Introduction:

[« 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***, ainsi que la fonction asymptotique du TNP qu'il faut modifier dans ces ***8 Fam (i)*** de nombres premiers **P' > 5**, représentant 26,666… % des entiers naturels non nuls; soit ***n / 3,75 .***

Car en effet, un nombre premier **q** = ***n – m dépend statistiquement de la congruence de m*** *autrement dit* ***q à pour antécédent m*** et donc si et seulement si **m** est un nombre premier **P'** ≤ ***m*** **et tel que P' ≢ n [P] alors q a pour antécédent P', car q** ne peut être un multiple de **P,** mais son complémentaire par rapport à **n ou 2m.**

On peut déjà affirmer : que ces fonctions ci-dessus modifiées deviennent : (**1**/Ln **m***Ln **2m**) d'être premier et d'être ***non congru à 2m (mod P)*** , on obtient : **m** / (**Ln m . Ln 2m)** *le nombres d'entiers* ***A*** ≢ **2m [P]** *qui précède un entier* ***A' = p'*** ⇔ **p' + q = 2m ,** avec **A** non nul de **1** à **m**, car le nombre d'entiers ***m non congru 2m mod P*** *est équivalent à* ***m*** / **Ln 2m,** *lorsque* ***m*** → **+** ∞ *ce qui implique les nombres premiers* **q tel que q =** ***2m − m*** **,** en utilisant l'algorithme de Goldbach**.**
Ce que l'on verra ci-après avec les fonctions du TNP et les définitions relatives à ces algorithmes dans ces 8 Fam(i).**»]**

Document complémentaire sur l'illustration de l'algorithme de Goldbach et ses propriétés :

les entiers A sont les entiers non nuls en progression arithmétique de raison 30, dans les deux cribles, représentés par des 1 ou des 0.
On veut prouver que quelque soit une limite N vérifiée, tel que 2N est somme de deux nombres premiers P' + q , alors la conjecture serra prouvée pour la limite 2N + 2 pour toute limite N ≥ 150 à l'aide d'une famille d'entiers A, non nuls en progression arithmétique de raison 30 et de premier terme (i) ∈ (1, 7, 11, 13, 17, 19, 23 ,29). La vérification de la conjecture pour une limite N⇒N+1; « plus généralement pour N ≥ 10⁹, *La vérification de la conjecture pour une limite* ***N⇒N+1+2+3+…..n; sur plusieurs limites successives »***

Le Décalage d'un rang des congruences sur les ,1, ou les ,0, ***relatif à la propriété de l'algorithme, vérifiera la conjecture pour 30(k+1) + 2i soit 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 1sont les P' non congruent à P
fam1 crible G : et : Fam 1 Ératosthène EGCrible 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, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 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, 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 précédente qui a été vérifiée : 15(k-1) + 1 = 2985 + 1, avec sa Fam (i = 1) pour affirmer que la conjecture serra vérifiée pour la limite suivante 15k + 1 , tel que 30k + 2 est somme de deux premiers P'+ q !

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; les 1 et les 0 sont ≡ 2n [P]
[1, 1, 1, 0, 0, 1, 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, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 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 $\not\equiv$ 2n[P]

[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, 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, 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 17 couples P'+ q au minimum qui vérifieront 30(k+1) + 2i = 6002. Vérification avec EGCrible:

[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, 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 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 ! *L.g* §

--

Pour 2n +2+2 = 6004 , il suffit de regarder la limite N précédente 15(k-1) + 2 = 2985 + 2, avec sa Fam (i = 17 ou 11 ou 23) qui a aussi été vérifiée.

