```
import numpy as np
import time

def multicirculante(n):
    """
    Crée l'opérateur G sous forme de liste d'entiers (bitsets).
    Chaque entier représente une ligne de la matrice.
    """
    taille_totale = sum(range(2, n + 1))
    G = np.zeros(taille_totale, dtype=object)
    index = 0
    for p in range(2, n + 1):
        # Pour chaque bloc circulaire de taille p
        for i in range(p):
            # Le 1 se déplace à la colonne (i+1)%p
            col = index + ((i + 1) % p)
            G[index + i] = 1 << col
        index += p
    return G, taille_totale

def multipliematbool(A, B_t, taille):
    """
    Multiplication de matrices booléennes optimisée.
    B_t est la transposée de B pour un accès rapide aux colonnes.
    """
    C = np.zeros(taille, dtype=object)
    for i in range(taille):
        row_A = A[i]
        # On utilise le bitwise pour multiplier la ligne i par toutes les
colonnes
        for j in range(taille):
            if row_A & B_t[j]:
                C[i] |= (1 << j)
    return C

def puissancematbool(G, k, taille):
    """Calcule G^k par exponentiation rapide."""
    # Matrice Identité en format bitset
    res = np.array([(1 << i) for i in range(taille)], dtype=object)
    base = G
    while k > 0:
        if k % 2 == 1:
            # Transposée de la base pour l'accès aux colonnes
            base_t = [(sum(1 << i for i in range(taille) if (base[i] & (1 <<
j)))) for j in range(taille)]
            res = multipliematbool(res, base_t, taille)
        # Carré de la base
        base_t = [(sum(1 << i for i in range(taille) if (base[i] & (1 << j))))
for j in range(taille)]
        base = multipliematbool(base, base_t, taille)
        k //= 2
    return res

def latrace(k, n):
    G, taille = multicirculante(n)
    G_k = puissancematbool(G, k, taille)
    # La trace est la somme des bits situés sur la diagonale (1 << i)
    trace = 0
    for i in range(taille):
        if G_k[i] & (1 << i):
            trace += 1
    return trace

tic = time.time()
```

```
for n in range(40,42,2):
    print(n,' :::::')
    operateur = multicirculante(n)
    for k in range(3,n//2+1,2):
        if latrace(k,n) == k and latrace(n-k,n) == n-k:
            print(k,' --> dg')
tac = time.time()
print(tac-tic,' s.')
```