Goldbach's Second Conjecture

locked

by *IEEEXtreme*

We want to find three primes that sum up to *N*, where *N* is an odd integer greater than 5. The first insight towards solving this problem is to realize that we can reduce Goldbach's second conjecture on three primes to Goldbach's original conjecture on two primes. If we take any *odd* prime *p* < *N-2* and assume that is the first prime in our answer, then what remains is to find two primes that sum to *N' = N-p*. And since *N'* > 2 and *N'* is even, we now have an instance of Goldbach's original conjecture. By searching for Goldbach's conjecture on the internet, we can see that the conjecture has been proven for all integers below 4×10^{18}. Assuming we can find such a *p*, we can conclude that there is always an answer, and we don't have to worry about printing "counterexample". And indeed it is easy to see that *p=3* always works given the lower bound on *N*.

Now, the hardest part about this problem is the magnitude of *N*. If it were much smaller we could either use brute force or dynamic programming to find these three primes. But we can use our above reduction to Goldbach's conjecture. In particular, notice that we're not assuming much about which prime *p* we pick. So if we pick a very large *p*, then the remaining *N' = N-p* will be small. To make *N'* as small as possible, we can pick the largest prime below *N-2*. But how far below *N-2* is the next prime? Searching the internet for prime gaps reveals that the distance between consecutive primes is a bit less than 2000 when dealing with numbers below 10^{18}. Thus *N'* will be less than 2000, which makes a brute force approach for finding the remaining two primes sufficient.

Now the last remaining part is to quickly find the largest prime below *N-2*. Since the distance from *N-2* down to the next prime isn't that large, we can simply try the numbers *N-3*, *N-4*, *N-5*, ..., until we find a prime. But these numbers can be really large, on the order of 10^{18}. Even an optimized trial division won't be quick enough. Instead we can use a probabilistic primality test such as the Miller-Rabin primality test. Note that, although unnecessary for this problem, this test can be made deterministic when *N* is smaller than 10^{18}.

Speaking of randomization, we will conclude this editorial with a pseudocode of a very randomized solution to this problem, which has an expected running time of *O(log(N)^3)*:

```
n = input()
while True:
    a = random(1,n)
    b = random(1,n)
    c = n - a - b
    if is_probably_prime(a) and is_probably_prime(b) and is_probably_prime(c):
        print a, b, c
        break
```

Statistics

Difficulty: Hard
Publish Date: Aug 05 2016