


Algorithmics	Student information	Date	Number of session
	UO: 299751	05/02/2025	01
	Surname: Rodriguez Fernandez	 Escuela de Ingeniería Informática Universidad de Oviedo	
	Name: Marcos		



Activity 1. [Measuring execution times]

1. Given that the total number of milliseconds in a year is $3,154 \times 10^{10}$. And the greatest number that a long can represent is a 64-bit Integer, which is:

$$2^{63} - 1 = 9.223.372.036.854.775.807$$

We can divide the maximum long number by the milliseconds in a year and obtain 292.471.208,7 years. Since we have already used around 55 years, we subtract 55

to that number. This gives the result of, approximately 292.471.153 years until this method stops working.



Activity 2. [Measuring execution times]

1. The measured time is in some cases 0. This is because it is measured in milliseconds, and some of the executions are so fast that they last less than one millisecond. Thus, the represented value is zero milliseconds.
2. The problem starts to get reliable times after the size reaches 156250. Before reaching this value, the times are below 50 milliseconds and we will not use them for lack of reliability.

Algorithmics	Student information	Date	Number of session
	UO: 299751	05/02/2025	01
	Surname: Rodriguez Fernandez		
	Name: Marcos		

Activity 3. [Taking small execution times]

1. When n is multiplied by 2 in each iteration, we can observe that times almost double as n doubles, especially when values get higher. This shows that the algorithm is $O(n)$.
2. The exact same thing happens as we increase the value. If instead of 2, we try with 3 or 4, the times will be multiplied by 3 and 4, respectively, as can be seen in the following images:

K = 3

```
SIZE=10 TIME=0 milliseconds SUM=37 NTIMES=100
SIZE=30 TIME=0 milliseconds SUM=86 NTIMES=100
SIZE=90 TIME=0 milliseconds SUM=-931 NTIMES=100
SIZE=270 TIME=0 milliseconds SUM=-136 NTIMES=100
SIZE=810 TIME=1 milliseconds SUM=-205 NTIMES=100
SIZE=2430 TIME=2 milliseconds SUM=234 NTIMES=100
SIZE=7290 TIME=3 milliseconds SUM=-3556 NTIMES=100
SIZE=21870 TIME=12 milliseconds SUM=-5262 NTIMES=100
SIZE=65610 TIME=39 milliseconds SUM=7571 NTIMES=100
SIZE=196830 TIME=118 milliseconds SUM=-29787 NTIMES=100
SIZE=590490 TIME=350 milliseconds SUM=-31185 NTIMES=100
SIZE=1771470 TIME=1051 milliseconds SUM=53194 NTIMES=100
SIZE=5314410 TIME=3143 milliseconds SUM=27679 NTIMES=100
SIZE=15943230 TIME=9484 milliseconds SUM=-91477 NTIMES=100
SIZE=47829690 TIME=28410 milliseconds SUM=-270179 NTIMES=100
SIZE=143489070 TIME=85451 milliseconds SUM=-1166369 NTIMES=100
```

k = 4

```
SIZE=10 TIME=0 milliseconds SUM=271 NTIMES=100
SIZE=40 TIME=0 milliseconds SUM=258 NTIMES=100
SIZE=160 TIME=0 milliseconds SUM=-213 NTIMES=100
SIZE=640 TIME=0 milliseconds SUM=288 NTIMES=100
SIZE=2560 TIME=1 milliseconds SUM=1476 NTIMES=100
SIZE=10240 TIME=6 milliseconds SUM=6027 NTIMES=100
SIZE=40960 TIME=25 milliseconds SUM=-7470 NTIMES=100
SIZE=163840 TIME=97 milliseconds SUM=-33025 NTIMES=100
SIZE=655360 TIME=394 milliseconds SUM=-18543 NTIMES=100
SIZE=2621440 TIME=1551 milliseconds SUM=90081 NTIMES=100
SIZE=10485760 TIME=6213 milliseconds SUM=-56176 NTIMES=100
SIZE=41943040 TIME=24794 milliseconds SUM=-436422 NTIMES=100
SIZE=167772160 TIME=99361 milliseconds SUM=-651954 NTIMES=100
SIZE=671088640 TIME=402609 milliseconds SUM=-897163 NTIMES=100
```

3. Given this information, we can surely state that the algorithm's time complexity is $O(n)$. Since it grows as fast as n grows.

Algorithmics	Student information	Date	Number of session
	UO: 299751	05/02/2025	01
	Surname: Rodriguez Fernandez		
	Name: Marcos		

4. I filled this table using: **repetitions = 100; n*=2.**

n	Tsum	Tmaximum
10000	6	3
20000	12	1
40000	24	1
80000	49	3
160000	96	3
320000	193	11
640000	386	20
1280000	796	50
2560000	1539	97
5120000	3158	196
10240000	6211	401
20480000	12403	760
40960000	24726	1163
81920000	50273	2639

I filled this table using: **repetitions = 10; n*=2.**

n	Tmatches1	Tmathes2
10000	7615	0
20000	30410	0
40000	OoT	1
80000	OoT	1
160000	OoT	1
320000	OoT	1
640000	OoT	3
1280000	OoT	5
2560000	OoT	11
5120000	OoT	24
10240000	OoT	49
20480000	OoT	111
40960000	OoT	173
81920000	OoT	351

Processor: 1.80 GHz

Memory: 16.0 GB

Algorithmics	Student information	Date	Number of session
	UO: 299751	05/02/2025	01
	Surname: Rodriguez Fernandez		
	Name: Marcos		

CONCLUSION

In the first comparison, we see that both algorithms have a complexity of $O(n)$. However, due to the nature of the performed operations, the *maximum* method is faster than the *sum* method. In the tables we can observe that times grow as fast as n grows. Thus, the times obtained meet what was expected, given the computational time complexity of the different operations.

In the second case, both algorithms do not have the same complexity. *Matches1* has an $O(n^2)$ complexity and *matches2* an $O(n)$ complexity. This is why, even though they perform the same operations, the difference in speed and obviously in execution time is huge. This reflects the importance on complexity in algorithms, since the execution of an implementation of a method may last minutes while another only milliseconds.