[Théorème G : Pour tout $\mathbf{A \in A_n}$ ensemble des entiers impairs $\leq \mathbf{n}$, avec $\mathbf{P \in P_{2n}}$ ensemble des nombres premiers $\leq \mathbf{\sqrt{2n}}$; tel que **si : $2n \not\equiv A$ [P]** où **A** précède **A + 2 = p'** un nombre premier, la **CG** sera vérifiée quel que soit la limite **n = + 1** , succédant à la limite ***n*** ayant été vérifiée ; ce qui explique cette **propriété et égalité** récurrente de l'algorithme de Goldbach, ***qui agit selon le principe d'une preuve par récurrence*** .

Preuve :

si le complémentaire de 2n – 7 est 193 cela implique **$7 \not\equiv 200$[P] ; d'où ,** par la propriété récurrente du décalage d'un rang des congruence. **Le successeur de 7qui pour complémentaire de 193 par rapport** à 2n+2 = 202 serra 7+2 = 9 ⇒ **$9 \not\equiv 202$[P], toujours non congru à P** et où 9 même si il n'est pas un nombre premiers, cela n'a aucune importance, car dans cette propriété récurrente c'est le décalage d'un rang des congruences qui importe, que l'on utilisera.

Inversement : si 202 – 7 =**195** ⇒ **7 ≡ 202[P]** où **195 ≠ p' , par récurrence le successeur de 7 est 9** et si il n'y avait pas ce décalage d'un rang des congruence, on pourrait se retrouver avec **9**, non congru à **P** ; alors que **9 ≡ 204[P] = 195 ≠ p' ;** qui est bien son successeur par récurrence et le même sans contestation possible, qui **est toujours non divisible par les mêmes facteurs premiers P..!** *Le contraire est absurde.*

$A \not\equiv 2n$[P] = (A+2) $\not\equiv$ (2n+2) [P] et inversement, si : **$A \equiv 2n$[P]** alors **(A+2) ≡ (2n+2) [P] .**

Exemple : de la propriété inverse que l'on utilisera : 15 ≡ **30[P]** , cela implique 13 ≡ **28[P]** ⇒ 11≡ **26[P]** ...etc cette différence de **15** serra toujours la même, le contraire serait absurde.

Inversement : 13 $\not\equiv$ **30[P]** , cela implique 11 $\not\equiv$ **28[P]** ⇒ 9 $\not\equiv$ **26[P]** mais 9 **≠ p' ;** cette différence **q = 17** ne peut pas changer, là aussi ce serait absurde**.**

Or , *ce qui nous importe pour la suite, c'est la propriété récurrente du décalage d'un rang des congruence* ***dans les deux sens.***

D'où il ressort : *Si un entier* ***A≠ 2n[P]*** *premier ou pas, précède* ***son successeur A+2 et que A+2 = p'*** *un nombre premier* ***, 2n +2*** *se décomposera obligatoirement en somme de deux nombres premiers. Le contraire, contredirait le TFA , ainsi que le TNP et cela serait aussi, contraire au décalage récurrent d'un rang des congruences lorsque la limite* ***n*** *, augmente de* ***1****.*
Cette propriété permet aussi de prendre en compte tous les entiers ***A≠ 2n[P] premiers ou pas,*** *lors des limites* ***n+1*** *suivantes, ce qui n'a jamais été fait.]*

Exemple :

secteur des entiers impairs représentés par les congruence =[1,] ou [0,] où à chaque augmentation de 1, limite n les congruences se décalent d'un rang ; conséquence de l'égalité de l'algorithme de Goldbach A $\not\equiv$ 2n[P] **= (2n+2) $\not\equiv$ (A+2)[P]**.

