TP4

RAJA GANAPATHY Srinivas ENGUIX Précillia

December 21, 2017

1 Les nombres premiers

En Python, le quotient et le reste de la division euclidienne d'un entier par un entier s'obtient de la façon suivante: pour le quotient on utilise a/b et pour le reste on cherche son modulo. Un nombre premier est un entier naturel qui admet exactement deux diviseurs distincts entiers et positifs (qui sont alors 1 et lui-même). Ainsi, 1 n'est pas premier car il n'a qu'un seul diviseur entier positif et 0 non plus car il est divisible par tous les entiers positifs.
Parmi les nombres suivants : 1001, 2017, 3001, 49999, 89999, les nombres qui sont premiers sont les suivants : 2017, 3001, 49999.
Nous avons écrit une fonction is prime qui prend en argument un entier n et qui renvoie true si n est premier et false si n n'est pas premier. Les nombre de Fermat F_0, F_1, F_2, F_3, F_4 sont premiers mais F_5 n'est pas premier.

2 Crible d'Erasthotène, distribution des nombres premiers

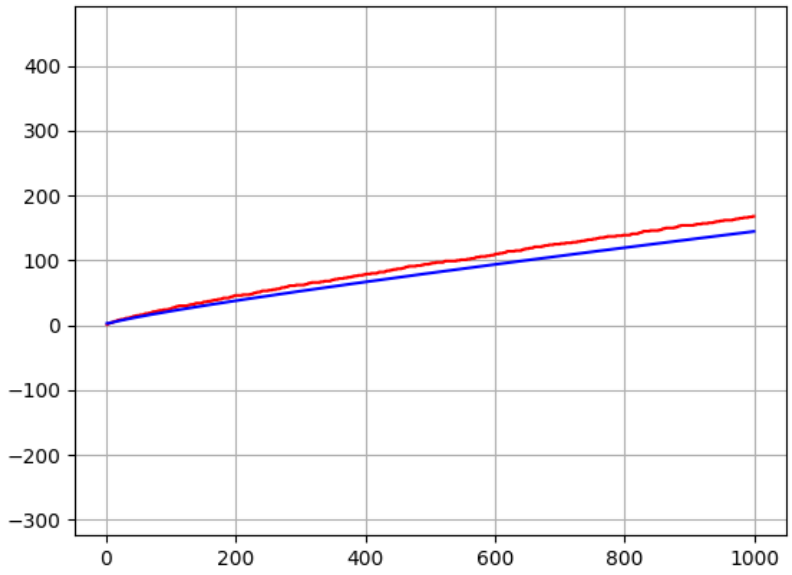
Nous avons calculé à l'aide du crible d'Erasthostène, la liste de tous les nombres premiers inférieurs à 200. Voici le tableau représentant :

1	**2**	**3**	4	**5**	6	**7**	8	9	10
11	12	**13**	14	15	16	**17**	18	**19**	20
21	22	**23**	24	25	26	27	28	**29**	30
31	32	33	34	35	36	**37**	38	39	40
41	42	**43**	44	45	46	**47**	48	49	50
51	52	**53**	54	55	56	57	58	**59**	60
61	62	63	64	65	66	**67**	68	69	70
71	72	**73**	74	75	76	77	78	**79**	80
81	82	**83**	84	85	86	87	88	**89**	90
91	92	93	94	95	96	**97**	98	99	100
101	102	**103**	104	105	106	**107**	108	**109**	110
111	112	**113**	114	115	116	117	118	119	120
121	122	123	124	125	126	**127**	128	129	130
131	132	133	134	135	136	**137**	138	**139**	140
141	142	143	144	145	146	147	148	**149**	150
151	152	153	154	155	156	**157**	158	159	160
161	162	**163**	164	165	166	**167**	168	169	170
171	172	**173**	174	175	176	177	178	**179**	180
181	182	183	184	185	186	187	188	189	190
191	192	**193**	194	195	196	**197**	198	**199**	200

L'algorithme procède par élimination : il s'agit de supprimer d'une table des entiers de 2 à 200 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. On commence par rayer les multiples de 2, puis à chaque fois on raye les multiples du plus petit entier restant. À la fin du processus, tous les entiers qui n'ont pas été rayés sont les nombres premiers inférieurs à 200.

Suite à ça, on a écrit une fonction python primes qui prend en argument un entier n et qui renvoie la liste de tous les nombres premiers inférieurs à n. Puis on a utilisé cette fonction pour calculer la liste de tous les nombres premiers inférieurs à 1000, on a écrit cette liste dans un fichier primes.txt. La liste sera affichée dans le terminal et dans le fichier texte (10 nombres par ligne).

