Essai sur les nombres premiers :

Formule1 :

P(n)=2n+PGCD (2n+1, div_1(2n+1)) ;

*div (12)={1, 2, 3, 4, 6, **12**} ; div_1(12)={2} ; div_2(12)={3} ;*

Formule2.1 :

q=div_1(2n+1) ;

b=q ;

P(n)=2n+ {q >1 =;b=q ; div_1(2n+q) } ;

Formule3 :

si nbdiv(2xi+1)>2 ;

A_i=i-1+ PGCD(2i,div_1(2i)); ou tout simplement A_i=i+1;

si nbdiv(2xi+1)=2 ; n= A_i ; i=n ; avec nbdiv :nombre élément du diviseur

P(n)=2n+1 ;

AV[] = {
[0] = 1, [1] = 6, [2] = 5, [3] = 4, [4] = 3, [5] = 2,
[6] = 1, [7] = 4, [8] = 3, [9] = 2, [10] = 1, [11] = 2,
[12] = 1, [13] = 4, [14] = 3, [15] = 2, [16] = 1, [17] = 2, [18] = 1, [19] = 4, [20] = 3, [21] = 2, [22] = 1, [23] = 6,
[24] = 5, [25] = 4, [26] = 3, [27] = 2, [28] = 1, [29] = 2,
}

Formule 2.2:

$\forall n, x \in N$

P(n)=2n+AV[(2n)Modulo(x)]

X est le module (ici, x=30) ;

Formule 2.3 :

$\forall n, x \in N$

P(n)=2n-1 +AV[(2n-1)Modulo(x)]

X est le module (ici, x=30) ;

Les nombres premiers sont beaucoup utilisés en informatique notamment dans la cryptographie, tels que le chiffrement RSA pour le commerce électronique (confidentialité des échanges de données)

Premier essai :

Formulation du nombre premier.

Prédire les nombres premiers est complexe. Comme l'explique Interesting Engineering, ils ne semblent suivre aucun modèle. Identifier un nombre premier ne permet pas de prédire le suivant. Il faut étudier d'autres nombres entre temps. Pour résoudre ce problème, Georg Frierich Bernhard a cherché à étudier les nombres premiers dans l'autre sens : peut-on prédire les nombres entiers premiers plus petits qu'un nombre entier donné ?

Pour répondre à cette question j'ai fait une remarque pertinente en ce qui concerne les nombres premiers, d'après mes analyses, on se rend compte que les nombres premiers sont tous impaires ; mais comme tous les nombres impaires ne sont pas tous premiers ex : 9, 15, 21 etc j'ai donc découverts une formule non parfaite mais corrigée plus loin dans quatre autres formules et cette formule permet de calculer les nombres premiers parmi les nombres impairs.

Formule1 :

Comme le meilleur moyen de vérifier si un nombre entier est premier ou non est la division successive de ce nombre avec 2…….n-1 ; dont la prise en compte du pgcd(a,b), d'où la formule :

$P(n)=2n+PGCD\ (2n+1,\ div_1(2n+1))$;

A noté que d'après ma remarque, le $pgcd(2n+1,div_1(2n+1))= div_1(2n+1)$*; seulement le concept* div_1 *est nouveau en mathématique dont vous aurez tout le loisir de valider après avoir constater la véracité de son utilisation ici ; et le pgcd permet de vous montrer comment je suis parvenus à obtenir la formule, je pense laisser des pistes pour la correction ou pour son optimisation . Donc vous pouvez toujours remplacer avec plaisir cette expression par*

$P(n)=2n+ div_1(2n+1)$;

div_1*: premier diviseur ; ex:* $div\ (12)=\{1, 2, 3, 4, 6, \mathbf{12}\}$

$div_1(12)=\{2\}$ *car 12 n'est pas un nombre premier et en plus 1 est l'élément neutre de la multiplication et de la division ;*

$div_2(12)=\{3\}$; $div_3(12)=\{4\}$;

Un autre exemple, cette foi-ci relative d'un nombre premier ;

div (5)= {1, 5} ; div_1(5)= {1} et div_2(5)= {5} car nous n'avons que ces deux valeurs ;

En algorithme :

12mod(2)=0 ; div1=2 ; 12mod(3)=0 ; div1=3 ; on a donc
si ((2(i)+1)mod(2)=0) alors
div_1=2
sinon si((2(i)+1)mod(3)=0) alors
div_1=3 sinon
etc...
Sinon
div_1=1 ;

Exemple de calcul :

n=50 ; P(50)=2x50+PGCD (2x50+1,div_1(2x50+1))
P(50)=100+PGCD (101,div_1(101)) ;
101=1x101
div (101)={1, 101} et div_1(101)={1}
P(50)=100+PGCD (101, 1) ; PGCD (101, 1) =1
P(50)=100+1 ;
P(50)=101

Ou encore :
P(50)=2x50+ div_1(2x50+1)
P(50)=100+div_1(101)
P(50)=100+1 ;
P(50)=101

n=6 P(6)=2x6+PGCD (2x6+1,div_1(2x6+1))
P(6)=12+PGCD (13,div_1(13)) ;
13=1x13
div (13)={1, 13} et div_1 (13)={1}
P(6)=12+PGCD (13, 1) ; PGCD (13, 1) =1
P(6)=12+1 ;
P(6)=13

Ou encore :
P(6)=2x6+ div_1(2x6+1)

P(6)=12+div$_1$(13)
P(6)=12+1 ;
P(6)=13

Si le concept div$_1$ est accepté et validé par la communauté scientifique, je pense que tous les nombres premiers de A….Z seront calculés avec les trois méthodes que je vais vous proposer ; ainsi la première formule moins performante que les deux autres est plus consistante que ce que me prédécesseurs ont déjà comme solution et trouve tous les nombres premiers sauf qu'elle mélange le résultat avec un nombre restreint des nombres composés ; les deux autres ont aussi besoin d'un réglage qui me donnerait une obligation de vous faire un détail que je vais toute suite vous expliquer.

Réponse :

Pour répondre à cette question ; peut-on prédire les nombres entiers premiers plus petits qu'un nombre entier donné ?

Exemple:

p=55; on a
n=(p/2)-1 ;
n=(55/2)-1; on a
n=27.5-1
n=26
P(26)=2x26+ div$_1$(2x26+1)
P(26)=52+div$_1$(53)
P(26)=52+1 ;
div$_1$(53)=1 ;
P(26)=53 est premier et p<55
Donc on peut calculer de P(1), P(2), …..,P(26)

Formule 1 : je vais calculer les nombres premiers entre 3 (trois) et cent (100) pour éclairer la lanterne des lecteurs.

Cette méthode mélange un nombre restreint des nombres impairs aux nombres premiers qui serra corrigé dans la deuxième formule ;

$P(n)=2n+pgcd(2n+1,div_1(2n+1))$;

$n=1$; $P(1)=2x1+$ $pgcd\,(2x1+1,\ div_1\,(2x1+1))$

$P(1)=2+\ pgcd\,(3,\ div_1\,(3))$;

$3=1x3$; $div_1(3)=\{1,\ 3\}$ et $dv_1(3)=\{1\}$

$P(1)=2+\ pgcd\,(3,\ 1)$;

$pgcd\,(3,\ 1)=1$

$P(1)=2+1$;

$P(1)=3$;

$P(n)=2n+\ div_1(2n+1)$;

$n=2$ $P(2)=2x2+\ div_1(2x2+1)$

$P(2)=4+\ div_1(5)$;

$5=1x5$;$div\ (5)=\{1,\ 5\}$ et $div_1(5)=\{1\}$

$P(2)=4+1$;

$P(2)=5$

$P(n)=2n+\ div_1(2n+1)$;

$n=3$ $P(3)=2x3+div_1(2x3+1)$

$P(3)=6+\ div_1(7)$;

$7=1x7$

$div\ (7)=\{1,\ 7\}$ et $div_1(7)=\{1\}$

$P(3)=6+1$;

$P(3)=7$

$n=4$

$P(n)=2n+\ div_1(2n+1)$;

$P(4)=2x4+\ div_1(2x4+1)$

$P(4)=8+\ div_1(9)$;

$9=1x3x3$

$div\ (9)=\{1,\ 3,\ 9\}$ et $div_1\ (9)=\{3\}$

$P(4)=9+3$;

$P(4)=11$

$n=5$;

$P(5)=10+div_1(11)$;

$11=1x11$

$div\ (11)=\{1,\ 11\}$ et $div_1(11)=\{1\}$

$P(5)=10+1$;

$P(5)=11$

$P(6)=13$ est premier

$P(7)=17$ est premier

$P(8)=17$ est premier

$P(9)=19$ est premier

$P(10)=23$ est premier

$P(11)=23$ est premier

$P(12)=29$ est premier

$P(13)=29$ est premier

$P(14)=29$ est premier

$P(15)=31$ est premier

P(16)=35 n'est pas premier

P(17)=39 n'est pas premier

P(18)=37 est premier

P(19)=41 est premier

P(20)=41 est premier

P(21)=43 est premier

P(22)=47 est premier

P(23)=47 est premier

P(24)=55 n'est pas premier

P(25)=53 est premier

P(26)=53 est premier

P(27)=59 est premier

P(28)=59 est premier

P(29)=59 est premier

P(30)=61 est premier

P(31)=65 n'est pas premier

P(32)=69 est premier

P(33)=67est premier

P(34)=71 est premier

P(35)=71 est premier

P(36)=73 est premier

P(37)=77 n'est pas premier

P(38)=83 est premier

P(39)=79 est premier

P(40)=83 est premier

P(41)=83 est premier

P(42)=89 est premier

P(43)=89 est premier

P(44)=89 est premier

P(45)=97 est premier

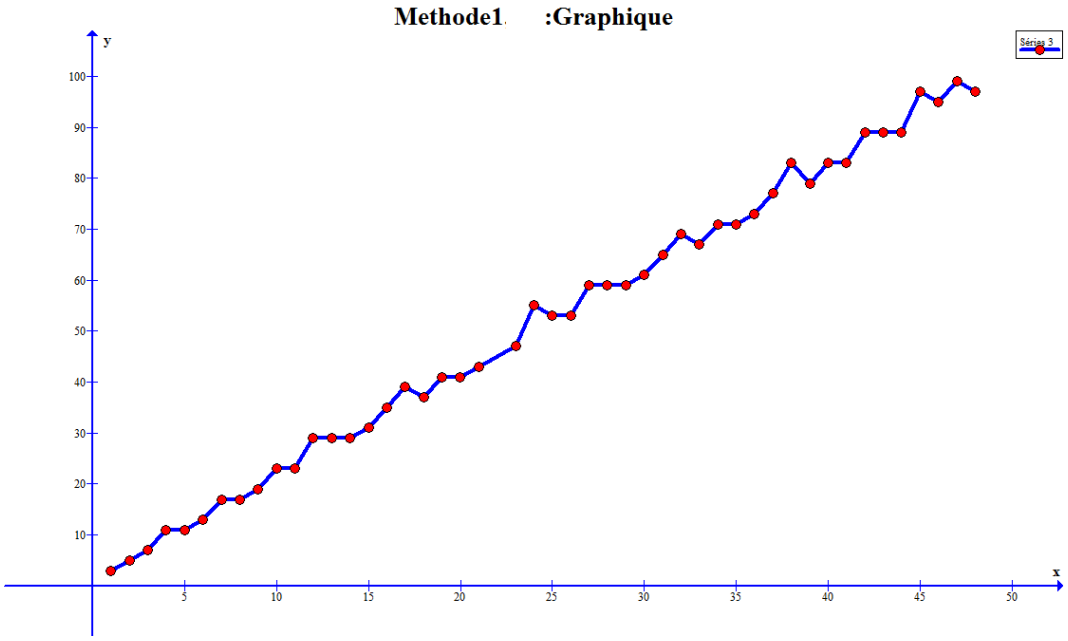
P(46)=95 n'est pas premier

P(47)=99 n'est pas premier

P(48)=97 est premier

P(49)=101 est premier

Représentation graphique Formule1 :



Méthode 2 :

Cette méthode corrige les imperfections de la première méthode en trouvant la correspondance exacte en nombre premier de chaque rang n.

