# Activity 1.1 [For what value of n do the Subtraction1 and Subtraction2 classes stop giving times (we abort the algorithm because it exceeds 1 minute)? Why does that happen?]

It stops giving values at 8192. Because it gives a StackOverflowError.

Activity 1.2 [How many years would it take to complete the Subtraction3 execution for n=80? Reason the answer.]

Escuela de Ingeniería Informática



Since a = 2, b = 1 and k = 0

Complexity is a^(n/b)

So for n = 80 it would take around 2^80ms \* TimePerInstruction =

43.152.365.729.557.250.048ms OR 1368352540 years.

Activity 1.3 [Implement a Subtraction4.java class with a complexity O(n3) and then fill in a table showing the time (in milliseconds) for n=100, 200, 400, 800, ... (until OoT).

To obtain n^3 we look for an algorithm such as a = 1 and k = 2 by substraction

|  |  |  |
| --- | --- | --- |
| *n* |  | *Time ms* |
| 100 | 1 |  |
| 200 | 11 |  |
| 400 | 79 |  |
| 800 | 622 |  |
| 1600 | 4905 |  |
| 3200 | 38856 |  |

# Activity 1.4 [Implement a Subtraction5.java class with a complexity O(3n/2) and then fill in a table showing the time (in milliseconds) for n=30, 32, 34, 36, … (until OoT).

To obtain 3^(n/2) we look for an algorithm such as a = 3 and b = 2 by substraction

|  |  |  |
| --- | --- | --- |
| *n* |  | *Time ms* |
| 30 | 307 |  |
| 32 | 874 |  |
| 34 | 2617 |  |
| 36 | 7828 |  |
| 38 | 23742 |  |

# Activity 1.5 [How many years would it take to complete the Subtraction5 execution for n=80? Reason the answer.)

Since complexity is 3^(n/2) it would take 3^40ms \* TimePerInstruction =

Escuela de Ingeniería Informática

Taking into account 3^15 \* TimePerInstruction = 307

So 3^40 \* 2,139535784851069144151537117078e-5 =

260.117.603.099.001ms = 8248 years

# Activity 2.1 [Implement a Division4.java class with a complexity O(n2) (with a<b^k) and then fill in a table showing the time (in milliseconds) for n=1000, 2000, 4000, 8000, … (up to OoT).]

K = 2

b = 2

a = 1

|  |  |  |
| --- | --- | --- |
| *n* |  | *Time ms* |
| 1000 | 6 |  |
| 2000 | 21 |  |
| 4000 | 78 |  |
| 8000 | 308 |  |
| 16000 | 1219 |  |
| 32000 | 4817 |  |
| 64000 | 19854 |  |

# Activity 2.2 [Implement a Division5.java class with a complexity O(n2) (with a>b^k) and then fill in a table showing the time (in milliseconds) for n=1000, 2000, 4000, 8000, … (up to OoT).]

K = 0

b = 2

a = 4

|  |  |  |
| --- | --- | --- |
| *n* |  | *Time ms* |
| 1000 | 19 |  |
| 2000 | 74 |  |
| 4000 | 293 |  |
| 8000 | 1158 |  |
| 16000 | 4563 |  |
| 32000 | 18428 |  |

# Activity 3.2 [After analyzing the complexity of the various algorithms within the two classes, executing them and after putting the times obtained in a table, compare the efficiency of each algorithm.

VectorSum2.java

1

(IMPORTANT: TIMES ARE MULTIPLIED BY 100000)

|  |  |  |
| --- | --- | --- |
| *n* |  | *Time ms* |
| 3 | 5 |  |
| 6 | 7 |  |
| 12 | 10 |  |
| 24 | 14 |  |
| 48 | 23 |  |
| 96 | 42 |  |
| 192 | 77 |  |
| 384 | 146 |  |
| 768 | 285 |  |
| 1536 | 565 |  |
| 3072 | 1129 |  |
| 6144 | 2244 |  |
| 12288 | 4499 |  |
| 24576 | 9017 |  |
| 49152 | 18033 |  |
| 98304 | 36408 |  |