N = 2987; fam 11 ; Ecrible P' = 1 ou 0 ⇔ $\not\equiv$ 2n [P] et en dessous Gcrible Entier A = 1 ⇔ $\not\equiv$ 2n [P] et P≤ √2n

*avec le décalage des congruences des A non congrus **1** ou **0 se reportent sur les** A+30 = P'= **1**ou **1** congrus ou pas **(30(k-1)+ 2) [P].***

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, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0]

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]

vérification 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] réel 21 couples p+q = 2N

--

*nous venons de vérifier le décalage récurrent des congruences des A=**1** ou A=**0 non congruent (mod P) qui se reportent** sur leur successeur A'+30 = **1**ou **1**, **qui ont pour antécédent A,** le contraire contredirait le TNP et Le TFA, (cqfd)*

La conjecture se montre avec l'une des 8 Fam(i) en fonction de la forme de N et pour toute limite N ≥ 150 :

*1) les nombres A **$\not\equiv$ 2n[P]** ⇔ **q** ∈ **[N;2N]** est donc défini pour **2N + 2** à une exception près !*
*Car les restes R de **30(k-1)+2i** par P ne peuvent plus être utilisés du simple fait qu'ils ne correspondent pas **aux nouveaux** R de **30k + 2i** par **P** qui eux sont les mêmes, le contraire serait absurde.*
*Si on utilisait les R de **30(k -1)+ 2i et les** R de 30k + 2i, le cardinal du nombre de nombres premiers **q** serait faux pour la limite **2N+2** qui a déjà été fixé et vérifié avec les nombres **P** qui ont criblés la limite N – 15 précédente.*

2) Si ce décalage d'un rang des congruences sur leur successeur A'+30 ne s'effectuait pas, le TFA serait contredit ce qui là aussi est absurde ! Du fait que:
L'égalité : (2n – A = B)≠ ((2n+30) – (A+30) = B) ce qui est idiot, B est toujours le même; si B est un nombres premiers P' il pourrait être multiple de P si ce décalage des non congruences ne se décalaient pas sur lui , et inversement un multiples de P pourrait se trouver premiers si la congruence ne se décalait pas sur lui, pour la limite suivante 15(k+1)+i.
*Car : Suivant l'exemple ci dessus avec la **fam 11**, nous avons le troisième élément **71 $\not\equiv$ 2N[P]** pour 2N = (2*2987) et tel que **5974 – 71 = q = 5903** premier et **101 est congruent** a P pour cette limite N, il est évident que pour la limite suivante, N =15(k+1)+2 = 3002 **⇒ 2N = 6004** – **(71+30)** = **5903 = q** il est bien toujours le même, par conséquent et suivant cette propriété de décalage d'un rang, **101** serra par obligation **non congruent** à P, le contraire est impossible !*
Conclusion:
*a) L'égalité récurrente du décalage des congruences tel que : **(2n – A = B)⇔ ((2n+30) – (A+30) = B).***

*b) Si un entier **A** en progression arithmétique de raison 30, de premier terme **i** , premiers P' ou pas, tel que A $\not\equiv$ **2N[P]** **précède A+30 = P' congrus ou pas à 2N [P] , suite au décalage d'un rang des congruences P' vérifiera la conjecture pour la limite N=15k, 15(k+1)+ i** et ainsi de-suite pour plusieurs limites successives.*
Autrement dit**, cela implique lors des limites précédentes N – 15k au rang n – k, on connaît déjà les couples qui vérifieront la conjecture pour cette limite N . Ce qui rend impossible l'infirmation de la conjecture pour la limite suivante N+1 sans avoir besoin de la cribler, 2N + 2 serra donc somme de deux nombres premiers. Ce que l'on peut montrer ci-dessous en **d).

c) L'impossibilité de réutiliser les restes ***R*** *de 2n par P pour la limite suivante =* ***15(k+1)+ 1*** *avec sa* ***fam(i)*** *confirme la véracité de la conjecture de Goldbach ; on en déduit alors qu'il est impossible de l'infirmer !*

Vérifions le troisième cas :
pour 15(k-1)+ 3 = 2988 ; soit pour 2n +2+2+2 = 6006 , on regarde la limite précédent n = 15(k-1)+ 3 = 2985 + 3 et sa fam (i = 7, ou 29 ou 17 ou 19 ou 13 ou 23):
N = 2988; fam 7 ; Ecrible P' = 1 ou 0 ⇔ ≢ 2n [P] et en dessous Gcrible Entier A = 1 ⇔ ≢ 2n [P] et P ≤ √2n

