

```
#!/usr/bin/env python3
"""Involution de Zagier pour  $p = 4k+1$  premier.
 $S = \{(x,y,z) \text{ entiers STRICTEMENT positifs : } x^2 + 4yz = p\}$ 
 $\tau$  : involution de Zagier (un seul point fixe  $(1,1,(p-1)/4)$ )  $\rightarrow |S|$  impair
 $\sigma$  :  $(x,y,z) \rightarrow (x,z,y)$  ; un point fixe ( $y=z$ ) donne  $p = x^2 + (2y)^2$ .
Usage : python3 zagier_involution.py [N] (défaut 100) ; verbose : python3
zagier_involution.py 30 -v
Corrections par rapport à la première version :
-  $x, y, z > 0$  (et non  $x, y$  dans  $[-\sqrt{n}, \sqrt{n}]$ ) ;
-  $p$  premier seulement : c'est la primalité qui exclut les cas frontière  $x = y-z$ 
( $p$  carré)
et  $x = 2y$  ( $p$  pair), les « images vides » des carrés parfaits ;
- les trois cas de  $\tau$  sont testés dans l'ordre de l'article, avec assertion si
un cas frontière survient.
"""

import sys
from math import isqrt

def isprime(m): return m > 1 and all(m % d for d in range(2, isqrt(m) + 1))

def ensemble_S(p):
    S = []
    for x in range(1, isqrt(p) + 1):
        r = p - x * x
        if r > 0 and r % 4 == 0:
            m = r // 4
            S += [(x, y, m // y) for y in range(1, m + 1) if m % y == 0]
    return S

def tau(t):
    x, y, z = t
    if x < y - z: return (x + 2 * z, z, y - x - z)
    if x > 2 * y: return (x - 2 * y, x - y + z, y)
    if y - z < x < 2 * y: return (2 * y - x, y, x - y + z)
    raise ValueError(f"cas frontière {t} :  $x = y-z$  ( $p$  carré) ou  $x = 2y$  ( $p$  pair)")

def sigma(t): return (t[0], t[2], t[1])

def main(N, verbose=False):
    for p in range(5, N + 1, 4):
        if not isprime(p): continue
        S = ensemble_S(p)
        assert all(x*x + 4*y*z == p and min(t) > 0 for t in S for x, y, z in
[t])
        img = {t: tau(t) for t in S}
        assert all(img[t] in S and img[img[t]] == t for t in S), "tau n'est
pas une involution"
        fix_tau = [t for t in S if img[t] == t]
        assert fix_tau == [(1, 1, (p - 1) // 4)], fix_tau
        assert len(S) % 2 == 1
        fix_sigma = [t for t in S if sigma(t) == t]
        assert fix_sigma, "sigma doit avoir un point fixe"
        x, y, _ = fix_sigma[0]
        print(f"p={p:3d} |S|={len(S):2d} (impair) point fixe de tau
{fix_tau[0]} "
f"point fixe de sigma {fix_sigma[0]} => {p} = {x}^2 + {2*y}^2")
        if verbose:
            for t in S: print(f" {t} -> tau {img[t]} sigma {sigma(t)}")
        print("Toutes les vérifications sont passées.")

if __name__ == "__main__":
    main(int(sys.argv[1]) if len(sys.argv) > 1 and sys.argv[1] != "-v" else 100,
"-v" in sys.argv)
```