Algorithmics	Student information	Date	Number of session
	UO: 300827	17/02/2025	1.2
	Surname: Leiras		
	Name: Sofía		

Escuela de Ingeniería Informática
Universidad de Oviedo

Universidad de Oviedo
Universidá d'Uviéu
University of Oviedo

Activity 1. Some iterative models

Table of times (in milliseconds) of execution of the program without optimization:

N	tLoop1	tLoop2	tLoop3	tLoop4
100	0.01	0.234	0.84	1.09
200	0.0209	0.905	3.64	10.58
400	0.044	4.275	15.41	60.63
800	0.1073	19.688	67.58	518.32
1600	0.2278	79	314.29	3682.8
3200	0.4683	362	1355.3	29284
6400	1.0052	1411	6081.6	OoT
12800	2.350	6355	24668	OoT
25600	4.870	28181	OoT	OoT
51200	10.325	OoT	OoT	OoT

Loop1: its theoretical complexity is O(n log n), so its expected growth is slightly more than linear. As the execution time in the table follows a slow increase, it is consistent with the expected complexity.

Loop2: its theoretical complexity is $O(n^2 \log n)$, so its expected growth is worse than loop1, slightly more than quadratic. The table shows a significant increase in time, confirming the expected complexity.

Loop3: its theoretical complexity is $O(n^2 \log n)$, similar to loop2. The table shows very large values, confirming the expected complexity. Despite having the same complexity as loop2, the times are bigger because the complexity has been simplified, ignoring some constants that may be bigger in loop3 and make its execution times larger. However, this does not prevent us from seeing that the growth is the expected one.

Algorithmics	Student information	Date	Number of session
	UO: 300827	17/02/2025	1.2
	Surname: Leiras		
	Name: Sofía		

Loop4: its theoretical complexity is $O(n^3)$, so its expected growth is rapid, leading to Out of Time (OoT) for large n. The table confirms this, as loop4 quickly becomes infeasible for large n.

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

Table of times (in milliseconds) of execution of the program without optimization:

N	tLoop5	tLoop6	tLoop7
100	7	93	1246
200	36.7	856	46358
400	179.4	7663	OoT
800	856.4	OoT	OoT
1600	4138	OoT	OoT
3200	19301	OoT	OoT
6400	OoT	OoT	OoT

Loop5: its theoretical complexity is $O(n^2 \log^2 n)$, so its expected growth is faster than $O(n^2)$, but much slower than cubic. The execution times in the table increase significantly but do not explode as fast as cubic. The function runs Out of Time (OoT) around n = 6400, indicating a high but not exponential complexity.

Loop6: its theoretical complexity is $O(n^3 \log n)$, so its expected growth is faster than $O(n^3)$. The function quickly becomes OoT for $n \geq 800$, which is expected for its complexity. The rapid increase in execution time is consistent with this complexity.

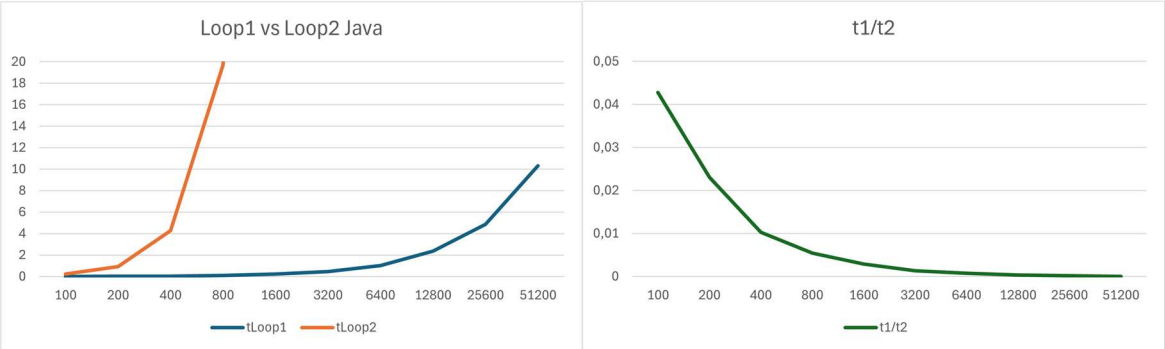
Loop7: its theoretical complexity is $O(n^4)$, so its expected growth is extremely fast, leading to OoT at small values of n. In the table, the function reaches OoT extremely quickly (n = 400), confirming its complexity. This is the worst of the three loops in terms of performance.

