Time complexity of modulo operator in Python

Asked 7 years, 7 months ago Modified 2 years, 11 months ago Viewed 11k times

8

I am trying to determine the time complexity of an algorithm that I have, but I need first to know the time complexity of the % (modulo) operator in Python.

According to this post on http://math.stackexchange.com, its time complexity could be something similar to `O(log m log n)`, and in some specific cases it could also be optimised to be constant, but I would like to know if someone really knows the time complexity of `%`, so that I can determine correctly the overall time complexity of my algorithm.

Of course I am aware that the complexity could change from implementation to implementation, but I am interested only in the standard implementation.

python python-3.x operators time-complexity modulo Edit tags

Share Edit Follow Close Flag

Protect

edited Apr 13, 2017 at 12:19

Community Bot

1 1

asked Feb 3, 2016 at 23:35

nbro

15.4k 32 113 197

1 As that post explains, the modulo operator on fixed-length integers is a single machine instruction, O(1). Is your algorithm some other use of modulo? – Prune Feb 3, 2016 at 23:47

1 Python supports arbitrarily long integers, it should slow down eventually – felixbade Feb 4, 2016 at 0:13

3 Answers

Sorted by: Reset to default Date modified (newest first)

4

It's not that easy to determine, because if we speak about integer math, cpython uses different optimizations(for example for integers not exceeding the machine word it may be O(1) and for others it may be other). So there are two ways: first is looking into cpython sources and the second is measuring performance(for example with timeit) and then building extrapolation curve based on experimental points. The second method is better, because you would get an exact result, rather than a guess. For simple purposes, building a plot of experimental points should be enough, and if you want more, you may also use some regression analysis methods(like least-squares polynomial fitting).

Here's source of int implementation in cpython (look for long_divrem and x_divrem routines): https://hg.python.org/cpython/file/tip/Objects/longobject.c

Added: For unsigned int modulo its used algorithm from Knuth's book, which is O(MN) where M+1 is number of machine words in the quotient and N is number of machine words in remainder. For signed it's used own implementation

Share Edit Follow Flag

edited Feb 4, 2016 at 0:44

answered Feb 4, 2016 at 0:25

thodnev

1,564 16 20

2

For large integers, Python division (and modulo) use an O(n^2) algorithm. Multiplication uses the Karatsuba multiplication which is O(n^1.585) but division uses basic "grade-school" division.

Share Edit Follow Flag

answered Feb 4, 2016 at 0:37

casevh

11.1k 1 24 35

3 you might mean `log n` (the number of digits in the number), not `n` here (assuming `n % m` expression) i.e., `O(log m log n)` time complexity. – jfs Feb 4, 2016 at 12:33

Yes, jfs is correct according to this answer: stackoverflow.com/a/18200092/1064565 – Jamie Mar 5, 2019 at 18:46

0

This post is hidden. It was deleted 2 years ago by Bhargav Rao.

My python sucks i dont know what to do lorem ipsum dolor sit amet consectetur adpiscing

Share Edit Follow **Undelete** Flag

answered Oct 25, 2020 at 4:05

user14514855

1

Comments disabled on deleted / locked posts / reviews
