3 ou 5 égal 7

Denise Vella-Chemla

1. Idée initiale

Dans de précédentes notes (voir [1], [2]), on a démontré qu'un nombre premier supérieur à $\sqrt{n}$ et inférieur ou égal à $n/2$, et qui n'est jamais congru à n modulo tout nombre premier inférieur à $\sqrt{n}$ est un décomposant de Goldbach de n.

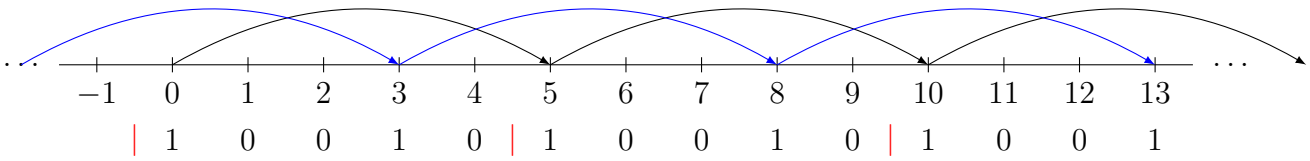
Il s'agit de démontrer qu'un tel nombre existe toujours.

À l'aide de 2 exemples, on montre d'abord comment implémenter la recherche de décomposants de Goldbach de n, un nombre pair, en utilisant des mots binaires, qu'on utilise comme des masques booléens[1].

2. Masques binaires

Supposons que l'on veuille trouver les nombres qui sont congrus à 0 modulo 5 (i.e. qui sont divisibles par 5) ou bien qui sont congrus à 3 modulo 5 (i.e. dont le reste lorsqu'on les divise par une division euclidienne par 5 est 3). On utilise un masque binaire de longueur 5 (**b**10010), que l'on reporte en regard de la droite des entiers relatifs, en le concaténant à ses copies (i.e. en mettant bout à bout des copies de ce masque). Les nombres en question "s'envoient" sur les différentes occurrences du booléen 1 dans les copies du masque de base.

Note : dans le texte, on accolera systématiquement à gauche la lettre **b** au début d'un mot binaire, pour distinguer un entier écrit en binaire d'un entier écrit dans la numération décimale habituelle.



Remarque : on utilisera la convention que l'entier 0 s'envoie sur le premier booléen positionné à 1 du masque de longueur un entier p donné, et par concaténation (i.e. par translation du masque), chaque nombre divisible par p s'enverra sur une occurrence de ce booléen 1 en position initiale dans une copie du masque de base. Les petits traits rouges verticaux sont destinés à faciliter la lecture des concaténations de mots binaires.

[1] On avait utilisé cette idée en 2009 (cf [3], [4]), on cherchait une démonstration par récurrence, plutôt que par étude combinatoire des mots possibles.

3. Deux exemples

La démonstration du fait qu'un nombre premier inférieur ou égal à $n/2$ incongru à n selon tout module premier inférieur à $\sqrt{n}$, s'il existe, est un décomposant de Goldbach de n est à lire dans [1].

Exemple du nombre 42

Au nombre pair 42, qui est congru à 0 (mod 2), 0 (mod 3) et 2 (mod 5), sont associés les 3 masques:

- **b**10, masque d'élimination de tous les nombres pairs ;
- **b**100, masque d'élimination de tous les nombres divisibles par 3, qui sont de fait congrus à 42 (i.e. qui ont leur complémentaire à 42 également divisible par 3, cela étant dû au fait que $x \equiv n \pmod{p} \iff n - x \equiv 0 \pmod{p}$) puisque 3 divise 42 ;
- **b**10100, masque d'élimination des nombres divisibles par 5 ou bien qui sont congrus à 2 modulo 5.

Exemple du nombre 98

Au nombre pair 98, qui est congru à 0 (mod 2), 2 (mod 3), 3 (mod 5) et 0 (mod 7) sont associés les 4 masques :

- **b**10, masque d'élimination de tous les nombres pairs ;
- **b**101, masque d'élimination de tous les nombres divisibles par 3 ou bien congrus à 2 (mod 3);
- **b**10010, masque d'élimination des nombres divisibles par 5 ou bien qui ont pour reste 3 quand on les divise (par division euclidienne) par 5 (c'est ce que signifie l'expression "congrus à 3 modulo 5" de Gauss).
- **b**1000000, masque d'élimination des nombres divisibles par 7, qui sont de fait congru à 98, puisque 7 divise 98.

Une recherche simple permet de trouver que le masque de base se calcule de la façon suivante ; il est égal au mot binaire du nombre suivant pour n pair :

- $\text{masque}(n, p) = 2^{p-(n \bmod p)-1}$ pour $p = 2$ ou pour tout p divisant n
- $\text{masque}(n, p) = 2^{p-(n \bmod p)-1} + 2^{p-1}$ pour tout p ne divisant pas n

On a $\text{masque}(40, 2) = \text{masque}(n, 2) = 2$ quel que soit n. $\text{masque}(40, 3) = 6$ et $\text{masque}(40, 5) = 32$.

