Solutions to Project Euler Problem 3: Largest Prime Factor

Alister McKinley

https://projecteuler.net/problem=3

The problem reads:

> The prime factors of 13195 are 5, 7, 13 and 29.
>
> What is the largest prime factor of the number 600851475143 ?

Dividing Out Approach

The Fundamental Theorem of Arithmetic states that any integer greater than 1 can be expressed as a unique product of prime factors. This product can be written in a canonical form as follows:

$$n = p_1^{r_1} p_2^{r_2} p_3^{r_3} \dots p_n^{r_n}$$

where:

$n, r_i \in \mathbb{Z}_{\geq 1}$	is a positive integer
$p_1 < p_2 < \cdots < p_i \in \mathbb{N}$	is prime

Our dividing out approach will repeatedly divide n by an increasing denominator d in order to discover the prime factors of n. Once this is complete, the largest prime factor found is the solution to the problem.

We start by setting our denominator d equal to the smallest prime:

$$d \leftarrow 2$$

Next, if d divides n we assign n the value of $\frac{n}{d}$ and we assign the highest factor found so far, $p_{\max}$, the value of d. If d does not divide n, we increment d by 1.

$$\text{If } d \mid n: \quad \begin{aligned} n &\leftarrow \frac{n}{d} \\ p_{\max} &\leftarrow d \end{aligned} \qquad \text{If } d \nmid n: \quad d \leftarrow d + 1$$

This process continues until d is larger than n. Once this occurs, $p_{\max}$ contains the answer to the problem.

The following table will demonstrate this process for the number 13195. As per the example we expect the highest prime factor to be 29.

n	d	$n/d \in \mathbb{Z}$	n/d	$p_{\max}$
13195	2	False		
13195	3	False		
13195	4	False		
13195	5	True	2639	5
2639	5	False		5
2639	6	False		5
2639	7	True	377	7
377	7	False		7
377	8	False		7
377	9	False		7
377	10	False		7
377	11	False		7
377	12	False		7
377	13	True	29	13
29	13	False		13
29	14	False		13
29	15	False		13
29	16	False		13
29	17	False		13
29	18	False		13
29	19	False		13
29	20	False		13
29	21	False		13
29	22	False		13
29	23	False		13
29	24	False		13
29	25	False		13
29	26	False		13
29	27	False		13
29	28	False		13
29	29	True	1	29

We can shorten this process by considering the maximum value of the denominator we will use when dividing. We will set this maximum value as follows:

$$d_{\max} = \lfloor \sqrt{n} \rfloor$$

The process now ends as soon as we are attempting to divide by a factor that is larger than this calculated maximum i.e. $d > d_{\max}$. At this point the number we are attempting to divide, n, is the largest prime factor we will find. The process becomes:

$$\text{If } d \mid n: \quad n \leftarrow \frac{n}{d} \qquad\qquad \text{If } d \nmid n: \quad d \leftarrow d+1$$
$$d_{\max} \leftarrow \lfloor \sqrt{n} \rfloor$$

The following table will demonstrate this shortened process for the number 13195. As per the example we expect the highest prime factor to be 29.

n	$d_{\max}$	d	$n/d \in \mathbb{Z}$	n/d
13195	114	2	False	
13195	114	3	False	
13195	114	4	False	
13195	114	5	True	2639
2639	51	5	False	
2639	51	6	False	
2639	51	7	True	377
377	19	7	False	
377	19	8	False	
377	19	9	False	
377	19	10	False	
377	19	11	False	
377	19	12	False	
377	19	13	True	29
29	5	13		

The process ended when d became larger than $d_{\max}$. As expected the number we were dividing, $n = 29$, is the largest prime factor.

Python Implementation

```python
def prime_factors(number: int) -> Set[int]:
    """Calculate the primes that will divide number.

    Prime numbers with a power of zero are not included.
    Prime numbers with a power greater than 1 are not repeated.
    """
    result = set()
    if number <= 1:
        return result
    current_numerator = number
    current_factor = 2
    max_factor = floor(sqrt(number))
    while current_factor <= max_factor:
        quotient, remainder = divmod(current_numerator,
                                     current_factor)
        if remainder == 0:
            current_numerator = quotient
```

```
            max_factor = floor(sqrt(current_numerator))
            result.add(current_factor)
        else:
            current_factor = current_factor + 1
    result.add(current_numerator)
    return result

print(max(prime_factors(number)))
```