Ł.g, **Conjecture ou algorithme de Goldbach** :

Supposition que l'algorithme de Goldbach pour une limite ***N = 615*** *fixée, l'entier* ***2N = 1230*** *n'a pas de décompositions en somme de deux nombres premier (****p'+ q****) ; ie) Aucun entier* ***A= p'*** *premier, de* ***7 à 615*** *n'est congru à* ***2N[P]*** *; Pour la famille 30k+7 ou 7 modulo 30, n'a aucune décomposition tel que ;* ***p'+ q = 2N = 1230****.*

(« *Plus généralement , on peut dire que pour toute Limite N criblée, ayant dénombrer le nombre de décompositions de 2N , alors on peut utiliser le résultat du criblage ≤ à cette limite N, pour dénombrer le nombre de décompositions de* ***2n + 2 = p'+q****, cela est* ***dû à la propriété récurrente de l'algorithme*** *de Goldbach, qui crée un système dynamique et va s'étendre sur plusieurs limites* ***N*** *successives lorsque N → ∞ . ie:) ⇒ les entiers* ***2N*** *successifs, d'où l'impossibilité de supposer que 2N +2, +4, +6, +X<N ; n'auraient pas de décompositions.*
*On peut réitérer indéfiniment cette limite N, qui donnera un résultat en repoussant de plus en plus loin ces limites N successives, où il est impossible de supposer la conjecture fausse.****»)***

Explication *:*

Si ***1 ≡ 2N[P]*** *lorsque la limite* ***N*** *augmente de 1, l'entier* ***2N*** *augmente de 2 , donc* ***(1+2) ≡ 2N+2 [P]****, avec* ***P*** *premier ≤√****2N****.*
(« Exemple : ***1 ≡ 28 [P] d'où (1+2) ≡ 30 [P], 28 – 1 ⇔ 30 – 3 ≠ q*** *premier, Inversement :*
si [1] tel que 1 ≢2N[P], il initialisera une diagonale d'entiers A non nul et non congrus à 2N modulo P, qui seront marquées 1.
Exemple : 1 ≢30 [P] d'où (1+2) ≢32 [P] , 30 – 1 ⇔ 32 – 3 = q premier. le contraire contredirait le TFA et le TNP. »)

Conséquence directe de ***(2N – A) ⇔ (2N +2) – (A + 2) on a un*** *décalage d'un rang des congruences, lorsque* ***N*** *augmente de 1 , donc 2N augmente de 2 , propriété récurrente , on a l'égalité : équivalente à (A+2)≢(2N+2) [P] ou à l'inverse (A+2) ≡ (2N+2) [P]..****»)***

On va travailler dans une des *8 familles* ***i*** *modulo* ***30*** *avec* ***i*** *appartenant à [* ***1.7.11.13.17.19.23.29*** *] sans perte de généralité. On aura bien par famille* ***30k + i :*** *cette égalité récurrente,* ***(2N – A) ⇔ (2N +30) – (A + 30),*** *lorsque* ***N*** *augmente de* ***15*** *et où* ***A*** *est un entier naturel positif de* ***i*** *à la limite* ***N fixée,*** *quelque soit une de ces 8 familles, en fonction* ***de la forme de N.***

1)
Par famille : On connaît le nombre de nombres premiers **p'** de **i** à **N = 600** et on connaît le nombre de nombres premiers **q** de **600 à 1200 , car on connaît** le nombre d'entiers **A ≢ 1200[P] , de i à 600** *que l'on peut nommer un* ***[champ*** *de congruences****]*** *représenté par des 1 ou des 0, tel que : [1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0] qui se décalera d'un rang pour toute limite N+1.*

2)
En augmentant la limite **N** ou **n de 15**, le nombre de nombres premiers **p'** de **i à 615** ainsi que le nombre de nombres premiers **q** de **615 à 1230** *ne peut varier tout au plus, que d'une unité entre ces deux intervalles ; de* ***[i à 600 ; i à 615] ce qui revient à dire, ∓1 entier A ≢ 1200[P].***

3)
On à vérifié la conjecture , « criblée par l'algorithme », jusqu'à la limite **n = 600** par pas de 15 , du fait que l'on travaille avec une famille , c'est à dire la famille **i = 7 modulo 30** . Tous les entiers pair de 2n = 1200 , 2n -30 , 2n - 60 …. etc ; se décomposaient bien en somme de deux nombres premiers **p' + q .**

4)
On suppose donc, que la conjecture « l'algorithme » est faux pour n +15, donc pour 2n+30 = 1230 ≠ p'+ q .

5) On va simuler cette supposition : 2N=1230 ne se décompose pas en la somme de deux nombres premiers, afin de constater les aberrations et conséquences qui en résulteraient !

(« *La première de ces aberrations est telle, qu'il faudrait que le postulat de Bertrand se réalise ; c'est à dire qu'il existe une limite* ***n > 14*** *où entre* ***n et 2n*** *, il existe effectivement un seul ou deux nombres premiers, autrement dit :*
un seul ou deux ***A≤ n, non congru à 2n[P].***

Ce qui est impossible, car pour les limites précédentes ***n – 1 ou n –15 par famille i,*** *on connaît déjà le résultat, qui par supposition , a été vérifié et contredirait la propriété récurrente de l'algorithme, sur plusieurs limites* ***n*** *successives.!*

D'où il est impossible, qu'il ne resterait qu'un seul nombre premier ***q appartenant à [615 ; 1230] . Autrement dit,*** *qu'il ne resterait qu'un seul nombre A ≢* ***2n[P].****») Ce que nous allons simuler , de 450 à 600.*

Résultat des 20 criblages précédant la limite n = 615 . Résultat du nombres de nombres premiers 1= **p'** de 7 à 600 de raison 30. (7,37,67,97….etc . N = 600. «*limites N précédentes diminue de raison 15 et* propriété récurrente inverse»
Ératosthène : premiers p' de 7 à 600, représenté par **1**, les **0** sont les multiples de **P ≤ √600.**
Donnez **n** : 600
crible E: [1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1]
Nombre premiers criblés famille **i = 7** : total = ***15***

Goldbach entiers **A** de 7 à 600 , **non congrus** à 1200 modulo **P ≤ √1200 ,** représenté par **1 , 0 =** congrus à **2n**[**P**].
Donnez n: 600
crible G: [1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0] *les 0 sont congrus a 1200[P]*
Nombres **A=1 non congrus à 2n[P]** de 7 à 600 *famille i=7*, (⇒ les nombres **q** de 600 à 1200) : total = ***11***

Donnez n – 15 = 585
crible **E**: [1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1]
Nombre premiers criblés famille 7 : 14

Donnez n: 585
crible **G**: [1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0]
Nombres non congrus 2n[P] 7 à 585 famille 7, ⇒ nombres q de 585 à 1170: 10

Donnez n – 15: 570
crible **E** : [1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1]
Nombre premiers criblés famille 7 : 14

Donnez n: 570
crible **G**: [0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1]
Nombres non congrus 2n[P] 7 à 570 famille 7, ⇒ nombres q de 570 à 1140: 10

Donnez n – 15: 555
crible **E**: [1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0]
Nombre premiers criblés famille 7 : 13

Donnez n: 555
crible **G**: [1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1]
Nombres non congrus 2n[P] 7 à 555 famille 7, ⇒ nombres q de 555 à 1110: 10

Donnez n: 540
crible **E**: [1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0]
Nombre premiers criblés famille 7 : 13

