A proof of the $O(n)$ time complexity of Euler's Prime-Sieving Algorithm

First, we present the code for the standard Euler's Prime-Sieving Algorithm:

```
bool isprime[maxn];
int primes[maxn];
int cnt = 0;

void euler(int n) {
    memset(isprime, true, sizeof(isprime));
    isprime[1] = false;
    for (int i = 2; i <= n; i++) {
        if(isprime[i]) {
            primes[++cnt] = i;
        }
        for (int j = 1; j <= cnt && i * primes[j] <= n; j++) {
            isprime[i * primes[j]] = false;
            if (i % primes[j] == 0) {
                break;
            }
        }
    }
}
```

Clearly, the time complexity of the algorithm is at least $O(n)$. The part of the code that determines the final time complexity is the inner loop about the variable `j`.

Define "visit" as marking a number with `isprime` using the inner loop. It is clear that the overall time complexity is the same as the number of times the inner loop is iterated; that is, the number of "visits" that occur in total.

Claim: The number of visits made to each integer from 2 to n is at most 1.

Proof:

If the number i is itself a prime, then it can only be visited by the outer loop, and cannot be visited by the inner loop. This is because to be visited by the inner loop, the number has to be decomposed into the product of two positive integers, namely `i` and `primes[j]`, which according to the code are both greater than 1. Hence, if a number is prime in the range $[1, n]$, it is not visited by the inner loop. That is, the number of visits to the primes is 0.

Now we show that for a composite integer m, it is visited at most once by the inner loop. Suppose that on the contrary this is not the case. Then, there must exist two different prime numbers a and c such that m is visited by both `isprime[b * a]` and `isprime[d * c]` in the inner loop, where $b = \frac{m}{a}$ and $d = \frac{m}{c}$ are integer factors of m. W.L.O.G., we can assume $a < c$, which implies $b > d$, as $ba = m = dc$. Since $d < b$, i iterates to d before it iterates to b. Consider the situation where $i = d$. Since m is updated by `isprime[d * c]`, the variable `j` must iterate to a value such that `primes[j]` is equal to c. However, because $a < c$, the prime a will be visited as a value of `primes[j]` before prime c. Looking back at the equation $ba = m = dc$, note that, since $a \nmid c$ and a, c are prime, by Euclid's Lemma we must have that $a \mid d$, or in C++ code, `i % primes[j] == 0`. This means the inner loop will break when `primes[j]` reaches a, and will therefore not reach c at all. This apparently contradicts with our definition of a, b, c, d.

Therefore, for the composite integer m, it is visited at most once by the inner loop.

Since a positive integers greater than 1 is either prime or composite, it is clear from above that in total, the number of visits made to the numbers in the range 2 to n (inclusive) by Euler's Prime-Sieving Algirthm is at least 1. $\square$