Les nombres premiers

Année 2023 – 2024

VUEBAT Hugo 1°5

Encadré par CASTAGNEDOLI Sébastien; FORT Élodie

Établissement : Institut Emmanuel d'Alzon – 30000 Nîmes

Chercheur: DUMONT Serge, LAMPS, Université de Perpignan Via Domitia

Table des matières

1. Présentation du sujet

Euclide a pu démontrer qu'il existe une infinité de nombres premiers. Sauriez-vous le montrer également ?
Mais beaucoup de questions peuvent encore se poser:
1. Existe-t-il des nombres premiers k tel que $n=k^2-1$ soit aussi un nombre premier ?
2. Un entier k étant préalablement choisi, existe-t-il des nombres premiers n tel que $n+k$ est encore un nombre premier ?
3. Existe-t-il beaucoup de nombres premiers congrus à 3 modulo 4 ?

2. Annonce des conjectures et résultats obtenus

Pour la première question, si k est un nombre premier est-ce-que $k^2 - 1$ est-il aussi premier, la réponse est non, sauf pour $k=2$.
Pour la deuxième question, si n est un nombre premier et k un nombre entier naturel alors $n + k$ n'est jamais premier quand k est impair, et si k est pair, il y a un programme python.
Pour la dernière question, s'il existe une infinité de nombres premiers $n \equiv 3[4]$, la réponse est oui, il en existe une infinité.

3. Preuve de principaux résultats

I. $n=k^2 - 1$ est-il premier?

a) $k=2$

k est un nombre premier. Si $k=2$, alors n est un nombre premier car $n=2^2 - 1=3$, donc si $k=2$, $n=3$ donc n est aussi un nombre premier quand $k=2.$

b) $k>2$

k est un nombre premier. $n=k^2 - 1 \Leftrightarrow n=(k-1)(k+1)$
Si $k>2$ alors $k - 1>1$ donc n possède un diviseur strict, $k - 1$
Donc n n'est pas que divisible par lui même et 1, alors n n'est pas un nombre premier pour $k>2.$

II. $n + k$ est-il premier ?

a) Avec k, un nombre impair

Si n est un nombre premier, alors n est un nombre impair, s'écrivant sous la forme $2l + 1$
k est un nombre impair, s'écrivant sous la forme $2l+1$
Donc en les additionnant:

$$n+k \Leftrightarrow (2l+1) + (2l+1)=4l + 2=2(2l+1)$$

$n + k$ est donc un nombre pair car il est multiple de deux.
Donc quand k est un nombre impair, la somme de n, un nombre premier, et de k n'est pas un nombre premier car la somme est paire, donc divisible par deux.

b) Avec *k*, un nombre pair

Si *n* est un nombre premier, alors *n* est un nombre impair, s'écrivant sous la forme *2l + 1.*
k est un nombre pair, s'écrivant sous la forme *2l.*
Donc en les additionnant:

$n + k \Leftrightarrow (2l+1) + 2l = 4l + 1$
$2L=4l$ donc $n + k \Leftrightarrow 2L + 1$

n + k est donc un nombre impair car il s'écrit de la forme *2L+1*.
Donc quand *k* est un nombre pair, la somme de *n*, un nombre premier, et de *k* est possiblement un nombre premier. Quand *k* est un nombre pair, il n'y a pas de cas général donc nous pouvons créer un programme informatique en langage python:

Les instructions:

- ➔ ligne 1 et 2, sont présentes pour rentrer("input"), les valeurs de *n* et de *k* choisies.

Les instructions (ligne 3 à 6) ont pour but de vérifier si *n* est bien un nombre premier.

- ➔ Ligne 3, créée une liste entre 2 et *n - 1.*
- ➔ Ligne 4, on vient faire une division de *n* par tous les nombres de la liste établie à la ligne 3. Si l'un des restes est égal à 0.
- ➔ Ligne 5, si la condition de la ligne 4 est respectée alors *n* n'est pas premier car il est divisible par un autre nombre que 1 et lui-même. Et donc le programme renvoie, "erreur: input a prime number" dont la traduction, "erreur: entrez un nombre premier".