Dans l'explication vous verrez que seul les valeurs de n dans la formule 1 qui n'ont pas de correspondances vrai à un nombre premier qui sont traité et tout le reste serra inchangé ;

Donc cette formule permet d'éliminer toutes les valeurs de la méthode 1 qui ne sont pas premiers.

Les rangs qui n'ont pas de correspondance ici tomberont dans un calcul infini qu'il faut lever dans une autre démanche notamment en algorithme.

Formule2.1 :

$q=div_1(2n+1)$; si $q=1 \Rightarrow b=q$; fin de calcul ; sinon $q=div_1(2n+q)$; $b=q$; jusqu'à $q=1$ et b égale à l'avant dernière valeur de q ;

$P(n)=2n+b$;

Exemple :

n=16 ;q=div$_1$(2x16+1) ; q= div$_1$(33) $\Rightarrow$q=3 !=1

b=3 ;q=div$_1$(2x16+3)=div$_1$(32+3)= div$_1$(35)=5 !=1 ; b=5 ;

q=div$_1$(32+5)= div$_1$(37)=1 ; q=1 fin

P(n)=2n+b;

P(16)=2x16+5; $\Rightarrow$P(16)=37 ;

---

n=17 ; q=div$_1$(2x17+1) ; q= div$_1$(35) $\Rightarrow$q=5 !=1

b=5 ;q=div$_1$(2x17+5)=div$_1$(34+5)= div$_1$(39)=3 !=1 ; b=3 ; q=

div$_1$(34+3)= div$_1$(37)=1 ; q=1 fin

P(n)=2n+b;

P(17)=2x17+3; $\Rightarrow$P(17)=37 ;

---

n=24 ; q=div$_1$(2x24+1) ; q= div$_1$(49) $\Rightarrow$q=7 !=1 ;b=7 ;

q=div$_1$(2x24+7)=div$_1$(48+7)= div$_1$(53)=1 ; b=7 ; fin

P(n)=2n+b;

P(24)=2x24+7; $\Rightarrow$P(24)=53 ;

---

n=31 ; q=div$_1$(2x31+1) ; q= div$_1$(63) $\Rightarrow$q=3 !=1

b=3 ;q=div$_1$(2x31+3)=div$_1$(62+3)= div$_1$(65)=5 !=1 ; b=5 ; q=

div$_1$(62+5)= div$_1$(67)=1 ; q=1 fin

P(n)=2n+b;

P(31)=2x31+5; $\Rightarrow$P(31)=67 ;

---

n=32 ; q=div$_1$(2x32+1) ; q= div$_1$(65) $\Rightarrow$q=5 !=1

b=5 ;q=div$_1$(2x32+5)=div$_1$(64+5)= div$_1$(69)=3 !=1 ; b=3 ;

q= div$_1$(64+3)= div$_1$(67)=1 ; q=1 fin

P(n)=2n+b;

P(32)=2x32+3; ⇒P(32)=67 ;

---

n=46 ; q=div$_1$(2x46+1) ; q= div$_1$(93) ⇒q=3 !=1

b=3 ;q=div$_1$(2x46+3)=div$_1$(92+3)= div$_1$(95)=5 !=1 ; b=5 ;

q= div$_1$(92+5)= div$_1$(97)=1 ; q=1 fin

P(n)=2n+b;

P(46)=2x46+5; ⇒P(46)=97 ;

---

n=47 ; q=div$_1$(2x47+1) ; q= div$_1$(95) ⇒q=5 !=1

b=5 ;q=div$_1$(2x47+5)=div$_1$(94+5)= div$_1$(99)=3 !=1 ; b=3 ;

q= div$_1$(94+3)= div$_1$(97)=1 ; q=1 fin

P(n)=2n+b;

P(47)=2x47+3; ⇒P(47)=97 ;

---

n=37 ; q=div$_1$(2x37+1) ; q= div$_1$(75) ⇒q=3 !=1

b=3 ;q=div$_1$(2x37+3)=div$_1$(74+3)= div$_1$(77)=7 !=1 ; b=7 ; q= div$_1$(74+7)= div$_1$(81)=3 ; q !=1 fin ; calcule infini

Donc p(37)=0 ;

Cette valeur entre dans un calcul infini dont l'ouverture d'une perspective pour l'étude de ces trois nombres 37, 75, 77 et 83 ;

à noté que la valeur recherchée qui est 79 correspond déjà avec le rang 39 càd P(39)=79 ; qui ne pose donc pas de problème.

La complexité du nombre premier réside du fait qu'un seul nombre premier peut correspondre à plusieurs rangs n et pour remédier à ces situations, un programme informatique peut bien sortir dans ce calcul infini, de sauter les doublons et de les rangé dans l'ordre croissant les valeurs de p;

En plus pour trouver une formule globale, je veux faire une suggestion aux lecteurs tout en laissant le soin à ceux qui ont des propositions pour un symbole plus commode à ce que je veux mettre tout de suit à disposition ;

En algorithme, dire:

Pour i de 1 à n faire

Traitement i, f(x), etc

Finpour

a comme équivalent en mathématique

$$\sum_{i=1}^{n} f(x)$$ n est traité dans f

ou pour ne pas avoir d'ambigüité

$$\sum_{i=1}^{n} nf(x)$$

Mais je ne vois pas d'équivalence pour

tantque (t !=1) faire

traitement t, f(x), etc..

fintantque

Celui qui a donc une meilleure idée ça serra la bienvenue car avec cette méthode de résolution sur les nombres premiers, on a un présent besoin d'un symbole pour matérialiser la formule générale; mais en attendant j'en propose un :

Formule2.1

$q=div_1(2n+1)$

$b=q$; $\{q>1 =$;$b=q$;$div_1(2n+q)$

Tantque q >1 faire

$b=q$;

$q=div_1(2n+q)$;

finTantque

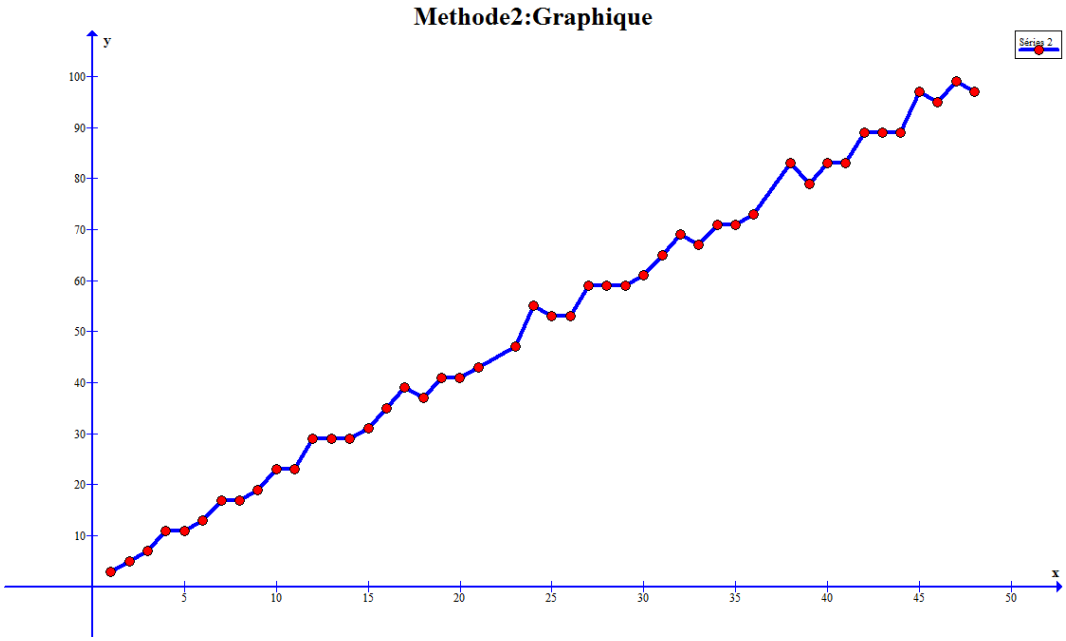
D'où la formule générale:

$q=div_1(2n+1)$;

$b=q$;

$P(n)=2n+ \{q>1 =$;$b=q$; $div_1(2n+q)\}$

Représentation graphique Formule 2.1 :



Le code java:

On entre le rang n pour calculer p(n)

```
package Premier;
import java.util.*;

public class Premier {


        public static void main(String[] args) {
                Premier P1 = new Premier();
                @SuppressWarnings("resource")
                Scanner keby = new Scanner(System.in);

                int s=0;
            System.out.println("Veuillez saisir un rang n :");
            s=keby.nextInt();
            System.out.printf("P : %d \n", +P1.Formule2(s));

        }

         public int Div1 (int n) {

        int i =2;

        int d=n-1 ;

        int div=1;
```

```
        while ((i<d) && ((n) % (i) != 0)) {
                i =i+1 ;
                    }

        if ((n) % (i) ==0) {
                  div = i  ;

                  }

        return div;
}

        public int Formule2 (int n){

        int i=0;
        int r=0;
        int control = 2 * n + 1 ;
        int Q = Div1(2 * n + 1);
        int b = Q;
        int sortie = 0;
        ArrayList<Integer> K= new ArrayList<Integer> ();
        K.add(control);

        while ((Q!=1) && (sortie == 0)) {

            b = Q ;
            control=2 * n + Q;
            Q = Div1(2 * n + Q);
            r=K.size();
```

```
            while (i<r)  {

                    if (control== K.get(i) ) {

                                sortie = sortie + 1;

                }

                    i=i+1;
                }


            K.add(2 * n + Q);

        }


if ((Q!=1) && (sortie == 1) ){

                return 0;
}
                        else {

return (2 * n + b);

                        }
```

```
        }

}
```

Code Java (les nombres premiers entre deux bornes)

Exe:

on entre les rang n1=1 et n2=100 pour calculer p(1), p(2), ,p(100) ;

```
package Premier;
import java.util.*;

public class Premier {


public static void main(String[] args) {
Premier P1 = new Premier();
@SuppressWarnings("resource")
Scanner keby = new Scanner(System.in);

System.out.println("Veuillez saisir le rang inferieur n1 et superieur n2
:");
long s=keby.nextLong();
long m=keby.nextLong();

for (long x = s ; x <= m; x++) {

System.out.printf("P%d  : %d \n",+x, +P1.Formule2(x));

}
```

```
}
public long Div1 (long n) {

long i =3;

long d=n-1 ;

long div=1;

if ((n) % (2)== 0) {
        div=2;

} else{

while ((i<=d) && ((n) % (i) != 0)) {

i =i+2 ;

}

if ((i<=d) &&(n) % (i) ==0) {
div = i ;

}

}

return div;
}
```

```
public long Formule2 (long n){

int i=0;
long r=0;
long control = 2 * n + 1 ;
long Q = Div1(2 * n + 1);
long b = Q;
long sortie = 0;
ArrayList<Long> K= new ArrayList<Long> ();
K.add(control);
while ((Q!=1) && (sortie == 0)) {

b = Q ;
control=2 * n + Q;
Q = Div1(2 * n + Q);
r=K.size();
while (i<r) {

if (control== K.get(i) ) {

sortie = sortie + 1;

}

i=i+1;
}
```

```
K.add(2 * n + Q);

}


if ((Q!=1) && (sortie == 1) ){

return 0;
}
else {

return (2 * n + b);

}

}

}
```