[0, 1, 0, 0, 1, 1, 0, 0, 1, 0]
[1, 3, 5, 7 ,9 , 11, 13, 15, 17, 19 ,]

Donnez N: 20
*[3, 5] <….***nombre P ≤ √2n**
1 *<…..reste r de 2n par P*
0
crible: [0, 1, 0, 0, 1, 1, 0, 0, 1, 0] de 1 à 19
Nombres non congru 2n[pi] 1 à 20 famille 1 premiers de 20 à 40: 4 ----- 0.0
--- Temps total: 0.0 sec ---
>>>
======== RESTART: /home/gilbert/Programmes/Python/3. G_crible . mod 2.py ========
Donnez ***N: 21*** *;de 1 à 19* ***, n=21, si 2n ≠ p'+ q ,*** *alors il n'existe pas ligne* ***n-1 = 20****d'entiers* ***A≠ (2n-2) [P] qui précède un premier p' au rang n+1, et 4 nombres premiers q , en moins dans [n,2n] alors qu'il y en avait 5***
[3, 5]
0
2
crible: [1, 0, 1, 0, 0, 1, 1, 0, 0, 1]
Nombres non congru 2n[pi] 1 à 21 famille 1 premiers de 21 à 42: 5 ----- 0.0
--- Temps total: 0.0 sec ---
>>>
======== RESTART: /home/gilbert/Programmes/Python/3. G_crible . mod 2.py ========
Donnez ***N: 22****de 1 à 19* ***, n=22, si 2n ≠ p'+ q ;*** *alors il n'existe pas ligne* ***n-1 =21*** *d'entiers* ***A≠ (2n-2) [P] qui précède un premier p' au rang n+1 et 4 nombres premiers q , en moins dans [n,2n] alors qu'il y en avait 6***
[3, 5]
[3, 5]
2

4
*crible: [**1, 1**, 0, 1, 0, 0, 1, 1, 0, 0, 1] de 1 à 21 n =21 **2n+2 ≠ p'+ q ,** alors il n'existe pas ligne **n-1 = 20** d'entiers **A≠ (2n-2) [P]….etc***
Nombres non congru 2n[pi] 1 à 22 famille 1 premiers de 22 à 44: 6 ----- 0.0
--- Temps total: 0.0 sec –

Simulateur de Goldbach , modulo 2:

Nombre premiers **p'** ≤ N où 1 n'est pas un nombre premier,

[1, **3**, **5, 7**, 9, **11, 13**, 15, **17, 19**, 21, **23**, 25, 27, **29, 31**, 35, **37**, 39, **41 43**, 45, **47**, 49, 51**,]** 53, 59, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97**]**

[ce qui se traduit par le secteur des nombres premiers suivant] **: n ≤ 30**
[1, 1, 1, 1, 0**, 1, 1,** 0**, 1, 1,** 0**, 1,** 0, 0**, 1] 15** ***p'** = **1,** ou multiples de P = 0*

N point k, de 15 à X conjecture fausse par supposition,
A : impairs, de 1 à 19 congrus = 0 , non congrus =1

***Note:** Le cas 2n+2 est évident à montrer qu'il serra toujours décomposable , du fait de l'égalité du nombre de nombres premiers qu'il y a entre n et 2n , par rapport à n+1 et 2n +2, qui ne peut que varier de 0 ou 1.*

point k =30 ou la conjecture est supposée se réduire en nombre de décomposition , **jusqu'à X= 60**

[entiers impairs de **1 à n**] et ils sont ***indicé de 0 à n.*** On calcule le reste **R** de 2n par **P**
P = 3 , 5 ; **R** = 0 , 0 :
S*i **R**%2= 0 on fait (**R+P**) // 2 et **on part de l'indice** en le marquant 0 , ainsi que tous ses suivants modulo P ,*
*si **R**%2 != 0 , **R**//2 on part de l'indice que l'on marque 0 , ainsi que tous ses suivants modulo P.*
*On aura marqué **[,0,]** tous les entiers **congrus à 2n [P]**, à la fin on relève les **[,1,]** qui sont les entiers **non congrus à 2n[P]***

