Préambule:

[« suivant l'affirmation de Wikipédia sur cette conjecture de Goldbach :

« Le théorème des nombres premiers affirme qu'un entier ***m*** sélectionné aléatoirement d'une manière brute possède (**1/** Ln **m**) chance d'être premier. Ainsi, si ***n*** est un grand entier pair et ***m***, un nombre compris entre **3** et ***n*/2**, alors on peut s'attendre à ce que la probabilité que ***m*** et ***n*** *–* ***m*** soient tous deux premiers soit égale à **1 /** (Ln **n .** Ln **m**) . Cet argument heuristique n'est pas rigoureux pour de nombreuses raisons; par exemple, on suppose que les événements que ***m*** et ***n – m*** **soient premiers sont statistiquement indépendants l'un de l'autre. »**

l'algorithme de Goldbach permet de prouver ***que cette affirmation est « Fausse »*** *pour une limite* ***n*** *fixée*, ainsi que la fonction asymptotique du TNP qu'il faudra donc modifier dans ces ***8 Familles (i)*** de nombres premiers **p' > 5**, « *ces entiers représentent 26,666… % des entiers naturels non nuls; soit* ***n / 3,75 .*** *»*

Car en effet, si on considère comme événement le calcul du nombre de nombres premier **q** ∈ **[*m ; n*]** ***il dépend statistiquement de la congruence des entiers m ≤ n/2*** *limite fixée, autrement dit* ***q à pour antécédent m*** et donc si et seulement si **m** est un nombre premier **p' ≤ *n/2*** et tel que **m ≢ n [P]** alors **q a pour antécédent m = p',** il s'agit donc d'un seul et même événement, dépendant de la **congruence de m ⇒ n – m** un nombre premiers **q.**

q ne peut être un multiple de **P,** mais le complémentaire de **m** par rapport à **n ;** c'est donc *une indépendance impossible.*

On peut déjà en déduire et affirmer, avec l'algorithme de Goldbach et celui d'Ératosthène, que cette fonctions ci-dessus (**1/**Ln **m** * Ln **n**) est parfaitement justifiée et permet d'estimer une quantité positive d'entiers **m** premiers mais aussi d'être ***non congru à n (mod P)*** , Conséquence directe du TNP, on obtient : **m /** Ln **n** une *proportion d'entiers* ***m*** **≢ n [P]** *et par conséquent une estimation non nulle d'entiers* ***m*** *qui précède un entier* ***m' = p'*** ⇔ **p' + q = n .** Le nombre d'entiers **m** *non congrus à* ***n*** *modulo* ***P*** *qui impliquent les nombres premiers* **q** ∈ **[*m ; n*]** *est équivalent à* ***m*** **/ Ln n,** *lorsque* ***m*** → **+ ∞ .«** *Ce que l'on verra ci-après avec ces deux cribles qui caractérisent les fonctions du TNP ainsi que les définitions relatives à ces algorithmes dans ces 8 Fam(i).***»**

On en déduit le raisonnement suivant : ces entiers **m = p'** ou pas et tel que **m ≢ n [P]** fait de par cette congruence, que ***statistiquement*** il s'agit d'un même événements que **m** et **n** soient premiers, *suivant le TNP on peut s'attendre à ce que la probabilité que* ***m*** *et* ***n*** *–* ***m*** *soient tous deux premiers soit égale à* ***1 / (Ln n . Ln m)***, caractérisé par les deux cribles, don celui de Goldbach que l'on va utiliser pour recribler les nombres **m** = **p' ≤ n/2** ayant été criblés au par avant par Ératosthène, selon le même principe expliqué ci dessous en **a :)** et **b :)**.**»]**

Objectif :

Démontrer que le crible de Goldbach qui agit dans les congruences et que l'on va utiliser pour résoudre la conjecture de Goldbach, est une variante du crible d'Ératosthène ; utilisant le même principe et une propriété bien connue des congruences : Propriété sur les entiers **A ≤ n** impairs, qu'il soit premiers **p'** ou pas et congrus ou pas à **2n** modulo **P** avec **P** ∈ $\mathbf{P_n}$ l'ensemble des nombres premiers ≤ **√2n ;** tout entier naturel ≤ **2n** qui n'est pas divisible par **P** est un nombre premier suivant le principe d'Ératosthène. Les **A** seront représenté par des **1** avant criblage : si **A** est congru **= 0** , sinon **= 1**.

(« ***Propriété connue des congruences :*** *pour 30k > 30, il existe* ***y/y'*** *tels que 30k =* ***P*y + R*** *on dit que 30k est congru à* ***R*** *modulo* ***P*** *et* ***A*** *=* ***P*y' + R*** *;* ***A*** *est là aussi congru à* ***R*** *modulo* ***P; ce qui implique 30k – A*** *=* ***P*(y – y')*** *d'où* ***P*** *divise* ***30k – A,*** *c'est à dire que* ***30k et A*** *partagent le même reste* ***R*** *dans la division par* ***P , on peut donc remplacer R par A*** *dans l'algorithme ; par conséquent et à l'inverse si* ***A*** *et* ***30k*** *ne partage pas le même reste* ***R,*** *tel que* ***A ≢ 30k [P],*** *alors* ***P*** *ne divise pas cette différence, d'où* ***30k – A = q*** *premier. Ce qui permet d'affirmer que pour tout A=****p'*** *et* ***q*** *ne sont pas deux événements indépendants l'un de l'autre, mais* ***q*** *à pour antécédent* ***p' ≢ 2n[P]*** *qui dépend donc de la congruence de* ***p',*** *plus généralement de* ***A≢ 2n [P].****»***)**

Montrer que l'interprétation de la comète de Goldbach est une boule de neige qui utilise la propriété récurrente du décalage d'un rang des congruences dans l'algorithme de Goldbach : ce qui permet de comprendre et d'expliquer pourquoi le nombre de solutions qui vérifient 2n augmentent de façon oscillatoire lorsque 2n tend ver ∞.

Goldbach indique que tout nombre pair 2n est la somme de 2 nombres premiers (p'+ q).

On veut utiliser la décomposition 30k + 2i ou de 2n en deux nombres premiers selon les principes d'un crible.