2

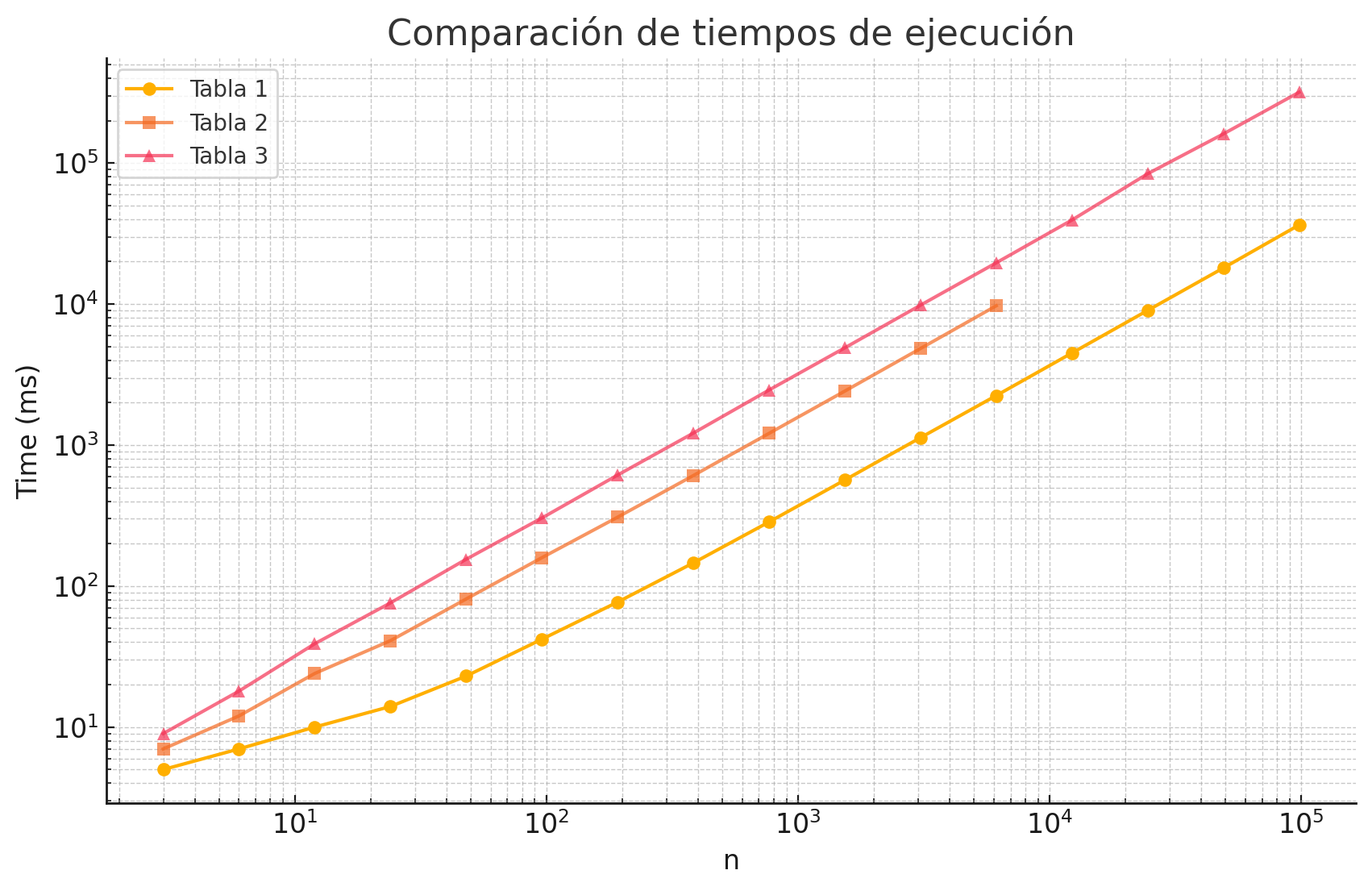
(IMPORTANT: TIMES ARE MULTIPLIED BY 100000)

|  |  |  |
| --- | --- | --- |
| *n* |  | *Time ms* |
| 3 | 7 |  |
| 6 | 12 |  |
| 12 | 24 |  |
| 24 | 41 |  |
| 48 | 81 |  |
| 96 | 159 |  |
| 192 | 308 |  |
| 384 | 608 |  |
| 768 | 1211 |  |
| 1536 | 2414 |  |
| 3072 | 4841 |  |
| 6144 | 9725 |  |
| 12288 | SO |  |
| 24576 | SO |  |
| 49152 | SO |  |
| 98304 | SO |  |

3

(IMPORTANT: TIMES ARE MULTIPLIED BY 100000)

|  |  |  |
| --- | --- | --- |
| *n* |  | *Time ms* |
| 3 | 9 |  |
| 6 | 18 |  |
| 12 | 39 |  |
| 24 | 76 |  |
| 48 | 155 |  |
| 96 | 304 |  |
| 192 | 613 |  |
| 384 | 1216 |  |
| 768 | 2458 |  |
| 1536 | 4893 |  |
| 3072 | 9821 |  |
| 6144 | 19616 |  |
| 12288 | 39462 |  |
| 24576 | 84000 |  |
| 49152 | 161000 |  |
| 98304 | 318000 |  |



Fibonacci2.java

1

(IMPORTANT: TIMES ARE MULTIPLIED BY 1000000)

|  |  |  |
| --- | --- | --- |
| *n* |  | *Time ms* |
| 10 | 86 |  |
| 11 | 89 |  |
| 12 | 92 |  |
| 13 | 99 |  |
| 14 | 102 |  |
| 15 | 106 |  |
| 16 | 111 |  |
| 17 | 120 |  |
| 18 | 117 |  |
| 19 | 122 |  |
| 20 | 126 |  |
| 21 | 135 |  |
| 22 | 137 |  |
| 23 | 142 |  |
| 24 | 145 |  |
| 25 | 148 |  |

2

(IMPORTANT: TIMES ARE MULTIPLIED BY 1000000)

|  |  |  |
| --- | --- | --- |
| *n* |  | *Time ms* |
| 10 | 112 |  |
| 11 | 117 |  |
| 12 | 126 |  |
| 13 | 127 |  |
| 14 | 134 |  |
| 15 | 143 |  |
| 16 | 149 |  |
| 17 | 157 |  |
| 18 | 165 |  |
| 19 | 170 |  |
| 20 | 179 |  |
| 21 | 188 |  |
| 22 | 190 |  |
| 23 | 200 |  |
| 24 | 205 |  |
| 25 | 210 |  |

3

(IMPORTANT: TIMES ARE MULTIPLIED BY 1000000)

|  |  |  |
| --- | --- | --- |
| *n* |  | *Time ms* |
| 10 | 172 |  |
| 11 | 184 |  |
| 12 | 221 |  |
| 13 | 238 |  |
| 14 | 256 |  |
| 15 | 266 |  |
| 16 | 282 |  |
| 17 | 297 |  |
| 18 | 315 |  |
| 19 | 325 |  |
| 20 | 343 |  |
| 21 | 358 |  |
| 22 | 370 |  |
| 23 | 389 |  |
| 24 | 402 |  |
| 25 | 415 |  |

