COMPTE-RENDU TP4

Université Paris 13

L2-Mathématiques

Maths AP

15 janvier 2018

NAIT-DAOUD Mohamed

Exercice 1

La division euclidienne est un outil très utile en arithmétique. Parmi ses intérêts il y a le fait de déterminer si un nombre est premier ou non. Un nombre premier est un entier supérieur ou égal à deux qui n'a pour diviseur que un et lui-même. Ainsi, en prenant un entier « *p* » au hasard, pour déterminer s'il est premier ou non, nous pourra regarder l'ensemble des diviseurs possibles inférieurs à sa racine carré puis on regardera le reste de *p* par ces diviseurs.
Sur ce principe, on a définie une fonction *« is_prime »* qui nous a permit de dire si oui ou non les nombres 1001, 2017, 3001, 49999 et 89999 sont premiers.
Ainsi on a le tableau suivant:

	1001	**2017**	**49 999**	**89 999**
Est-il premier ?	Non	Oui	Oui	Non

Ensuite on a travaillé sur les cinq premiers nombres de Fermat définis comme suit :
$F(n) = 2^{2^{**}n}+1$ pour *n* un entier naturel.

	F(0)	**F(1)**	**F(2)**	**F(4)**	**F(5)**
Est-il premier ?	Oui	Oui	Oui	Oui	Non

D'autres méthode, qui ne se base pas explicitement sur le reste d'un entier via la division euclidienne, permettent de déterminer des nombres premiers.

Exercice 2

Le théorème fondamental de l'arithmétique nous dit que tout entier *N* strictement positif admet une unique décomposition en facteur de nombres premier. Ainsi, il nous apparaît une utilité à en déterminer la liste des nombres premiers inférieur à *N*.
Le crible d'Ératosthène est une méthode permettant de déterminer une telle liste.
D'après Wikipédia, il s'agit de supprimer d'une table des entiers de 2 à *N* tous les multiples d'un entier. En supprimant tous les multiples, à la fin il ne restera que les entiers qui ne sont multiples d'aucun entier, et qui sont donc les nombres premiers.
Par exemple, pour *200* on a *46* nombres premiers qui lui sont inférieurs.

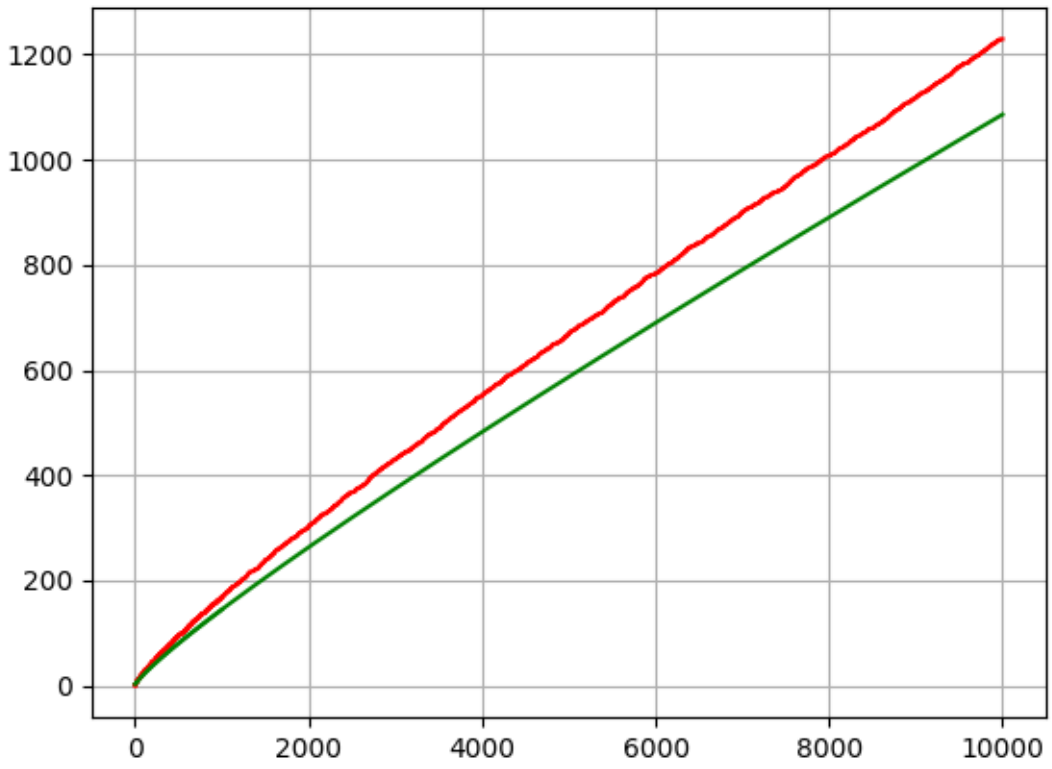
Tableau des nombres premier inférieur à 200 :

2	3	5	7	11
13	17	19	23	29
31	37	41	43	47

53	59	61	67	71
73	79	83	89	97
101	103	107	109	113
127	131	137	139	149
151	157	163	167	173
179	181	191	193	197
199				