Donnez n: 540
crible **G**: [0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1]
Nombres non congrus à 2n[P] 7 à 540 famille 7 ⇒ nombres q de 540 à 1080: 10

Donnez n: 525
crible **E:** [1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1]
Nombre premiers criblés famille 7 : 13

Donnez n: 525
crible **G**: [0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1]
Nombres non congrus à 2n[P] 7 à 525 famille 7 ⇒ nombres q de 525 à 1050: 10

Donnez n: 510
crible **E**: [1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1]
Nombre premiers criblés famille 7 : 13

Donnez n: 510
crible **G**: [1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0]
Nombres non congrus à 2n[P] 7 à 510 famille 7 ⇒ nombres q de 510 à 1020: 10

Donnez n: 495
crible **E**: [1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1]
Nombres premiers criblés famille 7 : 12

Donnez n: 495
crible **G:** [1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0]
Nombres non congrus à 2n[P] 7 à 495 famille 7 ⇒ nombres q de 495 à 990: 9

Donnez n: 480
crible **E**: [1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1]
Nombre premiers criblés famille 7 : 12

Donnez n: 480
crible **G**: [1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1]
Nombres non congru à 2n[P] 6 à 480 famille 7 ⇒ nombres q de 480 à 960: 9

Donnez n: 465
crible **E**: [1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0]
Nombre premiers criblés famille 7 : 11

Donnez n: 465
crible **G**: [0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1]
Nombres non congru à 2n[P] 6 à 465 famille 7 nombres q de 465 à 930: 8

Donnez n: 450
crible **E**: [1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0]
Nombre premiers criblés famille 7 : 11

Donnez n: 450
crible **G**: [0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0]
Nombres non congru à 2n[P] 6 à 450 famille 7 ⇒ nombres q de 450 à 900: 8

---

n = 435 ; crible **E**: [1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1] ; 11 Nombres **p'**
n = 435 ; crible **G**: [1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0] ; 8 nombres **q**

****

Résultat des nombres premiers **p'** criblés par l'algorithme de Goldbach, pour **n = 600 ;** ce qui donne pour **2n = 1200** le nombre de décompositions en sommes de deux nombres premiers **p' + q ⇔ p'≠ 2n[P].**

Donnez N ou n : 600 , on crée [un **champ** de congruences] que l'on décalera d'un rang, sur le champ d'Ératosthène.

crible Goldbach : [**1, 1**, 0, **1**, 0, 0, **1, 1, 1**, 0, 0, **1,** 0, 0, **1, 1**, 0, **1, 1**, 0,]]
Nombres **A** non congru 2n[P] **1 à 600** famille 7 **:** ⇔ premiers de **600 à 1200: 11**
[**1, 1**, 0, **1**, 0, 0, **1, 1, 1**, 0, 0, **1,** 0, 0, **1, 1**, 0, **1, 1**, 0,] *limite N+15* on décale d'un rang
crible Ératosthène : [1, 1, 1, **1,** 1, 1, **0, 0, 0,** 1, 1, **1,** 1, 1, **0, 1,** 1**, 0, 1,** 1**,]** 1] les **1** en **gras** = p' ≠2N[P]

Nombres premiers **p'** criblés **famille 7 : 16 ----- 0.0 et p' ≠2N[P] 6** ⇔ **(p'+ q =1200)** et 9 pour 1230 minimum

Pour la limite N+15 il suffit de décaler d'un rang le champ des congruences ci-dessus, de N = 600 , où le seul inconnu est le premier terme *, autrement dit : on ne peut supposer la conjecture fausse pour les limites suivantes de N + 1 à N +15..etc. («À noter que ce champ de congruences augmentera pour toute limite N+1 criblée, ce qui rendra impossible une conjecture fausse .!»)*

crible É et G unifié **:** [1, **1, 1**, 0, **1**, 0, 0, **0, 0, 1**, 0, 0, **1**, 0, 0, **1, 1**, 0, **1, 1**, 0] limite *N= 615*
Nombres de **p' non congru 2n[P] ou couple p'+ q = 2N = 1230 , de (i) à 607** famille i=7 **: 1 + 9**

Tableaux de la simulation en supposant que **1230** ne se décompose pas en sommes de deux nombres premiers **p' + q** :

Conséquences : cela veut dire que tous les nombres **A = p'** de 7 à 600 sont donc congrus à 1230 mod P,

ce qui se traduit par les **A = p'** représentez par **1**, sont donc remplacés par des 0 ou marqués en rouge et les entiers A de Goldbach représentez par des 1 deviennent 0, c'est à dire congrus à 2n [P] , de par cette supposition : conjecture fausse.

a) crible **G** : [0, 0, 0, 0, 0, 0, 0, 0, **1**, **1**, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] n = 615
crible E: [1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1]
il ne resterait plus que **2** nombre premiers **q = 1013** et **983** au lieux de **11** vérifiés précédemment de 600 à 1200 …
Ce qui est clairement impossible , car contraire au TFA et à la propriété récurrente de Goldbach , ces nombres premiers **q** ne peuvent pas devenir des multiples de **P, du fait que l'on est augmenté la limite n de 15** . ie :

(2N – A) ⇔ (2N +30) – (A + 30)

b) <u>Cela implique aussi :</u> que pour **n = 600 vérifié précédemment,** il ne peut y avoir d'entiers **A≠ 2n[P] qui précèdent** un nombre nombre premier **p' = A+30 ,** suite au décalage récurrent d'un rang des congruences, lorsque **n** augmente de **15; «** *propriété récurrente de l'algorithme de Goldbach* » !

Ce qui se traduit par des **0 à la place des** 1 *(« deviennent congrus à 2n[P] , au lieu de non congrus… »)*

crible G: [**0, 0**, 0, **0**, 0, 0, **1, 1**, **0**, 0, 0, **0**, 0, 0, **0**, **0**, 0, **0, 0**, 0] n = 600 il reste les **2** mêmes **1** ⇒ **q = 1013 , 983**
crible E: [1, 1, 1, 1, 1, 1, **0, 0**, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1] seul deux entiers **A = 0** =187 et 217 **sont non** congrus.

Il vient donc aussi, que la conjecture qui avait été vérifiée précédemment bonne, pour 2n =1200 ; elle ne l'ait plus ??
Paradoxe ou Miracle ...? *Imaginons alors, qu'il en serait de même pour les 7 autres famille* ***i[30]*** **...**

c) Il en est donc de même pour n = 585. les A = 1 qui précèdent de deux rangs, un nombre premier p' = A + 60 ne peut être non congrus à 2n [P] , donc ce 1 , devient congrus = 0. Ce qui se généralise sur les limites n précédentes… !

crible G: [0, 0, 0, 0, 0, **1**, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] n = 585
crible E : [1, 1, 1, 1, 1, 1, **1**, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1] il n'y a plus qu'une seule décomposition de 1170 au lieu de 10 , qui avaient été vérifiées précédemment ! Imaginons alors les conséquences, si **n = 10^{20} .**

d) Il est clairement impossible de supposer que pour **2n +30** ou plus généralement pour **2n + 2,+4,+6,+X** que la conjecture soit fausse !
<u>Seule « supposition »</u> qui pourrait exister : il existe une limite **n = y** suffisamment grande , à partir de laquelle le nombre d'entiers **A non congrus à 2n [P]** va chuter , afin que la conjecture soit fausse à la limite **X > y**, d'où **2X** ne se décomposera pas en somme de deux nombres premiers **p' + q** , afin de ne pas contredire le TNP ou la répartition des nombres premiers, ainsi que les 2 fonctions du TNP et la propriété récurrente de cet algorithme.

