**Program 1**

**Sought:** find all primes between 1 and 108 using 8 threads.

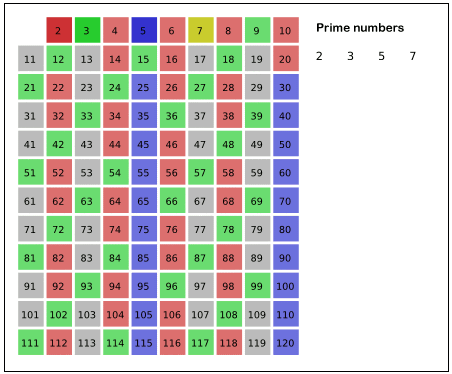
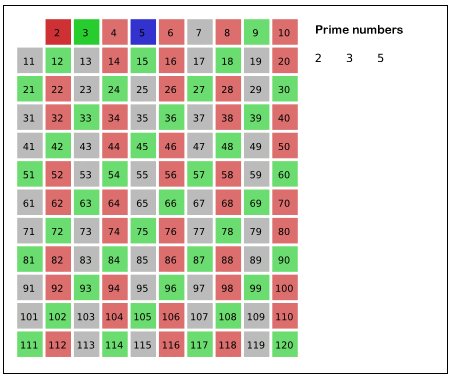
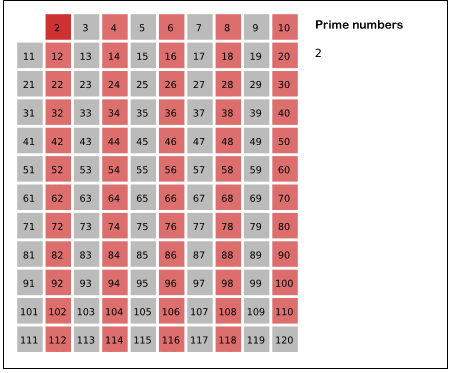
**Solution:**

I searched on the web to find a fast algorithm for finding prime numbers. Apparently, the sieve of Eratosthenes is one simple and efficient algorithm for finding all prime numbers up to any given limit.

To find all the prime numbers less than or equal to a given integer *n* by Eratosthenes' method:

1. Create a list of consecutive integers from 2 through *n*: (2, 3, 4, ..., *n*).
2. Initially, let *p* equal 2, the first prime number.
3. Starting from *p*, enumerate its multiples by counting to *n* in increments of *p*, and mark them in the list (these will be 2*p*, 3*p*, 4*p*, ... ; the *p* itself should not be marked).
4. Find the first number greater than *p* in the list that is not marked. If there was no such number, stop. Otherwise, let *p* now equal this new number (which is the next prime), and repeat from step 3.

When the algorithm terminates, the numbers remaining not marked in the list are all the primes below *n*. The main idea here is that every value for p is prime, because we have already marked all the multiples of the numbers less than p. Note that some of the numbers being marked may have already been marked earlier (e.g., 15 will be marked both for 3 and 5).



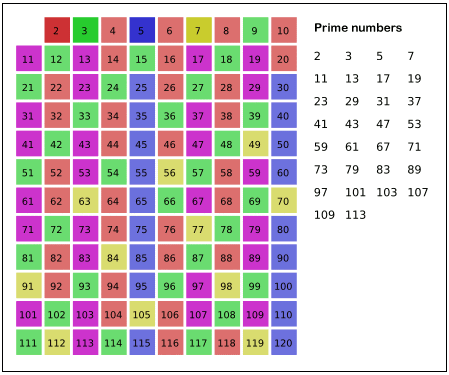


Figure Sieve of Eratosthenes algorithm steps for prime below 121

In order to benefit from multithread programming feature, three major computation areas have been parallelized using provided facility by OpenMP. In particular, the #pragma omp parallel has been utilized in the first round of algorithm to provide a private copy of the variable for each thread. These threads execute the code independently considering the highest rate of safety for guarantying the progress and correctness. Then, they return the results to the main thread. I also used the reduction clause which is a mix between the private, shared, and atomic clauses. It allows to accumulate a shared, but the type of accumulation must be specified. The reduction clause can be used to produce semantically different code when OpenMP is enabled.

**Efficiency:**

This algorithm produces all primes less than n. It includes a common optimization, which is to start looking for multiples of each prime i at i². The complexity of this algorithm is O(n log log n) [2].

**Experimental evaluation:**

The average time for executing sieve of Eratosthenes algorithm using sequential program was around 3.2 seconds while this run-time can be reduced to 1.4 seconds if eight threads utilized for finding prime numbers between 1 to 108. The program output has been verified by the expected value reported in [3] and also tested for other range of numbers.

**How to Compile the Code:**

I used Code::Blocks 13.12 tool to debug and compile my program. In order to enable OpenMP option in Code::Blocks, please follow below steps:

1. Enable C++ISO C++ language standard [std=c++11]
   1. Go to Toolbar -> Settings -> Compiler
   2. In the "Selected compiler" drop-down menu, make sure "GNU GCC Compiler" is selected
   3. Below that, select the "compiler settings" tab and then the "compiler flags" tab underneath
   4. In the list below, make sure the box for "Have g++ follow the C++11 ISO C++ language standard [-std=c++11]" is checked
   5. Click OK to save
2. add "-fopenmp" to "Settings -> Compiler ... -> Compiler setting tab -> Other options"
3. add “gomp” to "Settings -> Compiler … -> Linker setting

**References:**

[1] Wikipedia

[2] Jonathan Sorenson, An Introduction to Prime Number Sieves, Computer Sciences Technical Report #909, Department of Computer Sciences University of Wisconsin-Madison, January 2, 1990 (the use of optimization of starting from squares, and thus using only the numbers whose square is below the upper limit, is shown).

[3] https://primes.utm.edu/howmany.html