[1, 3, 5, 7, 9,11,13]
[*secteur des congruences représentant les A impairs **congru** = 0* **sinon** = 1]
[**1**, 0, 0, 1, 0, 1, 1,] **15**]n =15 // 2 = 7 +1 entiers impairs de 1 à 15
[1, **1**, 0, 0, 1, 0, 1, 1,] 16
[0, 1, **1**, 0, 0, 1, 0, 1,] **17**]
[0, 0, 1, **1**, 0, 0, 1, 0, 1,] 18
[1, 0, 0, 1, **1**, 0, 0, 1, 0,] **19**] si 2n = **p'+ p'** on ne peut supposer la conjecture fausse par évidence ,
[0, 1, 0, 0, 1, **1**, 0, 0, 1, 0,] 20
[1, 0, 1, 0, 0, 1, **1**, 0, 0, 1,] **21**] n=21 tout A ≠ p' tel que **A+A** ne peut être solution
[1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1,] 22
[0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0,] **23**]
[1, 0, 1, 1, 0, 1, 0, 0, 1, **1**, 0, 0,] 24
[0, 1, 0, 1, 1, 0, 1, 0, 0, 1, **1**, 0,] **25**] donc les cas n + n non premiers sont exclus
[0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0,] 26
[1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1,] **27**]
[0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, **1**,] **28** contradiction à supposition fausse , **27 ≠ p' et tous les autres sont congruent à P.**
[0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, **1**,] **29**] n=29 alors que la conjecture avait été validée.
on suppose que 2n+2 = 60 ne se décompose pas en la somme de deux premiers p+q
[1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1,] 30] entiers **A marqués en rouge = congrus à 2n[P]**
[1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1,]31] nombres premiers Ératosthène sont tous congrus !!!

1) certains entiers 2n – 2 qui ont vérifié la conjecture à l'exception de p'=p' = 2n , ils se sont trompés … ???

tous les entiers A de 1 à sont congrus à 2n[P] sauf 1 , il ne peut en rester qu'un, donc c'est la cellule 0 = 1.

Résultat , conséquence de la propriété récurrente , on fait la récurrence inverse , on marque en rouge tous les n – 1 des ligne n- 1 c'est à dire toutes les diagonales précédentes **!!! Ce qui est purement absurde car contradictoire.**
2) qu'un seul nombre premier de n à 2n ?
3) les entiers 2n -2 inférieur sont supposés avoir vérifiés la conjecture jusqu'à 2n , car elle est fausse à **2n+2**

Autre cas :

Simulation et illustration des entiers impair en progression arithmétique de raison 30 de premier terme 7 à partir d'une limite n = 300 , qui a vérifié tous les entiers pairs < 600 , avec les 8 familles, il est impossible d'infirmer la conjecture pour les entiers pairs < 780 .

Autrement dit il est impossible d'affirmer, que pour la limite suivante n+1 = 315 , aucun entiers pairs 2N ne s'écrirait pas comme la somme de deux nombres premiers (p'+q).

Simulation et vérification en utilisant simplement la famille **30k + 7** et les nombres **P ≤√2n**
les entiers pairs seront vérifiés → 2N = y = 780 , on **repousse la limite de 6**

Secteur gradué de : **1** = entier **non congrus** à **2n[P]** ; 0 = entier **congru à 2n[P]**

Donnez N: 300 , 600 →; [630; 660; 690; 720; 750; 780] y = 780

secteur des congruences représentant les entiers non congru =**1**,ou congrus à 2n[P] = 0
[**1, 1, 1, 1, 1, 1**, 0, 0, 0, **1**]
[**7,37,67,97,127,157**,187, 217, 247,**277**]

crible G: [**1**, 1, 0, 1, 0, 1, 0, 1, 1, 0] stop *fin du décalage vérifiant la conjecture jusqu'à 2n = 780* ,
crible E: [**1, 1, 1, 1, 1, 1**, 0, 0, 0, **1**] ⇒ entiers d'Ératosthène : [**7,37,67,97127,157**,187, 217, 247,**277**]

