Euler 0023

The Problem:

A perfect number is a number for which the sum of its proper divisors is exactly equal to the number.
A number is called deficient if the sum of its proper divisors is less than and it is called abundant if this sum exceeds.

As 12 is the smallest abundant number, the smallest number that can be written as the sum of two abundant numbers is 24. By mathematical analysis, it can be shown that all integers greater than 28123 can be written as the sum of two abundant numbers. However, this upper limit cannot be reduced any further by analysis even though it is known that the greatest number that cannot be expressed as the sum of two abundant numbers is less than this limit.

Find the sum of all the positive integers which cannot be written as the sum of two abundant numbers.

Considerations and Approach:

Based off of the question we know that we only have to go up to 28123.
We are going to generate all abundant numbers up to 28123 - 12, which is the the smallest abundant number.
After we generate all abundant numbers, we can generate all of the numbers that are sums of two abundant numbers. Then we subtract that from 28123 and voila.

The Code:

```
import math


def generate_primes(upper):
    prime_threshold = upper
    number_list = list(range(2, prime_threshold+1))
    prime_list = []
```

```
    current = 0
    total_iteration = 0
    while current < len(number_list):
        if number_list[current] != -1:
            prime = number_list[current]
            prime_list += [prime]

            increment = current + prime
            while increment < len(number_list):
                number_list[increment] = -1
                increment += prime
                total_iteration += 1

        #print(number_list)
        #print(prime_list)
        current += 1

    return prime_list

def divisor_finder(number, primes):
    upper_limit = number
    #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
    i = 0
    while primes[i] < sqrt_limit and i < len(primes):
        while current_upper_limit % primes[i] == 0:
            current_upper_limit = current_upper_limit/primes[i]
            prime_factors += [primes[i]]
        i += 1

    if current_upper_limit != 1:
        prime_factors += [current_upper_limit]

    #print(prime_factors)
```

```
    prime_counts = [[prime_factors[0], prime_factors.count(prime_factors[0])]]
    #print(prime_counts,prime_counts[-1][0],prime_factors[1])
    for i in range(1,len(prime_factors)):
        #print(prime_counts[-1][0],prime_factors[i])
        if  prime_counts[-1][0] != prime_factors[i]:
            prime_counts += [[prime_factors[i], prime_factors.count(prime_factors[i])]]

    #print(prime_counts)

    divisors = [1]
    counters = [x[1] for x in prime_counts]
    #print(counters)
    while sum(counters) > 0:
        new_divisor = 1
        for i in range(len(counters)):
            #print(prime_counts[i][0],counters[i], new_divisor)
            new_divisor *= prime_counts[i][0]**counters[i]
        divisors += [new_divisor]

        counters[0] -= 1
        for i in range(len(counters)):
            if counters[i] == -1:
                counters[i] = prime_counts[i][1]
                if i != len(counters) - 1:
                    counters[i+1] -= 1
            else:
                break

    divisors.sort()
    return divisors[:-1]

upper_limit = 28124

prime_list = generate_primes(upper_limit)

abundant_nums = []
for i in range(12, upper_limit-12):
    divisors = divisor_finder(i, prime_list)
```

```
    if sum(divisors) > i:
        #print(i,sum(divisors), divisors)
        abundant_nums += [i]

print(abundant_nums)

abundant_sum_nums = []
for num1 in abundant_nums:
    for num2 in abundant_nums:
        number = num1 + num2
        if number < upper_limit:
            abundant_sum_nums += [number]
        else:
            break

print(len(abundant_nums))
remove = set(abundant_sum_nums)
keep = [x for x in range(1,upper_limit)]
for number in remove:
    keep.remove(number)

print(len(keep))
```

Revision #2
Created 2025-12-29 20:54:28 UTC by Maxwell
Updated 2025-12-29 20:55:34 UTC by Maxwell