Ce qui contredit tout ce qui a été calculé sur cette conjecture et sur la répartition des nombres premiers à nos jours car :

1-) Cela contredirait l'estimation du TNP qui serait fausse, car il viendrait aussi que la fonction d'estimation du nombre de nombres **A ≠ 2n[P] ⇔ q ∈ [n ; 2n]** qui est une conséquence directe du TNP (« *théorème des nombres premiers* ») et qui est démontrée , serait fausse aussi …?

Fonction qui vaut ~ : $\lim_{n \to +\infty} \frac{n}{(\log 2\, n)}$.

2-) Mais aussi, où il est affirmé qu'il existe un entier **y** suffisamment grand à partir de laquelle la conjecture serait vraie !

Dire qu'il y a au moins plus d'un nombre premier **q** entre **n et 2n** , n'a aucun sens ; car supposons que pour la limite **n = 3000 000 000 030** et supposer qu'il n'y aurait pas de décomposition pour **2n = 6000 000 000 060** comment explique t'on la disparition de **15 253 010 888** nombres premiers **q** entre **n** et **2n, qu'ils y avaient lors de la limite précédente entre [n – 15 ; 2n – 30]** (« *au lieu d'une différence de* ***5*** *au maximum, en utilisant les 8 familles* **i** ») **?**

En utilisant le postulat de Bertrand ? Et cela**,** quelque soit les limites n – k30 vérifiées précédemment qui seraient fausses.
Alors que pour la ou les limites précédentes, entre **(n – 15)** et **(2n – 30)** , c'est à dire que **2n = 6000 000 000 030** qui se décomposait, il y avait pour les 8 familles de nombres premiers **A ≠ 2n[P]⇔ q**, *103 041 128 336* que l'on peut estimer avec :
La **fonction 2** du **TNP**, qui nous donnes au moins : *101 961 811 234 nombres premiers* **A ≠ 2n[P] ⇔** ***q de n à 2n.***

La **fonction 2 du théorème** de Goldbach**, corollaire du TNP en** est une conséquence directe de ce **TNP:**
(les.mathématique.net Mr Poirot); (log = logarithme naturel)*

G(n): la fonction de compte du nombre de nombres ***A≢ 2n[P] inférieur à n*** ⇔ **q** ∈ **[n;2n]**

*** Corollaire du TNP** : **G(n)** vaut $\lim_{n\to+\infty} \frac{n}{(\log 2n)}$

Le TNP dit que $\pi(n) = \frac{n}{(\log n)} + o(\frac{n}{\log n})$, donc le nombre de nombres premiers dans]*n*,2*n*] vaut

$$\pi(2n) - \pi(n) = \left(\frac{2n}{\log(2N)} - \frac{n}{\log N}\right) + o(\frac{n}{\log n})$$

$$\pi(2n) - \pi(n) = \left(\frac{2n}{\log(2N)} - \frac{n}{\log N}\right) + o(\frac{n}{\log n})$$

$$= n \times \left(\frac{2}{\log 2n} - \frac{1}{\log n}\right) + o(\frac{n}{\log n})$$

$$= n \times \frac{2\log n - \log(2n)}{\log(2n)\log n} + o(\frac{n}{\log n})$$

$$= \frac{n}{(\log 2n)} + o(\frac{n}{\log n})$$

Ceci termine cette simulation , qui va à l'encontre d'une conjecture de Goldbach Fausse, qui impacterait fondamentalement la répartition des nombres premiers ***q*** *et par la même le* ***TNP*** *, le* ***TFA*** *(« théorème Fondamental de l'Arithmétique ») …!*

On peut conclure que pour toute limite ***N ≥ 4*** *fixée, on peut considérer la conjecture de Goldbach comme un algorithme de décomposition d'un entier naturel positif* ***pair ≥ 6*** *en somme de deux nombres premiers* ***p' + q***.
Pour cela on utilise deux algorithmes, qui vont cribler les entiers naturels ***A*** *positifs de* ***1 à N*** *avec :*

1) *Celui d'Ératosthène en criblant les entiers A premiers* ***p',*** *de* ***1 à N*** *avec* ***P ≤ √N, P premier,*** *qui sont extrait en début de programme et* ***q****ui criblent avec P à partir de ses indexes défini par l'algorithme.*

Puis avec :

2)*Celui de Goldbach qui va recribler avec* ***P*** *à partir* ***de ses*** *indexes* ***différents*** *de ceux d'Ératosthène; les entiers naturels A ≢ 2N [P],* ***de 1 à N, mais*** *avec* ***P ≤ √2N,*** *ce qui ⇒* ***q ∈ [N;2N] .***

Par conséquent si A = p', tel que A ≢ 2N [P] ⇒ ***q*** *premier, on obtient obligatoirement la décomposition de* ***2N = p'+q ,*** *où* ***p' et q*** *ne sont donc pas indépendant l'un de l'autre, ils dépendent des congruences.*

Il est donc clair *: que cette supposition fausse («simulation») est impossible, pour tout* ***n > 10^{20}*** *, la rigueur du TFA et du TNP , rend impossible cette supposition pour* ***n > 10^{20} + 1 , ie :*** *le remplacement des nombres premiers* ***A= p' ≢ 2n[P]*** *par des entiers* ***A≡2n[P] sans contredire*** *la propriété récurrente de Goldbach, déjà vérifiés lors des limites précédentes, N – 1 , N – 2 … etc* **.**

Il est aussi impossible *d'utiliser les restes* ***R*** *de la division euclidienne de* ***2n par P ,*** *lors des ou du criblage des* ***limites n*** *précédentes, qui sont limités par les nombres* ***P ≤ √2n*** *qui criblent chaque limite* ***n*** *et où les restes* ***R*** *changent pour la limite suivante,* ***n+1****..!*
De la même façon que les indexes de départ des nombres ***P*** *qui criblent, sont différents entre Ératosthène et Goldbach, ce qui nous assure un minimum de solutions de* ***2n = p'+ q****, pour toute limite* ***n*** *fixée.*

Il en est de même de la rigueur du TNP, qui interdit le changement du nombre de nombres premiers ***q,*** *limité par* ***les nombres premiers P≤√2n qui criblent en partant de leur index,*** *pour toute limite* ***n*** *fixée qui augmenterai de* ***1 , ie 2n + 2*** *conséquence du TFA et cette égalité par famille i :* ***(2N – A) ⇔ (2N +30) – (A + 30)*** *.*

Conclusion : À moins de démontrer, qu'il existe effectivement une démonstration rigoureuse à cette conjecture, les arguments évoqués sont suffisamment convaincants, pour dire que cette conjecture est un algorithme de décomposition de 2n en somme de deux nombres premiers et ne peut donc être supposée fausse, pour toutes limites **2n + 2 > 300** quelque soit une des 8 Famille **i** fixée, par rapport à la forme de **n** (« *voir en fin de document* »).!

On peut d'ailleurs admettre une troisième fonction , donnant un minimum de couple de premiers (p'+ q) pour tout 2n ≥ à 300 par famille **i**, avec les fonctions déduite du TNP : ($\frac{n}{(\log n)}$ et $\frac{n}{(\log 2n)}$) ⇒ $\frac{n}{(\log n)^2}$ *en divisant par 8 ce résultat, si on travaille que sur une des 8 familles.*

Ce n'est qu'une conséquence du TNP et de la propriété récurrente de cet algorithme de Goldbach .
La probabilité de tirer un nombre premier ***q*** *, c'est à dire un entier* ***A ≢ 2n[P]*** *vaut environ* $\frac{1}{\log 2n}$.

