

Euler 0003

The Problem:

The prime factors of 13195 are 5, 7, 13, and 29.

What is the largest prime factor of the number 600951475143?

Considerations:

There are a lot of approaches, of which I will probably use later.

We are going to start with some basic sieving and then work out a more general algorithm:

Iterate from 2 to the upper limit:

- Is it divisible by 2:
 - add 2 to the prime factors list
 - set the loop to iterate up to $\text{upper_limit}/2$
 - Is it divisible by 2:
 - Keep doing above operation
 - Is it divisible by 3:
 - add 3 to the prime factors list
 - set the loop to iterate up to $\text{upper_limit}/3$
 - We don't care about 4.
 - Is it divisible by 5:
 - add 5 to the prime factors list
 - set the loop to iterate up to $\text{upper_limit}/5$
 - We don't care about 6.
 - Is it divisible by 7:
 - add 5 to the prime factors list
 - set the loop to iterate up to $\text{upper_limit}/7$
- etc.

Something to note: We don't care about any multiples of 2 and 3 after we check them, so a quick way to get around checking these multiples we can iterate through the loop by adding 6 and checking the before and after numbers.

The Code:

```

upper_limit = 6009515658416
#upper_limit = 20023110541 #, good to use to double check. Is 2 [2,2],3,167
current_upper_limit = upper_limit
prime_factors = []

#we are going to hardcode 2 and 3 for looping sake
if upper_limit % 2 == 0:
    while current_upper_limit % 2 == 0:
        current_upper_limit = current_upper_limit/2
        prime_factors += [2]
if upper_limit % 3 == 0:
    while current_upper_limit % 3 == 0:
        current_upper_limit = current_upper_limit/3
        prime_factors += [3]
current = 6

#go up to the upper limit of numbers, add 2 for edge case (I'm not sure that part is
correct)
while current < current_upper_limit + 2:
    #print(prime_factors, current_upper_limit)
    #print(prime_factors, current_upper_limit)
    #check the numbers before and after the increment of 6
    #add them to the prime factors as many times as necessary
    #keep dividing the new number to use.
    if current_upper_limit % (current - 1) == 0:
        while current_upper_limit % (current - 1) == 0:
            current_upper_limit = current_upper_limit/(current-1)
            prime_factors += [current-1]
    if current_upper_limit % (current + 1) == 0:
        while current_upper_limit % (current + 1) == 0:
            current_upper_limit = current_upper_limit/(current+1)
            prime_factors += [current+1]
    current += 6

print(prime_factors)

```

The Twist:

While I was testing this it took insanely long because I was calculating for 600951475143 instead of 600851475143.

The largest prime factor of 600851475143 is 6857. This is not very high, and easy enough to brute force.

The largest prime factor of 600951475143 is 1552846189. This is a lot of iterations to get too and the code would timeout.

After reading the solution provided by Project Euler I understood what I was missing to get the solution to this typo'd part.

Every number can have at most one prime factor that is greater than its square root. I have modified the code below to reflect this change and break out of the while loop if the `current_upper_limit` is greater than the square root of the original number.

This means that we only have to iterate through 775210 (square root of 600951475143) before we (can rightfully) call it quits and accept that:

- This is prime
- This is the last prime factor

Updated Code:

```
import math

upper_limit = 600951565841652
#upper_limit = 20023110541 #, good to use to double check. Is [2,2,3,167]
current_upper_limit = upper_limit
sqrt_limit = math.sqrt(upper_limit)
prime_factors = []

#we are going to hardcode 2 and 3 for looping sake
if upper_limit % 2 == 0:
    while current_upper_limit % 2 == 0:
        current_upper_limit = current_upper_limit/2
        prime_factors += [2]
if upper_limit % 3 == 0:
    while current_upper_limit % 3 == 0:
```

```

        current_upper_limit = current_upper_limit/3
        prime_factors += [3]
current = 6

#go up to the square root
while current < sqrt_limit:
    #print(prime_factors, current_upper_limit)
    #check the numbers before and after the increment of 6
    #add them to the prime factors as many times as necessary
    #keep dividing the new number to use.
    if current_upper_limit % (current - 1) == 0:
        while current_upper_limit % (current - 1) == 0:
            current_upper_limit = current_upper_limit/(current-1)
            prime_factors += [current-1]
    if current_upper_limit % (current + 1) == 0:
        while current_upper_limit % (current + 1) == 0:
            current_upper_limit = current_upper_limit/(current+1)
            prime_factors += [current+1]
    current += 6

#if we looped beyond the square root limit and the number
#we are left with isn't 1, then it is the final prime factor.
if current_upper_limit != 1:
    prime_factors += [current_upper_limit]

print(prime_factors)

```

Revision #3

Created 2025-06-17 00:51:46 UTC by Maxwell

Updated 2025-07-17 14:49:04 UTC by Maxwell