda demostrado que la complejidad del algoritmo es $O(1.6^n)$.


t fib1
0,008
0,006
0,004
0,002
0
0
5000
10000
15000
20000


t fib2
0,08
0,06
0,04
0,02
0
0
5000
10000
15000
20000

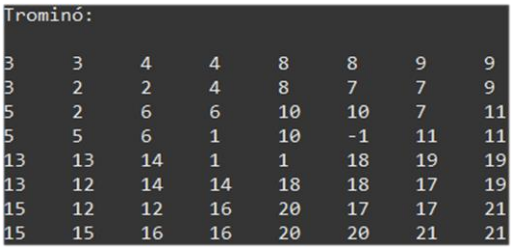

t fib3
0,014
0,012
0,01
0,008
0,006
0,004
0,002
0
0
5000
10000
15000
20000


t fib4
0,00007
0,00006
0,00005
0,00004
0,00003
0,00002
0,00001
0
0
5
10
15
20
25
30
35

Actividad 2:

Tiempos del Trominó:

N	*t Trominó*
16	0,00090
32	0,00251
64	0,00826
128	0,02872
256	0,11047
512	0,43949
1024	1,78248
2048	8,09016
4096	45,6000
8192	205,500
16384	673,200
32768	2784,50
Complejidad	***O(n^2)***



La complejidad del algoritmo de *Trominó* es O(n^2), tal y como se había pedido, para ello, se ha realizado mediante la técnica de *Divide y Vencerás*, creando un número de 4 subproblemas (a=4) y reduciendo en cada uno de estos el problema a la mitad (b=2), de esta forma, y como la complejidad base del algoritmo es O(1) (k=0), extraemos que la complejidad es: O($n^{\log_2 4}$) => O(n^2).

Tomamos $n_1 = 16384$ y $n_2 = 32768$, con $t_1 = 673{,}2$, para demostrar que la complejidad del algoritmo es O(n^2) aplicamos la siguiente formula:

$$t_2 = \frac{f(n_2)}{f(n_1)} * t_1 = \frac{{n_2}^2}{{n_1}^2} * t_1 = \frac{32768^2}{16384^2} * 673{,}2 => \boldsymbol{t_2 = 2692{,}8\ s}$$

Como el tiempo resultante es aproximado al que se ha obtenido en la gráfica queda demostrado que la complejidad del algoritmo es O(n^2).