En vérifiant pour N = 390 , les entiers pairs < 2N = 780 , entiers pairs vérifiée → y = 1140 on **repousse la limite de 12**

Donnez N: 390, **2N = 780** → 810, 840, 870, 900, 930, 960, 990 , 1020 , 1050 ...1080 ,1110 , 1140 , y = 1140
crible G: [[**1**, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0]
crible E: [1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, **1**]

Donnez N: 1140, 2N = 1140 → entiers pairs vérifiés jusqu'à y = (1140+1110) **on repousse la limites de 37** , quasiment le double , C'est à dire : que lorsque l'on vérifie les entiers pairs inférieur à une limite 2n fixée , on vérifie par la même que tous les entiers pairs < à (2n/30)*2 seront vérifié

crible G: [[**1**, 1, 1, 0,
1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1]
crible E: [1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, **1**].

n=2250 , 2250/30 = 37*2 = 74 et **74*2** → donne la prochaine limite n 3840 et tous les entiers pairs < 7680 sont vérifiés
G :[**[1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1]**

É :[1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0]

Lorsque l'on décale *les congruences de A, on ne s'occupe absolument pas de la primalité ou pas de l'entier congru ou pas , car il s'agit uniquement des congruence que l'on décalent d'un rang….*

Simulation :
principe de base , à chaque fois que l'algorithme met un 0 à la ligne n+1, correspondant à son successeur ligne n et rang n – 1, on supprime donc la diagonal en amont car par le principe de la récurrence inverse on doit respecter l'égalité inverse dans les deux sens,
si l'algorithme met un 1 sous un 0 en amont , décalé d'un rang au rang n - 1 de la ligne N – 1 , on remplace donc le 1 par 0 , car (2n – A) ⇔ (2n+2 – A +2)
il est impossible de supposer la conjecture Fausse !

De 600 a 690 ligne 1 à ligne 28 propriété inverse suite au décalage d'un rang des congruence 1 ou 0
entiers impairs congru = 0 non congru = 1
crible: [0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0] j'ai enlevé 2 cellule [1] remplacé par une cellule [0] z
crible: [0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1]
crible: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1] ici aussi et ensuite on laisse faire l'algorithme
crible: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, , 0, 0, 1]
crible: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1] 660
crible: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0] 675

crible: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0] 690
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0] 705
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0] 720
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0] 735 ici on remplace un 0 au premier rang une fois sur deux
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0] 750 si l'algorithme met un zéro 0 on laisse faire
[1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1] , Ératosthène Fam 7[30] 7,37,67,97,127,157,**187,217** ,
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 0, 0, 0, 0, 1, 0, 0]…>décroissance des nombres premiers **q** il ne peut en rester qu'un
[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0]
[0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0]
[0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0]
[0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1] 825
[0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]
[0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] 855
[0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] **870** point **X,** où la conjecture est Fausse **187** n'est pas premier
[0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] **885**
[0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, **0**, 0, 0, 0] 900
[0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] 915G
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, **0**, 0, 0] 930G
crible: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 0, 0, 0, **0, 0**, 0] 945 **0 0**
[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 0, 0, 0]
[1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] 990
[0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, **1**, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] 1005 à 1245 la dernière diagonal de 0 disparaît

[0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, **0**, 0, 1]

par contre cela ne supprime pas le **dernier nombre premier q** entre n et 2n , mais dire qu'il n'en resterait qu'un est absurde
Autant dire que HR est fausse !