D'où, la probabilité de tirer une entier ***A < n****, tel que* ***A≢ 2n[P]*** *est un nombre premier* ***p',*** *vaut environ :*
$$\frac{1}{(\log 2n)(\log n)}$$

Ce que l'on peut aisément vérifier, jusqu'à de grandes limites ***n*** *…*

Donnez n: 300
P≤√600 =[7, 11, 13, 17, 19, 23]
p' = 7, 37, 67,... *crible Ératosthène :* ***[1, 1, 1, 1, 1, 1****, 0, 0, 0,* ***1, 1****] Nombre premiers* ***p'*** *criblés pour la famille 7[30] total* ***8***

crible ***É*** *et* ***G****: [1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1]*
Nombres ***p'*** *non congru* ***2n[P]*** *ou couple* ***(p'+ q)*** *= 2n, de 7 à 300 famille 7[30] :* ***5***

Estimation minimum par cette fonction, pour cette famille : (300÷(ln 300^2))÷8 = 1,1527…
Cette estimation ne sera jamais = 0, elle serra toujours positive pour tout n.

Annexe :
Illustration précédente sur une supposition fausse pour **2n = 1230 ,** impacte de la conséquence pour cette famille complémentaire 23[30]sur le nombre de nombres premiers **q** qui viendraient à disparaître sur plusieurs intervalles **n ; 2n**
Limite **n** criblée **de 435 à 615 et rectifiée en supposant** *la conjecture fausse, on remplace donc les 1 par 0* **:**

n = 435 ; crible **G**: [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0]
n = 450 ; crible **G**: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0]
n = 465 ; crible **G**: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1]
n = 480 ; crible **G**: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1]
n = 495 ; crible **G:** [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0]
n = 510 ; crible **G**: [1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0]
n = 525 ; crible **G**: [0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1]
n = 540 ; crible **G**: [0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1]

n = 555 ; crible **G**: [0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]
n = 570 ; crible **G**: [0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]
n = 585 ; crible **G**: [0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
n = 600 ; *crible* **G**: [0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] *conjecture* ***serait fausse*** *aussi pour* ***1200***
n = 615 ; ***crible*** **G**: [0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] *supposition conjecture Fausse pour cette limite*
n = 615; **crible E**: [1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1] *entiers* 1= p' qui seraient ≡ 2n[P] *par supposition*

On remplace les A = 1 non congrus à 2n [P] par 0 congrus à 2n [p] suivant cette supposition, ce qui permet de constater l'impacte, par rapport à la conjecture vérifiée jusqu'à ***n = 600*** *en début de document et deviendrait fausse pour* ***2n = 1200*** *; alors qu'elle était vraie. Le* ***[champ des congruences]*** *s'en trouverait faussé ce qui est clairement impossible.*

Il est le clair que le nombre de premiers ***q ∈ [n;2n] deviendrait faux, s****uite à ce décalage d'un rang des congruence, lorsque la limite* ***n*** *augmente de 15, on réitère le décalage inverse , pour être conforme à cette supposition de conjecture fausse et à la propriété récurrente de l'algorithme, dans les deux sens…*

Quand serait il, si pour la limite ***n+1*** *, donc* ***2n+2 = 1232*** *ne se décomposait pas en somme de deux nombres premiers ? Il n'y a que trois familles qui décomposent* ***1232 = p'+ q ; les Fam : 30k +1 ; 30k + 19 et 30k + 13***

Illustration pour ces trois Familles suivant le même principe 1202 ne se décompose pas en somme de deux premiers p' + q :

Donnez N: 601
Nombres non congru ***2n[P] , 1*** *à 601 famille* ***1[30]*** *premiers de 601 à 1202:* ***11*** *nombres q = 1[30]*

crible: [1, 1, 0, 0, 0, **1, 1, 1,** 0, 0, 0, 0, 0, **1,** 0, **1,** 0, **1, 1, 1, 1, 1]*** ...11 nombre premiers **q = 31[30]**
crible:[1[**1, 1, 1, 0, 0, 1, 1,** 1, 1, 1, 1, 0, 1, 0, 0, **1,** 0, 0, 0, **1, 1]** 1 n'est pas un nombre premier. Pas de couples p'+ q
Alors que pour la limite précédente N -15 = 586 et vérifiée , il en était prévue 3 .
31 ; 61 ; 91 ; 121 ...etc
Pour N : 601Nombre premiers p' criblés famille 1 : 12

Donnez N: 601 fam 13[30] [13,43,63,73,...etc.]
crible: [0, 0, **1,** 0, **1, 1, 1,** 0, 0, **1,** 0, **1, 1,** 0, **1, 1, 1,** 0, 0, **1,** 0***]*** ...11 nombre premiers **q = 19[30]**
crible: [1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0] Pas de couples de premiers p'+ q par supposition , or : il en a été vérifiée pour la limite précédente N = 586 : 6 … ?
Nombre premiers criblés famille 13 : 13

Donnez N: 601, Fam 19[30] , [19,49,79,109...etc.]
crible: [0, **1, 1, 1, 1,** 1, 0, 0, 0, 0, **1, 1, 1,** 0, 0, **1,** 0, **1, 1, 1,** 0***]*** … 12 nombres premiers **q = 13[30]**
crible:[**1**, 0, **1, 1, 1**, 0, **1, 1**, 0, 0, 0, **1, 1, 1, 1,** 0, **1**, 0, 0, 0***]*** Pas de couples de premiers p'+ q par supposition , or : il en a été vérifiée pour la limite précédente N = 586 : 5 … ?
Nombre premiers criblés famille 19 : 11

Constat : pour la limite **N = 586** , on avait comptabilisé et vérifiée 35 nombres **q** entre 601 et 1202 , or si la conjecture était fausse pour cet entier **2N = 1202** , il manquerait 14 nombres **q … ?**

Résultat avec le crible E,G
Donnez n: 601 crible E,G Fam 1[30] 1 n'étant pas premier , il ne peut être un coupe **p'+ q**
crible Ératosthène: [1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1]
Nombre premiers criblés famille 1 : 12 ----- 0.0
crible **G** sur **É**: [1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1]
Nombres **P'** non congruent à **2n[P]** de 1 à 601 = 7; famille 1, nombre de couples **p + q** = 2n: 7-1 = 6.

Alors que dans le meilleur des cas possibles, il ne pourrait y avoir qu'une différence de 1 nombre q , lorsque la limite N augmente de 15 ..!