Algorithmics	Student information	Date	Number of session
	UO: 300827	17/02/2025	1.2
	Surname: Leiras		
	Name: Sofía		

Activity 3. Comparison of two algorithms with different complexity

Table of times (in milliseconds) of execution of the program without optimization:

N	tLoop1	tLoop2	t1/t2
100	$7 \cdot 10^{-3}$	0.234	0.042735
200	$1.52 \cdot 10^{-2}$	0.905	0.0230939
400	$3.38 \cdot 10^{-2}$	4.275	0.0102924
800	$7.73 \cdot 10^{-2}$	19.688	0.00545
1600	0.1577	79	0.0028835
3200	0.3334	362	0.0012936
6400	0.707	1411	0.0007124
12800	1.6076	6355	0.0003698
25600	3.531	28181	0.0001728
51200	7.343	OoT	--



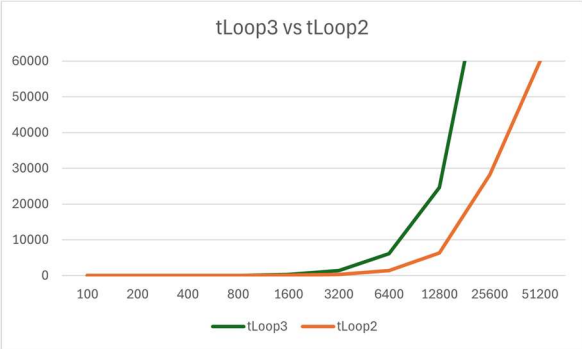
The complexity of loop1 is O(n log n), the complexity of loop2 is O(n^2 log n) and we are calculating the ratio t1/t2. In the table, we can see that the ratio decreases as n growths, which is consistent as the ratio should tend to 0, because the algorithm associated with the numerator (loop1) is less complex than the one in the denominator (loop2).

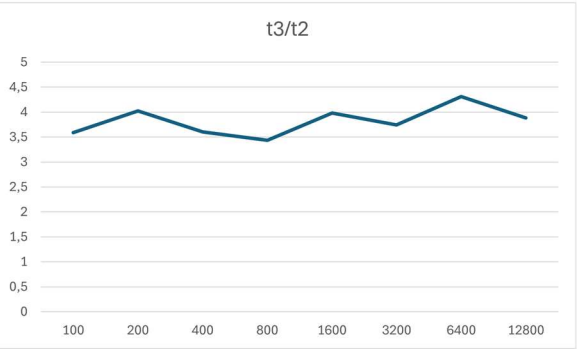
Algorithmics	Student information	Date	Number of session
	UO: 300827	17/02/2025	1.2
	Surname: Leiras		
	Name: Sofía		

Activity 4. Comparison of two algorithms with the same complexity

Table of times (in milliseconds) of execution of the program without optimization:

N	tLoop3	tLoop2	t3/t2
100	0.84	0.234	3.5897436
200	3.64	0.905	4.0220994
400	15.41	4.275	3.6046784
800	67.58	19.688	3.4325477
1600	314.29	79	3.9783544
3200	1355.3	362	3.7439227
6400	6081.6	1411	4.3101347
12800	24668	6355	3.881668
25600	OoT	28181	--
51200	OoT	OoT	--





The complexity of loop3 is $O(n^2 \log n)$, as well as the complexity of loop2, and we are calculating the ratio t3/t2. In the table, we can see that the ratio tends to a constant (between 3.5 and 4), which is consistent as the ratio should tend to a constant greater than 1, because the algorithm associated with the denominator (loop2) is slightly better than the one associated with the numerator (loop3).