E:{7,37,67,97,127,157…..2977} = N // 30 donne les nombres d'entiers (mod 30) < 2988 qui sont criblés, soit 99.
[1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0]
G:
[0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0]
vérification EG , N = 3003:
[0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0]

d) On pourrait supposer: qu'en serait il de la limite **N = 15k + 6 = 3006 ?**
Étant donné la conjecture vérifiée pour tout **N ≤ 3000** il suffit de s'apercevoir qu'au rang 6; c'est à dire
la fam **(i)** qui a vérifiée N – 90 =15(k-6)+6 = **2916** avec l'une des 6 fam **(i) = (1,11 ; 13,29 ; 19,23)**:
il y avait obligatoirement des entiers **A** ***≢ 2N[P] qui précédaient P' au rang 6; car cela est dû à la propriété récurrente de l'algorithme, afin de préserver l'égalité du point*** *a).*

vérification:
E: *pour N =* ***2916*** ***et fam 1 {****1,31,61,91,121,151,181,211 ..etc;* ***A*** *précède au rang 6 un P'=* ***1 ou 1*** ***donc :*** *tout entier* ***A= 1 ou 0 non congruent à P, qui précède un nombre premier*** *p' ou* ***p' congru*** ***ou pas*** *au rang 6,* ***vérifiera la conjecture pour la limite N= 15(k+6) + 6 = 3006.***

[1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0]
G:
[0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0] *soit au minimum 23 sur 29 couples* ***P'****+q, qui vérifieront* ***2N = 6012*** ***; il n'y a rien d'aléatoire.***
Donnez N: 3006
[1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1]. Nombres P' non congru 2n[pi], de 1 à 3006 famille 1 : 29 ----- 0.01

En règle générale la conjecture est vérifiée sur plusieurs limites 15(k+1,2,...n) successives lorsque n tend vers l'infini, avec un effet boule de neige, («à l'inverse d'une comète qui se désagrège»)
La formule d'estimation du nombre de couples p+q = 2n est une conséquence directe du TNP caractérisée par les deux fonction relatives aux deux algorithmes :

G(n) nombre d'entiers A ≢2n[P] ≤ n, vaut ~ $\lim_{n \to +\infty}$ ¿ $\frac{n}{(\log 2n)}$

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

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

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

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

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

Conséquence des deux fonctions du TNP*, on en déduit que le nombre d'entiers* ***A≠ 2n [P]*** *qui précèdent un entier* ***A'=P'⇔ P'+ q = 2n*** *est équivalent à:*

$$\frac{n}{(\ln(n) * \ln(2n))} \quad \textit{lorsque} \quad \lim_{n \to +\infty} ¿$$

Autre estimation heuristique : Le nombre de couples de nombres premiers (p', q) ou le nombre de nombres premiers A = P' **non congrus à 2n modulo P** de 1 à n ; vaut environ, lorsque la $\lim_{n \to +\infty}$ ¿ pour n =15k + n' , avec n' ∈ [0 ; 14] et **k** entier naturel non nul :

simplement $\frac{\pi(n)}{\ln \pi(n)}$ mais plus précisément : $C_2 \frac{G(n)}{\ln G(n)}$ **;** où G(n) est la fonction qui compte le nombre

d'entiers A de 1 à n , non congruent modulo **P** et $C_2 \approx 1{,}320323$ constante des premiers jumeaux.