➔ Ligne 6, ferme le programme pour le recommencer.

Si n est bien un nombre premier(ligne 7), alors le programme continue.

➔ Ligne 8, on appelle p, le nombre égal à $n + k$.

Après le calcul p, les instructions qui suivent (ligne 9 à 15), nous permettent de déterminer si p est premier donc si $n + k$ est premier en utilisant la même technique que pour n.

➔ Ligne 9, créée une liste entre 2 et $p - 1$.
➔ Ligne 10, on vient faire une division de p par tous les nombres de la liste établie à la ligne 9. Si l'un des restes est égal à 0.
➔ Ligne 11, si la condition de la ligne 10 est respectée alors n n'est pas premier car il est divisible par un autre nombre que 1 et lui-même. Et donc le programme renvoie, "p it is not a prime number" dont la traduction, "p n'est pas un nombre premier".
➔ Ligne 12, Ligne 6, ferme le programme pour le recommencer.
➔ Ligne 13, si la condition de la ligne 10 n'est pas respectée.
➔ Ligne 14, alors p est premier car il est divisible que par 1 et lui-même. Et donc le programme renvoie, "p it is a prime number" dont la traduction, "p est un nombre premier".
➔ Ligne 15, ferme le programme pour le terminer.

```python
k = input("input a natural number")
n = input("input a prime nuber")

for j in range(2, int(n)-1):
    if (int(n) % j) == 0:
        print("erreur: input a prime number")
        exit()
else:
    p = int(n) + int(k)
    for i in range(2, int(p)-1):
        if (int(p) % i) == 0:
            print(p, "It is not a prime number")
            exit()
    else:
        print(p,"It is a prime number")
        exit()
```

III. La congruence des nombres premiers

a) $n \equiv 3[4]$

Un nombre, n, congrus à 3 modulo 4 se note $n \equiv 3 \bmod 4$ ou $n \equiv 3[4]$.
Un nombre, n, est congrus à a modulo b si, et seulement si $n - a$ est divisible par b.
Ainsi un nombre, n, est congrus à 3 modulo 4 si, et seulement si $n - 3$ est divisible par 4.

$\mathcal{P}$ est l'ensemble des nombres premiers, $p \equiv 3[4]$.

Démonstration que $\mathcal{P}$ est infini:

Pour la démonstration, nous devons faire un lemme.
Comme $n \in \mathbb{N}$, il admet une décomposition en facteurs premiers:

$$n = p_1^{\alpha 1} \times p_2^{\alpha 2} \times \ldots \times p_n^{\alpha n} \text{ avec } n \equiv 3[4]$$

Donc pour tout i, $p_i \geq 2$, p_i est un nombre premier et $p_i \equiv 1[4]$ ou $p_i \equiv 3[4]$.

Pour démontrer que $\mathcal{P}$ est infini il faut que n ait un facteur premier $p_i \equiv 3[4]$, et pour le démontrer, on utilise un raisonnement par l'absurde:

On suppose $p_i \equiv 1[4]$. Si tous les $p_i \equiv 1[4]$ alors $n \equiv 1[4]$.
Sauf que d'après la lemme, le résultat est *ad absurdum* donc n a un facteur premier $p_i \equiv 3[4]$.

Si on prend un nombre, $s \equiv 3[4]$, il peut s'écrire:

$$s = 4(p_1 \times p_2 \times \dots \times p_n) - 1$$

ce nombre, s, peut être soit un nombre premier soit pas.

Si s est un nombre premier, $s \equiv -1[4]$ donc $s \equiv 3[4]$.
Donc $s \equiv 3[4]$, $s \in \mathcal{P}$. Sauf que d'après la théorie, $s \notin \mathcal{P}$.
C'est *ad absurdum* donc il existe une infinité de nombres premiers, $n \equiv 3[4]$.

Si s n'est pas un nombre premier, $s \equiv -1[4]$ donc $s \equiv 3[4]$.
D'après la lemme: s a au moins un facteur premier, $p_i \equiv 3[4]$.
Donc p_i divise s avec $p_i \in \mathcal{P}$ et p_i divise $s+1 = 4(p_1 \times p_2 \times \dots \times p_n)$.
p_i divise la différence $s - (s + 1) = -1$. Donc pi divise -1.
C'est *ad absurdum*. Donc il existe une infinité de nombres premiers, $n \equiv 3[4]$.

