

Algorithmics	Student information	Date	Number of session
	UO: 293860	06/02/2024	1.1
	Surname: López Álvarez		
	Name: Juan		

Activity 1. currentTimeMillis()

I computed the number of years that we can continue using this way of counting in two different but similar ways. The first one is computing the maximum positive integer with 64 bits, which is 2^{63} (the first bit is used for the sign), with this value we have the biggest value that the `currentTimeMillis()` method can return. We divide this value by 1000, then by 3600, then by 24 and finally by 365 to get the number of years. Finally, we subtract the number of years we've been using it (54) and we get the result: 292471154 years.

The other way is using eclipse to make the previous computation the following way:

```

1 package vectors;
2 |
3
4 public class ReamainingYears {
5     public static void main(String[] args) {
6         long max = Long.MAX_VALUE;
7         long currentTime = System.currentTimeMillis();
8
9         long yearMillis = 365L * 24 * 60 * 60 * 1000;
10
11         long remainingYears2 = (max-currentTime)/yearMillis;
12         System.out.println("There are " + remainingYears2 + " years remaining of currentTimeMillis()");
13     }
14 }
15
16 }
17

```

Problems Javadoc Declaration Console X

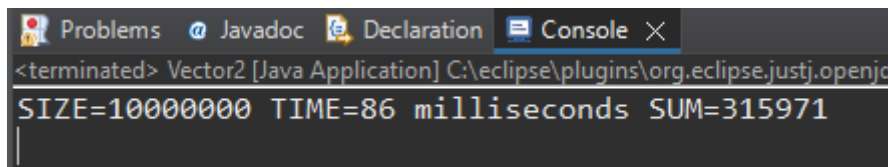
<terminated> ReamainingYears [Java Application] C:\eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64.17.0.2.v20220201-1208\jre\bin\javaw.exe (13 feb 2024 11:46:57 - 11:46:57)

There are 292471154 years remaining of currentTimeMillis()

Activity 2. Vector2 time measurement

Algorithmics	Student information	Date	Number of session
	UO: 293860	06/02/2024	1.1
	Surname: López Álvarez		
	Name: Juan		

Having a measurement of 0 milliseconds is not useful, the reason is that the size of the problem is not big enough and the computer is able to execute the program nearly instantaneously. When we have a size of the problem $n \geq 10000000$ we start having reliable measurements, even though they are still really small (87ms).



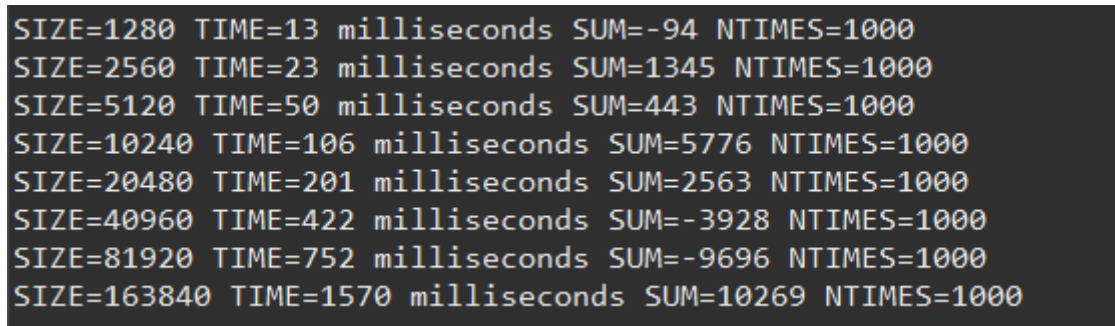
```

Problems  Javadoc  Declaration  Console X
<terminated> Vector2 [Java Application] C:\eclipse\plugins\org.eclipse.justj.openj
SIZE=10000000 TIME=86 milliseconds SUM=315971

```

Activity 3. Tables of measurements

When increasing the size of the problem by 2 we see that the time increases with the size of the problem (more or less), the same happens when increasing the size by 3, 4 and any other k.



```

SIZE=1280 TIME=13 milliseconds SUM=-94 NTIMES=1000
SIZE=2560 TIME=23 milliseconds SUM=1345 NTIMES=1000
SIZE=5120 TIME=50 milliseconds SUM=443 NTIMES=1000
SIZE=10240 TIME=106 milliseconds SUM=5776 NTIMES=1000
SIZE=20480 TIME=201 milliseconds SUM=2563 NTIMES=1000
SIZE=40960 TIME=422 milliseconds SUM=-3928 NTIMES=1000
SIZE=81920 TIME=752 milliseconds SUM=-9696 NTIMES=1000
SIZE=163840 TIME=1570 milliseconds SUM=10269 NTIMES=1000

```

Algorithmics	Student information	Date	Number of session
	UO: 293860	06/02/2024	1.1
	Surname: López Álvarez		
	Name: Juan		

Next tables 1 and 2 are presented in the same table:

n	Tsum (V4)	Tmax (V5)	Tmatches1 (V6)	Tmatches2(V7)
1000	8	22	985	35
2000	17	25	4357	49
4000	36	84	15928	48
8000	70	126	Oot	130
16000	154	355	Oot	275
32000	343	554	Oot	473
64000	570	1049	Oot	898
128000	1194	2021	Oot	2181
256000	2387	4062	Oot	3958
512000	5162	8413	Oot	8415
1024000	9804	15713	Oot	15878
2048000	19703	31075	Oot	33154
4096000	38908	Oot	Oot	Oot
8192000	Oot	Oot	Oot	Oot

The computer has the following specs:

- Processor: Inter Pentium CPU G4560 @ 3.50GHz
- Memory: 8GB @2400MHz

The first two algorithms sum and max match the complexity $O(n)$ as they are both composed by a single for loop which start in 0 and 1 respectively and end in n adding one each iteration. The first version of the matches algorithm has a complexity of $O(n^2)$ and that is the reason why even when we only execute the algorithm once the time spent to execute it with a reasonably big n is too high. On the other side, the second implementation has a complexity of $O(n)$ and we have similar times as in the max algorithm.