****

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**

N = 15(k+1); ***l'une des 8 Fam*** ***2n = 30 (k + 1)***

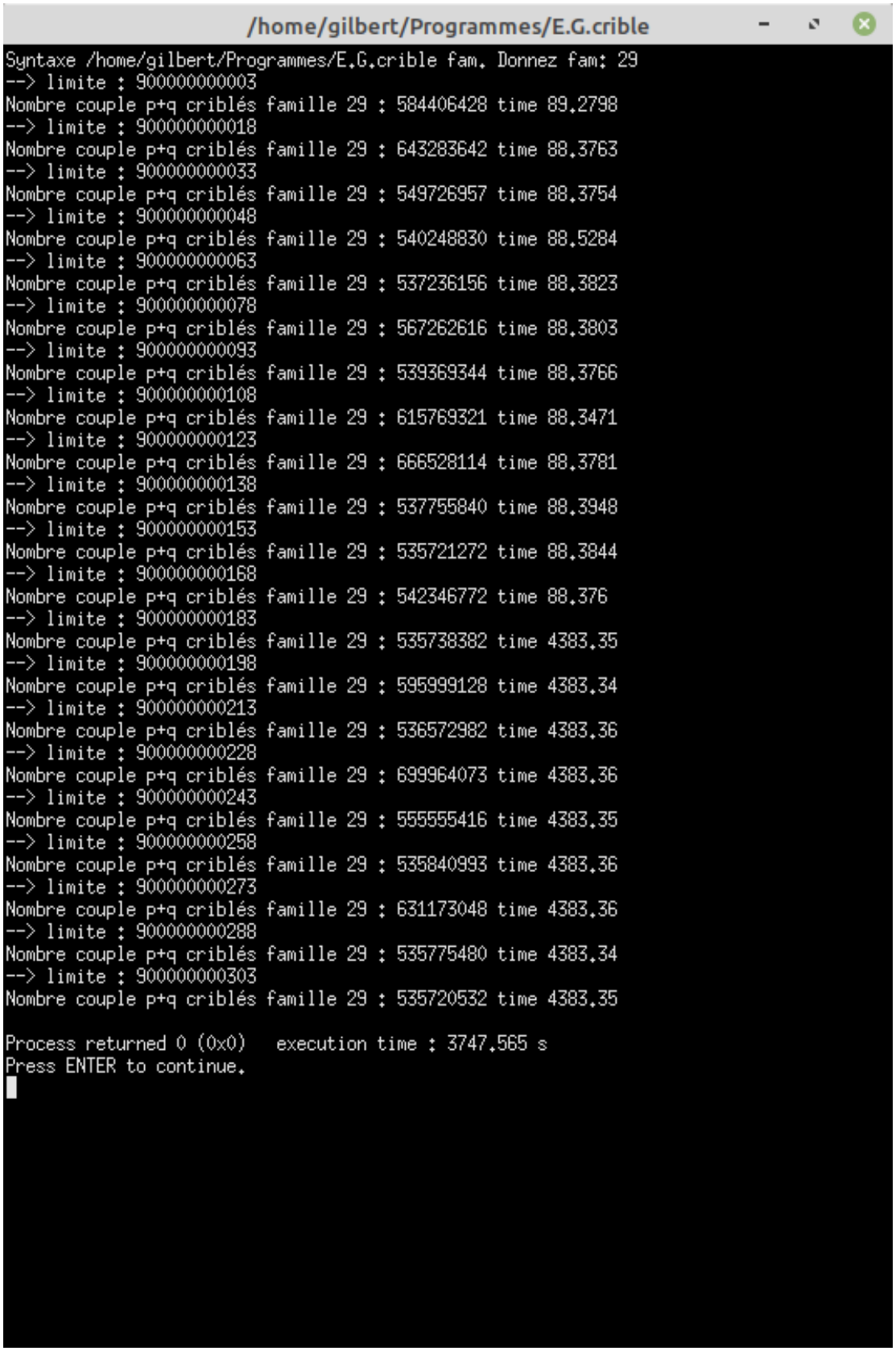
****

Connaissant le nombre de premiers P' de 1 à N; on peut aussi estimer par fam(i) comme ci dessous :

