Euler's project problem 12

John Fox

January 24, 2023

Problem statement

The sequence of triangle numbers is generated by adding the natural numbers.
So the 7th triangle number would be 1 + 2 + 3 + 4 + 5 + 6 + 7 = 28.
The first ten terms would be: 1, 3, 6, 10, 15, 21, 28, 36, 45, 55, ...
Let us list the factors of the first seven triangle numbers:
1: 1
3: 1,3
6: 1,2,3,6
10: 1,2,5,10
15: 1,3,5,15
21: 1,3,7,21
28: 1,2,4,7,14,28
We can see that 28 is the first triangle number to have over five divisors.
What is the value of the first triangle number to have over five hundred divisors?

Answer

The first triangular number with 500 divisors is 76576500

Idea

From geeks for geeks I got an idea on how to check divisors efficiently. If you are only checking up to the square root of the number any divisor that isnt the square root of the number actually should add 2 divisors because the divisor evenly divides the number and the other number in the factorization also evenly divides the number. If the divisor is the square root of the number just count it once.
To get the triangular numbers I did not re calculate the entire series every time. Instead I used a simplified version of memoization to save the last triangular number.
With these functions written efficiently the main function firstToNDivisors(n) was easy to write. Starting at 1 keep checking if the ith triangular number has 500 divisors.

Python code

```
import math
def getDivisors(n):
   numDiv = 0
   for i in range(1,math.floor(1 + math.sqrt(n))):
      if n % i == 0:
         if n / i == i:
            numDiv = numDiv + 1
         else:
            numDiv = numDiv + 2
   return numDiv

def trianglular(curNumber, index):
   return(curNumber + index)
```

```
def firstToNDivisors(n):
   i = 1
   curTriangle = 0
   while getDivisors(trianglular(curTriangle, i)) <= n:
      curTriangle = trianglular(curTriangle, i)
      i = i + 1
   return trianglular(curTriangle, i)

print(firstToNDivisors(500))
```