Deuxième version plus rapide :

Ici on cherche Div1 entre 3 et racine carré de (n)

```
package Premier;
import java.util.*;

public class Premier {


public static void main(String[] args) {
```

```
Premier P1 = new Premier();
@SuppressWarnings("resource")
Scanner keby = new Scanner(System.in);

System.out.println("Veuillez saisir le rang inferieur n1 et superieur n2
:");
long s=keby.nextLong();
long m=keby.nextLong();

for (long x = s ; x <= m; x++) {

System.out.printf("P%d  : %d \n",+x, +P1.Formule2(x));

}

}
public long Div1 (long n) {

long i =3;

/*long d=n-1 ;*/
double d=Math. sqrt(n) ;

long div=1;

if ((n) % (2)== 0) {
        div=2;

} else{
```

```
while ((i<=d) && ((n) % (i) != 0)) {

i =i+2 ;

}

if ((i<=d) &&(n) % (i) ==0) {
div = i ;

}

}

return div;
}



public long Formule2 (long n){

int i=0;
long r=0;
long control = 2 * n + 1 ;
long Q = Div1(2 * n + 1);
long b = Q;
long sortie = 0;
ArrayList<Long> K= new ArrayList<Long> ();
K.add(control);
```

```
while ((Q!=1) && (sortie == 0)) {

b = Q ;
control=2 * n + Q;
Q = Div1(2 * n + Q);
r=K.size();
while (i<r) {

if (control== K.get(i) ) {

sortie = sortie + 1;

}

i=i+1;
}


K.add(2 * n + Q);

}


if ((Q!=1) && (sortie == 1) ){

return 0;
}
else {
```

```
return (2 * n + b);

}

}

}
}
```

Programme VBA obtenu sur forum d'entraide :

```
Function Div1 (n As Long) As Long
'-------------------------------------
-------------------------------------
Dim i As Long
If TestMillerRabin(n) = True Then
     Div1 = 1
   ElseIf (2*n + 1) Mod (2) = 0 Then
     Div1 = 2
    Else
     For i = 3 To 2*n + 1 Step 2
        If (2*n + 1) Mod (i) = 0 Then
            Div1 = i
            Exit For
        End If
     Next i
  End If
End If

End Function
```

```
'-------------------------------------
'-------------------------------------
Function Main(n As Long)
'-------------------------------------
'-------------------------------------
Dim q As Long
Dim b As Long
Dim i As Long

i = n / 2 - 1 pour calculer un nbre
premier  P < n on calcul avant tout
son rang i.

q = Div1(2*I + 1)
b=q
z1= 2*I + 1
z2=0

If q > 1 Then
    Do
                b = q
                q = Div1(2*I + q)
          If ((z1=(2*n + q)) then
                Z2=z2 + 1;
           End If

     Loop While ((q > 1) and (z2 !=
1))

End If
```

```
    If    z2 = 1 Then
          P = 0
       Else
          p = 2 * i + b
       End If
  Main = p  (s'il faut calculer p1,
p2, ...., pi-1, pi  on les range dans
un tableau dans ce cas).
End Function

'-------------------------------------
-------------------------------------
-------------
Function TestMillerRabin(ByVal n As
Variant) As Boolean
'-------------------------------------
-------------------------------------
-------------
' Test MILLER RABIN. Retourne Vrai si
n est un nombre premier
'-------------------------------------
-------------------------------------
-------------
Dim A As Long, BaseMaxi As Long
Dim Base

' Optimisation nb base à tester:
Base = Array(2, 3, 5, 7, 11, 13, 17,
19, 23)
```

```
BaseMaxi = Int(4 + Log(n) / 8 - Abs(4
- Log(n) / 8))

If IsMultiple(n, 2) = True Then Exit
Function

' Algo:
A = 0
Do
    If IsMultiple(n, Base(A)) = False
Then
        If MillerRabin(n, Base(A)) =
False Then Exit Function
    End If
A = A + 1
Loop Until A > BaseMaxi

TestMillerRabin = True
End Function

'-------------------------------------
-------------------------------------
-------------
Function MillerRabin(ByVal n As
Variant, ByVal A As Variant) As
Boolean
'-------------------------------------
-------------------------------------
-------------
```

```
' RQ : IMPOSSIBILITE DE REPONDRE SI n multiple de a
'-------------------------------------------------------------------------
' Tests triviaux:
If MOD2(n, 2) = 0 Or n < 3 Then Exit Function

' Paramètres:
Dim h As Long, d As Variant, i As Long

' Convertit les variants en Decimal:
n = CDec(n): A = CDec(A): d = CDec(d)

' Calcul de n-1 = 2^h * d avec d impair:
Do
    h = h + 1
    d = (n - 1) / 2 ^ h
Loop Until d = Int(d) And MOD2(d, 2) = 1

' Test a^d=1 mod n:
MillerRabin = (ExpoMod(A, d, n) = 1)
If MillerRabin Then Exit Function

' Test s'il existe un 0<= i <= h-1 tel que a^(h*2^i)=-1 mod n:
```

```
For i = 0 To h - 1
    MillerRabin = (ExpoMod(A, d * 2 ^
i, n) = n - 1)
    If MillerRabin Then Exit For
Next i

End Function

'-----------------------------------
-------------------------------
Function ExpoMod(ByVal Nb As Variant,
ByVal Expo As Variant, _
                          ByVal Modulo
As Variant) As Variant
'-----------------------------------
-------------------------------
' EXPONENTIATION MODULAIRE RAPIDE :
Nb^Expo MOD Modulo.
'-----------------------------------
-------------------------------
' Convertit les variants en Decimal:
Nb = CDec(Nb): Expo = CDec(Expo):
Modulo = CDec(Modulo)

' Traitement:
ExpoMod = 1
Do
    If MOD2(Expo, 2) = 1 Then
        ExpoMod = MODProd(Nb,
ExpoMod, Modulo)
```

```
            Expo = (Expo - 1) / 2
            Nb = MODProd(Nb, Nb, Modulo)
        End If

If MOD2(Expo, 2) = 0 Then
            ExpoMod = MODProd(ExpoMod, 1,
Modulo)
            Expo = Expo / 2
            Nb = MODProd(Nb, Nb, Modulo)
        End If

Loop Until Expo = 0
End Function

'-------------------------------------
-------------------------------
Function MOD2(ByVal d As Variant,
ByVal n As Variant) As Variant
'-------------------------------------
-------------------------------
' Renvoie le modulo de d et n:
'-------------------------------------
-------------------------------
d = CDec(d): n = CDec(n)
MOD2 = CDec(d - n * Int(d / n))
End Function

'-------------------------------------
-------------------------------
```

```
Function IsMultiple(ByVal Nb1 As
Variant, ByVal Nb2 As Variant) As
Boolean
'-------------------------------
-----------------------------
' Teste si Nb1 est multiple de Nb2.
'-------------------------------
-----------------------------
Nb1 = CDec(Nb1): Nb2 = CDec(Nb2)
If Nb2 = 0 Then IsMultiple = True:
Exit Function
IsMultiple = ((Int(Nb1 / Nb2) = Nb1 /
Nb2) And Nb1 <> 0)
End Function

'-------------------------------
-----------------------------
Function MODProd(ByVal Nb1 As
Variant, ByVal Nb2 As Variant, _
                              ByVal Modulo
As Variant) As Variant
'-------------------------------
-----------------------------
' Renvoie le modulo du produit
"nb1*nb2 MOD Modulo" sans la limite
Double.
'-------------------------------
-----------------------------
' Convertit les variants en Decimal
et teste la grandeur du produit:
```

```
Nb1 = CDec(Nb1): Nb2 = CDec(Nb2):
Modulo = CDec(Modulo)
If Nb1 * Nb2 < 10 ^ 15 Then MODProd =
MOD2(Nb1 * Nb2, Modulo): Exit
Function

' Paramètre les variables en Decimal:
Dim r As Variant, C As Variant,
Facteur As Variant, d As Variant
r = CDec(r): C = CDec(C): Facteur =
CDec(Facteur): d = CDec(d)

' Prend le mini => plus rapide:
If Nb1 < Nb2 Then r = Nb2: Nb2 = Nb1:
Nb1 = r

' Optimisation facteur:
Facteur = 9
r = MOD2(Nb1, Modulo)
Do
    If IsMultiple(Nb2, Facteur) Then
        r = MOD2(Facteur * r, Modulo)
        Nb2 = Nb2 / Facteur
    Else
        d = MOD2(Nb2, Facteur)
        C = MOD2(C + r * d, Modulo)
        Nb2 = Nb2 - d
    End If
Loop Until Nb2 = 0
MODProd = MOD2(C, Modulo)
```

```
End Function

'-------------------------------------
-------------------------------------
--------------
Function InversMod(ByVal Nba As
Variant, ByVal Nbb As Variant) As
Variant
'-------------------------------------
-------------------------------------
--------------
Dim A, b, Q, t, x, y

' Convertit les variants en Decimal:
A = CDec(Nbb): b = CDec(Nba): Q =
CDec(Q): t = CDec(t): x = CDec(x): y
= CDec(y)

x = 1: y = 0
While (b <> 0)
    t = b
    Q = Int(A / t)
    b = A - Q * t
    A = t
    t = x
    x = y - Q * t
    y = t
Wend
InversMod = CDec(y)
```

```
If y < 0 Then InversMod = CDec(y + Nbb)
End Function
```

Formule 2.2 :

Des discussions sur des forums d'entraide m'ont permit d'avoir la connaissance de l'existence d'une table de calcule permettant de sauter sur des nombres composés et l'un de ce tableau ci-dessous ont été une bonne nouvelle pour l'optimisation de la formule 2 donnant naissance à deux autres formules, Formule 2.2 et Formule 2.3 qui doivent être tenus compte en combinant son calcul avec la formule 2.1:

$AV[] = \{$
$[0]=1,\ [1]=6,\ [2]=5,\ [3]=4,\ [4]=3,\ [5]=2,$
$[6]=1,\ [7]=4,\ [8]=3,\ [9]=2,\ [10]=1,\ [11]=2,$
$[12]=1,\ [13]=4,\ [14]=3,\ [15]=2,\ [16]=1,\ [17]=2,$
$[18]=1,\ [19]=4,\ [20]=3,\ [21]=2,\ [22]=1,\ [23]=6,$
$[24]=5,\ [25]=4,\ [26]=3,\ [27]=2,\ [28]=1,\ [29]=2,$
$\}$

Formule 2.2 :

$\forall\ n,\ x \in N$

P(n)=2n+AV[(2n)Modulo(x)] div1(p(n))=1

X est le module (ici, x=30) ;

Ce tableau fonctionne avec $2n \geq 5$ *et* $n \geq 3$ *ce qui veut dire que p(1),p(2) sont tous égaux à p(3)=7 donc inutile de les calculés ; L'utilisation en générale de ces tableaux nécessite de beaucoup de maitrise des nombres premiers entre 1 et la base x, ici x=30 ; car on fait la différence avec chaque indice i trouvé par le*

calcule càd (2n)Mod30 et le nombres premiers suivant qui constitue ici le saut consistant pour un seconde nombre premier ; Exemple :

Si (2n)Mod30 avec n=3 on a : AV[(6)Mod(30)]= V[6] $\Rightarrow$ *i=6 La différence entre le nombre premier suivant (7) et 6 est 1 donc V[6]=1 ; c'est ce qui est placé dans ce fameux tableau au cas où on ne comprend pas cette règle, on travaillera avec tout simplement pour copier et coller la correspondance i ;*

A noté que le p(n) $>$ *x n'est pas tout à fait sûr pour être premier ce qui nous amène à chaque fois de faire le test de* $div_1(p)$ *et si* $div_1(p)=1$ *c'est la plus part des cas on est donc assuré dans le cas contraire on se tourne vers la formule 2.1 pour trouver le p ; Ensuite cette formule reste valable même si le tableau et x changent de valeurs ;*

