Algorithmics	Student information	Date	Number of session
	UO:300809	12/03/2025	3
	Surname: González Bajo		
	Name: Javier		

Escuela de Ingeniería Informática Universidad de Oviedo

Universidad de Oviedo
Universidá d'Uviéu
University of Oviedo

Activity 1. Divide and Conquer by subtraction

n	Subtraction 1	Subtraction 2
1	0,000035	0,0000545
2	0,000051	0,000101
4	0,0000795	0,0002295
8	0,0001395	0,000643
16	0,0002975	0,001853
32	0,0005605	0,005427
64	0,001094	0,0172315
128	0,0021445	
256	0,0042585	
512	0,008445	
1024	0,0166435	

- Subtraction1: O(n)

  When n is multiplied by 2, the time is also multiplied by 2.

- Subtraction2: O(n^2)
  Theoretically when n is multiplied by 2, the time is multiplied by 4. But the times obtained in Subtraction2 are multiplied a little less than 4.

Algorithmics	Student information	Date	Number of session
	UO:300809	12/03/2025	3
	Surname: González Bajo		
	Name: Javier		

Escuela de Ingeniería Informática Universidad de Oviedo

Universidad de Oviedo
Universidá d'Uviéu
University of Oviedo

n	Subtraction3
1	0,000053
2	0,000118
3	0,000258
4	0,000542
5	0,001129
6	0,00226
7	0,004545
8	0,009073
9	0,018419
10	0,036701

- Subtraction3: $O(2^n)$
  Each time we add an n, the time is multiplied by 2 (2^1).
  For Subtraction3 to complete the execution for n = 80:

  n = 1 => t1 = 0,000053

  n = 80 => t80 = t1 * 2^(80-1) = 3,2*10^19 ms

$$\frac{3{,}2 * 10^{19}}{36 * 10^5 * 24 * 365{,}5} = 1013325227\ years$$

n	Subtraction4
100	3
200	23
400	163
800	1230
1600	9837
3200	OoT

- Subtraction4 O(n^3):
  When n is multiplied by 2, the time is multiplied by 8 (2^3).

n	Subtraction5
30	394
32	1162
34	3537
36	10705
38	10705
40	32550

Algorithmics	Student information	Date	Number of session
	UO:300809	12/03/2025	3
	Surname: González Bajo		
	Name: Javier		

- Subtraction5 O(3^(n/2)):
  Each time we add two n, the time is multiplied by 3 (3^(2/2)).

n = 38 => t38 = 32550

n = 80 => t80 = t38 * 3^((80-38)/2) = 3,4*10^14 ms

$$\frac{3{,}4 * 10\text{^}14}{36 * 10^5 * 24 * 365{,}5} = 10782\ years$$

Activity 2. Divide and Conquer by division

n	Division1	Divison2
1	0,0000375	0,000058
2	0,000045	0,000141
4	0,0000765	0,0003295
8	0,0001085	0,0007445
16	0,0001925	0,0015805
32	0,00029	0,0033175
64	0,0004615	0,006885
128	0,0008815	0,014204
256	0,001395	0,0293625
512	0,002621	
1024	0,00496	
2048	0,0097	

- Division1 O(n):
  At first, it doesn't seem to be have a O(n) complexity but when n is bigger, each time we multiply n by 2, the times are also multiplied by 2.

- Division2 O(nlogn)
  When we multiply n by 2, the times are multiplied by a number a little bigger than 2 (because of the logarithm in the complexity).

n	Divison3
1	0,0000545
2	0,000126
4	0,0002745
8	0,0005795
16	0,001183

Algorithmics	Student information	Date	Number of session
	UO:300809	12/03/2025	3
	Surname: González Bajo		
	Name: Javier		

32	0,0024
64	0,004804
128	0,0097165
256	0,0194115

- Division3
  When n is multiplied by 2, the time is also multiplied by 2.

n	Divison4	Divison5
1000	5	30
2000	18	116
4000	66	453
8000	257	1843
16000	1021	7239
32000	4113	28807
64000	16463	

- Division4 O(n^2)
  When n is multiplied by 2, the time is multiplied by 4.

- Division5
  When n is multiplied by 2, the time is multiplied by 4.