Imaginons alors les conséquences qu'ils en seraient, pour de grande limite ***N*** *vérifiée ..!*

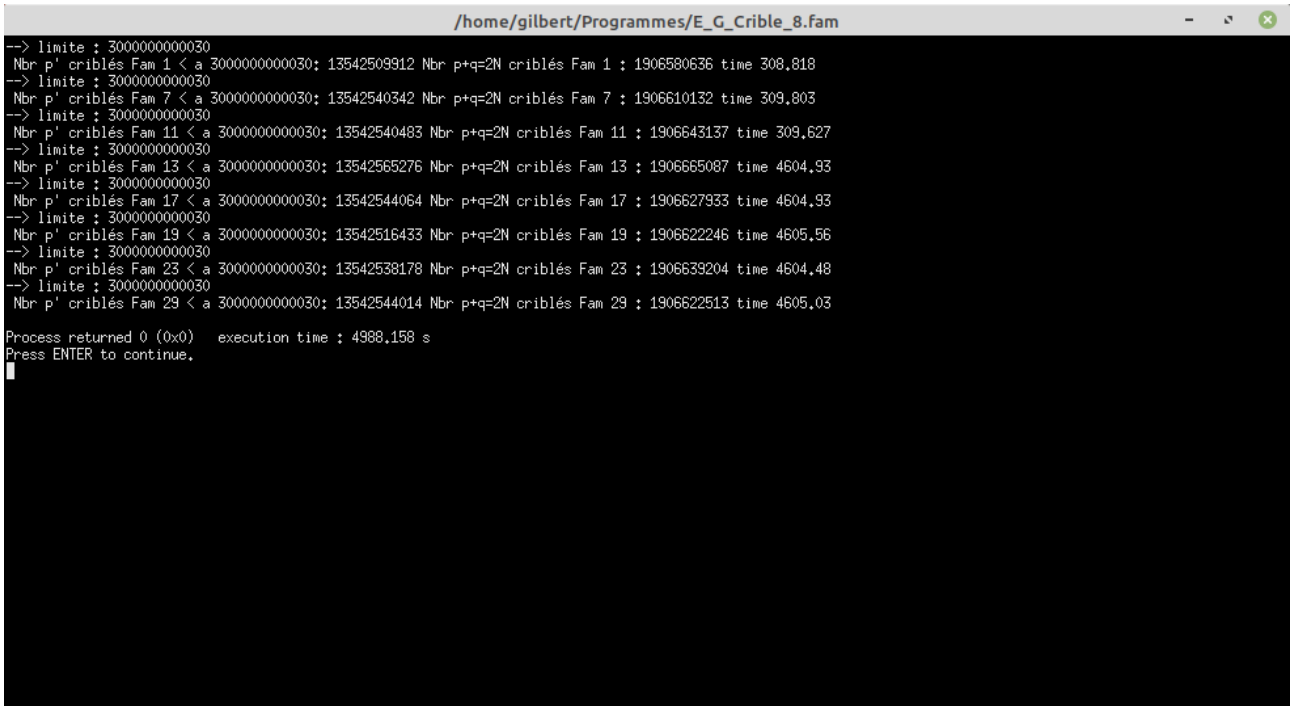
****

Résultat de la décomposition de ***2n*** *en somme de deux premiers par famille* ***i modulo 30***
Soit un total de 15 253 010 888 décompositions *de 6000 000 000 060 en somme de deux premiers p'+ q.*

Ce qui au minimum, « car cela se répercuterait sur les limites ***n*** *précédentes criblées », supprimerait autant de nombres premiers* ***q*** *entre* ***n*** *et* ***2n*** *pour les 8 familles, si la conjecture était fausse … (Supposition pour le moins absurde, dans l'état actuel ... qui contredirait aussi les 2 fonctions d'estimation du TNP.)*

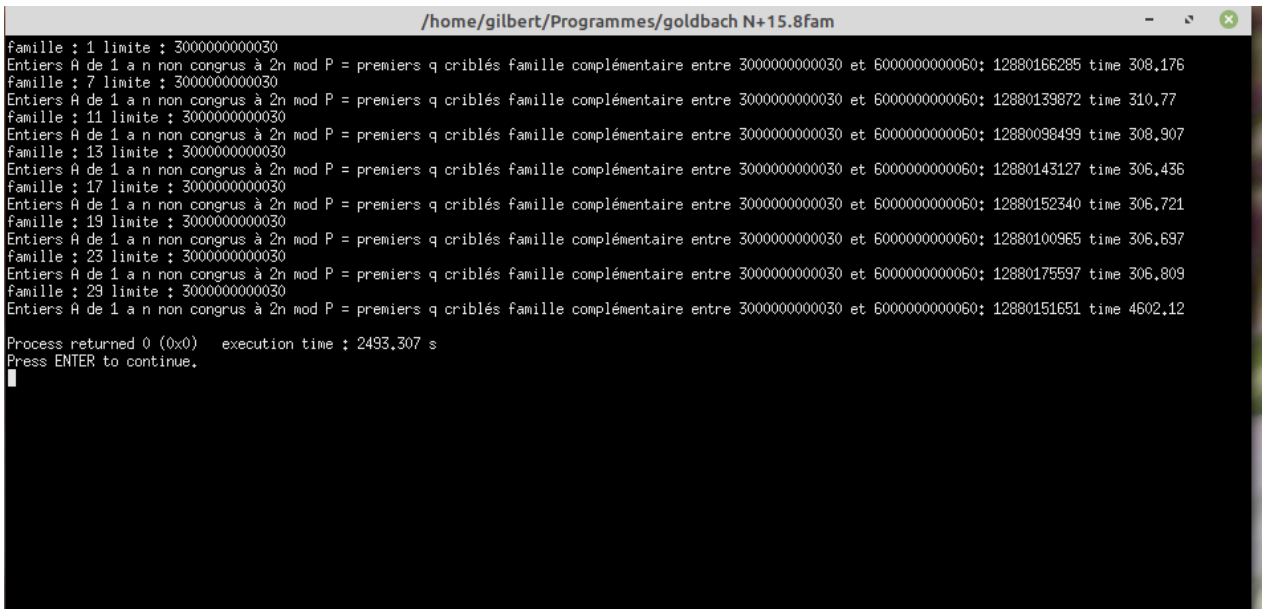
Alors qu'il y en avait: ***15 253 010 888 de plus*** *entre* ***(n – 15) et (2n – 30)*** *, qui ont été vérifiés et il ne devrait y avoir au maximum par famille* ***i*** *: qu'un écart* ***de un seul*** *nombre premiers* ***q*** *entre ces deux intervalles : [(n – 15) et (2n – 30)]* ***et*** *[(n) et (2n)]****. 103 041 128 336*** *nombres* ***q*** *entre* ***n et 2n.***

(« Pour la famille ***i = 7,*** *cela donnerait* ***1 906 610 132*** *nombres* ***q*** *en moins, correspondant au nombres de couples p+****q*** ***qui décomposent 2n*** *en moins, au lieu de* ***un seul*** *au maximum, pour la limite n +15 = 3000 000 000 030 criblée ..! »)*



fonction d'estimation minimum de couples p'+ q pour les 8 familles :
(3000 000 000 000 ÷ (ln 3000 000 000 000)^2) = **3 634 637 357**,36549…. < ***15 253 010 888 total réel***
Ci dessous :
Résultat du nombre de nombres d'entiers **A≠ 2n[P]** ⇔ au nombre de nombres premiers **q** par famille :
entre **n** = 3000 000 000 030 et **2n** = 6000 000 000 060

Soit pour les 8 Famille **i** modulo 30 : **103 041 128 336** nombres premiers **q ∈ [n , 2n]**



la fonction 2 relatif au corollaire du TNP ***donne au minimum :***
3000000000030 ÷ ln 6000000000060 =101 961 811 234,8783.. nombres premiers q.

