SÉANCE 2. AUTOUR DES NOMBRES PREMIERS

Objectifs : prendre en main Python. *Manipuler la notion de nombre premier et étudier leur répartition.*

1. Décomposition en produit de facteurs premiers.

Rappelons qu'un entier naturel $p \geq 1$ est premier si et seulement s'il admet exactement deux diviseurs (distincts) : 1 et lui-même.

(a) Soit $n \geq 1$ un entier. Montrer que s'il existe deux entiers naturels a et b tels que $n = ab$, alors soit $a \leq \sqrt{n}$, soit $b \leq \sqrt{n}$.

(b) (**Théorème fondamental de l'arithmétique**). Montrer que tout entier $n \geq 1$ s'écrit comme produit de nombres premiers et que cette décomposition est unique à l'ordre des facteurs près.

Indication : Pour l'existence de la décomposition, on procédera par récurrence sur n. Pour l'unicité, on pensera à utiliser le lemme d'Euclide : si un nombre premier p divise un produit ab, alors p divise a ou p divise b.

(c) Montrer qu'il existe une infinité de nombres premiers.

Indication : Raisonner par l'absurde et considérer $p_1 p_2 \cdots p_k + 1$ où $p_1, \ldots, p_k$ sont premiers.

(c) Déduire des questions (a) et (b) un algorithme `Decompo` prenant en entrée un entier $n \geq 1$ et donnant un sortie la liste de ses facteurs premiers. Implémentez cet algorithme en Python. On pourra penser à coder une fonction intermédiaire `mindiv` renvoyant le plus petit diviseur premier d'un entier n.

Indication : On pourra éventuellement commencer par définir un algorithme `mindiv` produisant le plus petit diviseur $d > 1$ d'un entier $n \geq 1$.

(d) Implémentez en Python un algorithme `ProduitListe` prenant en entrée une liste de nombres entiers et renvoyant leur produit. Vérifiez que pour des valeurs arbitraires de n on a bien `ProduitListe(Decompo(`n`))` $= n$.

(e) En utilisant les algorithmes `Decompo` (ou `mindiv`) et `ProduitListe`, écrire un algorithme `Euclide` prenant en entrée un entier $n \geq 1$ et renvoyant une liste de n nombres premiers produits par le raisonnement de la question (c). Implémenter cet algorithme en Python.

2. Crible d'Eratosthène.

Soit $N \geq 1$ un entier. Le crible d'Eratosthène est un algorithme dont le but est de lister tous les nombres premiers inférieurs ou égaux à N.

- On commence par lister tous les entiers compris entre 2 à N (rappelons que 1 n'est pas premier). On modifie cette liste étape par étape.
- On commence par éliminer tous les multiples de 2 strictement supérieurs à 2. On obtient une liste tronquée $2, 3, 5, 7, 9, \ldots, N$.
- L'entier suivant 2 dans la liste modifiée est 3. On élimine alors tous les multiples de 3 différent de 3. On obtient une nouvelle liste tronquée $2, 3, 5, 7, 11, 13, 17, \ldots, N$.
- On supprime ensuite tous les multiples de 5 (l'entier suivant 3 dans la nouvelle liste) qui lui sont strictement supérieurs. Et ainsi de suite.

- La liste finale est constituée de tous les nombres premiers compris entre 2 et N.

(a) Justifier que l'algorithme d'Eratosthène renvoie la liste des nombres premiers entre 2 et N.

(b) Justifier que dans le Crible d'Eratosthène on peut s'arrêter lorsque le carré de l'entier p dont on veut supprimer les multiples est strictement supérieur au plus grand entier restant, autrement dit, que tous les entiers $> p$ restant dans la liste sont des nombres premiers.

(c) Ecrire en Python un algorithme itératif `crible` prenant en entrée la borne N et renvoyant un tableau d de taille $n+1$ tel que $d[i] = i$ si i est premier et 0 sinon.

(d) Ecrire en Python un algorithme récursif `cribleRec` prenant en argument n et renvoyant la liste des nombres premiers compris entre 2 et n. Pour cela, on pourra créer les fonctions intermédiaires suivantes :

— `supprMult` qui prend un argument un entier p et une liste L et renvoie la liste L privée des multiples de p (utiliser la commande $L.remove(i)$ pour supprimer la valeur i de la liste L)

— `cribleRecListe` qui prend un argument une liste d'entiers L et qui renvoie la liste d'entiers obtenus en appliquant le crible à partir de L.

(e) En utilisant l'algorithme implémenté en (c), créez un nouvel algorithme `PiIteratif` prenant en entrée un entier N et renvoyant le nombre de nombres premiers $\leq N$.

(f) En vous inspirant de l'algorithme implémenté en (d), coder un algorithme récursif `PiRecursif` prenant en entrée un entier N et renvoyant le nombre de nombres premiers $\leq N$.

(g) Deux nombres premiers $p < q$ sont dits *jumeaux* si $q = p+2$. Programmer une fonction `Jumeaux`, qui associe à un entier $n \geqslant 1$ la liste des nombres premiers jumeaux inférieurs à n.

3. La fonction de comptage des nombres premiers.

Pour tout réel $x \geq 2$, on définit $\pi(x)$ comme le nombre de nombres premiers $p \leq x$.