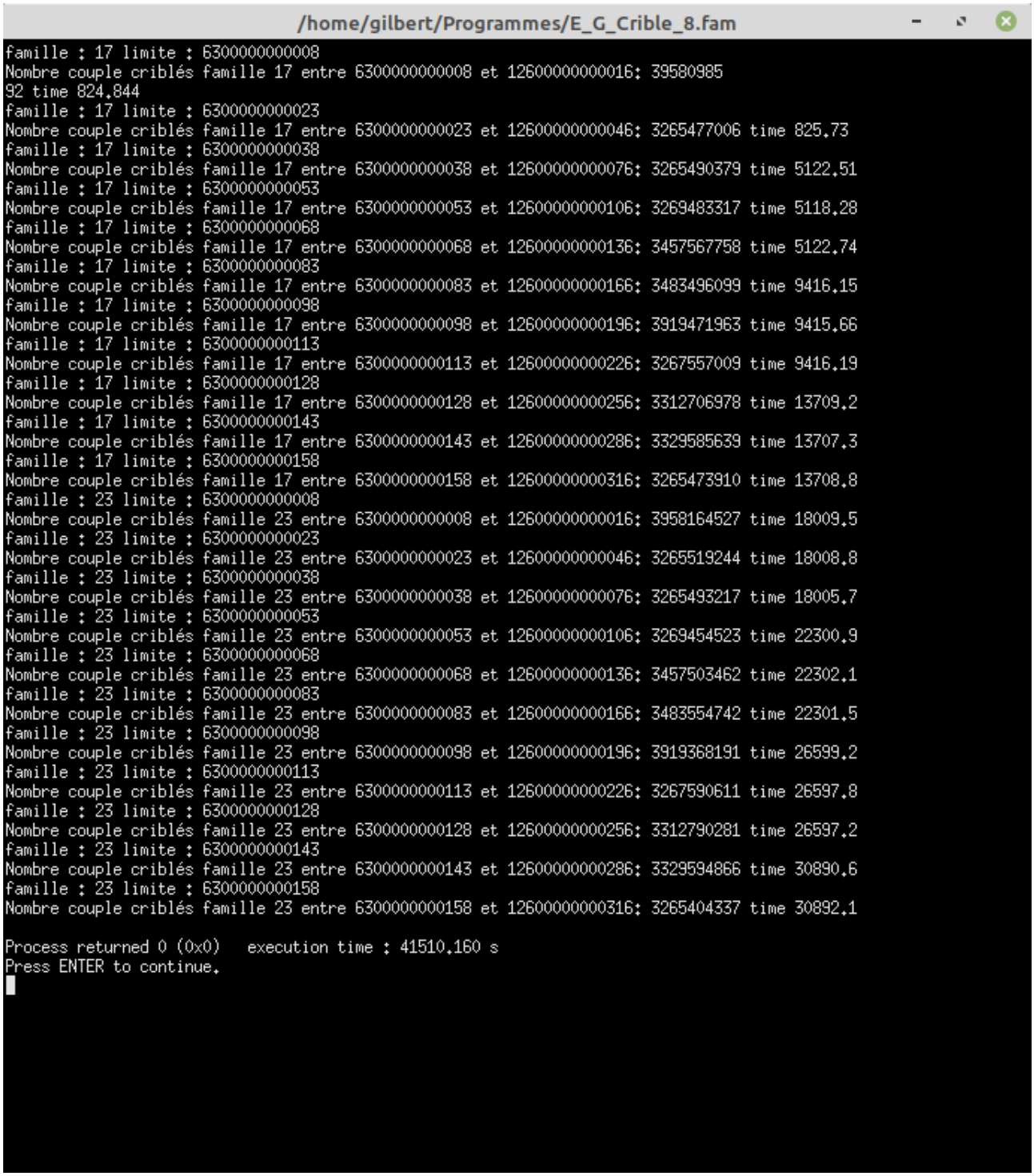
En utilisant les deux fonctions de π(n)= n /log n et G(n)= n /log 2n ; puis on fait la différence.