Calculons les cent(100) premiers nombres premiers pour la représentée graphiquement :

P(n)=2n+AV[(2n)Modulo(x)]

n=3 ;

*p(3)=2(3)+AV[(2*3)Mod(30)]*

p(3)=6+AV[(6)Mod(30)]

p(3)=6+AV[6]

AV[6]=1; dans le tableau

p(3)=6+1

p(3)=7 ; $div_1(p(3))=1$

++++++++++++++++++

n= 4;

*p(4)=2(4)+AV[(2*4)Mod(30)]*

p(4)=8+AV[(8)Mod(30)]

p(4)=8+AV[8]

AV[8]=3; dans le tableau

p(4)=8+3

p(4)=11; $div_1(p(4))=1$

++++++++++++++++++

n= 5;

p(5)=2(5)+AV[(2*5)Mod(30)]

p(5)=10+AV[(10)Mod(30)]

p(5)=10+AV[10]

AV[10]=1; dans le tableau

p(5)=10+1

p(5)=11; $div_1(p(5))=1$

++++++++++++++++++++++

n= 6;

p(6)=2(4)+AV[(2*6)Mod(30)]

p(6)=12+AV[(12)Mod(30)]

p(6)=12+AV[12]

AV[12]=1; dans le tableau

p(6)=12+1

p(6)=13; $div_1(p(6))=1$

++++++++++++++++++++++

n= 7;

p(7)=2(7)+AV[(2*7)Mod(30)]

p(7)=14+AV[(14)Mod(30)]

p(7)=14+AV[14]

AV[14]=3; dans le tableau

p(7)=14+3

p(7)=17; $div_1(p(7))=1$

++++++++++++++++++++++

n= 8;

p(8)=2(8)+AV[(2*8)Mod(30)]

p(8)=16+AV[(16)Mod(30)]

p(8)=16+AV[16]

AV[12]=1; dans le tableau

p(8)=16+1

p(8)=17; $div_1(p(8))=1$

++++++++++++++++++++

n= 9;

p(9)=2(9)+AV[(2*9)Mod(30)]

p(9)=18+AV[(18)Mod(30)]

p(9)=18+AV[18]

AV[18]=1; dans le tableau

p(9)=18+1

p(9)=19; $div_1(p(9))=1$

++++++++++++++++++++++

n= 10;

p(10)=2(10)+AV[(2*10)Mod(30)]

p(10)=20+AV[(20)Mod(30)]

p(10)=20+AV[20]

AV[20]=3; dans le tableau

p(10)=20+3

p(10)=23; div_1(p(10))=1

++++++++++++++++++++++

n= 11;

*p(11)=2(11)+AV[(2*11)Mod (30)]*

p(11)=22+AV[(22)Mod(30)]

p(11)=22+AV[22]

AV[22]=1; dans le tableau

p(11)=22+1

p(11)=23 ; div_1(p(11))=1

++++++++++++++++++++++

n= 12;

*p(12)=2(12)+AV[(2*12)Mod (30)]*

p(12)=24+AV[(24)Mod(30)]

p(12)=24+AV[24]

AV[24]=5; dans le tableau

p(12)=24+5

p(12)=29 ; div_1(p(12))=1

++++++++++++++++++++++

n= 13;

*p(13)=2(13)+AV[(2*13)Mod (30)]*

p(13)=26+AV[(26)Mod(30)]

p(13)=26+AV[26]

AV[26]=3; dans le tableau

p(13)=26+3

p(13)=29; div_1(p(13))=1

++++++++++++++++++++++

n= 14;

*p(14)=2(14)+AV[(2*14)Mod (30)]*

p(14)=28+AV[(28)Mod(30)]

p(14)=28+AV[28]

AV[28]=1; dans le tableau

p(14)=28+1

p(14)=29; div_1(p(14))=1

++++++++++++++++++++++

n= 15;

*p(15)=2(10)+AV[(2*15)Mod (30)]*

p(15)=30+AV[(30)Mod(30)]

p(15)=30+AV[0]

AV[0]=1; dans le tableau

p(15)=30+1

p(15)=31 ; div_1(p(3))=1

++++++++++++++++++++++

P16 = 37

++++++++++++++++++++++

P17 = 37

++++++++++++++++++++++

P18 = 37

++++++++++++++++++++++

P19 = 41

++++++++++++++++++++++

P20 = 41

++++++++++++++++++++++

P21 = 43

++++++++++++++++++++++

P22 = 47

++++++++++++++++++++++

P23 = 47

++++++++++++++++++++++

n= 24;

p(24)=2(24)+AV[(2*24)Mod(30)]

p(24)=48+AV[(48)Mod(30)]

p(24)=48+AV[18]

AV[18]=1; dans le tableau

p(24)=48+1

p(24)=49 ; $div_1(p(24))=7$

n'est pas premier

P25 = 53

++++++++++++++++++++++

P26 = 53

++++++++++++++++++++++

P27 = 59

++++++++++++++++++++++

P28 = 59

++++++++++++++++++++++

P29 = 59

++++++++++++++++++++++

P30 = 61

++++++++++++++++++++++

P31 = 67

++++++++++++++++++++++

P32 = 67

++++++++++++++++++++++

P33 = 67

++++++++++++++++++++++

P34 = 71

++++++++++++++++++++++

P35 = 71

++++++++++++++++++++++

P36 = 73

++++++++++++++++++++++

n= 37;

p(37)=2(37)+AV[(2*37)Mod(30)]

p(37)=74+AV[(74)Mod(30)]

p(37)=74+AV[14]

AV[14]=3; dans le tableau

p(37)=74+3

p(37)=77 ; div_1(p(37))=7 non premier

++++++++++++++++++++++

n= 38;

*p(38)=2(38)+AV[(2*38)Mod (30)]*

p(38)=76+AV[(76)Mod(30)]

p(38)=76+AV[16]

AV[16]=1; dans le tableau

p(38)=76+1

p(38)=77 ; div_1(p(38))=7 non premier

P39 = 79

++++++++++++++++++++++

P40 = 83

++++++++++++++++++++++

P41 = 83

++++++++++++++++++++++

P42 = 89

++++++++++++++++++++++

P43 = 89

++++++++++++++++++++++

P44 : 89

++++++++++++++++++++++

n= 45;

*p(45)=2(45)+AV[(2*45)Mod (30)]*

p(45)=90+AV[(90)Mod(30)]

p(45)=90+AV[0]

AV[0]=1; dans le tableau

p(45)=90+1

p(45)=91 ; div_1(p(45))=7 non premier

P46 = 97

++++++++++++++++++++++

P47 = 97

++++++++++++++++++++++

P48 = 97

++++++++++++++++++++++

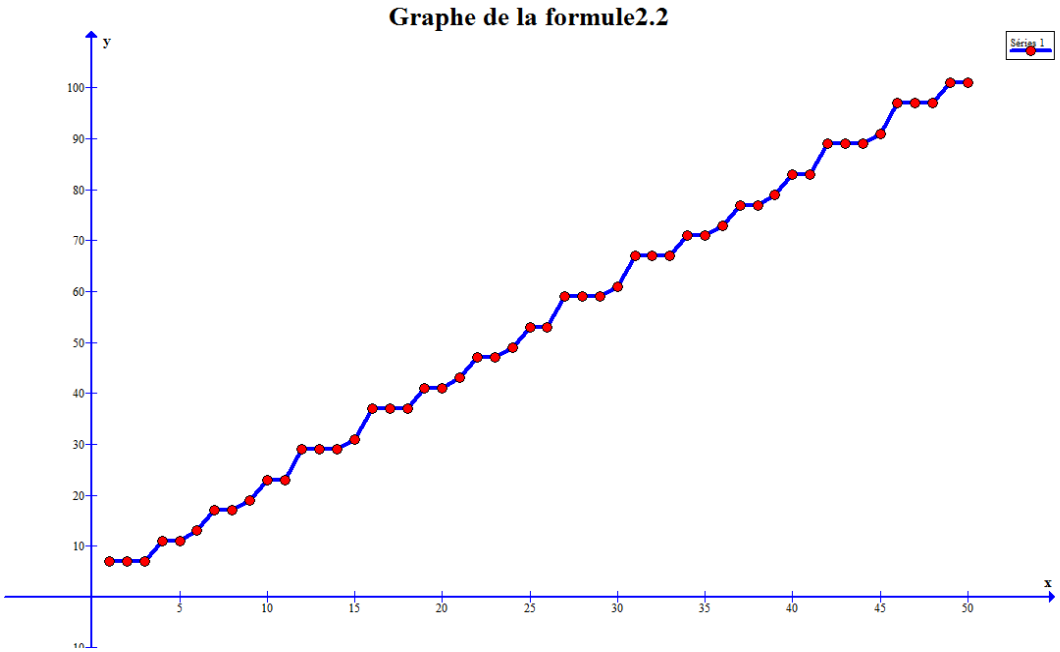
P49 = 101

++++++++++++++++++++++

P50 = 101

++++++++++++++++++++++

Représentation graphique Formule 2.2 :



Le code java en fonction de Int:

```
package PremierFormule3;

import java.util.Scanner;



public class PremierFormule3 {


public static void main(String[] args) {
        PremierFormule3  P1 = new PremierFormule3 ();
Scanner keby = new Scanner(System.in);

System.out.println("Veuillez saisir le rang inferieur n1 et
superieur n2 :");
```

```
int s=keby.nextInt();
int m=keby.nextInt();
for (int x = s ; x <= m; x++) {

System.out.printf("P%d  : %d \n",+x, +P1.Formule3(x));
}
keby.close();
}
public int Formule3 (int n){
        //long p=0;
        int AV[] = {
                    1,   6,   5,   4,   3,   2,
                    1,   4,   3,   2,   1,  2,
                    1,  4,  3,  2, 1, 2,
                    1,  4, 3, 2,  1,  6,
                    5,  4,  3,  2,  1, 2
                };


        return (2*n+ AV[(2*n)%30]);

}
}
```

Code java avec combinaison avec la formule2.1

```
package PremierFormule3;

import java.util.ArrayList;
```

```
import java.util.Scanner;



public class PremierFormule3 {


public static void main(String[] args) {
      PremierFormule3  P1 = new PremierFormule3 ();
Scanner keby = new Scanner(System.in);

System.out.println("Veuillez saisir le rang inferieur n1
et superieur n2 :");
int s=keby.nextInt();
int m=keby.nextInt();
for (int x = s ; x <= m; x++) {

System.out.printf("P%d  : %d \n",+x,
+P1.Formule3(x));

}
keby.close();
}
public int Formule3 (int n){
      //long p=0;

      int AV[] = {
                1,   6,   5,   4,   3,   2,
                 1,   4,   3,   2,   1,  2,
                 1,  4,  3,  2, 1, 2,
                 1,  4, 3, 2,  1,  6,
                 5,  4,  3,  2,  1, 2
            };

      if (n<=2){
            return Formule2 (n);
```

```
        }else {

        return (2*n+ AV[(2*n)%30]);

        }
}
public int Div1 (int n) {

        int i =3;

        /*long d=n-1 ;*/
        double d=Math. sqrt(n) ;

        int div=1;

        if ((n) % (2)== 0) {
              div=2;

        } else{

        while ((i<=d) && ((n) % (i) != 0)) {

        i =i+2 ;

        }

        if ((i<=d) &&(n) % (i) ==0) {
        div = i ;

        }

        }

        return div;
        }
```

```
public int Formule2 (int n){

        int p=0;
        loop1:
        for(int j=n;j<=2*n;j++) {
        n=j;
        int i=0;
        int r=0;
        int control = 2 * n + 1 ;
        int Q = Div1(2 * n + 1);
        int b = Q;
        int sortie = 0;
        ArrayList<Integer> K= new ArrayList<Integer> ();
        K.add(control);



        while ((Q!=1) && (sortie == 0)) {

        b = Q ;
        control=2 * n + Q;
        Q = Div1(2 * n + Q);
        r=K.size();
        while (i<r) {

        if (control== K.get(i) ) {

        sortie = sortie + 1;

        }

        i=i+1;
        }


        K.add(2 * n + Q);
```

```
        }

        if ((Q==1) && (sortie == 0) ){
        p=2 * n + b;  break loop1;


        }
            }
        return p;
        }
}
```