1. Représenter graphiquement $\pi(x)$ pour x compris entre 2 et 100 (resp. 1000, resp. 20000).
2. Comparer la courbe obtenue avec celle des fonctions $x \mapsto x$ et $\mapsto \sqrt{x}$. Qu'observez-vous ?
3. On désigne ici par log la fonction logarithme népérien. Calculer

$$\frac{\pi(10^n)}{10^n/\log(10^n)}$$

pour $n \in \{1,\dots,7\}$. Qu'observez-vous ?

4. Sur le même graphique, représenter $\pi(x)$ et $\frac{x}{\log x}$ pour x entre 2 et 20000.
5. A la question précédente : pourquoi préfère-t-on utiliser la fonction `crible` plutôt que la fonction `PiRecursif` ?
6. Tracez $\pi(x)\log(x)/x - 1$ pour x entre 2 et 10000.

C.F. Gauss et A.M. Legendre ont conjecturé à la fin du XVIIIe siècle que l'on a

$$\pi(x) \sim \frac{x}{\log(x)}$$

lorsque x tend vers $+\infty$. Ceci a été démontré à la fin du XIXe siècle par J. Hadamard et C. de la Vallée Poussin.

4. Estimations de $\pi(x)$ - Inégalités de Chebychev (majoration).

Le but de cette question est de montrer que

$$\pi(n) \leq c_1 \frac{n}{\log(n)} \quad \text{avec } c_1 = 6\log(2). \tag{1}$$

(remarque : ici log désigne le logarithme népérien). On fixe un entier $n \geq 1$. La lettre p désigne toujours un nombre premier.

(a) Montrez que tout nombre premier p vérifiant $n < p \leq 2n$ divise $\binom{2n}{n}$.

(b) En déduire que $n^{\pi(2n)-\pi(n)} \leq \prod_{n<p\leq 2n} p \leq \binom{2n}{n}$. On pourra commencer par étudier le nombre de facteurs du produit central.

(c) Montrez par récurrence sur n que $\binom{2n}{n} \leq 4^n$ pour tout $n \geq 1$. En déduire que $\pi(2n) - \pi(n) \leq 2\log 2 \times n/\log(n)$.

(d) Prouvez par récurrence sur k que $\pi(2^{k+1}) \leq 2^k$ pour tout $k \geq 0$.

Indication. Tout nombre premier > 2 est impair.

(e) En utilisant la question (c) avec $n = 2k$, montrer que pour tout $k \geq 0$ on a $(k+1)\pi(2^{k+1}) - 2^k\pi(2^k) \leq 3 \times 2^k$.

(f) En déduire que pour tout $n \geq 0$, on a $(n+1)\pi(2^{n+1}) \leq 3 \times 2^{n+1}$.

(g) Soit $x \geq 2$ un réel et n l'unique entier ≥ 1 vérifiant $2^n \leq x < 2^{n+1}$. Exprimez n en fonction de x, donnez une minoration de $n+1$ en fonction de x, puis utiliser la question précédente pour en déduire (1).

4. La formule de Legendre

P.Chebychev a également établi (en 1850) une minoration $c_2 \frac{n}{\log(n)} \leqslant \pi(n)$. Une première étape dans cette direction est l'étude de la factorisation de $n!$, qui fait l'objet des questions suivantes.

Etant donné un nombre entier $n \geq 1$, et un nombre premier p, on note $v_p(n)$ la valuation p-adique de n définie comme étant le plus grand entier v tel que p^v divise n.

(a) Quelle est la valuation 5-adique de $9625 = 5^3 \times 7 \times 11$?

(b) Ecrire un algorithme en python `valuation` prenant en entrée un entier n et un nombre premier p, et qui renvoie $v_p(n)$.

(c) Montrer qu'étant donnés des entiers $a, b \geq 1$ et un nombre premier p, on a $v_p(ab) = v_p(a) + v_p(b)$.

(d) Le but de cette question est de montrer que pour tout $n \geq 1$ et tout nombre premier p, on a (formule de Legendre)

$$v_p(n!) = \sum_{k\geq 1} \left\lfloor \frac{n}{p^k} \right\rfloor. \tag{2}$$

Ici, on désigne par $\lfloor x \rfloor$ la *partie entière* d'un réel x, définie comme étant le plus grand entier m vérifiant $m \leq x < m+1$.

(a) Pour $k \geq 0$, on note Ω_k l'ensemble des entiers m compris entre 1 et n et tels que $v_p(m) = k$. Justifier que Ω_k est l'ensemble vide pour $k > \log(n)/\log(p)$.

(b) Montrer que les Ω_k forment une partition de $\{1, \ldots, n\}$: autrement dit, pour tout entier m compris entre 1 et n il existe k tel que $m \in \Omega_k$, et les Ω_k sont deux à deux disjoints.

(c) Déduire des questions précédentes que

$$v_p(n!) = \sum_{k\geq 1} (\#\Omega_k) \times k.$$

(d) Justifiez que le nombres de multiples de p^k inférieurs à n est $\lfloor n/p^k \rfloor$.

(e) En déduire une expression pour $\#\Omega_k$, puis prouvez (2).