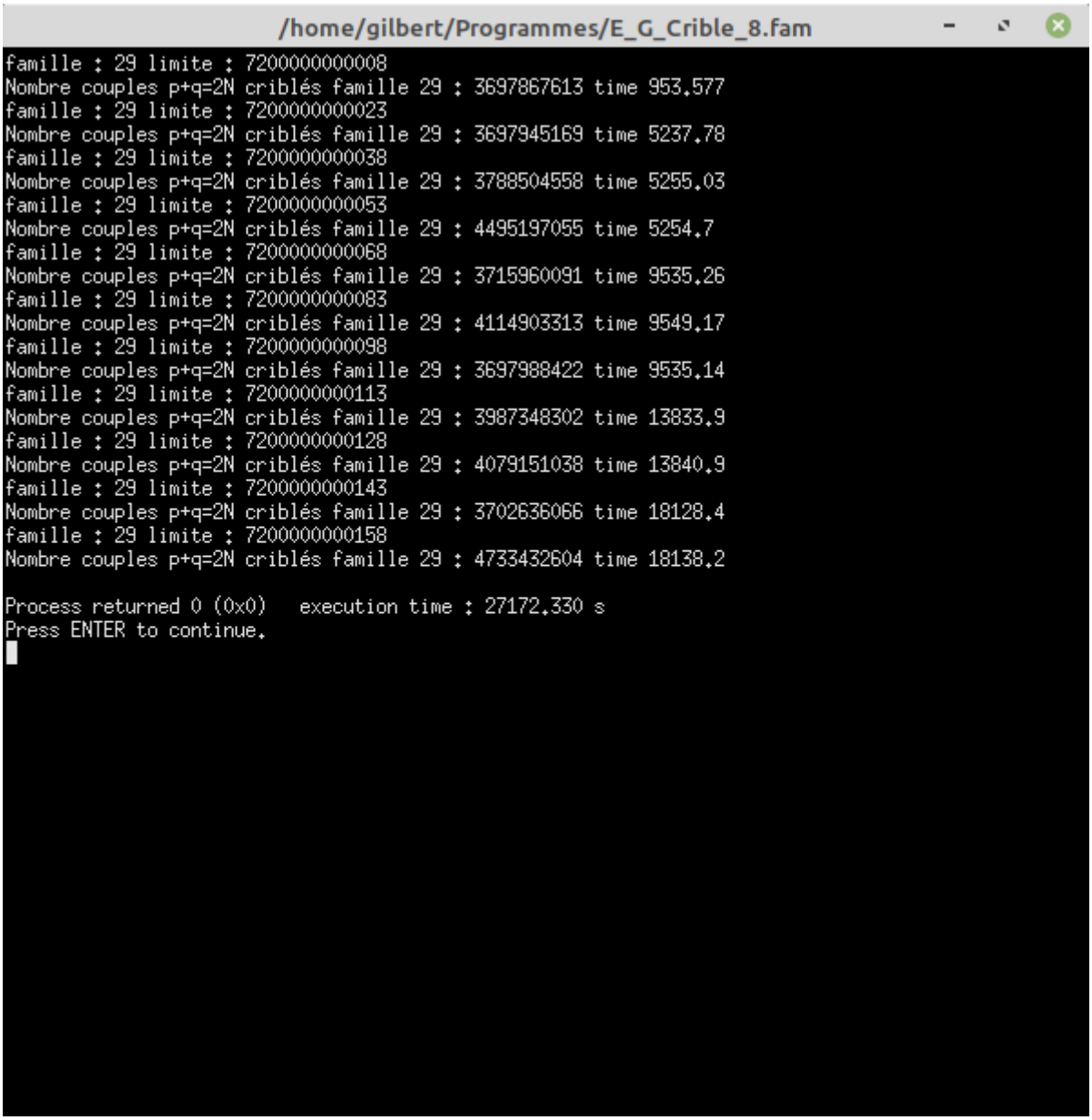
Extrait pour n = 900000000153 et fam 29 : (π(n) – G(n)) * log 15

