```
import math
import numpy as np
from numba import njit
import time
import matplotlib.pyplot as plt
import matplotlib.colors as mcolors

@njit(cache=True)
def crible_numba(N):
    est_p = np.ones(N+1, dtype=np.bool_)
    est_p[0] = False
    est_p[1] = False
    i = 2
    while i*i <= N:
        if est_p[i]:
            j = i*i
            while j <= N:
                est_p[j] = False
                j += i
        i += 1
    return est_p

@njit(cache=True, parallel=False)
def bosse(N, est_p, k):
    liste = []
    for n in range(6, N+1, 2):
        if n % 105 == k:
            G_n = 0
            for p in range(3, n//2 + 1, 2):
                if est_p[p] and est_p[n-p]:
                    G_n += 1
            liste.append(G_n)
    return liste

tic = time.time()
N = 10000
lescouleurs = ['AliceBlue' , 'AntiqueWhite' , 'aqua' , 'aquamarine' , 'azure' ,
'beige' , 'bisque' , 'black' , 'BlanchedAlmond' , 'blue' , 'BlueViolet' ,
'brown' , 'burlywood' , 'CadetBlue' , 'chartreuse' , 'chocolate' , 'coral' ,
'CornflowerBlue' , 'cornsilk' , 'crimson' , 'cyan' , 'DarkBlue' , 'DarkCyan' ,
'DarkGoldenrod' , 'DarkGray' , 'DarkGreen' , 'DarkKhaki' , 'DarkMagenta' ,
'DarkOliveGreen' , 'DarkOrange' , 'DarkOrchid' , 'DarkRed' , 'DarkSalmon' ,
'DarkSeaGreen' , 'DarkSlateBlue' , 'DarkSlateGray' , 'DarkTurquoise' ,
'DarkViolet' , 'DeepPink' , 'DeepSkyBlue' , 'DimGray' , 'DodgerBlue' ,
'firebrick' , 'FloralWhite' , 'ForestGreen' , 'fuchsia' , 'gainsboro' ,
'GhostWhite' , 'gold' , 'goldenrod' , 'gray' , 'green' , 'GreenYellow' ,
'honeydew' , 'HotPink' , 'IndianRed' , 'indigo' , 'ivory' , 'khaki' , 'lavender'
, 'LawnGreen' , 'LemonChiffon' , 'LightBlue' , 'LightCoral' , 'LightCyan' ,
'LightGoldenrodYellow' , 'LightGray' , 'LightGreen' , 'LightPink' ,
'LightSalmon' , 'LightSeaGreen' , 'LightSkyBlue' , 'LightSlateGray' ,
'LightYellow' , 'lime' , 'LimeGreen' , 'linen' , 'magenta' , 'maroon' ,
'MediumAquamarine' , 'MediumBlue' , 'MediumOrchid' , 'MediumPurple' ,
'MediumSeaGreen' , 'MediumSlateBlue' , 'MediumSpringGreen' , 'MediumTurquoise' ,
'MediumVioletRed' , 'MidnightBlue' , 'MintCream' , 'MistyRose' , 'moccasin' ,
'NavajoWhite' , 'navy' , 'OldLace' , 'olive' , 'OliveDrab' , 'orange' ,
'OrangeRed' , 'orchid' , 'PaleGoldenrod' , 'PaleGreen' , 'PaleTurquoise' ,
'PaleVioletRed' , 'PapayaWhip' , 'PeachPuff' , 'peru' , 'pink' , 'plum' ,
'PowderBlue' , 'purple' , 'red' , 'RosyBrown' , 'RoyalBlue' , 'SaddleBrown' ,
'salmon' , 'SandyBrown' , 'SeaGreen' , 'seashell' , 'sienna' , 'silver' ,
'SkyBlue' , 'SlateBlue' , 'SlateGray' , 'snow' , 'SpringGreen' , 'SteelBlue' ,
'tan' , 'thistle' , 'tomato' , 'turquoise' , 'violet' , 'VioletRed' , 'wheat' ,
'white' , 'WhiteSmoke' , 'yellow' , 'YellowGreen' ]
couleurs_a_eviter = {"white", "snow", "whitesmoke", "ghostwhite", "aliceblue",
"ivory", "azure", "floralwhite", "lavender", "blanchedalmond"}
```

```
lescouleurs = [c for c in lescouleurs if c.lower() not in couleurs_a_eviter]
indicecouleur = 0
est_p = crible_numba(N)
donnees_textes = []
for k in range(1, 105):
    indicecouleur = (indicecouleur + 1) % len(lescouleurs)
    couleur = lescouleurs[indicecouleur]
    liste = bosse(N, est_p, k)
    x_valeurs = [x for x in range(6, N+1, 2) if x % 105 == k]
    if liste and x_valeurs:
        plt.plot(x_valeurs, liste, color=couleur, alpha=0.7)
        donnees_textes.append((liste[-1], k, couleur, x_valeurs[-1]))
donnees_textes.sort(key=lambda item: item[0])
y_precedent = -9999
ecart_min_y = 1.8
compteur_x = 0
for y_reel, k, couleur, x_fin in donnees_textes:
    y_affichage = max(y_reel, y_precedent + ecart_min_y)
    palier_x = compteur_x % 20
    decalage_x = 80 + (palier_x * 100)
    x_affichage = x_fin + decalage_x
    plt.text(x_affichage, y_affichage, str(k), color=couleur, fontsize=7.5,
weight='bold', va='center', ha='left')
    y_precedent = y_affichage
    compteur_x += 1
print("\n=== CLASSEMENT DES RÉSIDUS K (DU PLUS BAS AU PLUS HAUT EN Y) ===")
residus_ordonnes = [item[1] for item in donnees_textes]
print(residus_ordonnes)
plt.title("Familles de congruence (mod 105) de la comète de Goldbach")
plt.xlabel("n")
plt.ylabel("G(n)")
plt.xlim(6, N + 1400)
plt.show()
tac = time.time()
print(f"Temps d'exécution global : {tac-tic:.4f} s.")
```