On a $\text{masque}(98, 3) = 5$, $\text{masque}(98, 5) = 18$ et $\text{masque}(98, 7) = 64$.

4. Concaténation de mots binaires

La concaténation s'effectue en multipliant le mot par les puissances de 2 adéquates, exactement comme on le fait en numération décimale.

Par exemple, le mot **b**110, écriture binaire de l'entier 5, peut être concaténé à lui même[2] ainsi :

$$5 + 5.2^3 + 5.2^6 + 5.2^9 + 5.2^{12} + 5.2^{15}$$

sera égal à **b**101101101101101101.

Présentons sur la droite des entiers relatifs, en regard des nombres de 0 à 21, l'opération de concaténation agir sur chacun des 3 masques : ils sont "étendus", i.e. concaténés à eux-mêmes autant de fois que nécessaire pour que tous les entiers de 0 à 21 soient envoyés sur un booléen d'un des 3 masques associés à 42.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1
	1	0	1	0	0	1	0	1	0	0	1	0	1	0	0	1	0	1	0	0	1	0
∨	1	0	1	1	1	1	1	1	1	1	1	0	1	0	1	1	1	1	1	0	1	1

On repère les booléens nuls associés aux entiers 1 (qu'on oublie), 11, 13 et 19 qui sont bien comme attendus des décomposants de Goldbach de 42.

On note que 5, qui est pourtant un décomposant de Goldbach de 42, a été éliminé par notre masque disjonctif, dans la mesure où il fait partie des nombres premiers inférieurs à $\sqrt{n} = \sqrt{42}$ dans ce cas, et où il est divisible par lui-même. C'est pour cette raison qu'a été précisé que l'on ne trouve ici que les nombres premiers décomposants de Goldbach de n qui sont supérieurs à $\sqrt{n}$.

5. L'opération ou logique (∨)

C'est cette opération qui a motivé le titre de la présente note : la table de vérité du ou logique (i.e. de la disjonction, représentée par le symbole ∨) est :

∨	0	1
0	0	1
1	1	1

On peut ainsi dire par boutade que "3 ou 5 égal 7" car 3 est représenté par le mot binaire **b**11, 5 est représenté par le mot binaire **b**101 et donc le ou logique de leurs 2 mots booléens est égal à **b**111 qui est la représentation en binaire du nombre 7.

C'est George Boole qui, souhaitant algébriser le raisonnement humain, a défini la loi de l'opération du ou logique (∨) sur l'ensemble $\mathbb{B} = \{0, 1\}$. Les mathématiciens insistent sur le fait que dans

[2] *Remarque* : en théorie des langages, on considère qu'on élève un mot à une certaine puissance, lorsqu'on concatène plusieurs copies de ce mot, l'exposant compte le nombre d'occurrences moins un du mot dans le mot résultant de l'opération de concaténation.

l'algèbre de Boole, $1 \vee 1 = 1 + 1 = 1$. Il faut bien distinguer cette égalité du résultat de l'addition de 1 et 1 dans $\mathbb{Z}/2\mathbb{Z}$, qui est nul, $1 + 1 = 0$.

Dans une note ultérieure, on utilisera non plus des mots périodiques, comme cela est adéquat pour étudier la conjecture de Goldbach, mais des mots binaires fractals, pour étudier l'espace des nombres premiers.

5. Existence d'un décomposant

Les exemples montrent, et c'est simple à comprendre, que les masques correspondant aux modules qui divisent n sont construits en concaténant un mot booléen de base, de longueur p, qui ne contient qu'un seul 1, tandis que le mot de base contient exactement 2 lettres 1 pour les modules qui ne divisent pas n (en effet, si p divise n, les 2 classes de congruence à éliminer n'en font qu'une, la classe nulle). Il semble légitime de penser qu'étudier combinatoirement uniquement des mots de base contenant systématiquement deux occurrences du booléen 1 plutôt qu'une seule occurrence est une sorte de pire des cas, puisque ce sont les booléens positionnés à 1 qui "éliminent" des nombres.

Si on était capable de démontrer que combinatoirement, toutes les disjonctions envisageables de mots binaires de toutes les formes possibles (qu'elles soient fictives ou pas) de masques pour les nombres de 2 à p contiennent toujours un zéro à une position minimale qui est inférieure à $\frac{p^2}{2}$, alors la conjecture de Goldbach serait démontrée.

Il y a $p-1$ mots différents de longueur p : le booléen en position 0 est systématiquement positionné à 1 (il représente la divisibilité par p). Le second booléen peut être positionné au choix sur l'une des $p-1$ positions possibles dans le mot de base.

Étudier la position minimale combinatoirement et démontrer que cette position minimale est toujours "suffisamment petite" pour assurer l'existence d'un décomposant de Goldbach reste à faire.

6. Décomposants de Goldbach, position la plus basse d'un zéro dans la disjonction des mots binaires

On considèrera d'abord combinatoirement tous les masques possibles de longueurs 2, 3 et 5, ce qui correspond à 8 combinaisons possibles. On étudiera le maximum des positions minimum du premier 0 dans les 8 cas possibles. Puis on essaiera d'agréger à ce qu'on aura trouvé les 6 masques possibles de longueur 7, etc.

