

Correctif (18 octobre 2026) au problème à élucider n° 1.
ia claude pour Denise Vella-Chemla, octobre 2026

Notons $2n = 4pk + 2a$ avec $0 \leq a < 2p$. Le nombre $f(2n, p)$ est bien le nombre d'impairs i , $3 \leq i \leq n$, tels que $i \equiv 0$ ou $i \equiv 2n \pmod{p}$ (deux classes de résidus, k impairs de chaque classe par période complète de $2p$). La récurrence donnée est exacte pour $a = 0$ et $a = p$, et $2 \cdot f(4pk, p)$ a le bon ordre de grandeur, mais le terme correctif du reste est faux dans certains cas (par exemple $2n = 28, p = 5$: on trouve 3 et non 2). La formule exacte est, pour $a \neq 0, p$:

$$f(2n, p) = \begin{cases} 2k + [2a > p] - [2a \equiv 1 \pmod{p}] & \text{si } 0 < a < p, \\ 2k + 1 + [2a > 3p] - [2a \equiv 1 \pmod{p}] & \text{si } p < a < 2p, \end{cases}$$

où $[\cdot]$ vaut 1 si la condition est vraie et 0 sinon (le dernier terme vient de l'exclusion de $i = 1$). Vérifiée sans aucun désaccord pour $2n \leq 3000$ et tout premier impair $p \leq \sqrt{2n}$.

Les deux remarques qui suivent le tableau sont fausses : pour $2n = 2q$ avec q premier, on n'a pas toujours $f(2q, p) \in \{f(2q - 2, p), 2f(2q - 2, p)\}$ (contre-exemple : $2q = 38, p = 5$: 4 et 3) ; pour $2j$ entre deux jumeaux, on n'a pas $f(2j, p) \in \{f(2j - 2, p), (f(2j - 2, p) + 1)/2\}$ (contre-exemple : $2j = 30, p = 3$: 3 et 4). Les échecs de la récurrence ne dépendent d'ailleurs pas de la primalité de n : ils tiennent à la position de a par rapport à $p/2$ et $3p/2$. Enfin f ne travaille que sur un seul premier p à la fois ; le « mélange » de tous les premiers $q \leq \sqrt{2n}$ (inclusion-exclusion du crible de Legendre) n'est pas fait, et la bijection jumeaux $\leftrightarrow$ premiers n'a donc aucune base.

Programme de l'ia claude pour vérifier ou corriger la Piste à creuser n° 1

```
"""Verification de la fonction f du 'Probleme a elucider no 1' (13/09/2013).
Notation : 2n est le pair, p un premier impair.
    compte(2n,p) = #{ i impair, 3<=i<=n : p | i ou p | 2n-i }    (definition
                        'combinatoire')
    f_rec(2n,p) = definition recursive du texte (avec les 2.f)
"""
from sympy import primerange, isprime

def compte(N, p):
    n = N // 2
    return sum(1 for i in range(3, n + 1, 2) if i % p == 0 or (N - i) % p == 0)

def f_rec(N, p):
    """f(4pk+2a,p) : N = 4pk + 2a, 0<=a<2p, texte litteral"""
    k, r = divmod(N, 4 * p)
    a = r // 2
    if a == 0:
        return k
    if a == p:
        return k + 1
    base = k
    if a < p:
        return 2 * base
```

```

    return 2 * base + 1

def f_exacte(N, p):
    """N = 4pk + 2a, 0<=a<2p. Deux classes d'impairs (i = 0 et i = 2a mod p),
    k elements chacune par periode complete de 2p, plus reste (t=1..a impairs),
    moins i=1 eventuel."""
    k, r = divmod(N, 4 * p)
    a = r // 2
    if a == 0:
        return k
    if a == p:
        return k + 1
    v = 2 * k
    if a > p:
        v += 1 # t = p (multiple de p dans le reste)
    if (a < p and 2 * a > p) or (a > p and 2 * a > 3 * p):
        v += 1 # t = 2a mod p, impair, dans le reste
    if (2 * a - 1) % p == 0:
        v -= 1 # i = 1 exclu (i >= 3)
    return v

if __name__ == "__main__":
    print("1) f_rec (texte litteral) vs compte (definition combinatoire)")
    bad = 0
    tot = 0
    for N in range(12, 101, 2):
        for p in primerange(3, int(N ** 0.5) + 1):
            tot += 1
            if f_rec(N, p) != compte(N, p):
                bad += 1
            if bad <= 12:
                print(f" 2n={N} p={p}: rec={f_rec(N,p)} compte={compte(N,p)}")
    print(f" desaccords : {bad}/{tot}")

    print("\n2) table du texte (compte) pour p=3,5,7, 2n=12..100 : quelques
    lignes")
    for N in (12, 14, 20, 62, 98, 100):
        print(N, [compte(N, p) for p in primerange(3, int(N**0.5)+1)])

    print("\n3) remarque 'N=2p premier' : f(2p,q) in {f(2p-2,q), 2 f(2p-2,q)} ?")
    ok = tot = 0
    for p in primerange(5, 2000):
        N = 2 * p
        for q in primerange(3, int(N ** 0.5) + 1):
            a, b = compte(N, q), compte(N - 2, q)
            tot += 1
            if a == b or a == 2 * b:
                ok += 1
            elif tot - ok <= 8:
                print(f" contre-exemple 2p={N}, q={q}: f={a}, f(2p-2)={b}")
    print(f" verifie {ok}/{tot}")

    print("\n4) remarque 'jumeaux' (2j entre deux premiers impairs jumeaux) :
    f(2j,q)=f(2j-2,q) ou (f(2j-2,q)+1)/2 ?")
    ok = tot = 0