On note $\pi(n)$ le nombre d'entiers premiers inférieurs à n, puis on a représenté graphiquement $\pi(n)$ en fonction de n pour n variant de 2 à 1000. Sur le même graphique, on a rajouté la fonction n / log(n). Voici le graphique ci-dessous :



Pour pouvoir interpréter le graphique, nous devons s'aider du théorème des nombres premiers, le voici :

Théorème des nombres premiers — Le nombre $\pi(x)$ de nombres premiers inférieurs ou égaux à x est équivalent, lorsque le réel x tend vers $+\infty$, au quotient de x par son logarithme népérien. Soit

$$\pi(x) \sim \frac{x}{\ln(x)} \ (x \to +\infty),$$

c'est-à-dire

$$\lim_{x \to +\infty} \pi(x) \frac{\ln(x)}{x} = 1.$$

Donc d'après le théorème des nombres premiers et du graphique, on peut dire que π(n) et la fonction n / log n, sont équivalents.

Nous avons ensuite crée en python la table donnée dans l'énoncé, nous l'avons remplie et écrit dans un fichier texte, la voici ci-dessous :

n	pi(n)	n/log n
10	4	4.34294
100	25	21.7147
1000	168	144.765
10000	1229	1085.74
100000	9592	8685.89
1000000	78498	72382.4

3 Factorisation d'un entier en premiers

En mathématiques, et en particulier en arithmétique élémentaire, le théorème fondamental de l'arithmétique ou théorème de décomposition en produit de facteurs premiers s'énonce ainsi : tout entier strictement positif peut être écrit comme un produit de nombres premiers d'une unique façon, à l'ordre près des facteurs.
Un nombre composé est un entier naturel différent de 0 qui possède un diviseur positif autre que 1 ou lui-même. La décomposition en facteurs premiers de 924 est 2**2 x 3 x 7 x 11.
Suite à ces recherches, nous avons écrit une fonction python factors qui prend en argument un entier n et qui renvoie la liste, dans l'ordre croissante, des facteurs premiers de n, chaque facteur étant répété autant de fois que nécessaire. Ainsi, nous avons fait le test pour n = 60, n = 80 et n = 30.

4 PGCD de deux entiers, identité de Bézout, algorithme d'Euclide

En arithmétique élémentaire, le plus grand commun diviseur ou pgcd de deux nombres entiers non nuls est le plus grand entier qui les divise simultanément. Pour calculer le pgcd, nous sélectionnons les facteurs communs, ensuite nous

effectuons le produit. Ici, 4864 = 2**8 x 19 et 3458 = 2 x 7 x 13 x 19 donc pgcd(4864, 3458) = 38.

Voici l'identité de Bézout :

Théorème de Bachet-Bézout (ou Identité de Bézout[6]**)** — Soient *a* et *b* deux entiers relatifs. Si *d* est le PGCD de *a* et *b*, alors il existe deux entiers relatifs *x* et *y* tels que $ax + by = d$.

Théorème de Bézout[6] — Deux entiers relatifs *a* et *b* sont premiers entre eux (si et) seulement s'il existe deux entiers relatifs *x* et *y* tels que $ax + by = 1$.

Claude-Gaspard Bachet de Méziriac (1581-1638).

L'algorithme d'Euclide permet de calculer le pgcd de deux entiers naturels non nuls a et b. On procède de la manière suivante : On effectue la division euclidienne de a par b. On note r le reste (on n'utilise pas le quotient). On remplace ensuite a par b et b par r. Tant que le reste est différent de 0, on réitère le procédé. Après un certain nombre d'itérations, on obtiendra un reste égal à 0. Le pgcd de a et de b est alors le reste précédent (c'est à dire le dernier reste non nul).

A l'aide de l'algorithme d'Euclide étendu, le pgcd des 2 nombres a = 4864 et b = 3458 est 34. Pour obtenir les coefficients de Bézout, il suffit de remonter l'algorithme d'Euclide à l'envers, on trouve alors les coefficients de Bézout : x = 32 et y = -45.

Nous avons écrit une fonction python euclide qui prend en arguments 2 entiers a et b, qui renvoie x, y, d, où d est le pgcd de a, b et x, y les coefficients de Bézout. On obtient d = 38, u = 32, et v = -45.

5 Chiffrement RSA

Le chiffrement RSA est un algorithme de cryptographie asymétrique, très utilisé dans le commerce électronique, et plus généralement pour échanger des données confidentielles sur Internet.

Nous n'avons pas eu le temps de poursuivre le reste de la partie RSA, car nous devions commencer le projet pour le partiel.

6 Analyse de la séance de TP

Le début du TP a été assez simple, nous avons manqué de temps pour la partie chiffrement RSA. Dommage, car c'était le TP le plus intéressant. Nous remarquons une forte évolution au niveau du codage python ainsi que du latex. C'est une matière très intéressant avec beaucoup de ressource.