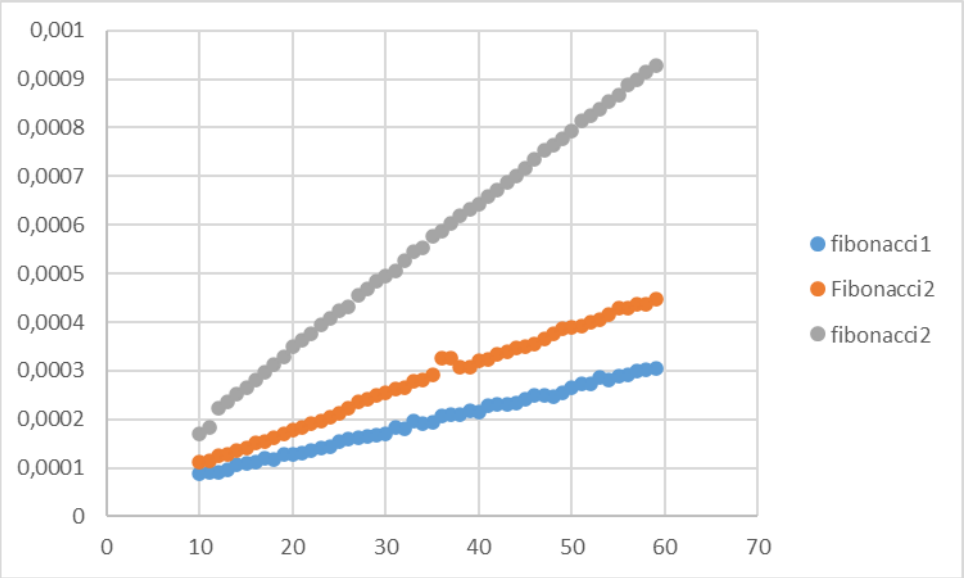
Activity 3. Two basics examples

n	Sum1	Sum2	Sum3
3	0,0000435	0,0000675	0,000089
6	0,000064	0,000113	0,0001785
12	0,000087	0,0002315	0,0003775
24	0,000131	0,00042	0,000759
48	0,000221	0,0008095	0,001548
96	0,0003965	0,001576	0,003048
192	0,000753	0,0031175	0,0061105
384	0,0014635	0,0061525	0,012243
768	0,00288	0,012232	0,0246975
1536	0,005783	0,024331	
3072	0,0115245		
6144	0,0226385		

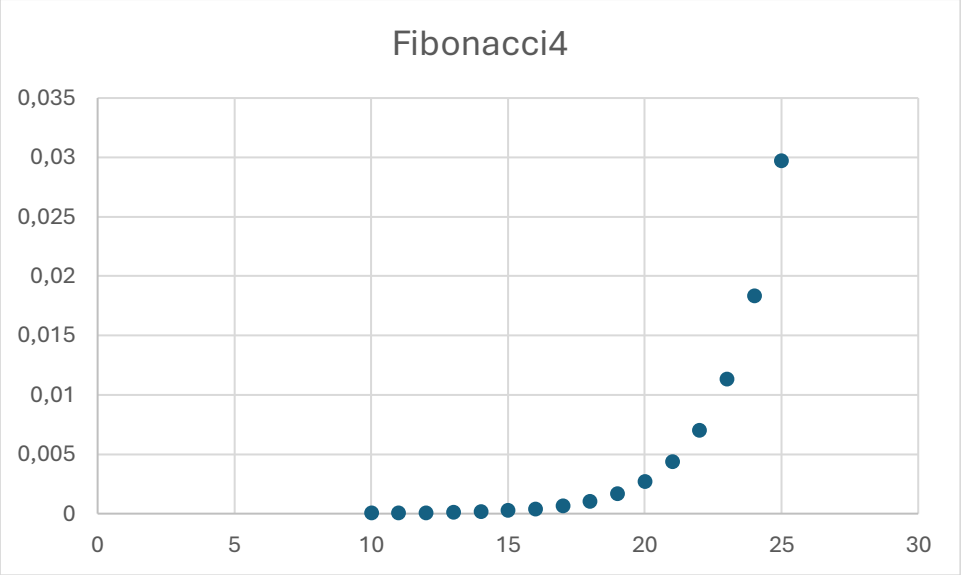
- Sum1O(n)
- Sum2 O(n)

Algorithmics	Student information	Date	Number of session
	UO:300809	12/03/2025	3
	Surname: González Bajo		
	Name: Javier		

- Sum3 O(n)
  Although the complexity is the same, the most efficient algorithm is sum1 because it is the one with the fastest times. The second one would be sum2 and then Sum3.



- Fibonacci1 O(n)
- Fibonacci2 O(n)
- Fibonacci3 O(n)



- Fibonacci4 O(1.6^n)

Algorithmics	Student information	Date	Number of session
	UO:300809	12/03/2025	3
	Surname: González Bajo		
	Name: Javier		

Fibonacci is the slowest because it has an exponential complexity. From the linear algorithms, the most efficient is Fibonacci 1 as it is the one with fastest times. The second most efficient is Fibonacci2 and then Fibonacci3.

Activity 4. Petanque championship organization

The algorithm complexity is O(n^2) with a = 2, b = 2, k =2.

n	Calendar
2	0
4	0
8	0,003
16	0,014
32	0,054
64	0,199
128	0,761
256	2,948
512	11,339
1024	44,739
2048	178,762

When n is multiplied by 2, the time is multiplied by 4 (2^2).