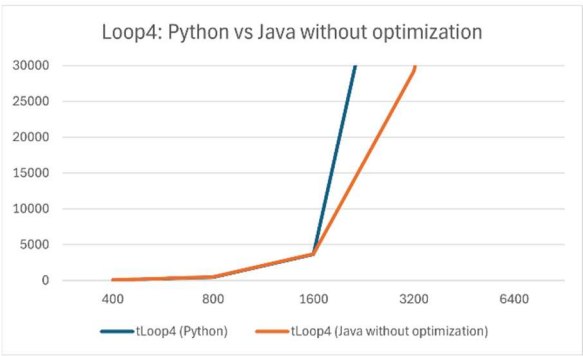
Algorithmics	Student information	Date	Number of session
	UO: 300827	17/02/2025	1.2
	Surname: Leiras		
	Name: Sofía		

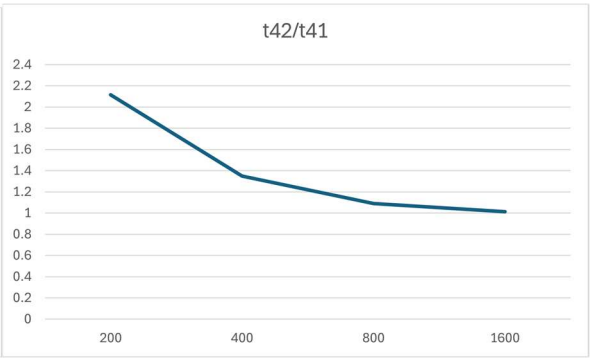
Activity 5. Comparison of the same algorithm in different development environments

Table of times (in milliseconds) of execution of the program:

n	tLoop4 (Python) t41	tLoop4 (Java without optimization) t42	tLoop4 (Java with optimization) t43	t42/t41	t43/t42
200	5	10.58	0.132	2.116	0.0124764
400	45	60.63	0.587	1.3473333	0.0096817
800	475	518.32	3.672	1.0912	0.0070844
1600	3632	3682.8	25.827	1.0139868	0.0070129
3200	OoT	29284	178.1	--	0.0060818
6400	OoT	OoT	12854.4	--	--

Comparison of loop4 in Python (t41) and loop4 in java without optimization (t42):

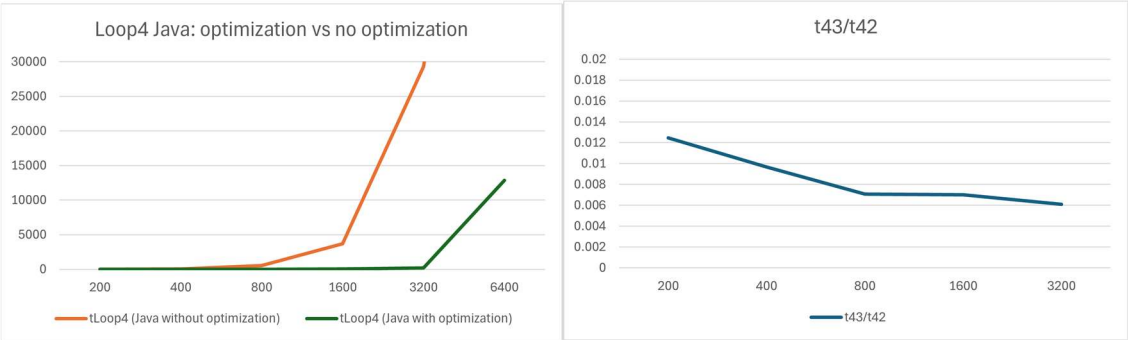




The complexity of the algorithm is $O(n^3)$, and we are calculating the ratio (t42/t41) between the algorithm being run in python and being run in java without optimization. In the table, we can see that the ratio tends to a constant (between 1 and 1.4, without taking into account the value at 200, which is far from the others), which is consistent as the ratio should tend to a constant greater than 1, because the algorithm associated with the denominator (t41) is better than the one associated with the numerator (t42).

Algorithmics	Student information	Date	Number of session
	UO: 300827	17/02/2025	1.2
	Surname: Leiras		
	Name: Sofía		

Comparison of loop4 in Java without optimization (t42) and loop4 in java with optimization (t43):



The complexity of the algorithm is $O(n^3)$, and we are calculating the ratio (t43/t42) between the algorithm being run in java with optimization and being run in java without optimization. In the table, we can see that the ratio tends to a constant (between 0.006 and 0.012which is consistent as the ratio should tend to a constant lower than 1, because the algorithm associated with the numerator (t43) is better than the one associated with the denominator (t42).