Algorithmics	Student information	Date	Number of session
	UO: 297383	03/03	3
	Surname: Herrero Sánchez		
	Name: Iván		

Escuela de Ingeniería Informática
Universidad de Oviedo

Universidad de Oviedo
Universidá d'Uviéu
University of Oviedo

Activity 1. [Subtraction]

For n = 8192 Subtraction1 and Subtranction2 stop giving times due to a Stack Overflow. We have a huge use of the stack memory using Subtraction with a=1.

Subtraction3 has a complexity of O (2^n) it would take 3.28 *10^4 years, the times grows very fast with this complexity.

TABLE 4

n	times
100	138*10^-2
200	103*10^-1
400	81
800	623
1600	4966
3200	38833
6400	OoT

TABLE 5

n	times
30	394
32	1172
34	3603
36	10578
38	31621
40	94612
42	OoT

For n = 80, the algorithm takes 2.105*10^13 milliseconds, so 16 019 years.

Algorithmics	Student information	Date	Number of session
	UO: 297383	03/03	3
	Surname: Herrero Sánchez		
	Name: Iván		

Activity 2. [Division]

Divisio1 a complexity O(n) but the trend line looks more exponential than linear, so the times don't match the theoretical time complexity. Something similar happens to Division2 and Division3, they have complexity O(n*log n) and O(log n), but the trend line of both times looks more exponential than logarithmic.

TABLE DIVISION 4

n	times
1000	48*10^-2
2000	196*10^-1
4000	79
8000	304
16000	1227
32000	4845
64000	19302
128000	76934

TABLE DIVISION 5

n	times
1000	250*10^-1
2000	103
4000	402
8000	1630
16000	6369
32000	25338
64000	OoT
128000	OoT

Algorithmics	Student information	Date	Number of session
	UO: 297383	03/03	3
	Surname: Herrero Sánchez		
	Name: Iván		

Activity 2. [Vector Sum and Fibonacci]

VECTOR SUM

n	Iteration O(n)	Substitution O(n)	Division O(n)
3	46*10^-6	71*10^-6	95*10^-6
6	66*10^-6	116*10^-6	189*10^-6
12	92*10^-6	232*10^-6	372*10^-6
24	132*10^-6	419*10^-6	771*10^-6
48	219*10^-6	803*10^-6	156*10^-5
96	396*10^-6	158*10^-5	306*10^-5
192	746*10^-6	308*10^-5	619*10^-5
384	1447*10^-6	606*10^-5	124*10^-4
768	285*10^-5	1214*10^-5	250*10^-4
1536	566*10^-5	242*10^-4	499*10^-4
3072	566*10^-5	489*10^-4	992*10^-4
6144	227*10^-4	960*10^-4	1975*10^-4

The first algorithm is faster is it just call one single method, when we have more calls to method the time tends to increase even if the complexity is the same. Like for the 2nd approach we have a recursive call and in the 3rd we have 2 recursive calls to the same method.

FIBONACCI

n	Iteration O(n)	Iteration with vector O(n)	Recursive O(n)	Recursive O(1.6^n)
10	89*10^-6	119*10^-6	176*10^-6	229*10^-5
11	94*10^-6	125*10^-6	187*10^-6	373*10^-5
12	96*10^-6	133*10^-6	223*10^-6	598*10^-5
13	101*10^-6	136*10^-6	237*10^-6	962*10^-5
14	106*10^-6	148*10^-6	258*10^-6	156*10^-4

15	111*10^-6	152*10^-6	269*10^-6	253*10^-4
16	114*10^-6	162*10^-6	288*10^-6	410*10^-4
17	119*10^-6	166*10^-6	300*10^-6	661*10^-4
18	121*10^-6	174*10^-6	329*10^-6	108*10^-3
19	131*10^-6	181*10^-6	336*10^-6	173*10^-3
20	139*10^-6	191*10^-6	359*10^-6	276*10^-3
21	144*10^-6	198*10^-6	373*10^-6	448*10^-3

The two iterative methods don't waste time accessing any method, but the vector one waste time accessing the indexes of the vector.

Activity 3. [Calendar]

n	t Calendar
2	
4	
8	
16	
32	
64	
128	