[1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1] Ératosthène 900
[1, 1, 1, 1, 1, 1, 0, 0, 0, **1**, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, **1**, 0, 0, 1] Ératosthène 915

conjecture fausse aucun impair n'est somme de deux premiers, conséquence est contradiction en amont les entiers 2n – a, 2n – b … etc qui ont pourtant vérifiés la conjecture se retrouverait non décomposables ce qui est idiot !
Par exemple pour 2n = 750,765,770 ou d'autres,
Mais aussi le nombre de nombres premier q ∈ [n;2n] qui a été dénombré il deviendrait faux !!! ce qui est complètement absurde ,

La supposition est donc fausse et la conjecture vraie
---

simulation à partir de N = 240 point K, où la conjecture est supposé fausse et va décroître = moins de décompositions
On remplace donc un [,1 ,] une fois sur deux rang 1 ligne n+2 si l'algorithme met un zéro on ne touche rien et on rectifie les diagonale , donc la ligne Ln+1 pour chaque rang n+1 , si son prédécesseur Ln , au rang n est un [0] on rectifie la cellule rang n+1 , Ln+1 si l'algorithme met un [1] pour respecter le décalage d'un rang récurrent de l'algorithme,

---

Famille 7 modulo 30 : on vérifiera avec Ératosthène à quel moment la conjecture devient fausse ligne Ln+1 = point X

1) crible réel on ne touche rien on fera une copie du tableau réel dans une 20ème de lignes
Et on verra les dégâts que cela aurait entraînés, si la conjecture était fausse.

crible: entiers impairs non congru = 1, **sinon** 0
(7 ,37 ,67 ,97 127 ,157 ,187]
1[0, 1, 1, 0, 1, 1, 1] n=210 ; 2n = 420 on l'algorithme progresse donc modulo 15(k),
2[1, 0, 1, 1, 0, 1, 1] n=225 à partir de cette ligne Ln+2 on remplace la cellule [1]par [0] , si il y a 0 on touche pas
3[0, 1, 0, 1, 1, 0, 1, 1]
4[1, 0, 1, 0, 1, 1, 0, 1]
[0, 1, 0, 1, 0, 1, 1, 0, 1]
: [1, 0, 1, 0, 1, 0, 1, 1, 0]
: [1, 1, 0, 1, 0, 1, 0, 1, 1, 0] n=300
: [0, 1, 1, 0, 1, 0, 1, 0, 1, 1]
: [1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1]
: [1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1]
: [0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1]
: [1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0] 375
: [1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0] 390
: [0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1]
: [0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1]420
: [1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0]435
: [0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0] 450
: [0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1] 465
: [1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1]
: [1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0]
: [1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0]510
: [0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1]525
: [0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1] **X = 540 conjecture vraie ,**

----------simulation----------

on va la supposer fausse en utilisant bien évidemment les mêmes entiers de la même famille,

Départ on l'a suppose fausse donc à partir de ce point K=2n =210 vers le point **X = 435 ou elle est fausse et on regarde inversement les contradictions que cela a entraînée sur les 2n -30, -60 …. jusqu'au point -30k =210**

1[0, 1, 1, 0, 1, 1, 1] n=210 ; 2n = 420 où l'algorithme progresse donc modulo 15(k),
2[**1,** 0, 1, 1, 0, 1, 1] n=225 à partir de cette ligne Ln+2 on remplace la cellule [1]par [0] , si il y a 0 on touche pas
3[0, 1, 0, 1, 1, 0, 1, 1]
4[0, 0, 1, 0, 1, 1, 0, 1]
[0, 0, 0, 1, 0, 1, 1, 0, 1]
: [0, 0, 0, 0, 1, 0, 1, 1, 0]
7[0, 0, 0, 0, 0, 1, 0, 1, 1, 0] n=300
: [0, 0, 0, 0, 0, 0, **1**, 0, 1, 1]…315……...le[0] est mis par l'algo, on ne fait rien
: [1, 0, **0**, 0, 0, 0, 0, 1, 0, 1, 1]
: [0, 1, 0, **0**, 0, 0, 0, 0, 1, 0, 1]
: [0, 0, 1, 0, **0**, 0, 0, 0, 0, 1, 0, 1]
: [0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0] n,375 …..> : [**1, 1, 1, 1, 1**, 0, 0, 0, **1, 1, 1**] **1=P =7,37,67,97,127,157,**187,217,247,**377,307,337 ok**
: [**0**,0, 0, 0, 1, 0, **0**, 0, 0, 0, 0, 1, 0]
: [0, **1**, 0, 0, 0, 1, 0, **0**, 0, 0, 0, 0, 1]
: [0, 0, **1**,0, 0, 0, **1, 0,** 0, 0, 0, 0, 0, 1]**420** on aurait pu la supposer fausse à ce point X en enlevant le **1** cellule Ligne **7 et 13**