L'objectif est de montrer **1):** que ces nombres premiers par famille forment l'ensemble des nombres premiers > 5 pour une limite **n** fixée et **2):** qu'il est impossible pour la limite suivante **n+1**, d'infirmer la conjecture ayant été vérifiée lors de la limite **n** précédente.

La méthode du crible à utiliser suit la méthode d'Ératosthène, en utilisant les congruences pour toutes limite **n ≥ 15k** où **n** progresse modulo 15 lorsque l'on crible par famille avec **P** un nombre premier ∈ $\mathbf{P_{2n}}$ l'ensemble des nombres premiers ≤ **√2n** qui vont cribler en utilisant l'algorithme de Goldbach. Pour Ératosthène on utilise **p** ∈ $\mathbf{P_n}$ l'ensemble des nombres premiers ≤ **√n**

On crible de **1** à **n** et non de **1** à **2n :**
a :) les entiers non nuls **A = p' ≤ n** avec **Ératosthène**; on part d'un nombre premier **p ≤ √n** « *ou du carré de **p*** » et on marque tous ses multiples ≤ **n** par pas de **p.**
b:) les entiers **A≤ n,** tel que **A ≢ 2n[P]** avec **Goldbach** suivant le même principe **:** mais avec **P ≤ √2n,** où on part du reste **R = A** de **2n** par **P** et on marque tous les (**A≡ 2n [P]) ≤ n** par pas de **P**.

⇨ cinq parties à expliciter :
1/ : comment on construit ce crible, algorithme de Goldbach.

2/ : montrer que les nombres premiers ainsi trouvés avec les deux algorithmes, forment l'ensemble des nombres premiers >5.

3/ : montrer que quelque soit un entier pair 2n = 30k+2i, il est toujours décomposable en somme de deux nombres premiers **(p'; q)** en utilisant une des 8 familles 30k + 2i , i ∈ {1,7,11,13,17,19,23,29} avec **30k ≥ 300**; sachant que la conjecture est vérifiée jusqu'à 300. «O*n peut aussi montrer en utilisant simplement les entiers impairs non nul **de 1 à n**.*»

4/ : montrer que l'on peut utiliser le principe de fonctionnement de ce crible avec ses propriétés, à l'ensemble des entiers pairs en progression arithmétique de raison 30, selon des conditions particulières; afin d'obtenir une estimation minorée de couples premiers, qui décomposent un entier pair > 4 en somme de deux nombres premiers, sachant que si **A = p'≢ 2n[P] ⇒ q** premier est un seul et même événement puisque **q** à pour antécédant **p'** non congruent à **P.**

5/ : on montrera que ce crible a une propriété récurrente, le décalage d'un rang des congruences sur leur successeur, lorsque la limite 2n augmente de 2, **ou en utilisant une seule** famille pour la limite 30(k+1) + 2i; en la choisissant par rapport à l'une des 15 formes de **n = 15k + i**, la limite du criblage fixée. Cette propriété récurrente a un effet boule de neige.
«*Ce crible dans les congruences fera ressortir la famille complémentaire par rapport à **2n** du fait que les nombres premiers **q** ont pour antécédent les entiers **A ≢ 2n[P],** il est donc inutile de chercher le complémentaire **q** de **A.***»

6/ : pour 3/ et 4/, on utilisera le Théorème des nombres premiers noté TNP et son corollaire, conséquence directe du TNP; pour en déduire une troisième fonction, indiquant une estimation minimum de couples **p'+ q = 2n** ou plus précisément **: une estimation d'entier A'= p' ≢ 2n[P] .**

1/ : Le crible.

Le **<u>périmètre</u>** de travail du crible par famille pour une limite **n=15k + i** qui progresse de raison 15: on travaille sur les entiers circonscrits aux familles suivantes qui seront appelé fam **(i)**: **(«***en fonction de la forme de **n=15k+i → 2n = 30k +2i**, on fixe la fam (**i**) ; ce qui donnera la famille complémentaire par rapport à 2n selon* ***5/ :*** *, pas forcément la même. Par exemple si **n = 15k +1** on a **2n = 30k+2** on a que trois fam(**i**) possible {1 ,13 ,19} , 1+31=32 et 13+19 = 32. En fixant la **fam 13** on obtient les complémentaires **q ∈ fam 19** et pour la **fam 1; q ∈ fam 1** qui est la même.***»)**

*fam(**i**)* - 30k+1
- 30k+7
- 30k+11
- 30k+13
- 30k+17
- 30k+19
- 30k+23
- 30k+29

L'intérêt de travailler avec ces huit familles ou une seule, est qu'elles permettent de réduire le nombre d'entiers naturels avec lesquels on travaille sans perte de généralité, puisque tout entier de forme 30k + ***i*** avec ***i*** n'appartenant pas à {***1,7,11,13,17,19,23,29***} n'est pas premier à l'exception de 2, 3 et 5 .

[« *Suivant le point* ***2/ :*** *tout nombres premier > 5 et donc de la forme **30k + i**.*

Démontrons que tout nombre premier supérieur à 30 appartient à l'une de ces 8 familles.
*Soit **p** un nombre premier supérieur à 30. Il n'existe donc pas de décomposition en nombre premier de **p**.*
30 étant le produit de 2, 3 et 5. Tout nombre premier supérieur à 30 n'est pas divisible par 2, 3, ou 5 s'écrit donc de la forme
$2^i.3^j.5^k$ *+ **pr** avec **pr** non divisible par 2, 3 et 5 et inférieur à 30.*
*Donc **pr** $\in$ **[7 ; 29]** étant l'un des nombres premiers inférieurs à 30 et différents de 2, 3, ou 5.»*]

L'objectif est d'extraire de ces suites arithmétiques de raison 30, les nombres premiers supérieurs à 5 congrus à 1 [30] ou à p [30] avec p appartenant à {7,11,13,17,19,23,29} avec Ératosthène, et de décomposer les nombres pairs en somme de deux nombres premiers en utilisant les congruences avec le crible de Goldbach variante d'Ératosthène. *De manière générale et pour une limite 15k+(i) ≥150 , une seule famille est suffisante sans perte de généralité, pour prouver la conjecture.*

Construction de l'algorithme AG, crible de Goldbach:

[« ***Note:*** *On peut construire directement le crible en partant de 1 et faire ressortir ses propriétés, en utilisant les nombres impairs A et où ces entiers A congrus à 2n[**P**] noté A ≡ 2n[**P**], seront marqués en rouge, suivant le principe d'Ératosthène par pas de **P**.*

Ex**: 1,3,5,7,9,11,13,15,17,19...etc et avec **P** ≥ 3. lorsqu'un nombre sera **congru à 2n[P]** il serra remplacé par **0** ou marqué en rouge suivant les cas ci-dessous de raison **P.
***ex:** limite n =15 , 2n =30 pour **P** ≤**√2n** on a **3** et **5** le reste **R** de 2n par **P** donnera **0** et **0**.*

On part toujours de l'indice du reste: (ici) **0** avec **p = 3** et on marque les entiers en rouge de 1 à 15 congruents à **p** ; suivant le principe d'Ératosthène **modulo p** ou de raison **p**.
(*Ce qui est équivalent à marquer les multiples de **P** ∈ **[n;2n]**, si ce n'est que l'on marque les entiers **A** congrus à **2n[P]** de **1** à **n** de raison **P.***)

[*Si le reste R %2 = 0 on part de **P** puis += **2p ;** jusqu'à la limite **n** fixée*]
ex 1: liste à criblér **limite n = 15 :** [*1,3,5,7,9,11,13,15*] ; on va marquer en rouge les A congruents à **P.**
a) résultat (mod 3) ou (mod 2*3) [*1,3,**5**,7,9,**11**,**13**,15*] *les A ≡ 2n [3]*
b) résultat (mod 5) ou(mod 2*5) [*1,3,5,**7**,9,**11**,**13**,15*] *puis les A ≡ 2n [5]*

*Soit 3 couples **p'**+q qui décomposent 30. On en déduit directement une conséquence du TNP:*
*Comme on crible pour une limite **n** fixée, avec les nombres premiers **p** ≤ racine de **2n** , la fonction du nombre d'entiers **A** non congrus à **2n[P]** noté **A** ≢ **2n [P]**; devient **[n / log 2n]** ⇒ le nombre de nombres premiers **q** ∈ **[n;2n]**.*

Montrons le décalage d'un rang des congruences sur leur successeurs impair en progression arithmétique de raison 2, permettant d'affirmer que la conjecture serra vérifiée pour la limite n+1 suivante.

ex 2 : *Propriété récurrente ou théorème de Goldbach : si un entier **A ≢ 2n[P]** précède un nombre premier **p'= A'+2 congru ou pas,** alors **p'** serra donc **non congru à (2n+2) [P]** pour la limite suivante **2n +2** et vérifiera par conséquent la conjecture **suivant l'égalité** ci après : **30 – A** ⇔ **(30+2) – (A+2)***
Montrons pourquoi:
*Lorsque la limite **n** augmente de 1, donc **2n** augmente de 2 , cela n'a pas d'influence sur le nombre de premiers ≤ **n=(15+1)** qui ne bougent pas, ni sur le nombre de premiers **P** qui criblent. **Seul les congruences** vont se décaler par obligation sur leur successeur **A'+ 2,** afin de satisfaire **l'égalité** ci-dessus; le contraire serait absurde comme on peut le vérifier ci-dessous :*

ex 3*: liste à cribler n+1 = **16** [1,**3**,5,7,9,11,13,15,<17,]*
*les restes de 32 par 3 et 5, augmentent de 2. Mais l'égalité **30 – A** ⇔ **(30+2) – (A+2)** **aura obligatoirement la même différence pour la raison suivante :***
***1** qui était ≢ **2n [P]** la congruence se reporte sur **(1+2)** ≢ **(2n+2) [P],** car **29 qui était un nombre premier q tel que** 30 – 1 = **29** , serra toujours le même **32 - 3 = 29 sinon cela est contraire au TFA, 29** ne peut devenir un produit de facteurs premiers.*
*D'où 3 serra ≢ **(2n+2) [P]** et tel que 32 – 3 = **29** ce qui donnera pour 2n +2 un couple p'+ q = 32 .!*

on avait pour n = 15, avec ***P = 3 et*** *R = 0 : [**1**,3,**5**,**7**,9,**11**,**13**,15] trois nombres congruent à **P***

*on a pour **n+1**=16, avec **P = 3** et R=2 : [**1**,**3**,**5**, 7,9,**11**,**13**,15] décalage d'un rang des congruences*

a) *la **non** congruence de **1** se reporte sur **3,** celle de 3 qui était congru se reporte sur **5,** celle de **7** non congru se reporte sur 9 et celle de 9 congru se reporte sur **11. etc etc.***

b) pour *n = 15, avec **P = 5 et** R =0 : [1,3,5,7,9,**11**,**13**,15 ,]*
*pour n+1=16, avec **P = 5** et R=2 : [**1**,**3**,5,7,9, **11**, **13**,15 ,]*
*la **non** congruence de **1** se reporte sur **3,** celle de 5 s'est reporté sur **7, car 7 et 32 sont ≡ 2[5], fin du crible.***

*On a 2 couples **3+29 et 13 +19** qui décomposent 32 et on constate bien que 23 qui est premier à pour antécédent **9 ≢ (2n+2) [p];** si les congruences ne se décalaient pas d'un rang, il serait resté congru à (2n +2) [p] ce qui est absurde, car contraire au TFA et au TNP, ce qui garanti aussi la famille complémentaire faisant partie d'un même événement !*

Cela permet de garder *la propriété des entiers B ∈[n;2n]; si 2n – A = B est un multiple de P, (2n+2) – (A+2) = B serra toujours multiple de P; et inversement si 2n – A = **q** premier, (2n+2) – (A+2) = **q** serra toujours premier.*

*Autre constat, **9** qui **n'est pas** premier mais qui précède **11** un nombre premier, **sa non** congruence se reportera sur **11** lors de la limite suivante **n+1, ainsi** que celle de 15 sur **17.***
Ce qui permet d'affirmer et sans vérifier*, que pour (2n +2) +2 la conjecture serra encore vérifiée avec au minimum 34 – 11 = **23, qui a été utilisé précédemment et** on commence à voir cet effet boule de neige et par conséquent l'impossibilité d'infirmer la conjecture pour les limites suivantes **n+1, n+2 ...***

