Finding Pythagorean Triples by Brute Force Search

Jose Torres Postigo

2023-09-23

1 Objectives

This lab session outlines the problem to implement the brute-force algorithm to find all primitive Pythagorean triples, given a positive integer $l \in \mathbb{N}$. The objectives of this are not only implement it successfully with improvements, but also to confirm the theoretical number of triples and evaluate the running time of the algorithm. I will accomplish this just by verifying empirically the number of primitive Pythagorean triples, and checking the experimental complexity of the algorithm.

A Pythagorean triple consists of three positive integers a, b and c, such that $a^2 + b^2 = c^2$, $a < b < c$. This triple is represented like (a, b, c). A primitive Pythagorean triple is one which a and b are coprime, i.e., they have no common divisor larger than 1.

The main problem of this brute-force algorithm is that the complexity is $O(l^3)$ in the first approach because of its inner loops. It can be reduced to $O(l^2)$ with some improvements, so I am going to run the experiment with this last one so this experiment is feasible.

2 Experimental Setup

The experiment is conducted easily: I run the algorithm giving the parameter l of the algorithm a very high value: 100,000 in this case. This will lead me to know better how efficient is this algorithm with these kind of values.

Table 1: Computational environment considered.

CPU	Intel® Core™ i7-6700HQ × 8, 16.0 GiB
OS	Fedora Linux 38 (Workstation Edition)
Java	openjdk 17.0.8 2023-07-18

3 Empirical Results

A summary of the experimental results is provided in Table 2 in the Appendix, along with the statistical fitting of the data to different growth models.

The graph Figure 1 shows us that the total of pythagorean triples according to the value of l, the maximum value, it is correct in every case. This means that

the algorithm implemented obtains every primitive Pythagorean triple that can under the value of l.

The graph Figure 2 shows the fit of the data taken doing the experiment to the model. The experiment, as we can see, goes from a maximum value of 10 up to 100,000; this value is l. The figures are given in seconds.

Overall, there is a good trend at the beginning of the graph, from $l = 10$ up to $l = 50\,000$ approximately. From this value, the fit of the data start being loose regarding the model, and start taking the shape of another one. In this case, looks like is an hyperbolic tangent.

```
## Warning: package 'ggplot2' was built under R version 4.3.1
```

