Algorithmics	Student information	Date	Number of session
	UO:300599	28/01/2025	1
	Surname: Alvarez Fernandez		
	Name: Ruben		

3 MEASURING EXECUTION TIMES

Calculate how many more years we can continue using this way of counting. Explain what you did to calculate it.

First we need to calculate how many milliseconds are in a year: 1000ms*60*60*24*365=31,536,000,000ms in a year.

Then if we know that the year is in long data type we have a 2^64 range of time to express which is equivalent to 18,446,744,073,709,551,616.

then 18,446,744,073,709,551,616/31,536,000,000 = 584,942,417 approximated years can be represented since 1970.

Why does the measured time sometimes come out as 0?

Time comes out as 0 because the time measurement was too low.

From what size of problem (n) do we start to get reliable times?

In vector two the argument must be at least 160000000

Taking small execution times (<50 ms)

What happens with time if the problem size is multiplied by 2?

In this case since we are only duplicating the size the complexity stills O(n) then it takes double of time.

What happens with time if the problem size is multiplied by a value k other than 2? (try it, for example, for k=3 and k=4 and check the times obtained)

The size will grow k times but complexity will still be the same then O(n) and for it, it will take the original time measure times k time.

Algorithmics	Student information	Date	Number of session
	UO:300599	28/01/2025	1
	Surname: Alvarez Fernandez		
	Name: Ruben		

Explain whether the times obtained are those expected from the linear complexity O(n)

Yes they are since it will take as many time as repetitions are taken as an argument by the program.

TABLE1 (times in milliseconds WITHOUT OPTIMIZATION):

N	Tsum	Tmaximum
10000	84/10000 milliseconds	96/1000 milliseconds
20000	152/10000 milliseconds	183/1000 milliseconds
40000	297/10000 milliseconds	358/1000 milliseconds
80000	596/10000 milliseconds	719/1000 milliseconds
160000	1180/10000 milliseconds	1434/1000 milliseconds
320000	2362/10000 milliseconds	2876/1000 milliseconds
640000	4771/10000 milliseconds	5768/1000 milliseconds
1280000	9544/10000 milliseconds	11491/1000 milliseconds
2560000	19042/10000 milliseconds	22957/1000 milliseconds
5120000	38397/10000 milliseconds	46067/1000 milliseconds
10240000	77841/10000 milliseconds	92134/1000 milliseconds

Algorithmics	Student information	Date	Number of session
	UO:300599	28/01/2025	1
	Surname: Alvarez Fernandez		
	Name: Ruben		

20480000	153211/10000 milliseconds	184268/1000 milliseconds
40960000		368433/1000 milliseconds
81920000		734071/1000 milliseconds

In this table times meet expectations since both algorithms have a linear complexity, We can see this in the times measured since they tend to increase double it's size simultaneously as the size of repetitions does it too.

TABLE2 (times in milliseconds WITHOUT OPTIMIZATION):

N	*Tmatches1*	*Tmatches2*
10000	789 milliseconds	97/1000 milliseconds
20000	3120 milliseconds	190/1000 milliseconds
40000	12658 milliseconds	371/1000 milliseconds
80000	Oot	739/1000 milliseconds
160000	Oot	1484/1000 milliseconds
320000	Oot	2988/1000 milliseconds
640000	Oot	5963/1000 milliseconds

Algorithmics	Student information	Date	Number of session
	UO:300599	28/01/2025	1
	Surname: Alvarez Fernandez		
	Name: Ruben		

1280000	Oot	11916/1000 millisecond
2560000	Oot	23809/1000 milliseconds
5120000	Oot	47654/1000 milliseconds
10240000	Oot	96207/1000 milliseconds
20480000	Oot	191951/1000 milliseconds
40960000	Oot	383975/1000 milliseconds
81920000	Oot	767950/1000 milliseconds

The results in this table meet the expectations since the algorithm of matches1 has a complexity of O(n^2) then the time increase exponentially making it take too long to measure most of the executions. Matches1 has a linear complexity O(n) such as the algorithms from the other table and so we can see that it behaves the same way.

Pc characteristics:

RAM : 8.00 GB (7.88 GB usable)

CPU: Intel(R) Core(TM) i5-7500 CPU @ 3.40GHz 3.41 GHz