Sieve of Eratosthenes

(another Prime Pages' Glossary entries)

Glossary:

- Index
- Search
- Home

- Previous
- Next
- Random

- Mail Editor

Prime Pages:

- Home
- Search
- Index
- FAQ

Top 5000:

- Primes
- Provers
- Curios

The most efficient way to find all of the small primes (say all those less than 10,000,000) is by using a sieve such as **the Sieve of Eratosthenes** (ca 240 BC):

> Make a list of all the integers less than or equal to *n* (and greater than one). Strike out the multiples of all primes less than or equal to the square root of *n*, then the numbers that are left are the primes.

For example, to find all the primes less than or equal to 30, first list the numbers 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 2 is prime, so keep it (we will color it green) and cross out its multiples (we will color them red), so the red numbers are not prime.

> 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 left (still black) is 3, so it is the first odd prime. Keep it and cross out all of its multiples. We know that all multiples less than 9 (i.e. 6) will already have been crossed out, so we can start crossing out at $3^2=9$.

> 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

Now the first number left (still black) is 5, the second odd prime. So keep it also and cross out all of its multiples (all multiples less than $5^2=25$ have already been crossed out, and in fact 25 is the only multiple not yet crossed out).

> 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 left, 7, is larger than the square root of 30, so there are no multiples of 7 to cross off that haven't already

been crossed off (14 and 28 by 2, and 21 by 3), and therefore the sieve is complete. Therefore all of the numbers left are primes: {2, 3, 5, 7, 11, 13, 17, 19, 23, 29}. Notice we just found these primes without dividing.

This algorithm can be written in pseudo-code as follows

```
Eratosthenes(n) {
  a[1] := 0
  for i := 2 to n do a[i] := 1
  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)
}
```

This method is so fast that there is no reason to store a large list of primes on a computer--an efficient implementation can find them faster than a computer can read from a disk. In fact the problem with the algorithm as presented above is not really speed (it uses O(n(log n)log log n) bit operations), but rather space (it is O(n)). So for large n we usually use a segmented sieve. However, both the time and space theoretically can be improved; for example, Pritchard's "linear segmented wheel sieve" uses O(n log n) bit operations and O(sqrt(n)/log log n) space.

See Also: TrialDivision

Related pages (outside of this work)

- Programs and pseudocode for The Sieve of Eratosthenes

References:

BH77
C. Bayes and **R. Hudson**, "The segmented sieve of Eratosthenes and primes in arithmetic progression," *Nordisk Tidskr. Informationsbehandling (BIT)*, **17**:2 (1977) 121--127. **MR 56:5405**

Bressoud89
D. M. Bressoud, *Factorizations and primality*

testing, Springer-Verlag, New York, NY, 1989. ISBN 0387970401. **MR 91e:11150** [pseudocode implementation on page 19] [QA161.F3B73]

Pritchard87

P. Pritchard, "Linear prime-number sieves: a family tree," *Sci. Comput. Programming*, **9**:1 (1987) 17--35. **MR 88j:11087** [Comparison of various sieves] [A comparison of recent sieves such as the sieve of Eratosthenes.]

Riesel94

H. Riesel, *Prime numbers and computer methods for factorization*, Progress in Mathematics Vol, 126, Birkhäuser Boston, Boston, MA, 1994. ISBN 0-8176-3743-5. **MR 95h:11142** [a PASCAL implementation on page 6] [An excellent reference for those who want to start to program some of these algorithms. Code is provided in Pascal. Previous edition was vol. 57, 1985.]
