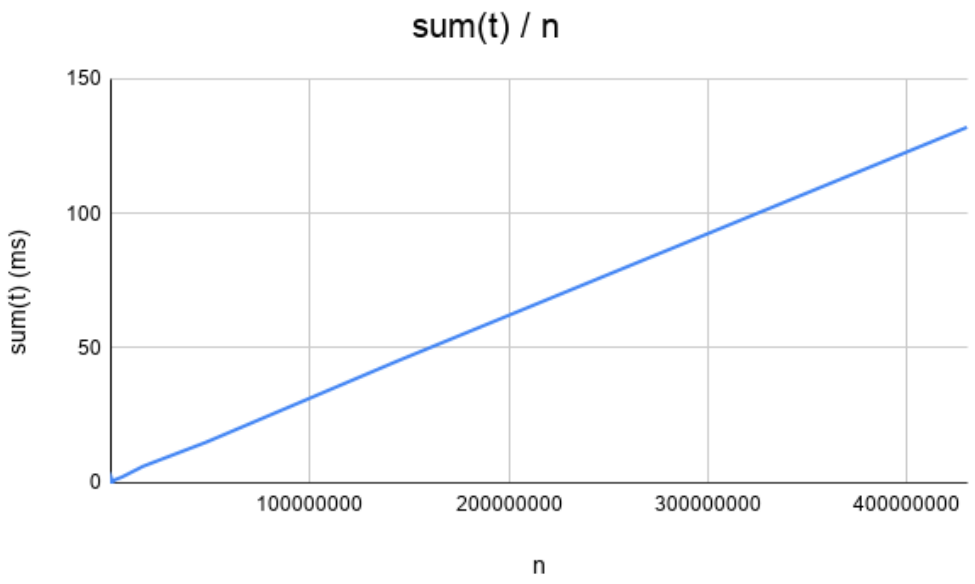
Algorithmics	Student information	Date	Number of session
	UO: 277921	9/2/2021	1.1
	Surname: García López		
	Name: Rosa		

Activity 1. Measuring execution times

1. The method will overflow at *new Date(Long.MAX_VALUE)*, which is 292278994. Thus, we can use it for than 292 million years more.
2. It means that the elapsed time is less than zero, and java rounds it to 0 when showing the measurement.
3. More or less from n = 110000000.

Activity 2. Grow of the problem size

1. The time is still the same, because the complexity has not changed.
2. Yes, they are.
3.



Algorithmics	Student information	Date	Number of session
	UO: 277921	9/2/2021	1.1
	Surname: García López		
	Name: Rosa		

Activity 3. Taking small execution times

n	*fillIn(t)*	*sum(t)*	*maximum(t)*
10	1	0	0
30	0	0	0
90	0	0	0
270	0	0	0
810	0	0	0
2430	0	0	0
7290	1	1	0
21870	1	0	0
65610	2	0	1
196830	3	1	1
590490	4	0	1
1771470	12	1	1
5314410	36	2	2
15943230	108	6	6
47829690	318	15	15
143489070	928	45	49
430467210	2802	132	244

The time is in milliseconds.

- What are the main components of the computer in which you did the work (process, memory)?
  The computer's memory and CPU.

- Do the values obtained meet the expectations?

  Yes, they do. The time complexity of the functions is O(n).

  ***fillIn(t)*:** t1 = 4 ms, n1 = 590490, n2 = 1771470, then, t2 = n2/n1 * t1 = 12 ms (the same as in the table).

  t1 = 12 ms, n1 = 1771470, n2 = 5314410, then, t2 = n2/n1 * t1 = 36 ms (the same as in the table).

***sum(t)*:** t1 = 6 ms, n1 = 15943230, n2 = 47829690, then, t2 = n2/n1 * t1 = 18 ms

t1 = 15 ms, n1 = 47829690, n2 = 143489070, then, t2 = n2/n1 * t1 = 45 ms (the same as in the table).

***maximum(t)*:** t1 = 2 ms, n1 = 5314410, n2 = 15943230, then, t2 = n2/n1 * t1 = 6 ms (the same as in the table).

t1 = 6 ms, n1 = 15943230, n2 = 47829690, then, t2 = n2/n1 * t1 = 18 ms

Activity 4. Operations on matrices

n	*sumDiagonal1(t)*	*sumDiagonal2(t)*
10	56	2
30	51	2
90	26	2
270	52	2
810	51	2
2430	50	2
7290	51	1
21870	51	1
65610	50	0
196830	51	1
590490	39	1
1771470	39	2
5314410	39	0
15943230	39	1
47829690	39	1
143489070	39	1
430467210	39	1

The time is in milliseconds.

Algorithmics	Student information	Date	Number of session
	UO: 277921	9/2/2021	1.1
	Surname: García López		
	Name: Rosa		

- What are the main components of the computer in which you did the work (process, memory)?
  The computer's memory and CPU.

- Do the values obtained meet the expectations?

  No, they do not.

  sumDiagonal1(t) $(\boldsymbol{O}(\boldsymbol{n}^2))$ **:** t1 = 52 ms, n1 = 270, n2 = 810, then,

  t2 = n2^2/n1^2 * t1 = 468 ms

  t1 = 51 ms, n1 = 810, n2 = 2430, then, t2 = n2^2/n1^2 * t1 = 459 ms

  sumDiagonal2(t) $(\boldsymbol{O}(\boldsymbol{n}))$ **:** t1 = 2 ms, n1 = 270, n2 = 810, then,

  t2 = n2/n1 * t1 = 6 ms

  t1 = 2 ms, n1 = 810, n2 = 2430, then, t2 = n2/n1 * t1 = 6 ms

Activity 5. Benchmarking

1. Because in Python it must first inspect the objects and find out their type, which is not known at compile time; in Java the type is declared beforehand so the compiler knows it.
2. Yes, in Java, the linear times at the beginning are greater with a smaller n than 3 iterations after; then the time increases normally.