Si la conjecture est fausse par supposition on aurait donc:
103 041 128 336 – 15 253 010 888 = 87 788 117 348 ***nombres premiers q.***
Soit un nombre de premiers ***q*** *inférieur au minimum de cette* ***fonction 2 , d'où ce corollaire démontré, serait faux aussi ..? Or,*** *plus la limite N tend vers l'infini, plus le champ des congruences tend vers l'infini , qui en se décalant d'un rang lors des limites suivante N+1… rend impossible l'infirmation de cette conjecture ; le contraire impacterait la répartition des nombres premiers et par la même les théorèmes en relation, avec cette répartition de nombres premiers.!*

****

Autre exemple :

La vérification de la conjecture pour une limite N ⇒ N+1; « plus généralement pour N ≥ 10^9, entraîne la vérification de la conjecture des limites N ⇒ N+1+2+3+..x; successives de plusieurs milliards, par famille ; **ie :** un effet boule de neige . »

Le Décalage d'un rang du champ des congruences sur les ,1, ou les ,0, d'Ératosthène, relatif à la propriété de l'algorithme, vérifiera la conjecture pour les nombres pairs 30(k+1) + 2i , soit la vérification des entiers **A ≠ 2N[P]**, pour la limite
N = 15k + 1 = 3001 , 2N = 6002; Ce qui rend impossible l'infirmation de la conjecture :

Donnez N = 15k = 3000 : on fixe la famille Fam(i) les **1**sont les **p'** non congruent à P
fam1 crible G : et : Fam 1 **É**ratosthène **E**G**C**rible pour N=3000 : conjecture vérifiée **pour tout n** ≤ 3000.
[0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 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, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0,0, 0, 0, 0, 1] **19** ∓ 1 couples **P'+ q** = 6000.

On suppose que pour **2N + 2** = 6002 il n'y a pas de solution : Or, il suffit de regarder la limite **N** =15(k-1) + 1 = 2985 + 1 précédente , qui a été vérifiée (ie : *on a criblé les* $A \neq 2N[P] , \leq N$) , relative à sa Fam (**i = 1**) pour affirmer que la conjecture serra vérifiée pour la limite suivante 15k + 1 , tel que 30k + 2 = 6002 est somme de deux premiers **p'+ q** !

Car le décalage d'un rang des congruences, sur leur successeur A+30, contredit la supposition!

ECrible les 1 sont premiers p' ≠ 2n[P] les 0 ≠ 2n[P] sont les multiple de P fam 30k+ i = 1;
[1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0]
Gcrible les 1 sont les entiers A ≠ 2n[P] [champ des congruences Famille 30k +1 , pour la limite N = 2986

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

Ce qui donnera au minimum 17 couples **p'+ q** qui vérifieront **30(k+1) + 2i = 6002**. Vérification avec **EG**Crible:
[0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1]

réel :18 couples p'+q = 2N. On peut montrer suivant cette propriété récurrente de l'algorithme de Goldbach, qu'il en sera de même sur les 14 limites successives suivantes.
Ce ne sont que les antécédents décalés d'un rang, relatif à la famille **i** concernée !

Autre cas EX avec 2N+2 = 6004

On vérifie **avec la famille i = 11** , pour la limite N = 2985 + 2 :

N = 2987; fam 11 ; **E**crible P' = 1 ⇔ ≢ 2n [P] et en dessous **G**crible Entier A = **1** ⇔ ≢ 2n [P] et P ≤ √2n
avec **le décalage des congruences de 1 rangs**, des **A** non congrus, **1** ou **0** (« *premiers ou pas* ») qui vont se reporter sur les A+30 = **P' = 1 que ce 1** soit congru ou pas à 2N modulo P, lors de la limite précédente , donc qui vérifiera au minimum cette limite N+1 suivante tel que 2N+2 = 6004 = p'+ q ; relatif à (15(k-1)+ 2) [P].

Crible E
[1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0] Soit : 21 p' ≢ 2N [P] .
Crible G
[0, 0, **1**, 0, 1, **1**, 1, 0, 0, **1**, 0, 0, 0, **1**, 0, 0, **1**, 0, 0, 1, 0, 1, **1**, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, **1, 1**, 0, 0, 0, 0, 1, 0, 1, **1**, 1, **1**, 0, 1, 0, **1, 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] Soit 21entiers **A** non congrus modulo P, qui ce sont décalés d'un rang sur leur successeur p'

Vérification avec le crible **EG**:
[0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0]
On a bien en réel : **21** couples **p+q** = **2N = 6004**

Nous venons de vérifier et de constater le décalage d'un rang qui est récurrent, des congruences des A = 1 ou A= 0 et où donc, on ne s'occupe pas de savoir si ce A est un nombre premier ou pas, **mais qu'il soit non congruent** (mod P) et les congruences = 1, qui vont se reporter sur leur successeur **A'+30 = 1 = p'** ; par conséquent qui ont pour antécédent A, le contraire contredirait cette propriété récurrente et donc le TNP et Le TFA, (cqfd).

Illustration jointe, on peut voir ci-dessous, pour une autre limite et 4 familles i ; le nombre de nombre premiers q qui disparaîtrait avec n = 6000 000 000 010 qui augmente de 15 , soit 2n = 12 000 000 000 020 en supposant que 2n ne se décompose pas en la somme : p'+ q = 2n, alors que pour n – 15 ce nombre de nombres premiers q ⇔ au nombre de A≢ 2n[p] ≤ n = 6000 000 000 010 – 15 a été vérifié ...!! Il est clairement impossible de supposer le contraire !

/home/gilbert/Programmes/E_G_Crible_8.fam

```
famille : 1 limite : 6600000000010
Nombre premiers criblés famille 1 plus petits que 6600000000010: 28967518720 time 840,754
Nombre couples p+q=2N criblés famille 1 : 3411256593 time 876,828
famille : 1 limite : 6600000000025
Nombre premiers criblés famille 1 plus petits que 6600000000025: 28967518720 time 830,226
Nombre couples p+q=2N criblés famille 1 : 3410685839 time 5179,47
famille : 1 limite : 6600000000040
Nombre premiers criblés famille 1 plus petits que 6600000000040: 28967518721 time 5127,24
Nombre couples p+q=2N criblés famille 1 : 3637444519 time 5170,46
famille : 1 limite : 6600000000055
Nombre premiers criblés famille 1 plus petits que 6600000000055: 28967518721 time 5119,43
Nombre couples p+q=2N criblés famille 1 : 4550506175 time 5171,94
famille : 1 limite : 6600000000070
Nombre premiers criblés famille 1 plus petits que 6600000000070: 28967518721 time 9418,66
Nombre couples p+q=2N criblés famille 1 : 3453976619 time 9464,75
famille : 7 limite : 6600000000010
Nombre premiers criblés famille 7 plus petits que 6600000000010: 28967578247 time 9415
Nombre couples p+q=2N criblés famille 7 : 3411253438 time 9463,73
famille : 7 limite : 6600000000025
Nombre premiers criblés famille 7 plus petits que 6600000000025: 28967578247 time 9417,21
Nombre couples p+q=2N criblés famille 7 : 3410707544 time 13763,8
famille : 7 limite : 6600000000040
Nombre premiers criblés famille 7 plus petits que 6600000000040: 28967578247 time 13710,6
Nombre couples p+q=2N criblés famille 7 : 3637421453 time 13766,3
famille : 7 limite : 6600000000055
Nombre premiers criblés famille 7 plus petits que 6600000000055: 28967578247 time 13717
Nombre couples p+q=2N criblés famille 7 : 4550474625 time 13766,1
famille : 7 limite : 6600000000070
Nombre premiers criblés famille 7 plus petits que 6600000000070: 28967578247 time 18011,9
Nombre couples p+q=2N criblés famille 7 : 3453890955 time 18055,1
famille : 13 limite : 6600000000010
Nombre premiers criblés famille 13 plus petits que 6600000000010: 28967565166 time 18000
Nombre couples p+q=2N criblés famille 13 : 3411240918 time 18056,1
famille : 13 limite : 6600000000025
Nombre premiers criblés famille 13 plus petits que 6600000000025: 28967565166 time 18010,7
Nombre couples p+q=2N criblés famille 13 : 3410706294 time 22348
famille : 13 limite : 6600000000040
Nombre premiers criblés famille 13 plus petits que 6600000000040: 28967565166 time 22311,5
Nombre couples p+q=2N criblés famille 13 : 3637301251 time 22356,4
famille : 13 limite : 6600000000055
Nombre premiers criblés famille 13 plus petits que 6600000000055: 28967565166 time 22297,1
Nombre couples p+q=2N criblés famille 13 : 4550423761 time 22354,6
famille : 13 limite : 6600000000070
Nombre premiers criblés famille 13 plus petits que 6600000000070: 28967565166 time 26598,1
Nombre couples p+q=2N criblés famille 13 : 3453932549 time 26639,2
famille : 19 limite : 6600000000010
Nombre premiers criblés famille 19 plus petits que 6600000000010: 28967477637 time 26597,7
Nombre couples p+q=2N criblés famille 19 : 3411186846 time 26646,6
famille : 19 limite : 6600000000025
Nombre premiers criblés famille 19 plus petits que 6600000000025: 28967477637 time 26604,8
Nombre couples p+q=2N criblés famille 19 : 3410694760 time 30942,2
famille : 19 limite : 6600000000040
Nombre premiers criblés famille 19 plus petits que 6600000000040: 28967477637 time 30894,6
Nombre couples p+q=2N criblés famille 19 : 3637331827 time 30943,8
famille : 19 limite : 6600000000055
Nombre premiers criblés famille 19 plus petits que 6600000000055: 28967477637 time 30895,5
Nombre couples p+q=2N criblés famille 19 : 4550384284 time 30940,3
famille : 19 limite : 6600000000070
Nombre premiers criblés famille 19 plus petits que 6600000000070: 28967477637 time 35188,6
Nombre couples p+q=2N criblés famille 19 : 3453881974 time 35237,1

Process returned 0 (0x0)   execution time : 51359,822 s
Press ENTER to continue.
```

Algorithme en python de Ératosthène et Goldbach : *« il existe aussi en C++ »*
On fixe la famille ***30k+ i****, par rapport à la forme de la limite* ***n*** *fixée donc pour* ***2n – n*** *= 30k + i' , pas forcément la même famille 30k + i.*

Pour toute Limite **N = 15k + n'**, avec **n' ∈ [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14]** on rentre la Fam(i) correspondante

N = 15k+1; Fam =(1,13,19); ⇒ **2n = 30k + 2** ⇒ **32 – 19 = 13** , ou **32 – 1 = 31** ⇒ la Fam complémentaire
N = 15k+2; Fam =(11,17,23); ⇒ **2n = 30k + 4** ⇒ **34 – 11 = 23** , ou **34 – 17 = 17**
N = 15k+3; Fam =(7,29,13,23,17,19); **2n = 30k + 6**
N = 15k+4; Fam =(1,7,19); 2n = **30k + 8**
N = 15k+5; Fam =(1,7,13,19)**;** **2n = 30k + 10**
N = 15k+6; Fam =(1,11,13,19,23,29); **2n = 30k + 12**
N = 15k+7; Fam =(1,7,13); 2n = **30k + 14** ⇒ **44 – 7 = 37** , ou **44 – 1= 43 = 30k+13**
N = 15k+8; Fam =(17,23,29); 2n = **30k + 16**
N = 15k+ 9; Fam =(1,7,11,17,19,29); 2n = **30k + 18**
N = 15k+10; Fam =(11,29,17,23); 2n = **30k + 20**
N = 15k+11; Fam =(11,23,29); **2n = 30k + 22**
N = 15k+ 12; Fam =(1,7,11,13,17,23); 2n = **30k + 24**
N = 15k+ 13; Fam =(7,13,19); 2n = **30k + 26**
N = 15k+ 14; Fam =(11,17,29); 2n = **30k + 28**

Pour les multiples de 30, avec N = 15k; l'une des 8 Fam , 2n = 30k

---

Par exemple dans le programme ci-dessous on fixe la limite n = 15k + 7 = 875017 début ; 875032 fin. On va cribler la famille 30k+7 ce qui donnera le nombre de décompositions de la famille complémentaire 30k +7 , qui décomposent 2n = 1750034.

```
from time import time
from os import system

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)  ##(si on fusionne les deux cribles il faudra rentrer, int((2n)**0.5) pour Goldbach.
    prime_E =[2,3]
    sieve_list = [True for _ in range(n+1)] ## c'est plus propre comme ça
    for each_number in candidate_range(n):
        if sieve_list[each_number]:
            prime_E.append(each_number)
            for multiple in range(each_number*each_number, n+1, each_number):
                sieve_list[multiple] = False
    print(prime_E[3:])
    return prime_E[3:]   ## on retourne les nombres Premiers qui vont cribler

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

    ## On génère un tableau de N/30 cases rempli de 1
    lencrible = ((n//30)+1)
```

```
    crible=[1 for _ in range(lencrible)] ## c'est plus propre comme ça
    GM = [7,11,13,17,19,23,29,31]
    ## On calcule les produits :
    for a in premiers:
        for b in GM:
            j = a * b
            if j%30 == fam:
                index = j // 30    ## on calcule l'index de départ et on crible directement à partir de cet index
                for idx in range(index, lencrible, a):  ## index qui est réutilisé ici...
                    crible[idx] = 0

    total = sum(crible)
    print("crible Ératosthène :", crible)  ## pour éditer le tableau Ératosthène criblé
    print(f"Nombre premiers criblés famille {fam} : {total} ----- {int((time()-start_crible)*100)/100}")
    return crible,lencrible

def GCrible_2N(premiers, crible, lencrible, n, fam):
    start_crible = time()
    ## On calcule les restes: ri = 2*n/pi
    nbpremiers = len(premiers)
    n2 = 2*n
    for premier in premiers:
        reste = n2 % premier
        #print(reste)
        if reste % 2 == 0:
            reste += premier
        pi2 = 2*premier
        ## tant que reste % 30 != fam on fait reste += pi2
        while reste % 30 != fam:
            reste += pi2
        ## Ensuite on divise reste par 30 pour obtenir l'index de départ
        reste //= 30
        ## On crible directement à partir de cet index
        for index in range(reste, lencrible, premier):
            crible[index] = 0

    total = sum(crible)
    print("crible É ET G:", crible) ## éditer le tableau criblé É et G
    print(f"Nombres P' non congru 2n[pi] ou couple P'+q = 2N, de (i) à {n} famille {fam} : {total} -----
{int((time()-start_crible)*100)/100}")

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

def main():
    ## On demande N a l'utilisateur
    n = demander_N()
    ## On récupère les premiers de 7 à √2N
    premiers = eratostene(n)
    start_time = time()
    ## On crible
    fam=7 ## ou 1, 7, 11, 13, 17, 19, 23, 29, au choix en fonction de n
    crible,lencrible=E_Crible(premiers, n, fam)
    GCrible_2N(premiers, crible, lencrible, n, fam)

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

En C++ *:*

```
// -*- compile-command: "/usr/bin/g++ -g goldbachs.cc" -*-
#include <vector>
#include <iostream>
#include <cmath>
#include <stdlib.h>
#include <time.h>
using namespace std;
// fill Erathosthene sieve crible for searching primes up to 2*crible.size()*32+1
// crible is a (packed) bit array, crible[i] is true if 2*i+1 is a prime
// crible must be set to true at startup
void fill_crible(vector<unsigned> &crible, unsigned p)
{
  crible.resize((p - 1) / 64 + 1);
  unsigned cs = crible.size();
  unsigned lastnum = 64 * cs;
  unsigned lastsieve = int(std::sqrt(double(lastnum)));
  unsigned primesieved = 1;
  crible[0] = 0xfffffffe; // 1 is not prime and not sieved (2 is not sieved)
  for (unsigned i = 1; i < cs; ++i)
    crible[i] = 0xffffffff;
  for (; primesieved <= lastsieve; primesieved += 2)
  {
    // find next prime
    unsigned pos = primesieved / 2;
    for (; pos < cs; pos++)
    {
      if (crible[pos / 32] & (1 << (pos % 32)))
        break;
    }
    // set mutiples of (2*pos+1) to false
    primesieved = 2 * pos + 1;
    unsigned n = 3 * primesieved;
    for (; n < lastnum; n += 2 * primesieved)
    {
      pos = (n - 1) / 2;
      crible[(pos / 32)] &= ~(1 << (pos % 32));
    }
  }
}
unsigned nextprime(vector<unsigned> &crible, unsigned p)
{
  // assumes crible has been filled
  ++p;
  if (p % 2 == 0)
    ++p;
  unsigned pos = (p - 1) / 2, cs = crible.size() * 32;
  if (2 * cs + 1 <= p)
    return -1;
  for (; pos < cs; ++pos)
  {
    if (crible[pos / 32] & (1 << (pos % 32)))
    {
      pos = 2 * pos + 1;
      // if (pos!=nextprime(int(p)).val) CERR << "error " << p << endl;
      return pos;
    }
```

```
  }
  return -1;
}

typedef unsigned long long ulonglong;

size_t ECrible(const vector<ulonglong> &premiers, ulonglong n, int fam, vector<bool> &crible, size_t len-
crible)
{ //on va contruire un tableau de 1 modulo 30 et rappeler les premiers p
  int cl = clock();
  // size_t lencrible = n / 30,
  size_t nbpremiers = premiers.size(); //on va contruire un tableau de 1 modulo 30 en divisant N par 30
  //vector<bool> crible(lencrible, true);                    // on rappelle les nombres premiers p d'Eratotene
  // ulonglong n2=2*n;
  vector<ulonglong> indices(nbpremiers);
  for (size_t i = 0; i < nbpremiers; ++i)
  {
    ulonglong p = premiers[i];
    ulonglong produit;
    int GM[] = {7, 11, 13, 17, 19, 23, 29, 31}; // on va calculer le produit de p par un element du groupe GM
    for (size_t j = 0; j < sizeof(GM) / sizeof(int); j++)
    {
      produit = p * GM[j]; // calcul du produit, jusqu'a ce que le produit soit égale à fam modulo 30
      if (produit % 30 == fam)
      {
        produit /= 30; // puis on va va calculer l'indice, afin de commencer à cribler de l'indice à n/30 et on réitère
        break;
      }
    }
    indices[i] = produit;
  }
  ulonglong nslices = lencrible / 1500000, currentslice = 0;
  if (nslices == 0)
    nslices = 1;
  for (; currentslice < nslices; ++currentslice)
  {
    size_t slicelimit = currentslice + 1;
    slicelimit = slicelimit == nslices ? lencrible : (currentslice + 1) * (lencrible / nslices);
    for (size_t i = 0; i < nbpremiers; ++i)
    {
      ulonglong p = premiers[i];
      size_t index;
      for (index = indices[i]; index < slicelimit; index += p)
        crible[index] = 0;
      indices[i] = index;
    }
  }
  size_t total = 0;
  for (size_t index = 0; index < lencrible; ++index)
    total += int(crible[index]);
  cout << " Nbr p' criblés Fam " << fam << " < a " << n << ": " << total; // << " time " << (clock() - cl) * 1e-
6 << endl;
  return total; // à la fin du crible on return le résultat est le temps mis
}

size_t GCrible(const vector<ulonglong> &premiers, ulonglong n, int fam, vector<bool> &crible, size_t len-
crible)
{
  int cl = clock();
  //size_t lencrible = n / 30,
```

```
  size_t nbpremiers = premiers.size(); //on rappelle le tableau d'eratosthene criblé
  //vector<bool> crible(lencrible, true);
  ulonglong n2 = 2 * n;
  vector<ulonglong> indices(nbpremiers);
  for (size_t i = 0; i < nbpremiers; ++i)
  {
    ulonglong p = premiers[i];
    ulonglong reste = n2 % p; // on calcule le reste de 2n par p
    if (reste % 2 == 0)
      reste += p;
    ulonglong pi2 = 2 * p;
    while (reste % 30 != fam) // tant que le reste += p n'est pas = à Fam % 30 on rajoute 2*p
      reste += pi2;
    reste /= 30; // on ensuite on va calculer l'indice pour commencer à cribler le tableau de 1.1.1.... avec p, de l'in-
dice à n/30
    indices[i] = reste;
  }
  ulonglong nslices = lencrible / 1500000, currentslice = 0;
  if (nslices == 0)
    nslices = 1;
  for (; currentslice < nslices; ++currentslice)
  {
    size_t slicelimit = currentslice + 1;
    slicelimit = slicelimit == nslices ? lencrible : (currentslice + 1) * (lencrible / nslices);
    for (size_t i = 0; i < nbpremiers; ++i)
    {
      ulonglong p = premiers[i];
      size_t index;
      for (index = indices[i]; index < slicelimit; index += p)
        crible[index] = 0;
      indices[i] = index;
    }
  }
  size_t total = 0;
  for (size_t index = 0; index < lencrible; ++index)
    total += int(crible[index]); // le criblage du tableau de 1 modulo 30 jusqu'a n/30 (1.1.1.1...etc) est fini on va
retourner le résultat
  cout << " Nbr couple p+q criblés Fam " << fam <<  " : " << total;// " time " << (clock() - cl) * 1e-6 << endl;
  return total;
}

int main(int argc, char **argv)
{
  vector<unsigned> temp;
  ulonglong debut = 875017;
  ulonglong fin = 875032;

  vector<int> familles;
  //familles.push_back(1);
  //familles.push_back(7);
  //familles.push_back(11);
  //familles.push_back(13);
  familles.push_back(17);
  //familles.push_back(19);
  //familles.push_back(23);
  //familles.push_back(29);

  for (int i = 0; i < familles.size(); i++)
  {
    int fam = familles[i];
```

```
  for (ulonglong limite = debut; limite < fin; limite += 15){
    cout << "--> limite : " << limite << endl;
    double sqrt2N = unsigned(std::sqrt(2 * double(limite)));
    fill_crible(temp, sqrt2N);
    vector<ulonglong> premiers;
    for (ulonglong p = 7; p <= sqrt2N;)
    {
      premiers.push_back(p);
      p = nextprime(temp, p);
      if (p == unsigned(-1))
        break;
    }

    size_t lencrible = limite/30;
    vector<bool> crible(lencrible, true);
    ECrible(premiers, limite, fam, crible, lencrible);
    GCrible(premiers, limite, fam, crible, lencrible);
  }
 }
}
```