


Algorithmics	Student information	Date	Number of session
	UO: 296503	17/02/2025	1.2
	Surname: Mulet	 Escuela de Ingeniería Informática Universidad de Oviedo	
	Name: Alonso		



Activity 1. Some iterative models

With a number of 10000 repetitions I got the following results:

As we are using this number of repetitions time units are tends of microseconds.

N	tLoop1	tLoop2	tLoop3	tLoop4
100	50	1649	8337	6840
200	96	6265	36415	50121
400	206	29324	OoT	OoT
800	485	OoT	OoT	OoT
1600	991	OoT	OoT	OoT
3200	2177	OoT	OoT	OoT
6400	4684	OoT	OoT	OoT
12800	10632	OoT	OoT	OoT
25600	22472	OoT	OoT	OoT
51200	47339	OoT	OoT	OoT

For a better measurement I will use less number of repetitions for Loop2, Loop3 and Loop4.

Loop2: 1000 repetitions, microseconds.

N	tLoop2
100	185
200	659
400	3092
800	14437
1600	55229
3200	OoT

For Loop3 and Loop4 I will use 100 repetitions, hundreds of milliseconds

Algorithmics	Student information	Date	Number of session
	UO: 296503	17/02/2025	1.2
	Surname: Mulet		
	Name: Alonso		

N	tLoop3	tLoop4
100	89	71
200	417	498
400	1730	3683
800	7152	29091
1600	28431	OoT
3200	OoT	OoT

Now let's take everything together, in milliseconds:

N	tLoop1	tLoop2	tLoop3	tLoop4
100	0.0050	0.185	0.89	0.71
200	0.0096	0.659	4.17	4.98
400	0.0206	3.092	17.30	36.83
800	0.0485	14.437	71.52	290.91
1600	0.0991	55.229	284.31	OoT
3200	0.2177	OoT	OoT	OoT
6400	0.4684	OoT	OoT	OoT
12800	1.0632	OoT	OoT	OoT
25600	2.2472	OoT	OoT	OoT
51200	4.7339	OoT	OoT	OoT

If we look at the code, we can see that Loop1 has a complexity of $n \log n^2$, which makes sense with the measured times.

The complexity of Loop2 is $O(n^2 \log_3 n)$ while Loop3 is $O(n^2 \log_2 n)$, both can be reduced to $O(n^2 \log n)$. Because of the complexity in detail, Loop3 get higher times than Loop2.

Finally, the complexity of Loop4 is $O(n^3)$ which is clearly shown in the times, for instance when $n = 400$ we get a time of 36830 and with $n = 800$ we get 290910, the times got almost 2^3 times higher (times 8).

Algorithmics	Student information	Date	Number of session
	UO: 296503	17/02/2025	1.2
	Surname: Mulet		
	Name: Alonso		

Activity 2. Creation of iterative models of a given time complexity

I am using 10 as number of repetitions so milliseconds to the power of 10^{-1}

N	tLoop5	tLoop6	tLoop7
100	50	102	4184
200	208	830	OoT
400	1007	7316	OoT
800	4782	OoT	OoT
1600	22488	OoT	OoT
3200	OoT	OoT	OoT
6400	OoT	OoT	OoT

As Loop7 is OoT by $n = 200$ I will measure it with just one repetition:

N	tLoop7
100	411
200	6638
400	OoT
800	OoT
1600	OoT
3200	OoT
6400	OoT

As the complexity of this method is n^4 , the times get too big too soon, but we can see that they follow the complexity, $6638/411$ is almost 16, which is 2^4 .

Putting it all together we have, in milliseconds:

N	tLoop5	tLoop6	tLoop7
---	--------	--------	--------

Algorithmics	Student information	Date	Number of session
	UO: 296503	17/02/2025	1.2
	Surname: Mulet		
	Name: Alonso		

100	5.0	10.2	411
200	20.8	83.0	6638
400	100.7	731.6	OoT
800	478.2	OoT	OoT
1600	2248.8	OoT	OoT
3200	OoT	OoT	OoT
6400	OoT	OoT	OoT

Activity 3. Two algorithms with different complexity

N	tLoop1	tLoop2	t1/t2
100	0.0050	0.185	0,027027
200	0.0096	0.659	0,0145675
400	0.0206	3.092	0,0066624
800	0.0485	14.437	0,0033594
1600	0.0991	55.229	0,0017943
3200	0.2177	OoT	

The result of dividing t1 by t2 is lower than 1, then we know that tLoop1 is better, this makes sense since it has a better complexity also.

As we can see the quotient gets divided by more or less two in each problem size increase, this happens because while t1 has a complexity of $O(n \log n^2)$, Loop2 has $O(n^2 \log n)$.

Activity 4. Two algorithms with the same complexity

Algorithmics	Student information	Date	Number of session
	UO: 296503	17/02/2025	1.2
	Surname: Mulet		
	Name: Alonso		

tLoop2	tLoop3	t2/t3
0,185	0,89	0,20786517
0,659	4,17	0,15803357
3,092	17,30	0,17872832
14,437	71,52	0,20185962
55,229	284,31	0,19425627
OoT	OoT	

As we can see by the times, there is a difference between these two algorithms despite having the same complexity at first glance. We can see how Loop2 is more or less 20% faster than Loop3. There is an explanation for this, if we get the exact complexity of both algorithms, Loop2 has $O(n^2 \log_3 n)$ while Loop3 $O(n^2 \log_2 n)$, so times grow a little bit less in Loop2.

Activity 5. Same algorithm in different development environments

Time in milliseconds

n	tLoop4.py	tLoop4.java no optimization	tLoop4.java optimization (rep = 1000)	t42/t41	t43/t42
100	3,39	0,77	0,090	0,22713864	0,1168831
200	24,87	5,31	0,537	0,21351025	0,1011299
400	200,40	41,37	3,526	0,20643713	0,0852308
800	OoT	326,32	24,807	No value	0,0760205
1600	OoT	OoT	179,37	No value	No value
3200	OoT	OoT	OoT	No value	No value

Algorithmics	Student information	Date	Number of session
	UO: 296503	17/02/2025	1.2
	Surname: Mulet		
	Name: Alonso		

These times have been taken with 100 as number of repetitions, for Loop4 I used 1000 reps to get more accurate times, all are converted to milliseconds in the table.

We can see how Java is quite better than python for this problem and how java with optimization is much better than without it.