Amélioration 2.2 :

*P(n)=2n+AV[(2*n)Mod30]+2[div$_1$(2*n+ AV[(2*n)Mod30])-1] /div$_1$(p)=1*

Cette amélioration permet de trouver correspondance première à chacune de ces valeurs :

p(24)=49 ; div$_1$(p(24))=7 n'est pas premier ;

p(37)=77 ; div$_1$(p(37))=7 non premier ;

p(38)=77 ; div$_1$(p(38))=7 non premier ;

p(45)=91 ; div$_1$(p(45))=7 non premier ;

Ainsi nous aurons :

P24 =: 61; div$_1$(p(24))=1 est premier

P37 = 89 $div_1(p(37))=1$ est premier

P38 = 89; $div_1(p(38))=1$ est premier

P45 = 103;$div_1(p(45))=1$ est premier

Code java avec combinaison avec la formule2.1

```
package formulecorriger1;

import java.util.ArrayList;
import java.util.Scanner;


public class formulecorriger1 {
	public static void main(String[] args) {
		formulecorriger1  P1 = new formulecorriger1 ();
	Scanner keby = new Scanner(System.in);

	System.out.println("Veuillez saisir le rang inferieur n1 et superieur n2 :");
	int s=keby.nextInt();
	int m=keby.nextInt();
	for (int x = s ; x <= m; x++) {

	System.out.printf("P%d  : %d \n",+x, +P1.Formule3(x));

	}
	keby.close();
	}
	public int Formule3 (int n){
```

```
//long p=0;
int i;
int j;
int form;
int AV[] = {
          1,   6,   5,   4,   3,   2,
           1,   4,   3,   2,   1,  2,
           1,  4,  3,  2, 1, 2,
           1,  4, 3, 2,  1,  6,
           5,  4,  3,  2,  1, 2
        };
i=AV[(2*n)%30];
j=Div1(2*n+i);
form=2*n+i+2*(j-1) ;

if (n<=2){

      return Formule2 (n);

}else if (Div1(form)==1){

      return form;

}else{

     return Formule2 (n);
```

```
    }

}

public int Div1 (int n) {

        int i =3;

        /*long d=n-1 ;*/
        double d=Math. sqrt(n) ;

        int div=1;

        if ((n) % (2)== 0) {
                div=2;

        } else{

        while ((i<=d) && ((n) % (i) != 0)) {

        i =i+2 ;

        }

        if ((i<=d) &&(n) % (i) ==0) {
```

```
        div = i ;

        }

        }

        return div;
        }
public int Formule2 (int n){

        int p=0;
        loop1:
        for(int j=n;j<=2*n;j++) {
        n=j;
        int i=0;
        int r=0;
        int control = 2 * n + 1 ;
        int Q = Div1(2 * n + 1);
        int b = Q;
        int sortie = 0;
        ArrayList<Integer> K= new ArrayList<Integer> ();
        K.add(control);



        while ((Q!=1) && (sortie == 0)) {
```

```
b = Q ;
control=2 * n + Q;
Q = Div1(2 * n + Q);
r=K.size();
while (i<r) {

if (control== K.get(i) ) {

sortie = sortie + 1;

}

i=i+1;
}


K.add(2 * n + Q);

}


if ((Q==1) && (sortie == 0) ){

p=2 * n + b;  break loop1;
```

```
        }

            }

        return p;

        }
}
```

Formule 2.3 :

```
AV[] = {
[0] = 1,  [1] = 6,  [2] = 5,  [3] = 4,  [4] = 3,  [5] = 2,
[6] = 1,  [7] = 4,  [8] = 3,  [9] = 2,  [10] = 1, [11] = 2,
[12] = 1, [13] = 4, [14] = 3, [15] = 2, [16] = 1, [17] = 2,
[18] = 1, [19] = 4, [20] = 3, [21] = 2, [22] = 1, [23] = 6,
[24] = 5, [25] = 4, [26] = 3, [27] = 2, [28] = 1, [29] = 2,
```

$\forall$ *n, x* $\in$ *N*

P(n)=2n-1 +AV[(2n-1)Modulo(x)] /div1(p(n))=1

X est le module (ici, x=30) ;

Ce tableau fonctionne également avec 2n-1 $\geq$ *5 et n* $\geq$ *3 ce qui veut dire que p(1),p(2) sont tous égaux à p(3)=7 donc inutile de les calculés ;*

L'utilisation en générale de ces tableaux nécessite de beaucoup de maitrise des nombres premiers entre 1 et la base x, ici x=30 ; car on fait la différence avec chaque indice i trouvé càd

(2n)Mod30 et le nombres premiers suivant qui constitue ici le saut consistant pour un seconde nombre premier ;
Exemple :

Si (2n)Mod30 avec n=3 on a : AV[(6)Mod(30)]= V[6] $\Rightarrow$ *i=6*
La différence entre le nombre premier suivant (7) et 6 est 1 donc V[6]=1 ; c'est ce qui est placé dans ce fameux tableau au cas où on ne comprend pas cette règle, on travaillera avec tout simplement pour copier et coller la correspondance i ;

A noté que le p(n) $>$ *x n'est pas tout à fait sûr pour être premier ce qui nous amène à chaque fois de faire le test de* $div_1(p)$ *et si* $div_1(p)=1$ *c'est la plus part des cas on est donc assuré dans le cas contraire on se tourne vers la formule 2.1 pour trouver le p. Ensuite cette formule reste valable même si le tableau et x changent de valeurs ;*

Calculons les cent(100) premiers nombres premiers pour la représentée graphiquement :

P(n)=2n-1 +AV[(2n-1)Modulo(x)];
*n= 3; p(3)=2(3)-1+AV[(2*3-1)Mod(30)]*
p(3)=+AV[(5)Mod(30)]
p(3)=5+AV[5]
AV[5]=2; dans le tableau
p(3)=5+2
p(3)=7 ; $div_1(p(3))=1$
++++++++++++++++++

*n= 4; p(4)=2(4)-1+AV[(2*4-1)Mod(30)]*
p(4)=7+AV[(7)Mod(30)]
p(4)=7+AV[7]
AV[7]=4; dans le tableau
p(4)=7+4
p(4)=11 $div_1(p(4))=1$

++++++++++++++++++

*n= 5; p(5)=2(5)-1+AV[(2*5-1)Mod(30)]*

p(5)=9+AV[(9)Mod(30)]

p(5)=6+AV[10]

AV[10]=1; dans le tableau

p(5)=10+1

p(5)=11; $div_1(p(5))=1$

++++++++++++++++++++++

*n= 6; p(6)=2(4)-1+AV[(2*6-1)Mod(30)]*

p(6)=11+AV[(11)Mod(30)]

p(6)=11+AV[11]

AV[11]=2; dans le tableau

p(6)=11+2

p(6)=13; $div_1(p(6))=1$

++++++++++++++++++++++

*n= 7; p(7)=2(7)-1+AV[(2*7-1)Mod(30)]*

p(7)=13+AV[(13)Mod(30)]

p(7)=13+AV[13]

AV[13]=4; dans le tableau

p(7)=13+4

p(7)=17; $div_1(p(7))=1$

++++++++++++++++++++++

*n= 8; p(8)=2(8)-1+AV[(2*8-1)Mod(30)]*

p(8)=15+AV[(15)Mod(30)]

p(8)=15+AV[15]

AV[115]=2; dans le tableau

p(8)=15+2

p(8)=17; $div_1(p(8))=1$

++++++++++++++++++++

*n= 9; p(9)=2(9)-1+AV[(2*9-1)Mod(30)]*

p(9)=17+AV[(17)Mod(30)]

p(9)=17+AV[17]

AV[17]=2; dans le tableau

p(9)=17+2

p(9)=19; $div_1(p(9))=1$

++++++++++++++++++++++

*n= 10; p(10)=2(10)-1+AV[(2*10-1)Mod(30)]*

p(10)=19+AV[(19)Mod(30)]

p(10)=19+AV[19]

AV[19]=4; dans le tableau

p(10)=19+4

p(10)=23 ; $div_1(p(10))=1$

++++++++++++++++++++++

*n= 11; p(11)=2(11)-1+AV[(2*11-1)Mod(30)]*

p(11)=21+AV[(21)Mod(30)]
p(11)=21+AV[21]
AV[21]=2; dans le tableau
p(11)=21+2
p(11)=23 ; div_1(p(11))=1
++++++++++++++++++++++
*n= 12; p(12)=2(12)-1+AV[(2*12-1)Mod(30)]*
p(12)=23+AV[(23)Mod(30)]
p(12)=23+AV[23]
AV[23]=0; dans le tableau
p(12)=23+0
p(12)=23 ; div_1(p(12))=1
++++++++++++++++++++++
*n= 13; p(13)=2(13)-1+AV[(2*13-1)Mod(30)]*
p(13)=25+AV[(25)Mod(30)]
p(13)=25+AV[25]
AV[25]=4; dans le tableau
p(13)=25+4
p(13)=29 ; div_1(p(13))=1
++++++++++++++++++++++
*n= 14; p(14)=2(14)-1+AV[(2*14-1)Mod(30)]*
p(14)=27+AV[(27)Mod(30)]
p(14)=27+AV[27]
AV[27]=2; dans le tableau
p(14)=27+2
p(14)=29; div_1(p(14))=1
++++++++++++++++++++++
*n= 15; p(15)=2(15)-1+AV[(2*15-1)Mod(30)]*
p(15)=29+AV[(29)Mod(30)]
p(15)=29+AV[29]
AV[29]=2; dans le tableau
p(15)=29+2
p(15)=31; div_1(p(15))=1
++++++++++++++++++++++
*n= 16; p(16)=2(16)-1+AV[(2*16-1)Mod(30)]*
p(16)=31+AV[(31)Mod(30)]
p(16)=31+AV[1]
AV[1]=6; dans le tableau
p(16)=31+6
p(16)=37; div_1(p(16))=1
++++++++++++++++++++++
*n= 17; p(17)=2(17)-1+AV[(2*17-1)Mod(30)]*
p(17)=33+AV[(33)Mod(30)]
p(17)=33+AV[3]
AV[3]=4; dans le tableau
p(17)=33+4

p(17)=37; div$_1$(p(17))=1

++++++++++++++++++++

*n= 18; p(18)=2(18)-1+AV[(2*18-1)Mod(30)]*

p(18)=35+AV[(35)Mod(30)]

p(18)=35+AV[5]

AV[5]=2; dans le tableau

p(18)=35+2

p(18)=37; div$_1$(p(18))=1

++++++++++++++++++++

*n= 19; p(19)=2(19)-1+AV[(2*19-1)Mod(30)]*

p(19)=37+AV[(37)Mod(30)]

p(19)=37+AV[7]

AV[7]=4; dans le tableau

p(19)=37+4

p(19)=41 ; div$_1$(p(19))=1

++++++++++++++++++++

P20 = 41

++++++++++++++++++++

P21 = 43

++++++++++++++++++++

P22 = 47

++++++++++++++++++++

P23 = 47

++++++++++++++++++++

*n= 24; p(24)=2(24)-1+AV[(2*24-1)Mod(30)]*

p(24)=47+AV[(47)Mod(30)]

p(24)=47+AV[17]