Que s soit premier ou pas, nous arrivons à la même conclusion. A l'aide d'une démonstration par l'absurde faisant en partie référence à la démonstration d'Euclide des nombres premiers. Donc il existe une infinité de nombres premier, $n \equiv 3[4]$ et donc $\mathcal{P}$ est infini.

b) Le programme

Pour savoir si *n*, un nombre premier est congrus à 3 modulo 4 ainsi, nous avons élaboré un programme informatique en langage python:

➔Ligne 1, fait appel à la bibliothèque « math ».

Les instructions:

➔Ligne 2, est présente pour rentrer(“input”), la valeur de *n.*

Les instructions (ligne 3 à 6) ont pour but de vérifier si *n* est bien un nombre premier.

➔Ligne 3, créée une liste entre 2 et *n - 1*.
➔Ligne 4, on vient faire une division de *n* par tous les nombres de la liste établie à la ligne 3. Si l’un des reste est égal à 0.
➔Ligne 5, si la condition de la ligne 4 est respectée alors *n* n’est pas premier car il est divisible par un autre nombre que 1 et lui-même. Et donc le programme renvoie, “erreur: input a prime number” dont la traduction, “erreur: entrez un nombre premier”.
➔Ligne 6, ferme le programme pour le recommencer.

Si *n* est bien un nombre premier(ligne 7), alors le programme continue.

➔Ligne 8, on vient définir *p* qui est égal à $n - 3$.

Les instructions qui suivent (ligne 9 à 15), nous permettent de déterminer si *p* est divisible par 4 et donc si *n*, un nombre premier est bien congrus à 3 modulo 4.

- ➔ Ligne 9, on vient définir une condition. Si le reste de la division de *p* par 4 est égal à 0.
- ➔ Ligne 10, si la condition établie de la ligne 9 est respectée alors le programme nous renvoie, "*n* it is a prime number congruent to 3 modulo 4" dont la traduction, "*n* est un nombre premier congrus à 3 modulo 4".
- ➔ Ligne 11, si la condition de la ligne 9 n'est pas pas respectée.
- ➔ Ligne 12, alors le programme renvoie, "*n* it is not a prime number congruent to 3 modulo 4" dont la traduction, "*n* n'est pas un nombre premier congrus à 3 modulo 4".

```
1  from math import *

2  n = input("input a prime nuber")#input a primpe number

3  for j in range(2, int(n)-1):#do a list numbers between 2 and n-1
4      if (int(n) % j) == 0:# if quotient is equal to 0
5          print("erreur: input a prime number")# if n is nota prime number
6          exit()
7  else:# if n is a prime number
8      p = int(n)-3#it is this number must be divisible by 4
9      if (p % 4) == 0:# if quotient is equal to 0
10         print(n,"is a prime number congruent to 3 modulo 4")
11     else:
12         print(n,"is not a prime number congruent to 3 modulo 4")
```

Remerciements

J’aimerais remercier mon lycée, Emmanuel D’Alzon, ainsi que les deux enseignants qui nous ont encadrés pendant cet atelier, M. Castagnedoli et Mme. Fort.
J’aimerais aussi remercier mon grand-père pour m’avoir expliqué les congruences mais aussi contrôlé et rectifié mes recherches et mon travail au fil de mon avancée.
J’aimerais remercier les membres de l’association de MATh.en.JEANS pour avoir créés et fait vivre cet atelier qui nous permet de mener nos recherches.
J’aimerais remercier la faculté des sciences de Montpellier pour nous avoir accueilli dans leurs locaux pour qu’on présente nos recherches .

Références

Graven-Développement, python:
https://www.youtube.com/watch?v=psaDHhZ0cPs&t=16s
Les bons profs, congruence:
https://www.youtube.com/watch?v=tTdHlpFERVQ
Yvan Monka, la congruence:
https://www.youtube.com/watch?v=uMSNllPBFhQ