Références

[1] http://denise.vella.chemla.free.fr/jade1.pdf

[2] http://denise.vella.chemla.free.fr/invariante.pdf

[3] http://denise.vella.chemla.free.fr/combimots.pdf

[4] http://denise.vella.chemla.free.fr/constructif.pdf

Annexe : Programme python et son résultat

Le programme

```
def renverse(n):
        return int(bin(n)[:1:-1], 2)

def lenieme(x, n):
        return x & 2**n != 0

print('les restes')
print('98 % 2 = ',98%2)
print('98 % 3 = ',98%3)
print('98 % 5 = ',98%5)
print('98 % 7 = ',98%7)
print('les mots de 98')
print('2 -> ', 2**(2-(98%2)-1))
print('3 -> ', 2**(3-(98%3)-1)+2**(3-1))
print('5 -> ', 2**(5-(98%5)-1)+2**(5-1))
print('7 -> ', 2**(7-(98%7)-1))
print('les masques')
modulo2 = 2+2*(2**2)+2*(2**4)+2*(2**6)+2*(2**8)+2*(2**10)
        +2*(2**12)+2*(2**14)+2*(2**16)+2*(2**18)+2*(2**20)
        +2*(2**22)+2*(2**24)+2*(2**26)+2*(2**28)+2*(2**30)
        +2*(2**32)+2*(2**34)+2*(2**36)+2*(2**38)+2*(2**40)
        +2*(2**42)+2*(2**44)+2*(2**46)+2*(2**48)
print('modulo 2')
print(bin(modulo2))
modulo3 = 5+5*(2**3)+5*(2**6)+5*(2**9)+5*(2**12)+5*(2**15)
        +5*(2**18)+5*(2**21)+5*(2**24)+5*(2**27)+5*(2**30)
        +5*(2**33)+5*(2**36)+5*(2**39)+5*(2**42)+5*(2**45)
        +5*(2**48)
print('modulo 3')
print(bin(renverse(modulo3)))
modulo5 = 18+18*(2**5)+18*(2**10)+18*(2**15)+18*(2**20)+18*(2**25)
        +18*(2**30)+18*(2**35)+18*(2**40)+18*(2**45)
print('modulo 5')
print(bin(renverse(modulo5)))
modulo7 = 64+64*(2**7)+64*(2**14)+64*(2**21)+64*(2**28)+64*(2**35)
        +64*(2**42)
print('modulo 7')
print(bin(renverse(modulo7)))
print('')
print('disjonction des puissances des masques')
res = renverse(modulo2) — renverse(modulo3) — renverse(modulo5) — ren-
verse(modulo7)
print(bin(res))
print('Les decomposants de Goldbach :')
for k in range(1,49):
if lenieme(res,k)==False and k != 1:
                print(k,' ',end='',)
print('')
```

Le résultat du programme

```
les restes
98 % 2 = 0
98 % 3 = 2
98 % 5 = 3
98 % 7 = 0

les mots de 98
2 -> 2
3 -> 5
5 -> 18
7 -> 64

les masques
modulo 2
0b10101010101010101010101010101010101010101010101010101010
modulo 3
0b101101101101101101101101101101101101101101101101
modulo 5
0b1001010010100101001010010100101001010010100101001
modulo 7
0b1000000100000010000001000000100000010000001

disjonction des puissances des masques
0b10111111111110111110111111111110111111111111111101

Les decomposants de Goldbach :
19 31 37
```

Programme généralisé

```
import math
import time

def prime(atester):
        k = 2 ;
        if (atester in [0,1]): return False ;
        if (atester in [2,3,5,7]): return True ;
        while (True):
                if ((k * k) ¿ atester): return True
                else:
                                        if ((atester % k) == 0): return False
                                        else: k=k+1
def renverse(n):
        return int(bin(n)[:1:-1], 2)
def lenieme(x, n):
                return x & 2**n != 0

start = time.time()
for n in range(6,10002,2):
        rac = int(math.sqrt(n))
        print('racine ',rac)
        res = 0
        for p in range(2,rac+1):
                if prime(p):
                        mot = 2**(p-(n%p)-1)
                        if p != 2 and n%p != 0:
                                mot = mot+2**(p-1)
                        print(p, ' –> ', mot)
                        modulop = mot
                        expoexposant = p
                        while expoexposant ¡= n/2:
                                modulop = modulop+mot*(2**expoexposant)
                                expoexposant = expoexposant + p
                        print(bin(modulop))
                        res = res — renverse(modulop)
        print('————————————-')
        print(bin(res))
        print('decomposants de Goldbach de ', n)
        moitie = int(n/2)
        for k in range(1,moitie):
                if lenieme(res,k)==False and k != 1:
                        print(k,' ',end='')
                        if not(prime(k)):
                                print('ratage')
                        else:
                                if prime(k) and not(prime(n-k)):
                                        print('ratage')
        print('')
end = time.time()
elapsed = end - start

print(f'Temps déxecution : elapsed:.2ms')
```