Vaut ~ (4247753690 – 4030718929) * log15 = 587741028,17


/home/gilbert/Programmes/E.G.crible
Syntaxe /home/gilbert/Programmes/E.G.crible fam. Donnez fam: 29
--> limite : 900000000003
Nombre couple p+q criblés famille 29 : 584406428 time 89,2798
--> limite : 900000000018
Nombre couple p+q criblés famille 29 : 643283642 time 88,3763
--> limite : 900000000033
Nombre couple p+q criblés famille 29 : 549726957 time 88,3754
--> limite : 900000000048
Nombre couple p+q criblés famille 29 : 540248830 time 88,5284
--> limite : 900000000063
Nombre couple p+q criblés famille 29 : 537236156 time 88,3823
--> limite : 900000000078
Nombre couple p+q criblés famille 29 : 567262616 time 88,3803
--> limite : 900000000093
Nombre couple p+q criblés famille 29 : 539369344 time 88,3766
--> limite : 900000000108
Nombre couple p+q criblés famille 29 : 615769321 time 88,3471
--> limite : 900000000123
Nombre couple p+q criblés famille 29 : 666528114 time 88,3781
--> limite : 900000000138
Nombre couple p+q criblés famille 29 : 537755840 time 88,3948
--> limite : 900000000153
Nombre couple p+q criblés famille 29 : 535721272 time 88,3844
--> limite : 900000000168
Nombre couple p+q criblés famille 29 : 542346772 time 88,376
--> limite : 900000000183
Nombre couple p+q criblés famille 29 : 535738382 time 4383,35
--> limite : 900000000198
Nombre couple p+q criblés famille 29 : 595999128 time 4383,34
--> limite : 900000000213
Nombre couple p+q criblés famille 29 : 536572982 time 4383,36
--> limite : 900000000228
Nombre couple p+q criblés famille 29 : 699964073 time 4383,36
--> limite : 900000000243
Nombre couple p+q criblés famille 29 : 555555416 time 4383,35
--> limite : 900000000258
Nombre couple p+q criblés famille 29 : 535840993 time 4383,36
--> limite : 900000000273
Nombre couple p+q criblés famille 29 : 631173048 time 4383,36
--> limite : 900000000288
Nombre couple p+q criblés famille 29 : 535775480 time 4383,34
--> limite : 900000000303
Nombre couple p+q criblés famille 29 : 535720532 time 4383,35
Process returned 0 (0x0) execution time : 3747,565 s
Press ENTER to continue.

N = 15k +8 ; Fam(i) = 17 ; 23 ou 29 ; fami (i) complémentaire 29 ;23 ; ou 17




/home/gilbert/Programmes/E_G_Crible_8.fam
famille : 29 limite : 7200000000008
Nombre couples p+q=2N criblés famille 29 : 3697867613 time 953.577
famille : 29 limite : 7200000000023
Nombre couples p+q=2N criblés famille 29 : 3697945169 time 5237.78
famille : 29 limite : 7200000000038
Nombre couples p+q=2N criblés famille 29 : 3788504558 time 5255.03
famille : 29 limite : 7200000000053
Nombre couples p+q=2N criblés famille 29 : 4495197055 time 5254.7
famille : 29 limite : 7200000000068
Nombre couples p+q=2N criblés famille 29 : 3715960091 time 9535.26
famille : 29 limite : 7200000000083
Nombre couples p+q=2N criblés famille 29 : 4114903313 time 9549.17
famille : 29 limite : 7200000000098
Nombre couples p+q=2N criblés famille 29 : 3697988422 time 9535.14
famille : 29 limite : 7200000000113
Nombre couples p+q=2N criblés famille 29 : 3987348302 time 13833.9
famille : 29 limite : 7200000000128
Nombre couples p+q=2N criblés famille 29 : 4079151038 time 13840.9
famille : 29 limite : 7200000000143
Nombre couples p+q=2N criblés famille 29 : 3702636066 time 18128.4
famille : 29 limite : 7200000000158
Nombre couples p+q=2N criblés famille 29 : 4733432604 time 18138.2
Process returned 0 (0x0) execution time : 27172.330 s
Press ENTER to continue.

On augmente la limite N de 15. On utilise pour cela, les trois fam (i) = 11 ; 17 et 23
nombre de premiers criblée ⇒ nombre de couples p+q = 2 * 4500 000 000 002 à 2 * 4500 000 000 107

/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.
```

Annexe 1) :

Programme des algorithmes {Ératosthène / Goldbarch} : Pour changer les fam(i) et la valeur de début et fin, on modifie les lignes de codes 156 et 157; puis on désactive les familles de 160 à 167 avec les deux //.

```
// -*- 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 << " plus petits que " << 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;
  ulonglong debut = 3000;
  ulonglong fin = 3225;

  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 << "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);
    }
  }
}
```