```
# -*- coding: utf-8 -*-

import math
from math import log
import numpy as np
import  matplotlib.pyplot as plt
import time

def prime_sieve(N):
    is_prime = np.full(N, True)
    is_prime[:2] = False
    for p in range(2, math.isqrt(N) + 1):
        if is_prime[p]:
            is_prime[p*p::p] = False
    return np.nonzero(is_prime)[0]

class Primes():
    def __init__(self, N):
        self.__primes = prime_sieve(N)
    def __str__(self):
        return str(self.__primes)
    def __iter__(self):
        return iter(self.__primes)
    def __len__(self):
        return self.__primes.size
    def __getitem__(self, k):
        return self.__primes[k]
    def __contains__(self, x):
        k = self.index(x)
        return k < len(self) and self.__primes[k] == x
    def index(self, x):
        return np.searchsorted(self.__primes, x, side='left')
    def count(self, x):
        return np.searchsorted(self.__primes, x, side='right')
    def range(self, start, stop, step=1):
        return self.__primes[self.index(start):self.index(stop):step]
    def factors(self, n):
        if n in self:
            return np.array([n])
        else:
            P = self.range(2, n//2 + 1)
            return P[n % P == 0]

def gb_count(N):
    P = Primes(N)
    G = np.zeros(N, int)
    if N > 4: G[4] = 1
    for p in P.range(3, N/2):
        G[p + P.range(p, N - p)] += 1
    return G

def hl_count(N):
    P = Primes(N)
    H = np.zeros(N)
    C2 = 0.660161815846869573927812110014
    for n in range(2, N//2):
        Q = P.factors(n)
        if Q[0] == 2: Q = Q[1:] # odd prime factors of n
        H[2*n] = 2*C2 * n/np.log(n)**2 * np.prod(1 + 1/(Q - 2))
    return H

def minidenise(x):
    return((((1+5**0.5)/(2*(5**0.5))))*(x/(log(x)*log(x))))
```

```
N = 10002
P = Primes(N)
HL = hl_count(N)
les6kplus1 = P[np.mod(P, 6) == 1]
les6kmoins1 = P[np.mod(P, 6) == 5]
fig, ax = plt.subplots(figsize=(15, 10))
ax.set_title("Goldbach's comet")
t0 = time.perf_counter()
G = gb_count(N)
dt = time.perf_counter() - t0
print(f"[{dt:5.2f} s] Golbach's conjecture is {all(G[4::2] >= 1)} for even
numbers < {N:_}.")
n = np.arange(0, N, 2)
ax.scatter(n, G[n], s=4, alpha=0.4)
plt.show()
print('nombre de decompositions de Goldbach des nombres de 6 à 102')
for x in range(6,104,2):
    print(x,' --> ',G[x])

# for k in
[6,8,12,38,68,98,128,152,326,332,398,488,632,668,692,992,1112,1412,1718,2048,225
2,2642,2672,2936,4412,5468,5948,7508,8042,8048,8552,9602]:
nombredepairs = 0
nombrederatagesdenise = 0
nombrederatagesHardyLittlewood = 0
for k in range(6,10002,2):
    nombredepairs += 1
    print(k,' --> ',minidenise(k),'  ',G[k],'  ',HL[k])
    if minidenise(k) > G[k]:
        print('rate denise')
        nombrederatagesdenise += 1
    if HL[k] > G[k]:
        print('rate Hardy-Littlewood')
        nombrederatagesHardyLittlewood += 1
print('ratages denise = ',nombrederatagesdenise/nombredepairs)
print('ratages Hardy-Littlewood =
',nombrederatagesHardyLittlewood/nombredepairs)
```