Algorithmics	Student information	Date	Number of session
	UO:300717	27/02/2025	3
	Surname: Almoina Iglesias		
	Name: Martín		

Escuela de Ingeniería Informática
Universidad de Oviedo

Universidad de Oviedo
Universidá d'Uviéu
University of Oviedo

Activity 1. Divide and Conquer by subtraction

Substraction1 has linear complexity meanwhile Substraction2 has quadratic complexity, while its hard to tell since the algorithms cause a stack overflow before we can get many reliable time measurements they appear to follow their theoretical complexities, every time n doubles Substraction2's time is more or less multiplied by 4 (2^2) as is expected of it's complexity.

- For what value of n do the Subtraction1 and Subtraction2 classes stop giving times (we abort the algorithm because it exceeds 1 minute)? Why does that happen?

There is a stack overflow when n is greater than 8192 for both Subtraction1 and Subtraction2.

- How many years would it take to complete the Subtraction3 execution for n=80? Reason the answer

The substraction3 algorithm has exponential theoretical complexity (2^n).

n1 = 20 t1 = 2ms | n2 = 80 t2 = ?

t2 = f(n2)/f(n1) * t1 = (2^80/2^20) * 2 = 2.305.843.009.213.693.952 ms =

73,117,802.2 years

Escuela de Ingeniería Informática

Algorithmics	Student information	Date	Number of session
	UO:300717	27/02/2025	3
	Surname: Almoina Iglesias		
	Name: Martín		

Substraction 4	
n	t (ms)
100	1
200	10
400	83
800	627
1600	4891
3200	39483
6400	311753

Substraction 5	
n	t (ms)
30	370
32	1089
34	3359
36	10219
38	30841
40	91114

How many years would it take to complete the Subtraction5 execution for n=80? Reason the answer.

Substraction5 has complexity O ($3^{n/2}$)

n1 = 30 t1 = 370ms | n2 = 80 t2 = ?

t2 = f(n2)/f(n1) * t1 = (3^40/3^15) * 370 = 313.496.785.493.910ms = 9,940.91786 years

Escuela de Ingeniería Informática

Algorithmics	Student information	Date	Number of session
	UO:300717	27/02/2025	3
	Surname: Almoina Iglesias		
	Name: Martín		

Activity 2. Divide and conquer by division

Division1 has linear complexity and Division2 has O(nlogn) complexity and Division3 also has linear complexity but obtained differently than in Division1 , the times obtained seem to line up with this, Division1's times double (more or less) every time n doubles, as do Division3's, meanwhile Division2's times more than double.

Division 4	
n	t(ms)
1000	5
2000	17
4000	68
8000	256
16000	1021
32000	4106
64000	16532
128000	65703

Division 5	
n	t(ms)
1000	24
2000	93
4000	375
8000	1477
16000	5942
32000	24422
64000	96196

Algorithmics	Student information	Date	Number of session
	UO:300717	27/02/2025	3
	Surname: Almoina Iglesias		
	Name: Martín		

Activity 3. Two basic examples

VectorSum	Iteration	Recursive Sub	Recursive Div
n	t1 (ms)	t2(ms)	t3(ms)
3	0,000042	0,000075	0,000088
6	0,000067	0,000125	0,000168
12	0,000093	0,00025	0,000356
24	0,000143	0,0004575	0,000752
48	0,000237	0,0008625	0,001536
96	0,00045	0,0016675	0,003098
192	0,000811	0,003295	0,006148
384	0,001576	0,00667	0,01213
768	0,003201	0,0136175	0,02467
1536	0,006373	0,026975	0,049574
3072	0,012738	0,053445	0,099126
6144	0,025056	0,105075	0,199182

All three methods have linear complexity however they are implemented in different ways The first method is implemented using a for loop, the second one using recursive subtraction, and the third one uses recursive division. From the times we can see that the best method is the iterative one.

Algorithmics	Student information	Date	Number of session
	UO:300717	27/02/2025	3
	Surname: Almoina Iglesias		
	Name: Martín		

Fibonacci				
n	t1(ms)	t2(ms)	t3(ms)	t4(ms)
10	0,000091	0,000113	0,000194	0,00255
15	0,000109	0,000147	0,000286	0,0276
20	0,000122	0,000176	0,000366	0,3036
25	0,000146	0,000212	0,000446	3,34
30	0,000166	0,000246	0,000526	37,51
35	0,000205	0,00028	0,0006	416,15
40	0,000228	0,000313	0,000684	4611
45	0,000238	0,000352	0,000762	51036
50	0,000267	0,000382	0,000846	Oot
55	0,000276	0,000421	0,000898	Oot
59	0,000297	0,000468	0,000938	Oot

Here we have four methods calculating the value number n of the Fibonacci sequence from 10 to 60,non-inclusive due to the second method causing a stack overflow when n=60, the first method uses a loop of linear complexity, the second one uses a vector and dynamic programming and also has linear complexity, and the third uses recursive calls with subtraction to also achieve linear complexity, finally the fourth method has an exponential complexity $O(1.6^n)$ this is because this method computes the previous two numbers in the sequence and then adds them making n's execution time the sum of the execution times of n-1 and n-2.

Algorithmics	Student information	Date	Number of session
	UO:300717	27/02/2025	3
	Surname: Almoina Iglesias		
	Name: Martín		

Activity 4. Calendar

The method used to assign the matches between the players has a complexity O(n^3) due to the use of two nested loops which contain a call to a method with O(n) complexity

Calendar				
n	t1	t2	t3	t4
2	0,0047	0,0022	0,00225	0,00265
4	0,0253	0,00535	0,00535	0,00645
8	0,04445	0,0208	0,01905	0,01845
16	0,0774	0,07955	0,0902	0,07715
32	0,2825	0,3038	0,3646	0,33395
64	1,29	1,60085	1,5713	1,7768
128	7,55	5,4	4,8	4,9
256	20,75	20,97	20,27	21,3
512	136,11	86,67	95,14	129,23
1024	372,71	423,33	384,14	335,01
2048	1602,08	1476,39	1576,67	1551,72

Although the time measurements aren't too reliable, possibly due to the hardware that the program was executed in, they do seem to line up with the theoretical complexity as they seem to grow too quickly for a lower complexity, however it is not an exact cubic growth possibly due to the method implementation and the unreliability of the measurements.