*On peut donc en déduire dès lors, une troisième fonction caractérisée par cet algorithme, relative au TNP par rapport aux deux fonctions **π(n)~[n / log n]** indique ~ le nombre de premiers ≤ **n** et **G(n)** qui vaut ~ [**n / log 2n**] indique le nombre d'entiers **A ≢ 2n[P] ≤ n** ⇔ nombre de premiers **q ∈ [n;2n].***

De ces deux fonctions on en déduira une troisième : $\frac{n}{(\ln(n)*\ln(2n))}$ *qui indiquera le nombre minimum de couples **p+q = 2n lorsque 2n → + ∞***

Ce sont les même nombres premiers ***P*** qui criblent une même limite **n**, selon le même principe avec ces deux algorithmes qui caractérisent les deux fonctions asymptotiques du TNP.
On prend en compte : les entiers ***A ≢ 2n[p]*** *et les nombres premiers **p'** pour une même limite **n ≥ 3** fixée.*

*On peut par conséquent estimer un minimum de couples p'+q pour **n > 149** en ne prenant en compte que **π(n)** est **G(n), où** la fonction **G(n)** ne tiend compte que des entiers **A premiers ou pas ≤ n , non congrus à 2n[P].***

Avec la fonction : $\frac{\pi(n)}{\ln(2n)}$ *ou encore par famille (i) et pour une limite N = n //30 :* $\frac{\pi(N)}{\ln(N)} > \frac{n}{(\ln(n)*\ln(2n))}$

Pour n =15 on avait 4 entiers non congruent[p] et 3 couples de premiers : on pourrait donc aussi utiliser simplement G(n) sur le log de G(n) *au lieu de 14 ÷(ln 14×ln 28) = 1,592* pour les petites limite n.

*C'est à dire : pour **n -1 = 14; G(14)** vaut 14/ln 28 = 4,2 ; qui est le nombres de **A≢ 2n[P], premiers ou pas,** ce qui implique par conséquent et suivant la propriété récurrente* **ex 2)** *ci-dessus : le résultat du nombre de couples pour la limite suivante **2n+2 = 30** , **serra*** le nombre de ***A précédent un entier A'+2 = p'*** *de la limite **n précédente** , que l'on vient de vérifier !* on aura donc ***4/ln 4 = 2*** *couples p'+q < 3 qui est le résultat réels.*

*Pour n+1 = 16 on a **2** couples de premiers qui vérifieront 32 («car pour la limite précédente n=15; G(n) donne : 15 / Ln 30 = 4,.. et **«G(n) sur le log de G(n)»** 4/ln 4 =2,... ≤ **2 ;** le nombre de A' qui précèdent **p'**»)*

*pour n+2 =17, p = 3 et 5; R = 1 et 4: donnera **[**1,**3**,**5**,7,9,**11**,13,15,**17] 4** couples de premiers p'+ q =34. («16 / Ln 32 = 4,.. et 4/ln 4 =2,... ≤ **4**. ou encore avec la fonction modifiée **17÷(ln 17× ln 34) = 1,70..** qui n'est qu'une conséquence du TNP.»)*

*pour **n+3** = 18, sans vérifier, il est clair que l'on aura 4 couples p'+ q = 36 car on a bien 4 A≢ **2n[p]*** **qui précèdent p'. {3, 5, 11 et 15} quand bien même 15 n'est pas un nombre p' mais il est non congruent à P;** Autrement dit l'effet boule de neige, vient du fait que l'on ne tient pas compte uniquement des nombres premiers **A' = p' ≤ n**; mais des *A≢ **2n[p]** précédent les **A' = p'.** «Cela n'a jamais été étudié.»*

*Ce qui permet avec cette propriété du décalage des congruences de ne pas tenir compte de la primalité des A qui précèdent **p'**; mais simplement du fait **qu'ils sont non congruent à P,** ceci rend impossible son infirmation pour la limite suivante **n+1** du fait qu'il y aura toujours des A≢ **2n[p]** précédent un nombre premier car on réplique la même image décalée d'un rang**,** a moins de supposer tous les **A congrus à 2n+2 (mod P)** ce qui est absurde**.***

***Cette fonction** caractérisée par le crible de Goldbach et aussi caractérisé par le TNP, car cela revient à cribler avec Ératosthène et Goldbach uniquement les entiers **A ≢2n[P]** pour la même Fam(i)et la même limite ! Elle serra donc inférieur au nombre réel de couples qui décomposent 2n en somme de deux premiers, lorsque la limite du crible **n → +** ∞.*

*Comme on peut le vérifier pour chaque limite **n**, le décalage des congruences se produit sur plusieurs limite **n + k** successives qui vérifieront la conjecture, ce qui explique cet effet boule de neige, **car on a** au par avant déjà vérifiées, plusieurs limites n – k successives, 2n + 2, 2n , 2n -2... etc ; et on ne peut redescendre indéfiniment puisque la conjecture aura été vérifiée.!*

Conclusion et théorème :** pour toute limite n ≥3 , tout entier A impair tel que **A ≢2n[P]** qui précède **A'= p'** pour la limite **n – 1, n – 2 ;** vérifiera la limite **n** et **n + 1 …** d'où **2n , 2n+2 et 2n + k avec k fini** seront somme de deux nombres premiers **p'+ q.

*Fin pour cette partie avec le crible dans les entiers npn nuls **A** impairs en progression arithmétique **de raison 2** ≤ n. **»]***
En fin de document on construit le programme pour cribler les entiers A impairs de 1 à n.

****

On va cribler dans les familles (i) en progression arithmétique de raison 30:

On va utiliser le même principe, mais en calculant l'index de départ des nombres premiers **P** qui cribleront suivant la famille **30k+(i)** noté fam(i) utilisé par rapport à une limite **n ≥150** en progression arithmétique de raison 15.
Les entiers **A** de **1 à n** sont en progression arithmétique de raison 30. Sont exclu les multiples de 2, 3 et 5.