: [0, 0, 0, **1**, 0, 0, **0, 1, 0**, 0, 0, 0, 0, 0] **435**…..**fausse**
: [0, 0, 0, 0, **1**, 0, **0, 0, 1**, 0, 0,0, 0, 0, 0] **450** à ce point **X =450 la conjecture deviendrait Fausse, alors que vérifiée vraie**
: [0, 0, 0, 0, 0, **1**, 0, 0, 0, 1, 0, 0, 0, 0, 0] 465 ……...>il ne reste plus que 3 nombre premiers **q[n,2n]** sur 7 ,
: [**0**, 0, 0, 0, 0, 0, **1**, 0, 0, 0, 1, 0, **0**, 0, 0, 0]480 …...on suppose conjecture fausse **X=555, on remplace cellule 1per 0**
: [0 ,**0**, 0, 0, 0, 0, 0, **1**, 0, 0, 0, 1, 0, 0, 0, 0]495 …..correspondant au **1** d'Ératosthène
: [**0**, 0, **0**, 0, 0, 0, 0, 0, **1**, 0, 0, 0, 1, 0, **0**, 0, 0]510
: [0, **0**, 0, **0,** 0, 0, 0, 0, 0, **1**, 0, 0, 0 1, 0, 0, 0] 525
: [0, 0, **0**, 0, **0**, 0, 0, 0, 0, 0, 0, **1**, 0, 0, 0, **1**, 0, **1**, 0] 540..>
: [1, 1, 1, 1, 1, 1, 1, **0, 0, 0**, 1, 1, 1, 1, 1, 1, **0**, 1, **1**, **0**] Ératosthène ,tous les **1** correspondant au 0 Goldbach sont marqué en rouge , car :

Conjecture supposée fausse , et, on remonte la diagonale des **1** que l'on remplace par **1= 0 , il faut en tout 10 cellules marquées [0] [0] dont 4 [0] supplémentaires ,** pour ne pas traîner en longueur
Conclusion :
1) Il reste bien un nombre **q[n,2n]** quelque soit la limite **n** ou on suppose la conjecture Fausse avec la conséquence que cela entraîne sur la répartition des nombres premier … donc aussi sur l'hypothèse de Riémann,

2) La conjecture serait aussi devenue fausse au point **X= 450** ce qui est absurde !

3) la répercutions des nombres premier se répercuterait aussi dans l'intervalle de **n à 1 en sens inverse.**

4) Donc impossible de la supposée fausse, la conjecture et donc varie !
= Simulateur par l'absurde

La simulation est faite par famille , on en ferait de même pour les 7 autre Fam , celle ci à l'avantage d'avoir beaucoup de nombres premiers de
7 à 157 et de 277 à 397….., ce qui rend le test plus long en supposant la conjecture fausse.

On peut aussi le faire avec le simulateur des entiers **Impairs 1[2]** …de **1 à n** , **et non de 1 à 2n , on utilise les congruences qui définissent par obligation le nombre premier q de la bonne Famille, ayant pour antécédent p'≤ n.**

Égalité récurrente *des algorithmes : **2n – A = (2n+2) – (A +2)** , ou :* **A≠ 2n[P] = (2n+2)≠ (A+2)[P]**.

