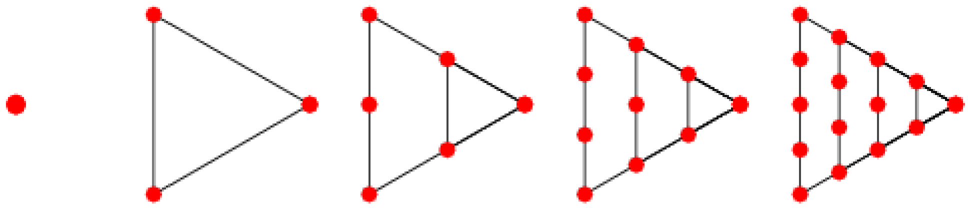
Highly Divisible Triangular Number — Project Euler (Problem 12)

pythonandr.com/2015/09/01/highly-divisible-triangular-number-project-euler-problem-12/

Anirudh September 1, 2015

All ∑**n** numbers are Triangle Numbers. They're called so, because they can be represented in the form of a triangular grid of points where the first row contains a single element and each subsequent row contains one more element than the previous one.



Problem 12 of Project Euler asks for **the first triangle number with more than 500 divisors.**

These are the **factors of the first seven triangle numbers**:

∑**1 = 1**: 1
∑**2 = 3**: 1,3
∑**3 = 6**: 1,2,3,6
∑**4 = 10**: 1,2,5,10
∑**5 = 15**: 1,3,5,15
∑**6 = 21**: 1,3,7,21
∑**7 = 28**: 1,2,4,7,14,28

Here's how I proceeded:

First Step: Find the smallest number with 500 divisors. Seems like a good starting point to begin our search.
Second Step: Starting at the number found in the previous step, search for the next triangle number. Check to see whether this number has 500+ divisors. If yes, this is the number we were looking for, else…
Third Step: Check n for which ∑n = triangle number found in the previous step
Fourth Step: Add (n+1) to the last triangle number found, to find the next triangle number. Check whether this number has 500+ divisors. If yes, this number is the answer. If not, repeat Fourth Step till the process terminates.

Now for the details:

The **First Step** isn't exactly a piece of cake, but necessary to reduce computation time. I solved this with a bit of mental math. The main tool for the feat is the **prime number decomposition theorem**:

> **Every integer N is the product of powers of prime numbers**

$N = p^{\alpha}q^{\beta}\cdot \ldots \cdot r^{\gamma}$

Where **p, q, …, r** are prime, while **α, β, …, γ** are positive integers. Such representation is unique up to the order of the prime factors.

If N is a power of a prime, **$N = p^{\alpha}$**, then it has **α + 1** factors:

$1, p, \ldots, p^{\alpha-1}, p^{\alpha}$

The total number of **factors of N** equals **(α + 1)(β + 1) … (γ + 1)**

500 = 2 x 2 x 5 x 5 x 5

So, the number in question should be of the form **$abq^4r^4s^4$** where **a, b, q, r, s** are primes that minimize **$abq^4r^4s^4$**. This is satisfied by **$7 \times 11 \times 2^4 \times 3^4 \times 5^4 = 62370000$**. This marks the end of the **First Step** which is where we start our search for our magic number.

The next 3 steps would need helper functions defined as below:

```
from math import *

# Function to calculate the number of divisors of integer n
def divisors(n):
limit = int(sqrt(n))
divisors_list = []
for i in range(1, limit+1, 1):
if n % i == 0:
divisors_list.append(i)
if i != n/i:
divisors_list.append(n/i)
return len(divisors_list)

# Function to check for triangle number
def isTriangleNumber(n):
a = int(sqrt(2*n))
return 0.5*a*(a+1) == n

# Function to calculate the last term of the series adding up to the triangle number
def lastTerm(n):
if isTriangleNumber(n):
return int(sqrt(2*n))
```

```
else:
return None
```

view raw euler12functions.py hosted with ❤ by GitHub

As can be seen from the above code, the algorithm to **calculate divisors of an integer** is as follows:

1. Start by inputting a number **n**
2. Let an int variable **limit = √n**
3. Run a loop from **i = 1** to **i = limit**
3.1 if **n** is divisible by **i**
3.1.1 Add **i** to the list of divisors
3.1.2 if **i** and **n/i** are unequal, add **n/i** to the list too.
4. End

Finally, executing the 4 steps mentioned earlier can be done like so (the code took less than **2***s* to arrive at the answer):

```
# First Step
# First number 'check' to have 500 divisors
check = 2**4 * 3**4 * 5**4 * 7 * 11

# Second Step
# Starting from 'check', iterate sequentially checking for the next 'triangle' number
while not isTriangleNumber(check):
check += 1

# Third and Fourth Steps
# Calculate the last term of the series ('seriesLastTerm') that adds up to the newly calculated triangle number 'check'
seriesLastTerm = lastTerm(check)

# Iterate over triangle numbers checking for divisors > 500
while divisors(check) <= 500:
# add the next term to check to get the next triangle number
check += (seriesLastTerm + 1)
seriesLastTerm += 1
print check
```

view raw euler12nonmeatypart.py hosted with ❤ by GitHub

Ans: *76576500*