Lycée Victor Hugo Spé PC* 24/25

- ITC PC* : Devoir Maison n° 1 -

à rendre le 16/9/24

Ce devoir est à rendre sous forme d'un fichier Python nommé `NomPrénomDM1.py` via la plateforme Moodle (Eclat puis Moodle puis cours nicolas Mary puis restitution travaux pc* puis restitution DM1) avant le 16 sept. 2024 23h59.

Il est demandé de compléter le fichier `trameDM1.py` et donc ensuite de le renommer. On pensera bien à insérer dans le fichier les tests et la documentation succincte de chacune des fonctions.

Si vous avez besoin d'un peu d'aide, ne pas hésiter à me demander par mail ou au lycée.

CONJECTURE DE GOLDBACH

I Préliminaire

1. Écrire une fonction `occurrences(L:list)->list` qui, étant donné une liste d'entiers naturels `L`, renvoie la liste des occurrences de chacun des entiers compris entre 0 et le maximum de `L`.

```
[In] occurences([1,1,5,2,1])
[Out] [0,3,1,0,0,1]
```

La création des occurrences peut être améliorée. D'une part, il est possible que certaines valeurs n'apparaissent pas dans la liste, et on stocke inutilement le nombre de leurs occurrences(qui est nul). De plus, il est nécessaire de chercher le maximum de la liste. Enfin, la fonction précédente ne convient que pour des listes d'entiers.

Pour éviter ces écueils, nous allons faire appel à la structure de dictionnaire.

2. Écrire une fonction `occurrences_dico(L:list)->dict` permettant de créer le dictionnaire ayant pour clés les valeurs distinctes de la liste `L` et pour valeurs le nombre d'occurrences de ces valeurs distinctes.

II Nombres Premiers

On rappelle qu'un nombre entier $n \geqslant 2$ est premier lorsqu'il n'est divisible que par deux entiers naturels : 1 et par lui-même.

La méthode du Crible d'Erathostène permet de rechercher des nombres premiers. Elle consiste à écrire tous les nombres entiers inférieurs ou égaux à n et à rayer successivement tous les multiples des nombres entiers $k \in [|2, \lfloor\sqrt{n}\rfloor|]$. A la fin, les nombres non rayés sont exactement les nombres premiers inférieurs ou égaux à n.

3. En se basant sur la méthode du crible, écrire une fonction `test_premier(n:int)->list` qui prend un entier $n \geqslant 1$ et qui qui renvoie une liste T de booléens telle que :

   pour tout entier $k \in [|0, n|]$, $T[k]$ vaut $True$ si et seulement si k est premier.

4. Écrire une fonction `premiers(n:int)->list` qui prend en entrée un entier $n \geqslant 1$ et qui renvoie en sortie la liste des nombres premiers inférieurs ou égaux à n

5. On note $\pi(n)$ le nombre d'entiers inférieurs ou égaux à n qui sont premiers.

   Par exemple, $\pi(5) = 3$ et $\pi(10) = 4$.

   Écrire une fonction `compte_premiers(n:int)->int` qui prend un entier $n \geqslant 1$ et renvoie une liste P telle que pour tout entier $k \in [|0, n|]$, $P[k] = \pi(k)$.

   Ici, seul le recours à la fonction `test_premier` est autorisé.

6. Le théorème des nombre premiers, conjecturé par Gauss et Legendre à la fin du 18^e siècle et démontré un siècle plus tard par Hadamard et De La Vallée Poussin, établit que :

$$\pi(n) \underset{n\to+\infty}{\sim} \frac{n}{\ln(n)} \qquad \text{c'est à dire} \qquad \frac{\pi(n)\ln(n)}{n} \underset{n\to+\infty}{\longrightarrow} 1$$

   Illustrer graphiquement cette dernière convergence et commenter votre illustration.

III Conjecture de Goldbach

La conjecture de Goldbach, énoncée en 1742, prétend que tout entier pair supérieur ou égal à 4 peut s'écrire comme la somme de deux nombres premiers. Il s'agit d'une des conjectures les plus célèbres et des plus difficiles de la théorie des nombres. Aucune preuve ni aucun contre-exemple n'a été trouvé à ce jour.

Nous allons tenter de la vérifier pour certaines valeurs.

7. Écrire une fonction `goldbach(n:int)->list` qui prend en entrée un un entier $n \geqslant 1$ et renvoie une liste T de booléens telle que pour tout entier $k \in [|0, n|]$,

   $T[k]$ vaut $True$ si et seulement si $2k$ est la somme de deux nombres entiers.

   On pourra utiliser la donnée `l=premier(2n)` et stocker dans une liste les valeurs $l(i) + l(j)$ lorsque $i \geqslant j$.