AV[7]=2; dans le tableau

p(24)=47+2

p(24)=49 ; div$_1$(p(24))=7 non pre,ier

++++++++++++++++++++

P25 = 53

++++++++++++++++++++

P26 = 53

++++++++++++++++++++

P27 = 59

++++++++++++++++++++

P28 = 59

++++++++++++++++++++

P29 = 59

++++++++++++++++++++

P30 = 61

++++++++++++++++++++

P31 = 67

++++++++++++++++++++

P32 = 67

++++++++++++++++++++

P33 = 67

++++++++++++++++++++++

P34 = 71

++++++++++++++++++++++

P35 = 71

++++++++++++++++++++++

P36 = 73

++++++++++++++++++++

n= 37; p(37)=2(37)-1+AV[(2*37-1)Mod(30)]

p(37)=73+AV[(73)Mod(30)]

p(37)=73+AV[13]

AV[13]=4; dans le tableau

p(37)=73+4

p(37)=77 ; div_1(p(37))=7 non premier

++++++++++++++++++++

n= 38; p(38)=2(38)-1+AV[(2*38-1)Mod(30)]

p(38)=75+AV[(75)Mod(30)]

p(38)=75+AV[15]

AV[13]=2; dans le tableau

p(38)=75+2

p(38)=77 ; div_1(p(38))=7 non premier

++++++++++++++++++++++

P39 = 79

++++++++++++++++++++++

P40 = 83

++++++++++++++++++++++

P41 = 83

++++++++++++++++++++++

P42 = 89

++++++++++++++++++++++

P43 = 89

++++++++++++++++++++

P44 = 89

++++++++++++++++++++++

n= 45; p(45)=2(45)-1+AV[(2*45-1)Mod(30)]

p(45)=89+AV[(89)Mod(30)]

p(45)=89+AV[29]

AV[29]=2; dans le tableau

p(45)=89+2

p(45)=91 ; div_1(p(45))=7 non premier

P46 = 97

++++++++++++++++++++

P47 = 97

++++++++++++++++++++++

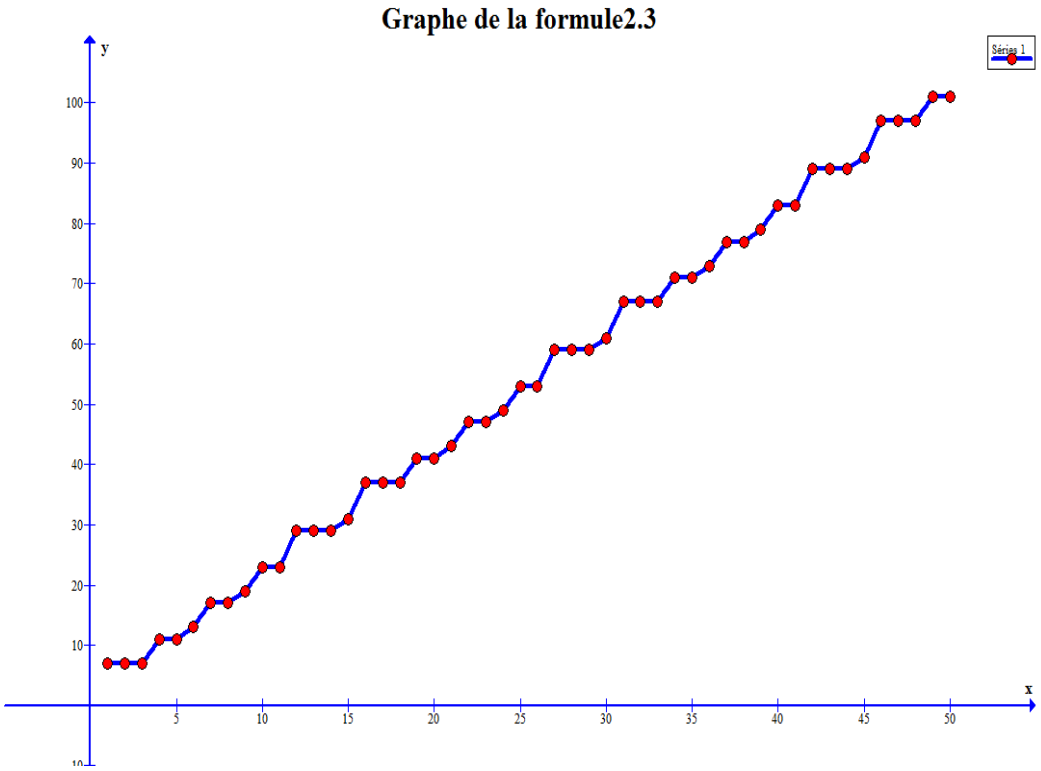
P48 = 97

++++++++++++++++++++++

P49 : 101

++++++++++++++++++++++++ P50 = 101

Représentation graphique Formule 2.3 :



Le code java en fonction de Int:

```
package Premierformule23;

import java.util.Scanner;


public class Premierformule23 {
	public static void main(String[] args) {
		Premierformule23  P1 = new Premierformule23 ();
	Scanner keby = new Scanner(System.in);
```

```
System.out.println("Veuillez saisir le rang inferieur n1 et superieur n2 :");
int s=keby.nextInt();
int m=keby.nextInt();
for (int x = s ; x <= m; x++) {

System.out.printf("P%d  : %d \n",+x, +P1.Formule3(x));

}
keby.close();
}
public int Formule3 (int n){
        //long p=0;

        int AV[] = {
                1,   6,   5,   4,   3,   2,
                 1,   4,   3,   2,   1,  2,
                 1,  4,  3,  2, 1, 2,
                 1,  4, 3, 2,  1,  6,
                 5,  4,  3,  2,  1, 2
               };


        return (2*n-1+ AV[(2*n-1)%30]);

}
```

}

Code java avec combinaison avec la formule2.1

```
package PremierFormule4;

import java.util.ArrayList;
import java.util.Scanner;


public class PremierFormule4 {

	public static void main(String[] args) {
		PremierFormule4  P1 = new PremierFormule4 ();
	Scanner keby = new Scanner(System.in);

	System.out.println("Veuillez saisir le rang inferieur n1 et superieur n2 :");
	int s=keby.nextInt();
	int m=keby.nextInt();
	for (int x = s ; x <= m; x++) {

	System.out.printf("P%d  : %d \n",+x, +P1.Formule4(x));

	}
	keby.close();
	}
```

```
public int Formule4 (int n){
        //long p=0;

        int [] AV = {
                    1,   6,   5,   4,   3,   2,
                     1,   4,   3,   2,   1,  2,
                     1,  4,  3,  2, 1, 2,
                     1,  4, 3, 2,  1,  6,
                     5,  4,  3,  2,  1, 2
                  };
        int l=(2*n-1)%30;
        int d=Div1(2*n-1+ AV[l]);
        if (n<=2){
              return Formule2 (n);
        }else if (d==1){
        return (2*n-1+ AV[(2*n-1)%30]);
        }else{

              return Formule2 (n);
        }
}
public int Div1 (int n) {

        int i =3;

        /*long d=n-1 ;*/
```

```
        double d=Math. sqrt(n) ;

        int div=1;

        if ((n) % (2)== 0) {
              div=2;

        } else{

        while ((i<=d) && ((n) % (i) != 0)) {

        i =i+2 ;

        }

        if ((i<=d) &&(n) % (i) ==0) {
        div = i ;

        }

        }

        return div;
        }

public int Formule2 (int n){
```

```
int p=0;
loop1:
for(int j=n;j<=2*n;j++) {
n=j;
int i=0;
int r=0;
int control = 2 * n + 1 ;
int Q = Div1(2 * n + 1);
int b = Q;
int sortie = 0;
ArrayList<Integer> K= new ArrayList<Integer> ();
K.add(control);



while ((Q!=1) && (sortie == 0)) {

b = Q ;
control=2 * n + Q;
Q = Div1(2 * n + Q);
r=K.size();
while (i<r) {

if (control== K.get(i) ) {
```

```
sortie = sortie + 1;

}

i=i+1;
}


K.add(2 * n + Q);

}


if ((Q==1) && (sortie == 0) ){

p=2 * n + b;  break loop1;



}

   }

return p;

}
```

}

Amélioration 2.3 :

$P(n)=2n-1+AV[(2*n-1)Mod30]+2[div_1(2*n-1+AV[(2*n-1)Mod30])-1]/div_1(p)=1$

Cette amélioration permet de trouver correspondance première à chacune de ces valeurs :

$p(24)=49$; $div_1(p(24))=7$ n'est pas premier ;

$p(37)=77$; $div_1(p(37))=7$ non premier ;

$p(38)=77$; $div_1(p(38))=7$ non premier ;

$p(45)=91$; $div_1(p(45))=7$ non premier ;

Ainsi nous aurons :

$P24 =: 61$; $div_1(p(24))=1$ est premier

$P37 = 89$ $div_1(p(37))=1$ est premier

$P38 = 89$; $div_1(p(38))=1$ est premier

$P45 = 103$;$div_1(p(45))=1$ est premier

Code java avec combinaison avec la formule2.1

```
package formulecorriger2;

import java.util.ArrayList;
import java.util.Scanner;


public class formulecorriger2 {
```

```
public static void main(String[] args) {
        formulecorriger2   P1 = new formulecorriger2  ();
Scanner keby = new Scanner(System.in);

System.out.println("Veuillez saisir le rang inferieur n1 et
superieur n2 :");
int s=keby.nextInt();
int m=keby.nextInt();
for (int x = s ; x <= m; x++) {

System.out.printf("P%d  : %d \n",+x, +P1.Formule3(x));

}
keby.close();
}
public int Formule3 (int n){
        //long p=0;
        int i;
        int j;
        int form;
        int AV[] = {
                1,   6,   5,   4,   3,   2,
                 1,   4,   3,   2,   1,  2,
                 1,  4,  3,  2, 1, 2,
                 1,  4, 3, 2,  1,  6,
```

```
                5,  4,  3,  2,  1, 2
            };
        i=AV[(2*n-1)%30];
        j=Div1(2*n-1+i);
        form=2*n-1+i+2*(j-1) ;

        if (n<=2){

            return Formule2 (n);

        }else if (Div1(form)==1){

            return form;

    }else{

            return Formule2 (n);

    }

}

public int Div1 (int n) {

        int i =3;
```

```
        /*long d=n-1 ;*/
        double d=Math. sqrt(n) ;

        int div=1;

        if ((n) % (2)== 0) {
                div=2;

        } else{

        while ((i<=d) && ((n) % (i) != 0)) {

        i =i+2 ;

        }

        if ((i<=d) &&(n) % (i) ==0) {
        div = i ;

        }

        }

        return div;
        }
public int Formule2 (int n){
```

```
int p=0;
loop1:
for(int j=n;j<=2*n;j++) {
n=j;
int i=0;
int r=0;
int control = 2 * n + 1 ;
int Q = Div1(2 * n + 1);
int b = Q;
int sortie = 0;
ArrayList<Integer> K= new ArrayList<Integer> ();
K.add(control);



while ((Q!=1) && (sortie == 0)) {

b = Q ;
control=2 * n + Q;
Q = Div1(2 * n + Q);
r=K.size();
while (i<r) {

if (control== K.get(i) ) {
```

```
sortie = sortie + 1;

}

i=i+1;
}


K.add(2 * n + Q);

}


if ((Q==1) && (sortie == 0) ){

p=2 * n + b;  break loop1;



}

   }

return p;

}
```

}

On ne ferra pas l'étude de la formule :