Ce qui provoque le décalage récurrent d'un rang des congruences sur leur successeur ***A+2,*** *lorsque* ***n*** *augmente de* ***1.*** ***«*** *On peut admettre que cela agit comme une preuve par récurrence.* ***»***

simulation : Ce qui permet grâce à cette propriété et cette égalité : qu'il n'est nul besoin de recribler, si on veut savoir que la conjecture est vraie pour le prochain de 2n +2 , relatif à n+1 . ! Donc par le processus inverse si elle est supposée fausse on refait le chemin inverse, car chaque* <u>ligne n</u> *, dépends des lignes inférieur* <u>n – 1</u> *… Ce qui rend la conjecture impossible à infirmer**.** cqfd **.**

programme ci-joins, python modulo 2 utilisé :

****

Programme Python (fam 1 **modulo 2):**

```
from time import time
from os import system
import math

def candidate_range(n):
    cur = 5
    incr = 2
    while cur < n+1:
        yield cur
        cur += incr
        incr ^= 6  # or incr = 6-incr, or however


def eratostene(n):
    n = int((2*n)**0.5)
    prime_list = [3]
    sieve_list = [True] * (n+1)
    for each_number in candidate_range(n):
        if sieve_list[each_number]:
            prime_list.append(each_number)
            for multiple in range(each_number*each_number, n+1, each_number):
                sieve_list[multiple] = False
    print(prime_list)
    return prime_list[0:]


def demander_N():
    n = input("Donnez N: ")
    n = int(n.strip().replace(" ", ""))

    return n


def G_crible(premiers, n, fam):
    start_crible = time()
    crible = (n//2)*[1]     # Ou: on rapelle le tableau Ératosthène criblé de N/2 cases
    lencrible = len(crible)

    # On calcule les restes: ri = 2*n/pi
    nbpremiers = len(premiers)
    n2 = 2*n

    for i, premier in enumerate(premiers):
        reste = n2 % premier
        #print(reste)
        if reste % 2 == 0:
            reste += premier
        pi2 = 2*premier
        while reste % 2 != fam:   ## tant que ri % 2 != fam on fait ri += 2pi
            reste += pi2
                        # Ensuite on divise Ri par 2 pour obtenir l'indexe
        reste //= 2
                        # On crible directement à partir de l'index que l'on marque 0
```

```
        for index in range(reste, lencrible, premier):
            crible[index] = 0

    total = sum(crible)
    print("crible:", crible)
    print(f"Nombres non congru 2n[pi] {1} à {n} famille {fam} premiers de {n} à {n2}: {total} ----- {int((time()-
start_crible)*100)/100}")


def main():
    # On demande N a l'utilisateur
    n = demander_N()

    # On récupère les premiers de 3 à √2N
    premiers = eratostene(n)
    #lprint("premiers:", premiers)
    #print(f"nombres premiers de3 à {int((2*n)**0.5)}: {len(premiers)}")

    start_time = time()
    # On crible
    G_crible(premiers, n, 1)
    temps = time()-start_time
    print(f"--- Temps total: {int(temps*100)/100} sec ---")


main()
system("pause")
```