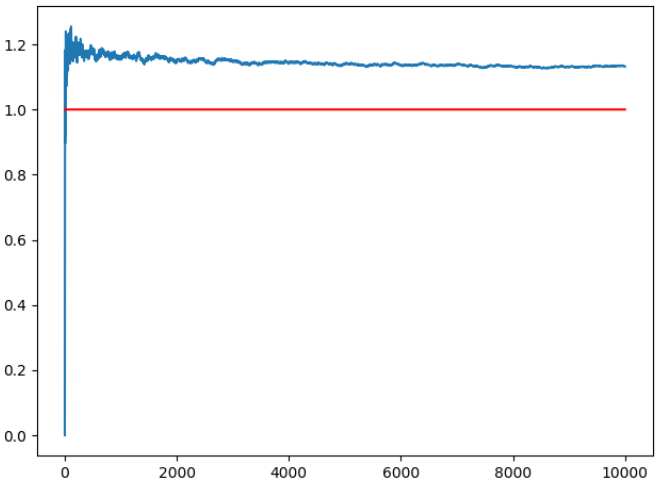
8. Écrire une fonction `comete_goldbach(n:int)->list` qui prend en entier un entier $n \geqslant 1$ et renvoie une liste C telle que pour tout entier $k \in [|0, n|]$, `C[k]` soit égal au nombre de façons différentes d'écrire l'entier $2k$ comme somme de deux nombres premiers.

9. Tracer ensuite, sans relier les points, le graphe de la fonction $k \mapsto C(k)$ pour $k \in [|0, 2000|]$ et observer la fameuse comète de Goldbach ! ! ! !

CORRECTION

```python
##1

def occurences(L:list)->list:
    '''entrée : liste d'entiers naturels
    sortie : liste des occurrences'''
    m=L[0]
    for i in range(len(L)):
        if L[i]>m:
            m=L[i]
    result=[0]*(m+1)
    for e in L:
        result[e]+=1
    return result

#test
print(occurences([1,2,3,4,4,4,4,6,6,9]))  #rép [0, 1, 1, 1, 4, 0, 2, 0, 0, 1] ok


##2

def occurences_dico(L:list)->dict:
    '''entrée : liste d'entiers naturels
    sortie : dictionnaire des occurrences'''
    result={}
    for e in L:
        if e in result:
            result[e]+=1
        else:
            result[e]=1

    return result

print(occurences_dico([1,2,3,4,4,4,4,6,6,9])) #rép {1:1, 2:1, 3:1, 4:4, 6:2, 9:1}

##3

def test_premier(n:int)->list:
    '''entrée : un entier naturel n
    sortie : liste des positions des nombres premiers entre 0 et n '''

    L=[1]*(n+1)
    L[0]=0
    L[1]=0
    for i in range(2,int((n)**(1/2)+1)):
        if L[i]==1:
            j=2*i
            while j<=n:
                L[j]=0
                j=j+i
    return L

print(test_premier(10))  #rép : [0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0] ok
```

```
# test_premier(10**7) prend qques secondes
# test_premier(10**8) reste encore raisonnable (1minute)


##4

def premiers(n:int)->list:
    '''entrée : un entier n
    sortie : liste des nbres premiers inférieurs ou égaux à n'''
    liste=test_premier(n)
    result=[]
    for i in range(n+1):
        if liste[i]==1:
            result.append(i)
    return(result)

print(premiers(40))   #rép [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37]


##5 uniquement avec la fct test_premier

def compte_premiers(n:int)->list:
    '''entrée: un entier n
    sortie: liste des valeurs de la fonction pi'''
    result=[0,0]
    liste=test_premier(n)
    compteur=0
    for i in range(2,n+1):
        if liste[i]==1:
            compteur=compteur+1
        result.append(compteur)
    return result

print(compte_premiers(10)) #rép [0, 0, 1, 2, 2, 3, 3, 4, 4, 4, 4]


##6

import matplotlib.pyplot as plt
from math import *

n=10000

plt.figure('th nb premiers')
temp=compte_premiers(n)
X=[i for i in range(1,n+1)]
Y=[temp[i]*log(i)/i for i in range(1,n+1)]
Z=[1 for i in range(1,n+1)]

plt.plot(X,Y)
plt.plot(X,Z,'r')
plt.show()

'''on observe une convergence extrêmement lente vers 1'''
```




```python
## 7

def goldbach(n:int)->list:
    '''entrée : un entier n
    sortie : liste T dont T[k] est vrai ssi 2k vérifie la conjecture'''

    l=premiers(2*n)
    T=[False]*(n+1)
    Liste_somme=[]
    for i in range(len(l)):
        for j  in range(i,len(l)):
            Liste_somme.append(l[i]+l[j])
    for k in range(n+1):
        if 2*k in Liste_somme:
            T[k]=True
    return T

print(goldbach(10))
#rép [False, False, True, True, True, True, True, True, True, True, True] ok

##8-9

def comete(n:int)->list:
    '''entrée : un entier n
    sortie : liste C tq C[k]=nombre de possibilités d'écrire 2k
    comme somme de deux nbres premiers'''
    p=premiers(2*n)
    T=[0]*(n+1)
    Liste_somme=[]
    for i in range(len(p)):
        for j  in range(i,len(p)):
            Liste_somme.append(p[i]+p[j])

    D=occurences_dico(Liste_somme)

    for k in range(n+1):
        if 2*k in D:
            T[k]=D[2*k]
    return T
```

```
40
41 C=comete(2000)
42
43
44 plt.figure('comete')
45 plt.clf()
46 plt.plot(C,'.',ms=3)
47 plt.show()
```