$p(n)=2(n-1)+div_1(2n-1) / div_1(p)=1$; *qui a la même représentation que les deux précédentes formules et qui pourra aussi être alternée avec la formule2.1*

càd $p(n)=2n+b.$ *quand* $div_1(p) \neq 1$;

Sans oublier la fonction à deux(2) variables :

Soit N différent de zéro :

$\forall n1, n2 \in N, f(n1,n2)=(n1+n2+2)/2+div_1(n1+n2+2)/2)-1 /$

$n1 < n2$; $n2=n1+1$; *n1 est pair ; n2 est impair ;* $div_1(p)=1$;

Code java combinaison Formule2.1et Formule2.2 avec biginteger :

```
package FormuleBigInteger1;

import java.math.BigInteger;
import java.util.ArrayList;
import java.util.Scanner;
public class FormuleBigInteger1 {

public static void main(String[] args) {
FormuleBigInteger1  P1 = new FormuleBigInteger1 ();
Scanner keby = new Scanner(System.in);
```

```
System.out.println("Veuillez saisir le rang inferieur n1 et
superieur n2 :");
 BigInteger s=keby.nextBigInteger();
 BigInteger m=keby.nextBigInteger();
 for (BigInteger x = s ; x.compareTo(m) == -1 ||
x.compareTo(m) == 0;x = x.add(new BigInteger("1")) ) {
System.out.printf("\n"+ P1.Formule3(x));

}
keby.close();
}
public BigInteger Formule3 (BigInteger n){

int  AV[] ={   1,   6,   5,   4,   3,   2,
    1,   4,   3,   2,   1,  2,
     1,  4,  3,  2, 1, 2,
     1,  4, 3, 2,  1,  6,
     5,  4,  3,  2,  1, 2
};
BigInteger deux = new BigInteger("2");
BigInteger bi1=(deux.multiply(n)).mod(new BigInteger("30"));
Integer i1 = bi1.intValue();
int v=AV[i1];
BigInteger j =
Div1((deux.multiply(n)).add(BigInteger.valueOf(v)));
```

```
BigInteger form =
deux.multiply(n).add(BigInteger.valueOf(v).add(deux.multiply((
j).subtract(new BigInteger("1"))))); //int form=2*n+i+2*(j-1) ;
if ( n.compareTo(BigInteger.valueOf(2))<=0 ){
return Formule2 (n);
}else if (Div1(form).compareTo(BigInteger.valueOf(1))==0){

return form;
  }else{
    return Formule2 (n);
   }
}
public BigInteger Div1 (BigInteger n) {

//BigInteger d=sqrt(n);

      int step;
    if ( n.compareTo(BigInteger.valueOf(5))<0 ) {
if((n.compareTo(BigInteger.valueOf (2)) == 0) ||
(n.compareTo(BigInteger.valueOf (3)) == 0))  {
 return BigInteger.valueOf (1) ;
 }
else {
return BigInteger.valueOf (2);
        }
   }
```

```
 else  if((n.mod(new
BigInteger("6")).equals(BigInteger.ZERO))||(n.mod(new
BigInteger("6")).equals( BigInteger.valueOf(2)))||(n.mod(new
BigInteger("6")).equals( BigInteger.valueOf(4))))
  {
return BigInteger.valueOf (2);
          }
      else if (n.mod(new BigInteger("6")).equals(
BigInteger.valueOf(3))) {
return BigInteger.valueOf (3);

       }
   else {
    step=2;
    int  i= 5;
 while
(BigInteger.valueOf(i).multiply(BigInteger.valueOf(i)).compare
To(n) <= 0)  {
if (n.mod(BigInteger.valueOf(i)).equals(
BigInteger.valueOf(0))) {
 return BigInteger.valueOf (i);
else {
i=i+ step;
    if ( step==2) {
          step= 4;
       }
```

```
      else {
step=2;
          }
}
    }
    return BigInteger.valueOf (1);

                    }
   }

public BigInteger Formule2 (BigInteger n){

BigInteger p=new BigInteger("0");
loop1:
for (BigInteger j = n ; j.compareTo(n.multiply(new
BigInteger("2"))) < 0 /*|| j.compareTo(n.multiply(new
BigInteger("2"))) == 0*/;j = j.add(new BigInteger("1")) ) {
n=j;
int i=0;
int r=0;
BigInteger deux = new BigInteger("2");
BigInteger control = (deux.multiply(n)).add(new
BigInteger("1"));
BigInteger Q = Div1((deux.multiply(n)).add(new
BigInteger("1")));
BigInteger b = Q;
```

```
long sortie = 0;
ArrayList<BigInteger> K= new ArrayList<BigInteger> ();
K.add((deux.multiply(n)).add(new BigInteger("1")));
while ((Q.compareTo(new BigInteger("1")) <0) && (sortie ==
0)) {
b = Q ;
control = (deux.multiply(n)).add(new BigInteger("Q"));
Q = Div1((deux.multiply(n)).add(new BigInteger("Q")));
r=K.size();
while (BigInteger.valueOf(i).compareTo(BigInteger.valueOf(r))
< 0)  {
if (control== K.get(i) ) {
sortie = sortie + 1;
}
i=i+1;
}
K.add((deux.multiply(n)).add(new BigInteger("Q")));
}

if ((Q.compareTo(new BigInteger("1")) ==0) && (sortie == 0))
{  p=(deux.multiply(n)).add(b);
  break loop1 ;
}
   }
return p;
```

}

}

Code java combinaison Formule2.1et Formule2.3 avec biginteger :

```
package formulebigInteger2;

import java.math.BigInteger;
import java.util.ArrayList;
import java.util.Scanner;

public class formulebigInteger2 {

    public static void main(String[] args)
{
    formulebigInteger2  P1 = new
formulebigInteger2 ();
     Scanner keby = new
Scanner(System.in);

    System.out.println("Veuillez saisir le
rang inferieur n1 et superieur n2 :");
     BigInteger s=keby.nextBigInteger();
     BigInteger m=keby.nextBigInteger();
     for (BigInteger x = s ;
x.compareTo(m) == -1 || x.compareTo(m) ==
0;x = x.add(new BigInteger("1")) ) {

System.out.printf("\n"+ P1.Formule3(x));
```

```
}
keby.close();
    }
public BigInteger Formule3 (BigInteger
n){

        int  AV[] ={
                1,   6,   5,   4,   3,   2,
                 1,   4,   3,   2,   1,  2,
                 1,  4,  3,  2, 1, 2,
                 1,  4, 3, 2,  1,  6,
                 5,  4,  3,  2,  1, 2
            };

        BigInteger un = new
BigInteger("1");
        BigInteger deux = new
BigInteger("2");
        BigInteger
bi1=(deux.multiply(n).subtract(un)).mod(n
ew BigInteger("30"));
        Integer i1 = bi1.intValue();
        int v=AV[i1];

        BigInteger j =
Div1((deux.multiply(n)).add(BigInteger.va
lueOf(v)));
```

```
        BigInteger form =
deux.multiply(n).add(BigInteger.valueOf(v
).add(deux.multiply((j).subtract(new
BigInteger("1"))))));  //int
form=2*n+i+2*(j-1) ;

        if (
n.compareTo(BigInteger.valueOf(2))<=0 ){

            return Formule2 (n);

        }else if
(Div1(form).compareTo(BigInteger.valueOf(
1))==0){

            return form;

      }else{

              return Formule2 (n);

        }

    }

    public BigInteger Div1 (BigInteger n)
{
```

```
        //BigInteger d=sqrt(n);


            int step;


             if (
n.compareTo(BigInteger.valueOf(5))<0 ) {



    if((n.compareTo(BigInteger.valueOf
(2)) == 0) ||
(n.compareTo(BigInteger.valueOf (3)) ==
0))  {

                    return
BigInteger.valueOf (1) ;
                }

                else {


                    return
BigInteger.valueOf (2);

                      }

                    }
```

```
                    else  if((n.mod(new
BigInteger("6")).equals(BigInteger.ZERO))
||(n.mod(new BigInteger("6")).equals(
BigInteger.valueOf(2)))||(n.mod(new
BigInteger("6")).equals(
BigInteger.valueOf(4))))
                    {

                     return
BigInteger.valueOf (2);

                    }

                    else if (n.mod(new
BigInteger("6")).equals(
BigInteger.valueOf(3))) {

                     return
BigInteger.valueOf (3);

                    }
                                    else {

                step=2;
                int  i= 5;

                while
(BigInteger.valueOf(i).multiply(BigIntege
r.valueOf(i)).compareTo(n) <= 0)  {
```

```
                if
(n.mod(BigInteger.valueOf(i)).equals(
BigInteger.valueOf(0))) {

                    return
BigInteger.valueOf (i);
                }
               else {

                  i=i+ step;

                     if ( step==2) {
                         step= 4;
                   }
                  else {

                      step=2;
                     }
                }
             }
                 return BigInteger.valueOf
(1);

                                      }
           }
```

```
    public BigInteger Formule2 (BigInteger
n){

    BigInteger p=new BigInteger("0");

    loop1:

    for (BigInteger j = n ;
j.compareTo(n.multiply(new
BigInteger("2"))) < 0 /*||
j.compareTo(n.multiply(new
BigInteger("2"))) == 0*/;j = j.add(new
BigInteger("1")) ) {

    n=j;

    int i=0;
    int r=0;

    BigInteger deux = new BigInteger("2");

    BigInteger control =
(deux.multiply(n)).add(new
BigInteger("1"));
    BigInteger Q =
Div1((deux.multiply(n)).add(new
BigInteger("1")));
    BigInteger b = Q;
```

```
    long sortie = 0;

    ArrayList<BigInteger> K= new
ArrayList<BigInteger> ();
    K.add((deux.multiply(n)).add(new
BigInteger("1")));


    while ((Q.compareTo(new
BigInteger("1")) <0) && (sortie == 0)) {
    b = Q ;
    control = (deux.multiply(n)).add(new
BigInteger("Q"));
    Q = Div1((deux.multiply(n)).add(new
BigInteger("Q")));
    r=K.size();
    while
(BigInteger.valueOf(i).compareTo(BigInteg
er.valueOf(r)) < 0)  {

    if (control== K.get(i) ) {

    sortie = sortie + 1;

    }

    i=i+1;
    }
```

```
    K.add((deux.multiply(n)).add(new
BigInteger("Q")));

    }

    if ((Q.compareTo(new BigInteger("1"))
==0) && (sortie == 0)) {



      p=(deux.multiply(n)).add(b);
      break loop1 ;



    }


       }

    return p;

    }


}
```

Formule 3 :

Ici on saute sur des rangs correspondants aux nombres composés ; Seul les nombres premiers qui sont calculés.

