|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| LEARNING PROFILE FOR FindMaxDivisors | | | | | |
| *Name* | *:* | *Tyler Lucas* | *Submission Date* | *:* | *N/A* |
| *Student ID* | *:* | *3305203* | *Created On* | *:* | *2017-05-16* |

# Problem Statement

Which integer between 1 and 10000 has the largest number of divisors, and how many divisors does it have? Write a program to find the answers and print out the results. It is possible that several integers in this range have the same, maximum number of divisors. Your program only has to print out one of them.

# Description of the Code

Iterates from highest number to lowest, determining how many divisors is in each and whether that is the most encountered thus far. The lowest number with the maximum number of divisors, and its list of divisors, is remembered. It starts from the highest number intentionally, so that the searching loop can skip those lower numbers that had already been found to be factors of those higher, as they must have fewer divisors (by at least 1). This speeds up the loop by about half.

It has the features of remembering the numbers, if any, that share the same number of divisors as that which has the most, and remembering the divisors of the lowest of these. Currently commented out but otherwise functional is a bit at the end that tells the user how many numbers the loop was able to skip as a result of finding them earlier to be factors of larger numbers within the search range.

# Errors and Warnings

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Errors / Warnings** | **Details** | **How I solved them** |
| 1 | java.lang.ArrayIndexOutOfBoundsException: 10000 | Tried accessing element #10000 with index 10000 instead of 9999. | Changed indexing to 1 less than previous implementation. |
| 2 | java.lang.RuntimeException: Uncompilable source code - variable strDivisorsOfMax might not have been initialized at FindMaxDivisors.main(FindMaxDivisors.java:75) | strDivisorsOfMax was not initialized. | strDivisorsOfMax = “”; |
| 3 | Incorrect behaviour. | Divisors tally was 1 below actual. I had forgotten to include ‘1’ as a factor. | Added 1. |

# Sample Input and Output

Version 1.0

The number under 10000 with the most divisors is 7560.

It has 64 divisors, including 1 and itself, as follows:

7560, 3780, 2520, 1890, 1512, 1260, 1080, 945, 840, 756, 630, 540, 504, 420, 378, 360, 315, 280, 270, 252, 216, 210, 189, 180, 168, 140, 135, 126, 120, 108, 105, 90, 84, 72, 70, 63, 60, 56, 54, 45, 42, 40, 36, 35, 30, 28, 27, 24, 21, 20, 18, 15, 14, 12, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1

These numbers were in the same range and had the same large number of divisors (though not the same ones): 9240

Version 1.1

The number under 100 with the most divisors is 60.

It has 12 divisors, including 1 and itself, as follows:

60, 30, 20, 15, 12, 10, 6, 5, 4, 3, 2, 1

These numbers are in the same range and also have 12 divisors:

( listed as <number>, <list of number's divisors> )

96, 48, 32, 24, 16, 12, 8, 6, 4, 3, 2, 1

90, 45, 30, 18, 15, 10, 9, 6, 5, 3, 2, 1

84, 42, 28, 21, 14, 12, 7, 6, 4, 3, 2, 1

72, 36, 24, 18, 12, 9, 8, 6, 4, 3, 2, 1

# Discussion

I copied the output (the list of divisors) to an Excel spreadsheet and verified that the numbers were correct. I also used a web calculator to verify the results (see [Divisors Calculator – 20-Digits!](http://www.javascripter.net/math/calculators/divisorscalculator.htm)[[1]](#footnote-1)).

It looks like there is quite a bit of room for improvement in the algorithm, using prime numbers, starting near the square root of the number, and likely some other techniques that I don’t know about. It works fine for our range of 1 to 10000, but seems to choke for larger ranges. I tried a few:

* 1 to 50000: 45360 with 100 divisors. It took about 4 seconds.
* 1 to 100000: 83160 and 98280 with 128 divisors. It took about 14 seconds.
* 1 to 250000: 221760 with 168 divisors. It took about 1 minute and 25 seconds.
* 1 to 500000: 498960 with 200 divisors. It took about 5 minutes and 39 seconds.

At this rate, my computer might take over 10 hours to find the result for the range up to 970000.

The fit line is approximately , shown below.

Figure 2: Approximate Program Run Time as a Function of Search Range

1. http://www.javascripter.net/math/calculators/divisorscalculator.htm [↑](#footnote-ref-1)