SIEVE USING WHEEL FACTORIZATION AND APPLICATIONS

V. Barbera

Abstract

This paper presents the use of a specific wheel factorization sieve algorithm in some applications.

Sieve using wheel factorization

A refinement of the sieve of Eratosthenes[1] is using the wheel factorization[2] with the basis $\{2, 3\}$ to find the prime numbers $p>3$ less than a given value of n.

The sieve is performed on two boolean vectors $Primes5mod6$ and $Primes1mod6$ where for each index $i\geq 1$ the value of $Primes5mod6$[i] is *True* if the number $-1+6\cdot i$ is prime or *False* otherwise, similarly for the value of $Primes1mod6$[i] in relation to the number $1+6\cdot i$

$$\begin{array}{rcccccccc} & (5 & 11 & 17 & 23 & 29 & 35 & \cdots) \\ Primes5mod6= & [1 & 1 & 1 & 1 & 1 & 0 & \cdots] \\ i & 1 & 2 & 3 & 4 & 5 & 6 & \cdots \end{array} \qquad \begin{array}{rcccccccc} & (7 & 13 & 19 & 25 & 31 & 37 & \cdots) \\ Primes1mod6= & [1 & 1 & 1 & 0 & 1 & 1 & \cdots] \\ i & 1 & 2 & 3 & 4 & 5 & 6 & \cdots \end{array}$$

Starting from two boolean vectors of size $\lfloor n/6 \rfloor$, initialized to *True*, this algorithm reset the elements corresponding to multiples of each prime $p\leq\lfloor\sqrt{n}\rfloor$. In both vectors, starting from $i=1$, if the element [i] is *True*, the value of the subsequent elements are reset from index corresponding to number $(1+6\cdot i)\cdot(-1+6\cdot i)$ for $Primes5mod6$ and from index corresponding to number $(\pm 1+6\cdot i)^2$ for $Primes1mod6$ increasing by $\pm 1+6\cdot i$ therefore:

- for vector $Primes5mod6$, considering that $(1+6\cdot i)\cdot(-1+6\cdot i)=-1+6\cdot(6\cdot i^2)$, elements are reset from indices $6\cdot i^2$ and subsequent increments of $\pm 1+6\cdot i$
- for vector $Primes1mod6$ considering that $(-1+6\cdot i)^2=1+6\cdot(6\cdot i^2-2\cdot i)$ elements are reset from indices $6\cdot i^2-2\cdot i$ and increments of $-1+6\cdot i$ and that $(1+6\cdot i)^2=1+6\cdot(6\cdot i^2+2\cdot i)$ elements are reset from indices $6\cdot i^2+2\cdot i$ and subsequent increments of $1+6\cdot i$

Python implementation of the algorithm

```
def sieve(n):
    Primes5mod6 = [True] * (n//6+1)
    Primes1mod6 = [True] * (n//6+1)
    for i in range(1,int((n**0.5+1)/6)+1):
        if Primes5mod6[i]:
            Primes5mod6[6*i*i::6*i-1]=[False]*((n//6-6*i*i)//(6*i-1)+1)
            Primes1mod6[6*i*i-2*i::6*i-1]=[False]*((n//6-6*i*i+2*i)//(6*i-1)+1)
        if Primes1mod6[i]:
            Primes5mod6[6*i*i::6*i+1]=[False]*((n//6-6*i*i)//(6*i+1)+1)
            Primes1mod6[6*i*i+2*i::6*i+1]=[False]*((n//6-6*i*i-2*i)//(6*i+1)+1)
    return Primes5mod6,Primes1mod6
```

Goldbach function

To find the value of the Goldbach function[1] $g(n)$ for even integers $n>2$ is sufficient to count the non-zero values of the vector *[A & B]* where A and B are two vectors obtained starting from the two vectors $Primes5 mod 6$ *and* $Primes1 mod 6$ as follows:

Case I $n/2 \equiv 0 (mod\ 3)$

$n=6\cdot k=(-1+6\cdot k_1)+(1+6\cdot k_2)$ with k_1 *from 1 to (k-1)* and k_2 *from (k-1) to 1 step -1*

$A=Primes\,5\,mod\,6[1:k-1]$ and $B=Primes\,1\,mod\,6[k-1:1:\ step\ -1]$

Case II $n/2 \equiv 1 (mod\ 3)$

$n=2+6\cdot k=(1+6\cdot k_1)+(1+6\cdot k_2)$ with k_1 *from 1 to* $\lfloor k/2 \rfloor$ and k_2 *from (k-1) to* $\lceil k/2 \rceil$ *step* -1

$A=Primes\,1\,mod\,6[1:\lfloor k/2 \rfloor]$ and $B=Primes\,1\,mod\,6[k-1:\lceil k/2 \rceil:\ step\ -1]$

or

$n=(-1+6\cdot k)+3$ therefore *[A & B]=Primes5mod6[k]*

Case III $n/2 \equiv 2 (mod\ 3)$

$n=4+6\cdot k=-2+6\cdot(k+1)=(-1+6\cdot k_1)+(-1+6\cdot k_2)$ with k_1 *from* 1 *to* $\lfloor (k+1)/2 \rfloor$ and k_2 *from k to* $\lceil (k+1)/2 \rceil$ *step* -1

$A=Primes\,5\,mod\,6[1:\lfloor (k+1)/2 \rfloor]$ and $B=Primes\,5\,mod\,6[k:\lceil (k+1)/2 \rceil:\ step\ -1]$

or

$n=(1+6\cdot k)+3$ therefore *[A & B]=Primes1mod6[k]*

Python implementation using numpy

goldbach function returns vector G with $G[i]=g(2\cdot i)$ for $0 \le i < n/2$

```
import math
import numpy as np
def goldbach(n):
    Primes5mod6 =np.ones((n//6+1), dtype=bool)
    Primes1mod6 =np.ones((n//6+1), dtype=bool)
    for i in range(1,math.isqrt(n)//6+1):
        if Primes5mod6[i]:
            Primes5mod6[6*i*i::6*i-1]=[False]
            Primes1mod6[6*i*i-2*i::6*i-1]=[False]
        if Primes1mod6[i]:
            Primes5mod6[6*i*i::6*i+1]=[False]
            Primes1mod6[6*i*i+2*i::6*i+1]=[False]

    G=[0,0,1,1,1,2]
    for ni in range(12,n-3,6):
        k=ni//6
        G.append(np.count_nonzero(Primes5mod6[1:k]&Primes1mod6[k-1:0:-1]))
        G.append(np.count_nonzero(Primes1mod6[1:math.floor(k/2)+1]&Primes1mod6[k-1:math.ceil(k/2)-1:-1])
+Primes5mod6[k])
        G.append(np.count_nonzero(Primes5mod6[1:math.floor((k+1)/2)+1]&Primes5mod6[k:math.ceil((k+1)/2)-
1:-1])+Primes1mod6[k])
    return G
```

Twin prime

Starting from the two vectors obtained from the previous sieve, to find indices i of twin prime pairs $(-1+6\cdot i\ ,\ 1+6\cdot i)$ less than n we must consider the indices of the elements with value $True$ of the vector $[Primes5 mod 6\ \&\ Primes1 mod 6]$.

Python implementation

```
def twin_prime(n):
    A,B=sieve(n)
    print("(3  ,  5)")
    for i in range(1,n//6+1):
        if A[i] and B[i]:
            print("({:d}  ,  {:d})".format(6*i-1, 6*i+1))
```

References

[1] https://en.wikipedia.org/wiki/Sieve_of_Eratosthenes

[2] https://en.wikipedia.org/wiki/Wheel_factorization

[3] https://en.wikipedia.org/wiki/Goldbach%27s_comet