P(1)=3 est premier

P(2)=5 est premier

P(3)=7 est premier

P(5)=11 est premier

P(6)=13 est premier

P(8)=17 est premier

P(9)=19 est premier

P(11)=23 est premier

P(14)=29 est premier

P(15)=31 est premier

Example:

i=16

nbdiv(2x16+1) >2;

A_{16}=16-1+ PGCD

(2x16,div_1(2x16))

A_{16}=15+ PGCD (32,div_1(32))

A_{16}=15+ PGCD (32 , 2)

A_{16}=15+2

A_{16}=17

Ou tout simplement:

i=16

nbdiv(2x16+1) >2;

A_i=i+1

A_{16}=16+1

A_{16}=17

nbdiv(2x17+1) = nbdiv(35)>2 ;

A_{17}=16-1+ PGCD

(2x17,div_1(2x17))

A_{17}=16+ PGCD (34,div_1(34))

A_{17}=16+ PGCD (34 , 2)

A_{17}=16+2

A_{17}=18

Ou tout simplement:

nbdiv(2x17+1) = nbdiv(35)>2 ;

A_i=i+1

A_{17}=17+1

A_{17}=18

nbdiv(2x18+1) = nbdiv(37)=2 ;

n=A_{17} ; n=18; i=18

P(n)=2n+1 ;

P(18)=2x18+1;

P(18)=36+1 ;

P(18)=37 ;

---

i=19

nbdiv(2x19+1)=nbdiv(39) >2 ;

A_{19}=19-1+ PGCD
(2x19,div_1(2x19))
A_{19}=18+ PGCD (38,div_1(38))
A_{19}=18+ PGCD (38 , 2)
A_{19}=18+2
A_{20}=20
Ou tout simplement:
A_i=i+1
A_{19}=19+1
A_{19}=20

nbdiv(2x20+1)= nbdiv(41)=2 ;
n=A_{20} ; n=20; i=20
P(n)=2n+1 ;
P(20)=2x20+1;
P(20)=40+1 ;
P(20)=41 ;

P(21)=43 est premier

P(23)=47 est premier
P(26)=53 est premier

P(29)=59 est premier

P(30)=61 est premier

P(33)=67 est premier

P(35)=71 est premier

P(36)=73 est premier

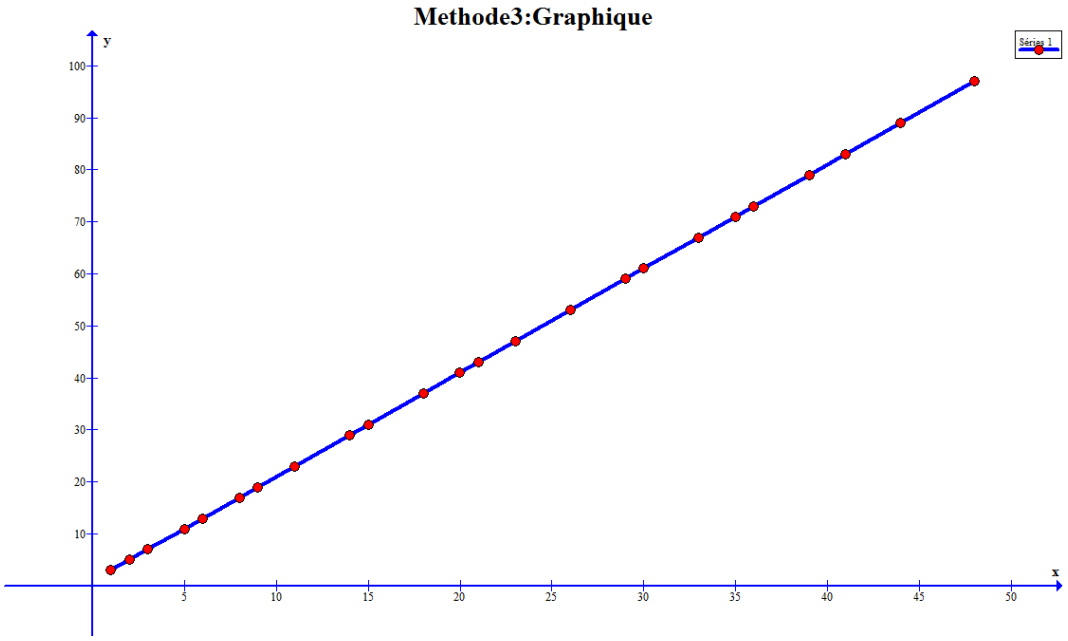
P(39)=79 est premier

P(41)=83 est premier

P(44)=89 est premier

P(48)=97 est premier

Représentation graphique Formule 6 :



Programme VBA obtenu sur forum d'entraide :

```
Function Div1(n As Long)
'-------------------------------------
-------------------------------------
---------------
Dim i As Long
If n Mod (2) = 0 Then
    Div1 = 2
Else     For i = 3 To n / 2 Step 2
for i=3 to n ici)
         If n Mod (i) = 0 Then
             Div1 = i
             Exit For
         End If
    Next i
End If
End Function
```

```
'-----------------------------------
-----------------------------------
--------------
Function Nbdiv(n As Long) As Long
'-----------------------------------
-----------------------------------
--------------
Dim i As Long
If TestMillerRabin(n) = True Then
    Nbdiv = 2
Else
    Nbdiv = 3
End If
End Function

'-----------------------------------
-----------------------------------
--------------
Function Pgcd(x1 As Long, x2 As Long)
As Long
'-----------------------------------
-----------------------------------
--------------
' Calcul le PGCD de deux nombres
' Variante optimisant l'Algo
d'Euclide
'-----------------------------------
-----------------------------------
--------------
Dim r As Long
```

```
' Validité paramètre:
If x1 < x2 Then r = x2: x2 = x1: x1 =
r 'inverse les valeurs

' Calcul:
Do While Abs(x2) >= 1
    r = x1 Mod x2
    x1 = x2
    x2 = r
Loop
Pgcd = Abs(x1)
End Function

'-------------------------------------
-------------------------------------
--------------
Function Main(n As Long)
'-------------------------------------
-------------------------------------
--------------
Dim i As Long
Dim A As Long
Dim p As Long

i = n
If Nbdiv(2 * i + 1) > 2 Then
    Do
        A = (i - 1) + Pgcd(2 * i,
Div1(2 * i))
```

```
            If Nbdiv(2 * A + 1) = 2 Then
                p = 2 * A + 1
            Else
                i = A
            End If
        Loop While p = 0
Else
    p = 2 * i + 1
End If
Main = p
End Function

'-------------------------------------
-------------------------------------
--------------
Function TestMillerRabin(ByVal n As
Variant) As Boolean
'-------------------------------------
-------------------------------------
--------------
' Test MILLER RABIN. Retourne Vrai si
n est un nombre premier
'-------------------------------------
-------------------------------------
--------------
Dim A As Long, BaseMaxi As Long
Dim Base

' Optimisation nb base à tester:
```

```
Base = Array(2, 3, 5, 7, 11, 13, 17,
19, 23)
BaseMaxi = Int(4 + Log(n) / 8 - Abs(4
- Log(n) / 8))

If IsMultiple(n, 2) = True Then Exit
Function

' Algo:
A = 0
Do
    If IsMultiple(n, Base(A)) = False
Then
        If MillerRabin(n, Base(A)) =
False Then Exit Function
    End If
A = A + 1
Loop Until A > BaseMaxi

TestMillerRabin = True
End Function

'-----------------------------------
-----------------------------------
--------------
Function MillerRabin(ByVal n As
Variant, ByVal A As Variant) As
Boolean
```

```
'-------------------------------------
-------------------------------------
--------------
' RQ : IMPOSSIBILITE DE REPONDRE SI n
multiple de a
'-------------------------------------
-------------------------------------
--------------
' Tests triviaux:
If MOD2(n, 2) = 0 Or n < 3 Then Exit
Function

' Paramètres:
Dim h As Long, d As Variant, i As
Long

' Convertit les variants en Decimal:
n = CDec(n): A = CDec(A): d = CDec(d)

' Calcul de n-1 = 2^h * d avec d
impair:
Do
    h = h + 1
    d = (n - 1) / 2 ^ h
Loop Until d = Int(d) And MOD2(d, 2)
= 1

' Test a^d=1 mod n:
MillerRabin = (ExpoMod(A, d, n) = 1)
If MillerRabin Then Exit Function
```

```
' Test s'il existe un 0<= i <= h-1
tel que a^(h*2^i)=-1 mod n:
For i = 0 To h - 1
    MillerRabin = (ExpoMod(A, d * 2 ^
i, n) = n - 1)
    If MillerRabin Then Exit For
Next i

End Function

'-----------------------------------
-------------------------------
Function ExpoMod(ByVal Nb As Variant,
ByVal Expo As Variant, _
                          ByVal Modulo
As Variant) As Variant
'-----------------------------------
-------------------------------
' EXPONENTIATION MODULAIRE RAPIDE :
Nb^Expo MOD Modulo.
'-----------------------------------
-------------------------------
' Convertit les variants en Decimal:
Nb = CDec(Nb): Expo = CDec(Expo):
Modulo = CDec(Modulo)

' Traitement:
ExpoMod = 1
Do
```

```
    If MOD2(Expo, 2) = 1 Then
        ExpoMod = MODProd(Nb,
ExpoMod, Modulo)
        Expo = (Expo - 1) / 2
        Nb = MODProd(Nb, Nb, Modulo)
    End If

If MOD2(Expo, 2) = 0 Then
        ExpoMod = MODProd(ExpoMod, 1,
Modulo)
        Expo = Expo / 2
        Nb = MODProd(Nb, Nb, Modulo)
    End If

Loop Until Expo = 0
End Function

'-------------------------------------
-----------------------------
Function MOD2(ByVal d As Variant,
ByVal n As Variant) As Variant
'-------------------------------------
-----------------------------
' Renvoie le modulo de d et n:
'-------------------------------------
-----------------------------
d = CDec(d): n = CDec(n)
MOD2 = CDec(d - n * Int(d / n))
End Function
```

```
'-------------------------------------
-------------------------------
Function IsMultiple(ByVal Nb1 As
Variant, ByVal Nb2 As Variant) As
Boolean
'-------------------------------------
-------------------------------
' Teste si Nb1 est multiple de Nb2.
'-------------------------------------
-------------------------------
Nb1 = CDec(Nb1): Nb2 = CDec(Nb2)
If Nb2 = 0 Then IsMultiple = True:
Exit Function
IsMultiple = ((Int(Nb1 / Nb2) = Nb1 /
Nb2) And Nb1 <> 0)
End Function

'-------------------------------------
-------------------------------
Function MODProd(ByVal Nb1 As
Variant, ByVal Nb2 As Variant, _
                          ByVal Modulo
As Variant) As Variant
'-------------------------------------
-------------------------------
' Renvoie le modulo du produit
"nb1*nb2 MOD Modulo" sans la limite
Double.
'-------------------------------------
-------------------------------
```

```
' Convertit les variants en Decimal
et teste la grandeur du produit:
Nb1 = CDec(Nb1): Nb2 = CDec(Nb2):
Modulo = CDec(Modulo)
If Nb1 * Nb2 < 10 ^ 15 Then MODProd =
MOD2(Nb1 * Nb2, Modulo): Exit
Function

' Paramètre les variables en Decimal:
Dim r As Variant, C As Variant,
Facteur As Variant, d As Variant
r = CDec(r): C = CDec(C): Facteur =
CDec(Facteur): d = CDec(d)

' Prend le mini => plus rapide:
If Nb1 < Nb2 Then r = Nb2: Nb2 = Nb1:
Nb1 = r

' Optimisation facteur:
Facteur = 9
r = MOD2(Nb1, Modulo)
Do
    If IsMultiple(Nb2, Facteur) Then
        r = MOD2(Facteur * r, Modulo)
        Nb2 = Nb2 / Facteur
    Else
        d = MOD2(Nb2, Facteur)
        C = MOD2(C + r * d, Modulo)
        Nb2 = Nb2 - d
    End If
```

```
Loop Until Nb2 = 0
MODProd = MOD2(C, Modulo)
End Function

'-----------------------------------
-----------------------------------
--------------
Function InversMod(ByVal Nba As
Variant, ByVal Nbb As Variant) As
Variant
'-----------------------------------
-----------------------------------
--------------
Dim A, b, Q, t, x, y

' Convertit les variants en Decimal:
A = CDec(Nbb): b = CDec(Nba): Q =
CDec(Q): t = CDec(t): x = CDec(x): y
= CDec(y)

x = 1: y = 0
While (b <> 0)
    t = b
    Q = Int(A / t)
    b = A - Q * t
    A = t
    t = x
    x = y - Q * t
    y = t
Wend
```

```
InversMod = CDec(y)
If y < 0 Then InversMod = CDec(y +
Nbb)
End Function
```

Représentation des trois méthodes les trois Formules: