Sieve of Eratosthenes

Dr. Minal Moharir, CSE, RVCE

Sieve of Eratosthenes

- In mathematics, the **sieve of Eratosthenes** is an ancient algorithm for finding all prime numbers up to any given limit.

- It does so by iteratively marking as composite (i.e., not prime) the multiples of each prime, starting with the first prime number, 2.

Overview

- A prime number is a natural number that has exactly two distinct natural number divisors: the number 1 and itself.

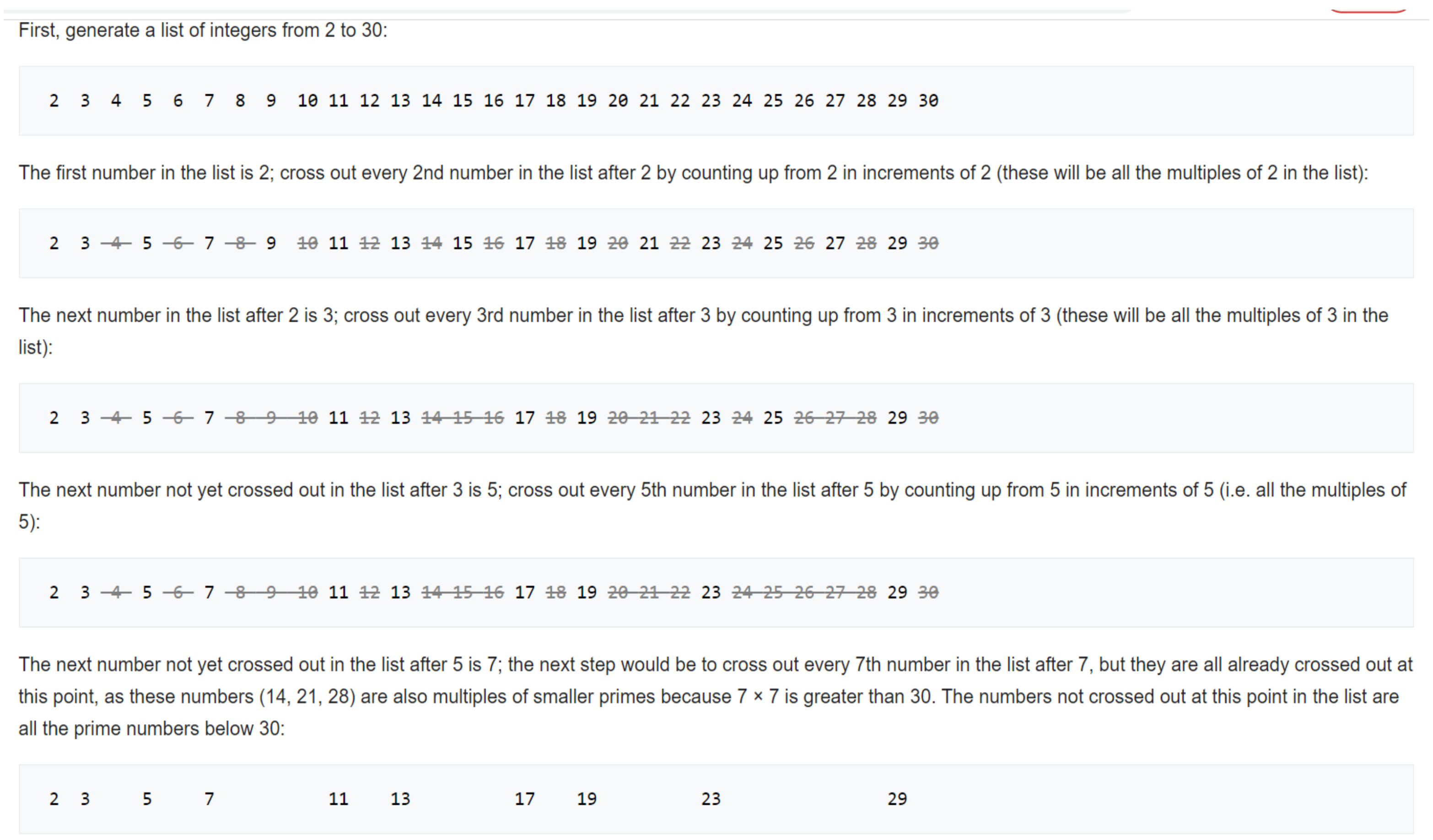
Overview

- To find all the prime numbers less than or equal to a given integer *n* by Eratosthenes' method:
- Create a list of consecutive integers from 2 through *n*: (2, 3, 4, ..., *n*).
- Initially, let *p* equal 2, the smallest prime number.
- Enumerate the multiples of *p* by counting in increments of *p* from 2*p* to *n*, and mark them in the list (these will be 2*p*, 3*p*, 4*p*, ...; the *p* itself should not be marked).
- Find the smallest number in the list greater than *p* 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*.

Example

- To find all the prime numbers less than or equal to 30, proceed as follows.
- First, generate a list of integers from 2 to 30:

Example



First, generate a list of integers from 2 to 30:

2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30

The first number in the list is 2; cross out every 2nd number in the list after 2 by counting up from 2 in increments of 2 (these will be all the multiples of 2 in the list):

2 3 ~~4~~ 5 ~~6~~ 7 ~~8~~ 9 ~~10~~ 11 ~~12~~ 13 ~~14~~ 15 ~~16~~ 17 ~~18~~ 19 ~~20~~ 21 ~~22~~ 23 ~~24~~ 25 ~~26~~ 27 ~~28~~ 29 ~~30~~

The next number in the list after 2 is 3; cross out every 3rd number in the list after 3 by counting up from 3 in increments of 3 (these will be all the multiples of 3 in the list):

2 3 ~~4~~ 5 ~~6~~ 7 ~~8~~ ~~9~~ ~~10~~ 11 ~~12~~ 13 ~~14~~ ~~15~~ ~~16~~ 17 ~~18~~ 19 ~~20~~ ~~21~~ ~~22~~ 23 ~~24~~ 25 ~~26~~ ~~27~~ ~~28~~ 29 ~~30~~

The next number not yet crossed out in the list after 3 is 5; cross out every 5th number in the list after 5 by counting up from 5 in increments of 5 (i.e. all the multiples of 5):

2 3 ~~4~~ 5 ~~6~~ 7 ~~8~~ ~~9~~ ~~10~~ 11 ~~12~~ 13 ~~14~~ ~~15~~ ~~16~~ 17 ~~18~~ 19 ~~20~~ ~~21~~ ~~22~~ 23 ~~24~~ ~~25~~ ~~26~~ ~~27~~ ~~28~~ 29 ~~30~~

The next number not yet crossed out in the list after 5 is 7; the next step would be to cross out every 7th number in the list after 7, but they are all already crossed out at this point, as these numbers (14, 21, 28) are also multiples of smaller primes because 7 × 7 is greater than 30. The numbers not crossed out at this point in the list are all the prime numbers below 30:

2 3 5 7 11 13 17 19 23 29

Segmented sieve (Cache Friendly)

- As Sorenson notes, the problem with the sieve of Eratosthenes is not the number of operations it performs but rather its memory requirements.
- For large *n*, the range of primes may not fit in memory; worse, even for moderate *n*, its cache use is highly suboptimal.
- The algorithm walks through the entire array *A*, exhibiting almost no locality of reference.

Segmented sieve (Cache Friendly)

- A solution to these problems is offered by *segmented* sieves, where only portions of the range are sieved at a time.
- Divide the range 2 through *n* into segments of some size $\Delta \leq \sqrt{n}$.
- Find the primes in the first (i.e. the lowest) segment, using the regular sieve.
- For each of the following segments, in increasing order, with *m* being the segment's topmost value, find the primes in it as follows:
  - Set up a Boolean array of size Δ, and
  - Mark as non-prime the positions in the array corresponding to the multiples of each prime $p \leq \sqrt{m}$ found so far, by calculating the lowest multiple of *p* between $m - \Delta$ and *m*, and enumerating its multiples in steps of *p* as usual. The remaining positions correspond to the primes in the segment, since the square of a prime in the segment is not in the segment (for $k \geq 1$, one has {\displaystyle (k\Delta +1)^{2}>(k+1)\Delta }).
- If Δ is chosen to be $\sqrt{n}$, the space complexity of the algorithm is $O(\sqrt{n})$,