Gems from the world of data structures and algorithms

Enumerating all primes less than n in $O(n)$ time

Difficulty level: *high*

The following algorithm also called "Sieve of Eratosthenes" algorithm for computing all prime numbers in range [1..n] is very well known.

Keep a boolean array $A[1..n]$ such that $A[i] = 1$ initially but $A[i] = 1$ at the end only if i is a prime number and zero otherwise.

```
Eratosthenes(n) {
A[1] ← 0
p ← 2
While p^2 < n do {
        j ← p^2
        while (j < n) do {
                A[j] ← 0
                j ← j + p
        }
        repeat p ← p + 1 until A[p] = 1
}
return(A)
}
```

The running time of the algorithm can be bounded as follows : Let us first count the number of basic (arithmetic, logical and assignment) operations. In each iteration of the outer while loop, we select a prime p and then cancel all numbers less than n which can be expressed as $p * x, x \geq p$. Convince yourself that each non prime number will get canceled in this way. So for prime number 2 the numbers which get canceled are $n/2$, for number 3 the numbers which get canceled are $n/3$, and so on. So the total number of these cancellations are

$$\frac{n}{2} + \frac{n}{3} + \frac{n}{5} + \frac{n}{7} + \frac{n}{11} + ... \quad = \quad n \sum_{p \leq \sqrt{n},\ p \text{ is prime}} \frac{1}{p} \quad = \quad \Theta(n \log\log n)$$

Here we are using the following Lemma whose proof is beyond the scope of this course (do not be afraid to see this Lemma. What you are supposed to do has got nothing to do with it :-)

Lemma 1. *The series* $\sum_{p \leq \sqrt{n}, p \text{ is prime}} \frac{1}{p}$ *approaches to* $\log\log n$ *asymptotically.*

Hence the total number of basic arithmetic, logical and assignment operations is $O(n \log\log n)$. Assuming each such operation takes $O(1)$ time, it follows that the time complexity of the above algorithm is $O(n \log\log n)$.

Challenge :
You need to design an algorithm for computing all prime numbers in the interval $[1..n]$. Your algorithm must have running time $O(n)$.
Hint :
You need to somehow modify the algorithm given above and use a careful data structure so that the number of cancellations reduce from $n \log \log n$ to n. This goal can be achieved if you make sure that a non-prime number is canceled only once. How will you achieve this goal? All you need is a few cute observations and then a simple data structure to materialize the idea based on these observations. Give complete description of the algorithm and the data structure and prove that your algorithm cancels any non prime number exactly once.

every art is beautiful and so is the art of algorithm design ...