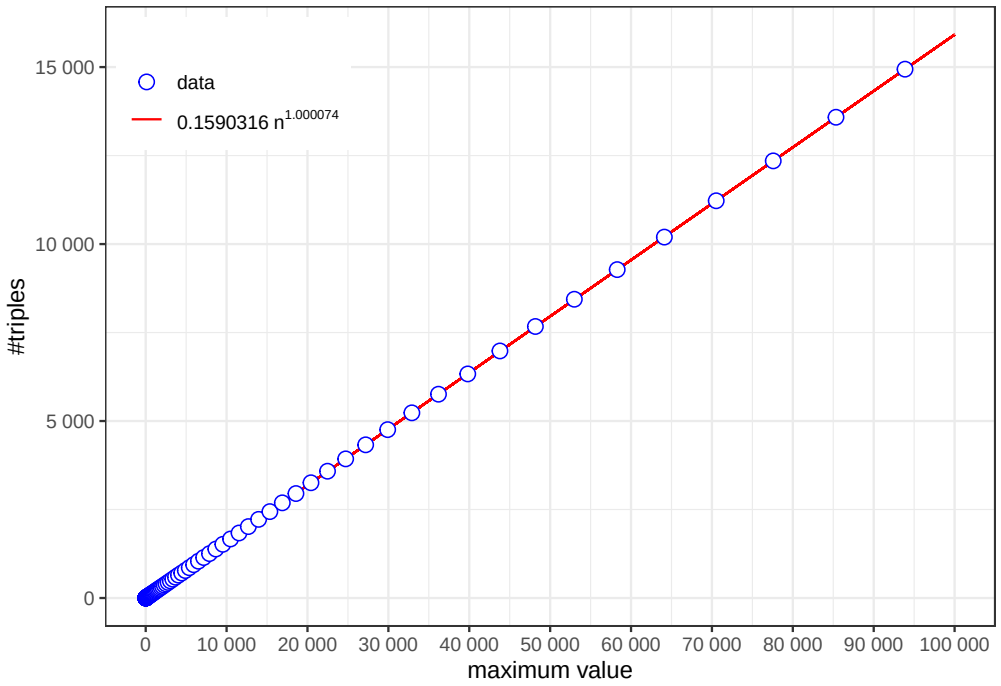

Figure 1: Number of primitive Pythagorean triples up to a maximum value

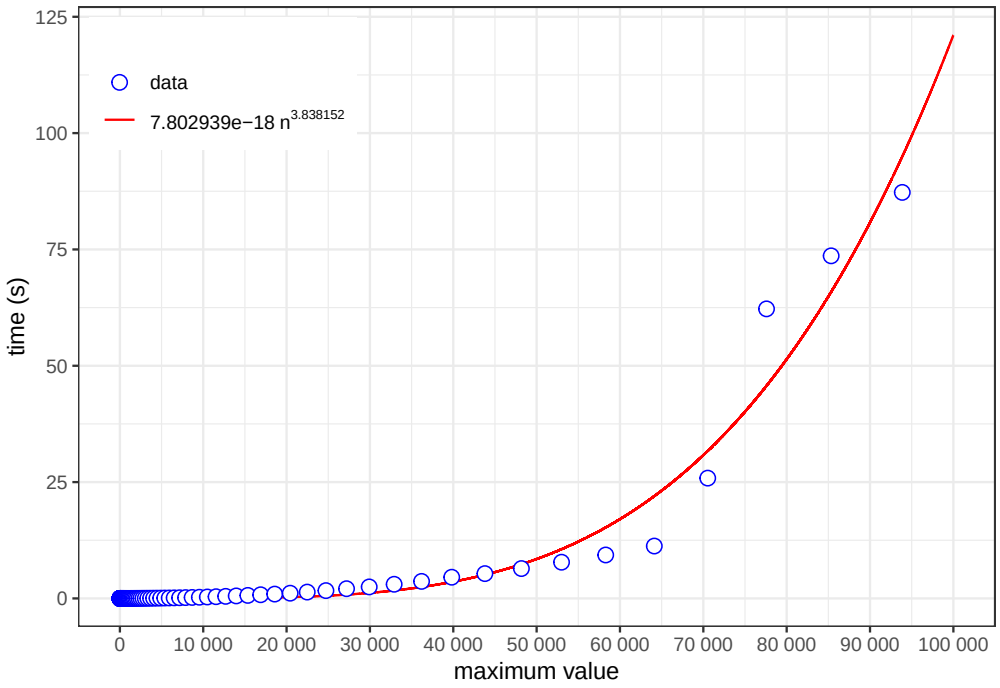

Figure 2: Time required for finding primitive Pythagorean triples up to a maximum value

4 Discussion

Even though in the graph Figure 1 I have got a perfect fit, in the graph Figure 2 the model does not look like is the correct one. Maybe is not this the actual problem, but the hardware i am using to accomplish this experiment.

In my opinion, this model should be reviewed and the data contrasted with another models to find a better one.

This is not the most efficient approach for finding Pythagorean triples, but the brute-force approach used in this project is the easiest to implement. Looking for more approaches to this problem, I found the Euclid's formula.

The Euclid's formula provides a more efficient solutions with a time complexity of $O(n)$. Seems to be the preferred choice for data scientists and software engineers due to its speed and scalability.

A Appendix

A.1 Data Summary

```
## Warning: package 'knitr' was built under R version 4.3.1
```

Table 2: Summary of the experimental results

maxval	#triples	time
10	1	0.0023478
11	1	0.0000184
12	1	0.0000207
13	2	0.0000405
14	2	0.0000370
15	2	0.0000408
16	2	0.0000300
17	3	0.0000381
18	3	0.0000593
19	3	0.0000621
20	3	0.0000622
22	3	0.0000498
24	3	0.0000473
26	4	0.0000652
28	4	0.0000660
30	5	0.0000836
33	5	0.0001445
36	5	0.0000893
39	6	0.0001271
42	7	0.0001216
46	7	0.0001319
50	7	0.0001723
55	8	0.0001762
60	8	0.0001841
66	11	0.0002632
72	11	0.0002688
79	12	0.0003827
86	14	0.0003508
94	15	0.0004619
103	17	0.0005291
113	19	0.0006496
124	19	0.0003380
136	20	0.0004477
149	24	0.0005353
163	25	0.0005718
179	27	0.0007061
196	31	0.0007467
215	34	0.0002341
236	38	0.0003621
259	40	0.0003173
284	45	0.0006480
312	49	0.0006928
343	54	0.0007435
377	61	0.0008964
414	65	0.0009771
455	72	0.0009318
500	80	0.0010564
550	89	0.0011808

maxval	#triples	time
605	96	0.0015593
665	104	0.0028234
731	116	0.0033733
804	128	0.0026417
884	140	0.0036977
972	154	0.0037054
1069	169	0.0042033
1175	187	0.0051306
1292	207	0.0061766
1421	227	0.0075964
1563	249	0.0086981
1719	273	0.0101665
1890	302	0.0124014
2079	329	0.0148046
2286	364	0.0176578
2514	399	0.0210273
2765	440	0.0245136
3041	484	0.0285644
3345	532	0.0343794
3679	587	0.0411193
4046	646	0.0494350
4450	706	0.0594633
4895	776	0.0687401
5384	856	0.0857040
5922	941	0.1024385
6514	1040	0.1230864
7165	1143	0.1479568
7881	1254	0.1801892
8669	1383	0.2108935
9535	1517	0.2533542
10488	1668	0.3076286
11536	1837	0.3693021
12689	2018	0.4458163
13957	2222	0.5372873
15352	2441	0.6485005
16887	2689	0.7881002
18575	2951	0.9311796
20432	3256	1.1232715
22475	3582	1.3645177
24722	3931	1.6783680
27194	4328	2.0902131
29913	4757	2.4753701
32904	5232	3.0407420
36194	5758	3.6593705
39813	6331	4.5552621
43794	6980	5.3330445
48173	7671	6.4175350
52990	8441	7.7793825
58289	9278	9.3376337
64117	10197	11.2555474

maxval	#triples	time
70528	11223	25.8541627
77580	12350	62.2259497
85338	13583	73.6247369
93871	14941	87.2577775

A.2 Model Fitting

A.2.1 Number of Pythagorean triples

```
##
## Formula: numtriples ~ a * maxval^b
##
## Parameters:
##    Estimate Std. Error t value Pr(>|t|)
## a 0.1590316  0.0002183   728.4   <2e-16 ***
## b 1.0000738  0.0001247  8017.8   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.234 on 100 degrees of freedom
##
## Number of iterations to convergence: 2
## Achieved convergence tolerance: 8.488e-09
```

A.2.2 Computational time

```
##
## Formula: time ~ a * maxval^b
##
## Parameters:
##    Estimate Std. Error t value Pr(>|t|)
## a 7.803e-18  1.291e-17   0.604    0.547
## b 3.838e+00  1.456e-01  26.357   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.446 on 100 degrees of freedom
##
## Number of iterations to convergence: 290
## Achieved convergence tolerance: 8.339e-06
```