Programme Python Goldbach : Par Famille 30k + i , déjà réglé sur la famille 30k +7 **,** avec i ∈ [1,7,11,13,17,19,23,29]

```
from time import time
from os import system
import math

def candidate_range(n):
    cur = 5
    incr = 2
    while cur < n+1:
        yield cur
        cur += incr
        incr ^= 6  # or incr = 6-incr, or however


def eratostene(n):
    n = int((2*n)**0.5)
    prime_list = [2, 3]
    sieve_list = [True] * (n+1)
    for each_number in candidate_range(n):
        if sieve_list[each_number]:
            prime_list.append(each_number)
            for multiple in range(each_number*each_number, n+1, each_number):
                sieve_list[multiple] = False
    #print(prime_list[3:])
    return prime_list[3:]


def demander_N():
    n = input("Donnez N: ")
    n = int(n.strip().replace(" ", ""))
    #n = int(30 * round(float(n)/30))
    return n


def GCrible(premiers, n, fam):
    start_crible = time()
    crible = (n//30)*[1]     # Ou: on rapelle le tableau Ératosthène criblé de N/30 cases
    lencrible = len(crible)

    # On calcule les restes: ri = 2*n/pi
    nbpremiers = len(premiers)
```

```
    n2 = 2*n

    for i, premier in enumerate(premiers):
        reste = n2 % premier
        #print(reste)
                # tant que ri % 30 != fam on fait ri += 2pi
        if reste % 2 == 0:
            reste += premier
        pi2 = 2*premier
        while reste % 30 != fam:
            reste += pi2
        # Ensuite on divise ri par 30 pour obtenir l'indexe
        reste //= 30
        # On crible directement avec l'index
        for index in range(reste, lencrible, premier):
            crible[index] = 0

    total = sum(crible)
    print("crible:", crible)
    print(f"Nombres non congru 2n[pi] {1} à {n} famille {fam} premiers de {n} à {n2}: {total} ----- {int((time()-start_crible)*100)/100}")


def main():
    # On demande N a l'utilisateur
    n = demander_N()

    # On récupère les premiers entre 7 et √2N
    premiers = eratostene(n)
    #lprint("premiers:", premiers)
    #print(f"nombres premiers entre 7 et {int((2*n)**0.5)}: {len(premiers)}")

    start_time = time()
    # On crible
    GCrible(premiers, n, 7)
    temps = time()-start_time
    print(f"--- Temps total: {int(temps*100)/100} sec ---")


main()
system("pause")
```

Programme Python Ératosthène : déjà réglé sur la famille 30k +7 , les deux famille doivent toujours être les mêmes car on crible Goldbach avec les congruence, ce qui évite de s'occuper de la famille complémentaire de p' premiers d'Ératosthène dont aucun intérêt,

lorsque 7 est congru à 2n modulo 30 , on sait très bien que q = 23 [30], autrement dit 60 – 7= 53 =23[30] ,

---

Ératosthène :

```
from itertools import product
from time import time
from os import system
import math

def candidate_range(n):
    cur = 5
    incr = 2
    while cur < n+1:
        yield cur
        cur += incr
        incr ^= 6  # or incr = 6-incr, or however


def eratostene(n):
    n = int(n**0.5)  #(si on fusionne les deux cribles il faudra rentrer, int((2n)**0.5) pour Goldbach.
    prime_list =[2,3]
    sieve_list = [True] * (n+1)
    for each_number in candidate_range(n):
        if sieve_list[each_number]:
```

```
            prime_list.append(each_number)
            for multiple in range(each_number*each_number, n+1, each_number):
                sieve_list[multiple] = False
    #print(prime_list[3:])
    return prime_list[3:]


def demander_N():
    n = input("Donnez N: ")
    n = int(n.strip().replace(" ", ""))
    #n = int(30 * round(float(n)/30))
    return n


def E_Crible(premiers, n, fam):
    start_crible = time()

    # On génère un tableau de N/30 cases rempli de 1
    crible = n//30*[1]
    lencrible = len(crible)
    GM = [7,11,13,17,19,23,29,31]
    # On calcule les produits: j = a * b

    for a in premiers:
        for b in GM:
            j = a * b
            if j%30 == fam:
                index = j // 30  # Je calcule l'index et On crible directement à partir de l'index
        for idx in range(index, lencrible, a):  # index qui est réutilisé ici...
            crible[idx] = 0
        #print(index)

    total = sum(crible)  #à la place, pour utiliser le tableau d'Ératosthène criblé dans le crible de Goldbach, on return "crible:", crible
    print("crible:", crible)
    print(f"Nombre premiers criblés famille {fam} : {total} ----- {int((time()-start_crible)*100)/100}")


def main():
    # On demande N a l'utilisateur
    n = demander_N()

    # On récupère les premiers de 7 à √N
    premiers = eratostene(n)
    #print(f"nombres premiers entre 7 et n: {len(premiers)}")

    start_time = time()
    # On crible
    E_Crible(premiers, n, 7)  # au choix (1,7,11,13,17,19,23,29)
    temps = time()-start_time
    print(f"--- Temps total: {int(temps*100)/100} sec ---")


main()
system("pause")
```
