Prime Number Generation: the Sieve of Eratosthenes

The Sieve of Eratosthenes is touted as one of the fastest method of determining prime numbers below 10^7. The Sieve of Eratosthenes was developed around 240 BC. (Visit http://primes.utm.edu for more information.) The sieve operates in this fashion:

1. Make a list of all the integers less than or equal to n (and greater than one).
2. Strike out the multiples of all primes less than or equal to the square root of n.
3. The remaining numbers are the primes.

The following program generates all the prime numbers from one to *n*.
The program on page 3 is optimized for evaluating large data sets and arbitrary starting points.

```
Sieve_one(n) :=
  for i ∈ 0 .. n                                  prepares an array from 1 to n
    data_i ← i
  j ← 2
  step ← 2
  while step ≤ floor(√n)                          removes the non-prime numbers
    step ← | while data_j = 0                     determines the next prime number
           |   j ← j + 1                          i.e., the first nonzero number
           | data_j
    k ← 2 · step
    while k ≤ n                                   replaces all non-prime numbers
      data_k ← 0                                  with 0
      k ← k + step
    j ← j + 1
  data ← sort(data)                               returns the nonzero numbers (primes)
  i ← 0
  while data_i = 0
    i ← i + 1
  submatrix(data, i, n, 0, 0)
```

n := 100

Primes := $Sieve_{one}(n)$

Primes =

	0
0	1
1	2
2	3
3	5
4	7
5	11
6	13
7	17
8	19
9	23
10	29
11	31
12	37
13	41
14	43
15	47

```
filename := concat("Primes up to ", num2str(n), ".dat")
filename = "Primes up to 100.dat"
WRITEPRN(filename) := Primes
```


Determining the prime numbers in an arbitrary range is only slightly more difficult. We need to determine the primes from one to the maximum of the data range. This is accomplished by calling $Sieve_{one}$.

```
Sieve_any(data) :=
    start ← data_0
    n ← rows(data) − 1
    end ← data_n
    primes ← Sieve_one(√end)                    obtains the necessary prime numbers
    for i ∈ 1 .. rows(primes) − 1               removes the non-prime numbers
                                                i.e., multiples of the primes.
        step ← primes_i
        j ← max(ceil(start / step), 2) · step − start      determines the start position
                                                           in the array
        while j ≤ n                             replaces all non-prime numbers
                                                with 0
            data_j ← 0
            j ← j + step
    data ← sort(data)                           returns the nonzero numbers (primes)
    i ← 0
    while data_i = 0
        i ← i + 1
    submatrix(data, i, n, 0, 0)
```

```
range :=
    for i ∈ 0 .. 50
        temp_i ← i + 24
    temp
```

range =

	0
0	24
1	25
2	26
3	27
4	28
5	29
6	30
7	31
8	32
9	33
10	34
11	35
12	36
13	37
14	38
15	39

$Sieve_{any}(range)$ =

	0
0	29
1	31
2	37
3	41
4	43
5	47
6	53
7	59
8	61
9	67
10	71
11	73

The array-size can slow the processing of large ranges. This routine breaks the job up into smaller pieces of up to $5 \cdot 10^5$ test numbers. Each set is sent to Sieve_{any} in order.

The routine aborts when the prime number array reaches $7.8 \cdot 10^6$ (MathCads array limit is $8 \cdot 10^6$ elements

```
Sieve(initial, range) :=
    number ← ceil(range / 5·10^5)                    determines the number of sub-arrays
    size ← ceil(range / number)                      determines the size of each sub-array
    primes ← 0
    for i ∈ 0 .. number − 1
        data ←  temp ← 0                             generates the sub-arrays
                for j ∈ 0 .. size − 1
                    temp_j ← i · size + j + initial
                temp
        temp2 ← Sieve_any(data)                      determines the primes in each sub-array
        primes ← stack(primes, temp2)                and combine the primes returned
        break if rows(primes) > 7.8 · 10^6
    submatrix(primes, 1, rows(primes) − 1, 0, 0)
```

initial := 0 initial number

range := $1 \cdot 10^8$ range of numbers to test

Primes := Sieve(initial, range)

$\text{Primes}_0 = 1$ first prime number

$\text{Primes}_{\text{last(Primes)}-1} = 99999971$ last prime number

$\text{rows(Primes)} - 1 = 5761455$ number of primes in range

Primes =

	0
0	1
1	2
2	3
3	5
4	7
5	11
6	13
7	17
8	19
9	23
10	29
11	31
12	37
13	41
14	43
15	47

```
filename := concat("Primes between ", num2str(initial), " and ", num2str(initial + range),
filename = "Primes between 0 and 100000000.dat"
WRITEPRN(filename) := Primes
```


Distribution of Primes

Primes := READPRN("<filename>")

load a pre-saved datafile instead of generating them with Seive.

bins := 100 i := 1 .. bins j := 1 .. bins – 1

$$space := \frac{Primes_{last(Primes)}}{bins} \qquad interval_i := space \cdot \left(i - \frac{1}{2}\right)$$

f := hist(interval , Primes)