Le problème de cette méthode est quelle perd son utilité pour des nombres premiers suffisamment grand.
Nous avons donc codé en python une fonction *Prime* qui renvoie la liste des nombres premier. Elle parcours tous les entiers entre 2 et *N* et lorsque la fonction *is_prime* renvoie « *TRUE* », alors elle ajoute cet élément à la liste.
Pour avoir une idée de l'évolution de cette liste selon le *N choisit*, on pourra se ramener à l'étude de la fonction *n/log(n)* comme nous l'indique le théorème des nombres premiers. En effet en prenant pi(n) = card(I) avec I = {1<k<n | div(k)={1,k}} on remarque une similitude entre les deux fonctions:

<u>Graphique de pi(n) et n/ln(n) en fonction de n</u>



En rouge, la courbe représentative de *pi(n)* et en vert la courbe représentative de *n/ln(n).*

Tableau de quelques valeurs prises par *pi(n)* et *n/ln(n)*

n	**pi(n)**	**n/log(n)**
10	4	4,34294
100	25	21,7147
1000	35	144,765
10 000	1229	1085,74
100 000	9592	8685,89
1 000 000	78498	72382,4

Ainsi on remarque ici que la quantité de nombres premiers augmente de manière logarithmique. Cela nous permet ainsi de conjecturer par exemple qu'il y a une infinité de nombre premier d'une part ; d'autre part cela permet de comprendre que l'accroissement de la quantité n'est pas très rapide et ainsi de supposer que trouver des nombres premiers très grand est « plus rare » ou plus difficile.
D'autres méthodes nous permettent de vérifier qu'un nombre est premier ou non comme la décomposition en facteurs premier bien qu'un de ses rôles essentiellement connus aujourd'hui a été de montrer qu'il existait une infinité de nombre premier.

Exercice 3

D'après le théorème fondamental de l'arithmétique, tout entier positif et non nul peut être écrit comme un produit de nombre premiers et de façon unique.
Exemple :
$924 = 2^2*3*7*11$

Afin de déterminer la décomposition en nombre premier d'un entier *N* on récupère la liste des nombres premiers qui lui sont inférieur d'une part. Puis, on parcours cette liste dans l'ordre croissant en testant si un parmi cette liste divise N. Si oui alors on réitère l'opération autant que possible jusqu'à ce qu'il ne divise plus le reste obtenu à chaque action. Ensuite, quand ça ne fonctionne plus on change d'entier jusqu'à parcourir toute la liste des premiers inférieurs a *N*.
Ainsi, si un nombre *p* à la décomposition *p = p*1 alors il sera premier.*

code python : obtenir une décomposition en facteur premier d'un entier N

```
def factors(n):
```

```
P = []
D = primes(n)
p = D[0]
l = int(sqrt(n))
N = n
for i in range(1,l+1):
        if (N % p == 0):
                while(N % p == 0):
                        N = N/p
                        P.append(p)

        p = D[i]


return P
```

Toutefois, cette méthode peut s'avérer très longue si le N choisit est suffisamment grand.
Il arrive parfois que ce qu'il nous intéresse n'est pas la primalité d'un entier mais la primalité entre deux entiers.

Exercice 4

Soit a et b deux entiers. On appel *pgcd(a,b) = d*, l'entier telle que pour tout *x* dans *div(a,b)*, *x* est inférieur à *d* et *d|a et d|b.*
Pour trouver *d* on peut s'aider de la décomposition en facteur premier de *a* et *b.*

Exemple :
a = 4864 = 2^8*19
b = 3458 = 2*7*13*19
pgcd(a,b) = 19

cela nous amène à l'évocation d'un certain théorème connu comme étant *l'identité de Bézout.* Il dit la chose suivante : soit a et b des entiers relatifs. Il existe un couple d'entiers relatifs (x,y) tel que :
ax+ by = pgcd (a,b)

Ainsi, en trouvant des x et y tel que ax + by = 1 on aura alors que pgcd(a,b)=1 et on pourra dire que a et b sont premiers entre eux.

Exemple : Reprenons le a et le b de l'exemple précédent.
4864 = 1* 3458 + 1406
3458 = 2*1460 + 646

1460 = 2*646 + 168
646 = 3* 168 + 142
168 = 1*142 + 26
142 = 5*26 + 12
26 = 2*12 + 2
12 = 6*2 + 0
pour trouver u et v il faut remonter le calcul en partant de la dernière égalité. Ici, on voit que le dernier reste non nul est deux ce qui indique que a et b ne sont pas premiers entre eux.

Cela peut s'avérer très long à faire à la main, ainsi on a coder en python une fonction nous permettant de déterminer *a,b, u* et *v.* Étant donné que nous étions intrigué par une erreur de notre code, mr.Cardinal, afin de nous aider, nous a indiquer un code disponible sur *wikipédia*.

Exercice 5

De part la longueur et la difficulté de l'exercice, l'enseignant nous a conseiller de ne pas nous lancer dans cet exercice au vu du peu de temps nous restant pour cela.

Conclusion

On a put observer différentes méthodes de test de primalité sur un entier seul ou entre deux entiers. Les nombres premiers ont une incidence particulière en cryptographie qui se révèle être très utile. Le chiffrement RSA en est un exemple d'application.