Avec Ératosthène en début de programme on extrait les nombres premiers ***P*** ∈ $\mathbf{P_n}$ ≤ √**2n** l'ensemble des nombres premiers qui vont cribler avec le crible **G**.
*-On établit un tableau de **1*** (n//30) représentant les **A.***
*-On calcul le reste **R** de 30k + 2i par **P** ∈* $\mathbf{P_n}$.
*-puis si R %2=0 on ajoute **P** tel que **R +P** = j ; ensuite + 2p tant que j%30 est différent de fam (i)*
-si j%30==Fam(i) on calcul l'index tel que j//30 = idx.
-Et on crible de l'idx qui serra marqué 0, par pas de **P**→ n // 30 en remplaçant les ***1*** par 0.
-les **0** seront les entiers **A ≡ 2n[P]**; à la fin on compte les **1** qui sont les **A ≢ 2n[P]** .

Ex: on fixe la limite **n = 15k** = 300, **la fam(i) = 7** progression arithmétique de raison 30 ;
-les **A** seront représenté par des **1: A** ∈ [*7,37,67,97,127………277 < 300*]
-tableau du crible n//30 **[1,1,1,1,1,1,1,1,1,1] p = 7,11,13,17** le **R** de 600 par **p = 5, 6, 2, 5**
-on calcule **j = R +P si R%2=0 puis + 2P** sinon **R** +=**2P** → **j%30 == fam(i)=7**

P=7 , R =5 va donner → 5+14 , 19+14 →33, 47, 61,75, 89, 103, 117,131, 145, 159, 173, **187**==7%30 ok.

on calcul l'index : idx = j//30 , **187 // 30 == 6**. et on va cribler en partant de cet idx,*« attention on compte en commençant par 0 ,1 ,2 , n... → n//30 »* on remplace le 1 par 0 puis par pas de 7 on remplace **1** par **0**.
ce qui va donner ***idx = 6*** *→* **[1,1,1,1,1,1,0,1,1,1]** puis *on réitère avec* ***P = 11, R = 6***
127 == j%30 et ***idx = 4***→ **[1,1,1,1,0,1,0,1,1,1]** puis *on réitère* ***. P = 13, R = 2,***
67 *== j%30 et* ***idx = 2****→* ***[1,1,0,1,0,1,0,1,1,1]*** *puis on réitère* ***. P = 17, R = 5*** *donnera 277 ,*
277*== j%30 et* **idx = 9**→ ***[1,1,0,1,0,1,0,1,1,0]*** *fin, on fait la somme des 1 = 7* ***A ≢ 2n[P]*** *⇒ 7 premiers* ***q∈[n;2n].***

Ératosthène pour la même limite criblée mais avec ***P ≤ √n*** *et la même fam(i) donnera le tableau des nombres :*
1= p'∈ [1,1,1,1,1,1,0,0,0,1] *⇔ [7,37,67,97,127………..277 < 300].*
En criblant le tableau de ***1*** *avec le crible (G) →* ***0 = A ≡2n [p]∈ : [1,1,0,1,0,1,0,1,1,0] ;*** *ce qui donne le résultat suivant, pour le tableau d'Ératosthène criblé : nombre de* ***,1, ⇒*** *couples* ***p'+ q*** *= 600; 4 couples ∈* ***[1,1,1,1,1,1,0,0,0,1]***

La propriété récurrente est la même, lorsque ***n*** *ou* ***n+i*** *augmente de 15 les congruences se décalent d'un rang sur leur successeur* ***A+30.*** *«On a nul besoin de se soucier de la fam i complémentaire pour vérifier la conjecture, l'algorithme utilise les congruences et* ***q*** *a pour antécédent* ***A ≢2n[P] .»***

Cela donnera par obligation le tableau suivant pour 2n= 30(k+1) = 630 ; ***4*** *couples, sans même avoir besoin de cribler: [7,37,67,97,127,157,187,217,247,277] avec le décalage d'un rang des congruences [****x,1, 1****, 1,* ***1****, 1, 0, 0, 0,* ***1****] seul le premier élément x est inconnu .*

résultat réel pour n = 15(k+1) = 315; vérifié :

Donnez N: 315; crible EG_2n_mod30*: [****1****,* ***1, 1****, 1,* ***1****, 1, 0, 0, 0,* ***1, 1****]* ***4*** *couples p+q = 2n. Et ~5 , pour 30(k+2) = 660*

Donnez N: 330, décalage d'un rang *[****1****, 1,* ***1, 1****, 1,* ***1****, 0, 0, 0,* ***1, 1****] réel* ***6*** *couples*

Ce qui permet d'affirmer formellement que la conjecture serra vérifié pour la limite ***2n = 30(k+1) + 2i****; sans avoir besoin de cribler cette limite* **n = 15(k+1)+i.**

En effet: si **A ≢** ***2n[p]*** premier ou pas précèdent **A+30 premier p',** congru ou pas à **2n [p]** alors ce dernier, qui par obligation serra **non congruent à P,** il formera un couple: **p'** + q qui décompose **2n = 30(k+1) +2i** en somme de deux premiers !

Il devient donc avec d'autres raisonnements à l'appui, **impossible d'infirmer la conjecture** pour la limite suivante **2n = 30(k+1) +2i ,** car il est aussi **impossible d'utiliser les restes R** de la limite **n** précédente pour cribler la limite **n+1;** le contraire infirmerait l'égalité que l'on a montrée *l'égalité* ***30 – A ⇔ (30+2) – (A+2)*** *conséquence de la propriété récurrente de l'algorithme : le décalage d'un rang des congruences pour tout n +15 , par fam(i).*

La fonction 2 du théorème de Goldbach est une conséquence directe du TNP: (log = *logarithme naturel)*
G(n): la fonction de compte du nombre de nombres ***A≢ 2n[P]*** ⇔ **q ∈ [n;2n]**

Corollaire: **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 $]n,2n]$ vaut

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

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

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

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

*Tout nombre pair 2n ≥ 180 peut s'écrire comme la somme de deux nombres premiers (**p'**+**q**)) appartenant à une famille Fam(i) tel que définie en début de document.*
*Conséquence des deux fonctions du TNP on déduit une troisième fonction qui indique : que le nombre de **A≢ 2n[P]** qui précèdent un entiers **A'= p'⇒** le nombre de couples **p'+q = 2n** est équivalent à*

$\frac{n}{(\ln(n)*\ln(2n))}$ *lorsque **n** tend vers + ∞. Et par famille on a pour une limite N =n//30 :* $\frac{\pi(N)}{\ln(N)}$

Heuristiquement on peut aussi : pour n ≥ 3000 : $C_2 \frac{G(n)}{\ln G(n)}$ **;** où C_2 ≈ 1,320323… constante premiers jumeaux.
*Pour que la conjecture soit fausse pour une limite **n,** il aurait fallu utiliser les restes R des limites **n-k** précédentes, mais aussi que la fonction* $\frac{n}{(\ln(n)*\ln(2n))}$ *soit nulle ; or pour une limite **n** qui est*

*vérifiée , cette fonction indique aussi le nombre de **A ≢2n[P]** qui précèdent **A'= p'** des limites précédentes aux rangs **n-1** et **n-2** qui par supposition ont été vérifiées et on a vérifié la limite **n;** le contraire serait contradictoire, par conséquent cette fonction sera toujours positive !*

Mais qui plus est, il est clairement impossible, que la fonctions du TNP ou son corollaire soit nulle.

Reste alors la solution**: elle est indécidable, en supposant **qu'il n'existe pas** de limites précédentes **n** – k , **n** – 2 etc **n** – 1 pouvant être vérifiées, qui permettent d'affirmer que la conjecture serra vraie pour la limite **n** et donc **n+1.
***Or : si la limite n – (k – 1)** a été vérifiée, alors successivement les limites **n** – k, **n** – 2 , **n** – 1 seraient aussi vérifiées suite à cette propriété récurrente, qui se propage sur un ensemble fini de plusieurs limites successives.*
Contradiction ou paradoxe ? On en déduit que la conjecture est donc vraie.

****

***Pour information annexe 1**:*

Illustration:

(p''+q) le nombre de couples qui vérifiera 30(k+1) = 2n +30 , illustré ci-dessous a une variation près.

Limite **N =15k** , en progression arithmétique de raison 15.
Les 1 montrent le début du décalage d'un rang des congruences de la **ligne G** pour chaque changement de limite: N = **15(k+1)** et jusqu'à **quelle limite N =15(k+ n) elle serra vérifiée ;** alors que la ligne **E,** ne se décale pas bien évidemment, mais augmente d'un élément lorsque N = 15(k+2) donc augmente de 30.

Dans cette illustration en partant de N = 300 elle sera vérifiée jusqu'à N = 450 suite à la propriété récurrente du décalage des congruences sur leur successeur A' = A+30.
Autrement dit on prend en compte les entiers **A**, **premiers ou pas**, **Mais non congruent à P, qui précède un A' = p'** congru ou pas à 2N [P].

Ce qui permet de prédire la vérification de la conjecture sur plusieurs limites 2n +30 successives, car on ne fait que dupliquer l'image de la limite n précédente décalée d'un rang.

Fam(i) = fam 30k +7 qui sont criblés par **G** et **E**: 7; 37 ;67 ;97 ;127 ;157 ;187…..etc 30k +7
« le résultat réel est vérifié par le crible EG »
les entiers **A**, représentés par les **1 ou 0 non congruent à P,** pour une limite **n=15k** vérifiée , implique le nombres de solution pour la limite **n = 15(k+1)** qui serra vérifiée avec le crible EG.

Ex** première lignes ci-dessous : pour la limite **n =15k = 300** ; on a 4 **A** **≢2n[P]** représentés par **1 et 0
*qui précèdent un **A = 1 = p'** premier ou un 1 = p' congruent modulo P ligne **E** ; c'est à dire qu'il soit congru ou **pas** :*
*il vient par conséquent, pour la limite n =15(k+1) = 315 , soit **2n = 30(k+1) = 630** on aura 4 couples p'+q = 630 à une exception près, à cause du dernier élément qui ne permet pas de voir si il précède ou pas un A' = p'.*
*Que l'on peut vérifier deuxième ligne **E** pour n=15(k+1) = 315, EG réel = 4.*

G[**1**, **1**, 0, **1**, 0, **1**, 0, **1**, **1**, 0] **n = 300** ; **pour** 630 EG: = 4 p'+q; **fonct** G(n) = 6; ***G(n) /ln G(n) = 3,348***
E[1, 1, 1, 1, 1, 1, 0, 0, 0, 1] EG réel = 4 et pour n+15, on aura avec G(n) = 4 : ***G(n) /ln G(n) = 2,885***

G[0, **1**, **1**, 0, **1**, 0, **1**, 0, **1**, **1**] n+15= 315; **pour** 660 EG: = 5 p'+q ; G(n) = 6; ***G(n) /ln G(n) = 6/Ln 6 = 3,348***
E[1, 1, 1, 1, 1, 1, 0, 0, 0, 1] EG réel = 4 *le dernier 1 ne permet pas de voir si il précède un* **1 = P' ou 0**

G[1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1]n=330 ; **pour (30k+1)** = 690 EG = 4 p'+q; G(n) = 7; G(n) /ln G(n)
E[1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1] EG réel = 6;

G[1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1]n=345; 690 EG: = 5 p'+q; G(n) = 7; G(n) /ln G(n)
E[1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1] EG réel = 5;

G[0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1] n=360; 720 EG: = 4 p'+q; G(n) = 7; G(n) /ln G(n)
E[1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1] EG réel = 6

G[1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0] n = 375; 750 EG: = 5 p'+q; G(n) = 7; G(n) /ln G(n)
E[1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1] EG réel = 5 [1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0]

G[1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0] n=390; 780 EG: = 6 p'+q; G(n) = 8; G(n) /ln G(n)
E[1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1] e EG réel = 6 [1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0]

G[0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1] n=405; 810 EG: = 5 p'+q; G(n) = 9; G(n) /ln G(n)
E[1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1]e EG réel = 6 [0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1]

G[0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1] n =420; 840 EG : = 5 p'+q; G(n) = 9 ; G(n) /ln G(n)
E[1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1] EG réel = 6 [0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1]

G[1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0] n =435; 870; EG: = 6 p'+q ; G(n) = 9; G(n) /ln G(n)
E [1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1] EG réel = 6 [1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0]

G[0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0] n =450; 900 EG: = 5 p'+q; G(n) = 9; G(n) /ln G(n)
E[1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0] EG réel =6 [0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0]

Pour n = 465 il y aura 5 p'+q en réel, car les deux derniers A = 0 ligne **G** et **E** sont congrus **2n [P].**

En générale le nombre de p'+q et différent d'une unité avec le nombre de p'+q de la limite précédente.
l.g

Les programmes sont disponibles en python ou C++

ci dessous quelques tests pour les statistiques que l'on peut vérifier avec la fonction $\frac{\pi(N)}{\ln(N)}$

avec N = 15k + 2 et 3 fam(i) : (11 ; 17 ; 23)

N = 15k +1 et 3 autres fam(i) : (1 ; 13 ; 19)

/home/gilbert/Programmes/E_G_Crible_8.fam

```
famille : 11 limite : 4500000000002
Nombre premiers criblés famille 11 plus petits que 4500000000002: 20020233398 time 517.285
Nombre couples p+q=2N criblés famille 11 : 2607250497 time 534.803
famille : 11 limite : 4500000000017
Nombre premiers criblés famille 11 plus petits que 4500000000017: 20020233398 time 519.102
Nombre couples p+q=2N criblés famille 11 : 2414789785 time 538.586
famille : 11 limite : 4500000000032
Nombre premiers criblés famille 11 plus petits que 4500000000032: 20020233398 time 517.226
Nombre couples p+q=2N criblés famille 11 : 2387299132 time 4831.89
famille : 11 limite : 4500000000047
Nombre premiers criblés famille 11 plus petits que 4500000000047: 20020233398 time 4812.35
Nombre couples p+q=2N criblés famille 11 : 2387419091 time 4831.23
famille : 11 limite : 4500000000062
Nombre premiers criblés famille 11 plus petits que 4500000000062: 20020233398 time 4812.37
Nombre couples p+q=2N criblés famille 11 : 2881665432 time 4833.6
famille : 11 limite : 4500000000077
Nombre premiers criblés famille 11 plus petits que 4500000000077: 20020233398 time 4812.28
Nombre couples p+q=2N criblés famille 11 : 2392874730 time 4831.57
famille : 11 limite : 4500000000092
Nombre premiers criblés famille 11 plus petits que 4500000000092: 20020233398 time 4812.37
Nombre couples p+q=2N criblés famille 11 : 2543413744 time 9127.56
famille : 11 limite : 4500000000107
Nombre premiers criblés famille 11 plus petits que 4500000000107: 20020233398 time 9108.35
Nombre couples p+q=2N criblés famille 11 : 2390888872 time 9125.38
famille : 17 limite : 4500000000002
Nombre premiers criblés famille 17 plus petits que 4500000000002: 20020230303 time 9106.88
Nombre couples p+q=2N criblés famille 17 : 2607237962 time 9125.06
famille : 17 limite : 4500000000017
Nombre premiers criblés famille 17 plus petits que 4500000000017: 20020230303 time 9103.24
Nombre couples p+q=2N criblés famille 17 : 2414764165 time 9122.85
famille : 17 limite : 4500000000032
Nombre premiers criblés famille 17 plus petits que 4500000000032: 20020230303 time 9107.6
Nombre couples p+q=2N criblés famille 17 : 2387361906 time 13418.7
famille : 17 limite : 4500000000047
Nombre premiers criblés famille 17 plus petits que 4500000000047: 20020230303 time 13402.3
Nombre couples p+q=2N criblés famille 17 : 2387391058 time 13418.2
famille : 17 limite : 4500000000062
Nombre premiers criblés famille 17 plus petits que 4500000000062: 20020230303 time 13402
Nombre couples p+q=2N criblés famille 17 : 2881684253 time 13418.8
famille : 17 limite : 4500000000077
Nombre premiers criblés famille 17 plus petits que 4500000000077: 20020230303 time 13403.6
Nombre couples p+q=2N criblés famille 17 : 2392893286 time 13417.8
famille : 17 limite : 4500000000092
Nombre premiers criblés famille 17 plus petits que 4500000000092: 20020230303 time 13403.8
Nombre couples p+q=2N criblés famille 17 : 2543414474 time 17711.5
famille : 17 limite : 4500000000107
Nombre premiers criblés famille 17 plus petits que 4500000000107: 20020230303 time 17694
Nombre couples p+q=2N criblés famille 17 : 2390985893 time 17713.4
famille : 23 limite : 4500000000002
Nombre premiers criblés famille 23 plus petits que 4500000000002: 20020235931 time 17700.1
Nombre couples p+q=2N criblés famille 23 : 2607229116 time 17718.3
famille : 23 limite : 4500000000017
Nombre premiers criblés famille 23 plus petits que 4500000000017: 20020235931 time 17694.9
Nombre couples p+q=2N criblés famille 23 : 2414889851 time 17712.8
famille : 23 limite : 4500000000032
Nombre premiers criblés famille 23 plus petits que 4500000000032: 20020235931 time 17699.9
Nombre couples p+q=2N criblés famille 23 : 2387362595 time 22010.5
famille : 23 limite : 4500000000047
Nombre premiers criblés famille 23 plus petits que 4500000000047: 20020235931 time 21992.9
Nombre couples p+q=2N criblés famille 23 : 2387344606 time 22011.9
famille : 23 limite : 4500000000062
Nombre premiers criblés famille 23 plus petits que 4500000000062: 20020235931 time 21990
Nombre couples p+q=2N criblés famille 23 : 2881730868 time 22011.8
famille : 23 limite : 4500000000077
Nombre premiers criblés famille 23 plus petits que 4500000000077: 20020235931 time 21991.4
Nombre couples p+q=2N criblés famille 23 : 2392862314 time 22013.1
famille : 23 limite : 4500000000092
Nombre premiers criblés famille 23 plus petits que 4500000000092: 20020235931 time 21993.5
Nombre couples p+q=2N criblés famille 23 : 2543470424 time 26306.2
famille : 23 limite : 4500000000107
Nombre premiers criblés famille 23 plus petits que 4500000000107: 20020235931 time 26287.3
Nombre couples p+q=2N criblés famille 23 : 2390956946 time 26305.6

Process returned 0 (0x0)   execution time : 28635.434 s
Press ENTER to continue.
```

/home/gilbert/Programmes/E_G_Crible_8.fam

```
 famille : 1 limite : 4500000000001
 Nombre premiers criblés famille 1 inférieur a 4500000000001: 20020196872 Nombre couples p+q=2N criblés famille 1 : 2901992180 f
00000016
 Nombre premiers criblés famille 1 inférieur a 4500000000016: 20020196872 Nombre couples p+q=2N criblés famille 1 : 2527795581
famille : 1 limite : 4500000000031
 Nombre premiers criblés famille 1 inférieur a 4500000000031: 20020196872 Nombre couples p+q=2N criblés famille 1 : 2387314194
famille : 1 limite : 4500000000046
 Nombre premiers criblés famille 1 inférieur a 4500000000046: 20020196872 Nombre couples p+q=2N criblés famille 1 : 2411945980
famille : 1 limite : 4500000000061
 Nombre premiers criblés famille 1 inférieur a 4500000000061: 20020196872 Nombre couples p+q=2N criblés famille 1 : 2421888865
famille : 1 limite : 4500000000076
 Nombre premiers criblés famille 1 inférieur a 4500000000076: 20020196872 Nombre couples p+q=2N criblés famille 1 : 2864885208
famille : 1 limite : 4500000000091
 Nombre premiers criblés famille 1 inférieur a 4500000000091: 20020196872 Nombre couples p+q=2N criblés famille 1 : 2556655085
famille : 1 limite : 4500000000106
 Nombre premiers criblés famille 1 inférieur a 4500000000106: 20020196872 Nombre couples p+q=2N criblés famille 1 : 2725089069
famille : 13 limite : 4500000000001
 Nombre premiers criblés famille 13 inférieur a 4500000000001: 20020216296 Nombre couples p+q=2N criblés famille 13 : 290199947
8 famille : 13 limite : 4500000000016
 Nombre premiers criblés famille 13 inférieur a 4500000000016: 20020216296 Nombre couples p+q=2N criblés famille 13 : 252777344
3 famille : 13 limite : 4500000000031
 Nombre premiers criblés famille 13 inférieur a 4500000000031: 20020216296 Nombre couples p+q=2N criblés famille 13 : 238734035
4 famille : 13 limite : 4500000000046
 Nombre premiers criblés famille 13 inférieur a 4500000000046: 20020216296 Nombre couples p+q=2N criblés famille 13 : 241193078
2 famille : 13 limite : 4500000000061
 Nombre premiers criblés famille 13 inférieur a 4500000000061: 20020216296 Nombre couples p+q=2N criblés famille 13 : 242190297
5 famille : 13 limite : 4500000000076
 Nombre premiers criblés famille 13 inférieur a 4500000000076: 20020216296 Nombre couples p+q=2N criblés famille 13 : 286480902
6 famille : 13 limite : 4500000000091
 Nombre premiers criblés famille 13 inférieur a 4500000000091: 20020216296 Nombre couples p+q=2N criblés famille 13 : 255660833
6 famille : 13 limite : 4500000000106
 Nombre premiers criblés famille 13 inférieur a 4500000000106: 20020216296 Nombre couples p+q=2N criblés famille 13 : 272515250
7 famille : 19 limite : 4500000000001
 Nombre premiers criblés famille 19 inférieur a 4500000000001: 20020198781 Nombre couples p+q=2N criblés famille 19 : 290201359
1 famille : 19 limite : 4500000000016
 Nombre premiers criblés famille 19 inférieur a 4500000000016: 20020198781 Nombre couples p+q=2N criblés famille 19 : 252784065
5 famille : 19 limite : 4500000000031
 Nombre premiers criblés famille 19 inférieur a 4500000000031: 20020198781 Nombre couples p+q=2N criblés famille 19 : 238738090
1 famille : 19 limite : 4500000000046
 Nombre premiers criblés famille 19 inférieur a 4500000000046: 20020198781 Nombre couples p+q=2N criblés famille 19 : 241196101
9 famille : 19 limite : 4500000000061
 Nombre premiers criblés famille 19 inférieur a 4500000000061: 20020198781 Nombre couples p+q=2N criblés famille 19 : 242193011
6 famille : 19 limite : 4500000000076
 Nombre premiers criblés famille 19 inférieur a 4500000000076: 20020198781 Nombre couples p+q=2N criblés famille 19 : 286477377
6 famille : 19 limite : 4500000000091
 Nombre premiers criblés famille 19 inférieur a 4500000000091: 20020198781 Nombre couples p+q=2N criblés famille 19 : 255656769
5 famille : 19 limite : 4500000000106
 Nombre premiers criblés famille 19 inférieur a 4500000000106: 20020198781 Nombre couples p+q=2N criblés famille 19 : 272508126
3
Process returned 0 (0x0)   execution time : 28623.772 s
Press ENTER to continue.
```

Annexe 1

En fonction de la limite **N = 15k** fixée, on fixe lune des **8 Fam (i)** correspondante à la forme de **N**

Pour **N** de la forme **15k**, on peut choisir n'importe la quelle des 8 Fam.

Le programme est fixé avec une limite **N** début = 3000000000 et Fin = 3000000330 il progressera de raison 15,
Il n'y a que la fam(i) à rentrer à la demande:

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**
N = 15k+2; Fam =(11,17,23); **2n = 30k + 4**
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**
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 N = 15(k+1) + 0; l'une des 8 Fam 2n = 30 (k + 1)

Programme c++ utilisé avec Code::blocks

```
// -*- 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 lencrible)
{ //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 ci dessus
  // 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 << " Nombre premiers criblés famille " << fam << " inférieur 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 lencrible)
{
  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 ci dessus
  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'indice 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 << " Nombre couples p+q=2N criblés famille " << fam << " : " << total;// << " time " << (clock() - cl) * 1e-6 << endl;
  return total;
}

int main(int argc, char **argv)
{
  vector<unsigned> temp;   // on modifie la limite N = début et fin de raison 15 ou k*15
  ulonglong debut = 100020;
  ulonglong fin = 100035;

  vector<int> familles;    //pour changer de Famille en fonction de N on active ou désactive les deux //
  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 << " famille : " << fam << " 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);
    }
  }
}
```