```

```

for j in range(4, 2000):
    if isprime(2*j-1) and isprime(2*j+1): # 2j est le pair entre deux jumeaux
        -> N = 2j
        N = 2 * j
        for q in primerange(3, int(N ** 0.5) + 1):
            a, b = compte(N, q), compte(N - 2, q)
            tot += 1
            if a == b or 2 * a - 1 == b:
                ok += 1
            elif tot - ok <= 8:
                print(f" contre-exemple 2j={N}, q={q}: f={a}, f(2j-2)={b}")
print(f" verifie {ok}/{tot}")

print("\n5) formule exacte (derivee a la main) vs compte")
bad = 0; tot = 0
for N in range(12, 3000, 2):
    for p in primerange(3, int(N ** 0.5) + 1):
        tot += 1
        if f_exacte(N, p) != compte(N, p):
            bad += 1
print(f" desaccords : {bad}/{tot}")

```

Son résultat :

```

1) f_rec (texte litteral) vs compte (definition combinatoire)
2n=28 p=5: rec=2 compte=3
2n=38 p=5: rec=3 compte=4
2n=48 p=5: rec=4 compte=5
2n=52 p=7: rec=3 compte=4
2n=54 p=7: rec=3 compte=4
2n=58 p=5: rec=5 compte=6
2n=66 p=7: rec=4 compte=5
2n=68 p=5: rec=6 compte=7
2n=68 p=7: rec=4 compte=5
2n=78 p=5: rec=7 compte=8
2n=80 p=7: rec=5 compte=6
2n=82 p=7: rec=5 compte=6
desaccords : 16/109

2) table du texte (compte) pour p=3,5,7, 2n=12..100 : quelques lignes
12 [1]
14 [2]
20 [3]
62 [10, 6, 4]
98 [16, 10, 4]
100 [16, 5, 7]

3) remarque 'N=2p premier' : f(2p,q) in {f(2p-2,q), 2 f(2p-2,q)} ?
contre-exemple 2p=38, q=5: f=4, f(2p-2)=3
contre-exemple 2p=58, q=5: f=6, f(2p-2)=5
contre-exemple 2p=94, q=7: f=7, f(2p-2)=6
contre-exemple 2p=118, q=5: f=12, f(2p-2)=11
contre-exemple 2p=122, q=7: f=9, f(2p-2)=8
contre-exemple 2p=146, q=11: f=7, f(2p-2)=6

```

```

contre-exemple 2p=158, q=5: f=16, f(2p-2)=15
contre-exemple 2p=178, q=5: f=18, f(2p-2)=17
verifie 3146/3404

```

4) remarque 'jumeaux' ($2j$ entre deux premiers impairs jumeaux) : $f(2j, q) = f(2j-2, q)$
ou $(f(2j-2, q) + 1) / 2$?

```

contre-exemple 2j=30, q=3: f=3, f(2j-2)=4
contre-exemple 2j=42, q=3: f=4, f(2j-2)=6
contre-exemple 2j=42, q=5: f=4, f(2j-2)=2
contre-exemple 2j=60, q=5: f=3, f(2j-2)=6
contre-exemple 2j=72, q=5: f=7, f(2j-2)=4
contre-exemple 2j=72, q=7: f=5, f(2j-2)=3
contre-exemple 2j=102, q=3: f=9, f(2j-2)=16
contre-exemple 2j=102, q=5: f=10, f(2j-2)=5
verifie 804/1091

```

5) formule exacte (derivee a la main) vs compte
desaccords : 0/15856

Piste à creuser n° 4 : c'était du hasard pur

Le programme

```

"""Piste n 4 : r(2^k.p) comme somme de r(2^j.p), j<k. r(n) = #{(q,q') premiers
    impairs, q<=q', q+q'=n}."""
import numpy as np, sys
from itertools import combinations

def crible(M):
    s = np.ones(M + 1, dtype=bool); s[:2] = False
    for i in range(2, int(M**.5) + 1):
        if s[i]: s[i*i::i] = False
    return s

def make_r(S):
    def r(n):
        h = n // 2
        a = np.arange(3, h + 1, 2)
        return int(np.count_nonzero(S[a] & S[n - a]))
    return r

def subset_sum_ok(target, vals):
    """target = somme d'un sous-ensemble de vals ? (DP)"""
    reach = {0}
    for v in vals:
        reach |= {x + v for x in reach if x + v <= target}
    return target in reach

if __name__ == "__main__":
    p = int(sys.argv[1]) if len(sys.argv) > 1 else 13
    K = int(sys.argv[2]) if len(sys.argv) > 2 else 19
    S = crible(p * 2 ** K + 1); r = make_r(S)

```

```

R = [r(p * 2 ** k) for k in range(K + 1)]
print(f"p={p} : r(2^k p) pour k=0..{K} :", R)
print("rapports r_k/r_(k-1) :", [round(R[k] / R[k-1], 2) for k in range(2, K + 1)])
print("\nCriterie : tout entier <= somme des precedents est somme d'un
sous-ensemble si r_k <= 1 + somme_(j<k) r_j")
for k in range(2, K + 1):
    crit = R[k] <= 1 + sum(R[:k])
    ok = subset_sum_ok(R[k], R[:k])
    print(f" k={k:2d} r={R[k]:7d} critere={crit} sous-ensemble={ok}")
print("\nTemoin : n'importe quel entier m de meme taille s'ecrit aussi comme
somme de r_j precedents :")
import random; random.seed(1)
k = K
tests = [random.randint(R[k-1], R[k]) for _ in range(200)]
print(" 200 entiers au hasard dans [r_(K-1), r_K] :", sum(subset_sum_ok(m,
R[:k]) for m in tests), "/ 200 representables")

```

Son résultat :

```

p=13 : r(2^k p) pour k=0..19 : [0, 3, 3, 5, 7, 10, 22, 28, 46, 80, 139, 230, 404,
688, 1222, 2146, 3874, 6972, 12558, 22769]
rapports r_k/r_(k-1) : [1.0, 1.67, 1.4, 1.43, 2.2, 1.27, 1.64, 1.74, 1.74, 1.65,
1.76, 1.7, 1.78, 1.76, 1.81, 1.8, 1.8, 1.81]

```

```

Criterie : tout entier <= somme des precedents est somme d'un sous-ensemble si r_k
<= 1 + somme_(j<k) r_j
k= 2 r=      3 critere=True sous-ensemble=True
k= 3 r=      5 critere=True sous-ensemble=False
k= 4 r=      7 critere=True sous-ensemble=False
k= 5 r=     10 critere=True sous-ensemble=True
k= 6 r=     22 critere=True sous-ensemble=True
k= 7 r=     28 critere=True sous-ensemble=True
k= 8 r=     46 critere=True sous-ensemble=True
k= 9 r=     80 critere=True sous-ensemble=True
k=10 r=    139 critere=True sous-ensemble=True
k=11 r=    230 critere=True sous-ensemble=True
k=12 r=    404 critere=True sous-ensemble=True
k=13 r=    688 critere=True sous-ensemble=True
k=14 r=   1222 critere=True sous-ensemble=True
k=15 r=   2146 critere=True sous-ensemble=True
k=16 r=   3874 critere=True sous-ensemble=True
k=17 r=   6972 critere=True sous-ensemble=True
k=18 r=  12558 critere=True sous-ensemble=True
k=19 r=  22769 critere=True sous-ensemble=True

```

```

Temoin : n'importe quel entier m de meme taille s'ecrit aussi comme somme de r_j
precedents :
200 entiers au hasard dans [r_(K-1), r_K] : 200 / 200 representables

```