Cryptographie 3
Arithmétique modulaire – Nombres premiers

La cryptographie asymétrique, ou à clef publique utilise essentiellement de l'arithmétique modulaire sur de grands entiers, et nécessite en particulier de grands nombre premiers. Python peut calculer sur des entiers arbitrairement longs :

```
>>> 2**512
13407807929942597099574024998205846127479365820592393377723561443721764030073546976801874298166903427690031858186486050853753882811946569946433649006084096L
>>>
```

L'opérateur modulo est le %. On veut pouvoir calculer des inverses dans $\mathbb{Z}/n\mathbb{Z}$.

Un nombre premier est un nombre qui n'admet comme diviseurs que 1 et lui-même. Ces nombres sont particulièrement appréciés en cryptographie du fait de l'arithmétique modulaire : $\mathbb{Z}/p\mathbb{Z}$ est un corps fini, tout élément posséde un inverse pour la multiplication, facilement calculable.

1. Écrire une fonction `gcd(a,b)` qui calcule le pgcd de deux entiers a et b.
2. Écrire une fonction `xgcd(a,b)` qui renvoie d, u, v où $d = ua + vb$ est le pgcd de a et b.
3. Écrire une fonction `inversemod(a, n)` qui renvoie l'inverse de a modulo n.
4. Écrire une fonction `primes(n)` qui renvoie la liste des nombres premiers inférieurs à n. On pourra utiliser le crible d'Eratosthène : Partir de deux listes $P = [2], M = [x|2 < x < n, x \not\equiv 0 \mod 2]$. A chaque étape, on insère $M[0]$ à la fin de la liste P, et on remplace M par la liste de ses éléments non divisibles par $M[0]$.
5. Test exact
   Écrire une fonction déterminant si un nombre est premier, en utilisant la recherche d'un diviseur. Etablir une liste des 200000 premiers nombres premiers.
   Sur ce seul critère, le nombre de divisions à effectuer est en $\mathcal{O}(n)$ ce qui est impossible pour des valeurs de n s'écrivant sur plusieurs centaines de bits.

6. Écrire une fonction `powermod(a,b,n)` qui calcule $a^b \mod n$ par l'algorithme rapide vu en cours.

7. *Test de Fermat.* – Pour diminuer le nombre d'opérations à effectuer, une première approche utilise le petit théorème de Fermat :
   Si p est premier et $\text{pgcd}(a,p) = 1$ alors $a^{p-1} = 1 \mod p$
   On choisit donc un entier $1 < a < p$ (donc premier avec p si p est premier) , puis on calcule $a^{p-1} \mod p$; si cette valeur n'est pas 1, on est sûr que p n'est pas premier.
   On recommence plusieurs fois. Si l'égalité est vérifiée pour toutes les valeurs de a testées, p est *probablement* premier. Établir une fonction de test de primalité puis la liste des 200000 premiers nombres premiers de Fermat en faisant varier le nombre de valeurs a testées.
   Comparez avec la liste précédente. On remarque que certains nombres ont été souvent reconnus comme premiers mais ne le sont pas.
   Vérifiez que 561 passe le test $a^{561} \equiv a[561]$ pour toutes les valeurs $1 < a < 561$, et vérifie $a^{560} \equiv 1[561]$ pour tous les a tels que avec $pgcd(a, 561) = 1$.

8. *Test de Miller Rabin.* – Cette approche utilise la structure de corps de $\mathbb{Z}/p\mathbb{Z}$: si p est premier. $\mathbb{Z}/p\mathbb{Z}^*$ est un groupe multiplicatif d'ordre $p-1$.
   Pour tout $1 < a < p$, $a^{p-1} = 1 \mod p$ et $a^{\frac{p-1}{2}}$ est une racine de l'équation $x^2 = 1$. Si p est premier, sa valeur est donc 1 ou $-1 (= p-1)$. Si cette valeur est 1, le même raisonnement s'applique à sa racine.
   On écrit donc $p - 1 = 2^s t$ avec t impair.Pour $1 < a < p - 1$ si p est premier avec p on a
   – ou bien $a^t = 1 \mod p$ (*on a trouvé 1 dès le début*)
   – ou bien il existe $i \in 0 \ldots s-1$ tel que $a^{2^i t} = -1 \mod p$ (*on ne trouve pas le premier* 1 *sans que sa racine carrée soit* -1)
   Écrire une fonction `miller_rabin(n, num_trials=4)` qui teste si un nombre n est premier par la méthode de Miller-Rabin.
   Utilisez la pour etablir une fonction de test de primalité puis la liste des 200000 premiers nombres premiers.

9. Le plus grand nombre premier connu est souvent un *nombre de Mersenne*. Ce sont les $M_p = 2^p - 1$. Montrez que si p n'est pas premier, M_p ne l'est pas non plus. Utilisez votre test de Miller-Rabin et votre liste de petits nombres premiers pour trouver un nombre de Mersenne (probablement) premier le plus grand possible.

10. Écrire une fonction `solve_chinese(yy,mm)` qui résout le système de conguences $x_i \equiv y_i \mod m_i$.
11. *Test de Solovay Strassen.* – Cette approche, utilise le symbole de Legendre $\left(\frac{a}{p}\right)$ qui peut être calculé en $\mathcal{O}(\log p)^2$ (de manière similaire à la recherche du pgcd). Défini pour un entier premier impair p :
    - $\left(\frac{a}{p}\right) = 0$ si $a = 0 \mod p$
    - $\left(\frac{a}{p}\right) = 1$ si $a =\neq 0 \mod p$ et il existe un entier $x, a = x^2 \mod p$
    - $\left(\frac{a}{p}\right) = -1$ si un tel x n'existe pas
    - $\left(\frac{a}{1}\right) = 1$

    il vérifie
    - Si $a = b \mod p$ alors $\left(\frac{a}{p}\right) = \left(\frac{b}{p}\right)$
    - $\left(\frac{a}{p}\right) = 0$ si $\text{pgcd}(a, p) \neq 1$ et 1 ou -1 si $\text{pgcd}(a, p) = 1$
    - $\left(\frac{ab}{p}\right) = \left(\frac{a}{p}\right)\left(\frac{a}{p}\right)$
    - $\left(\frac{a}{nm}\right) = \left(\frac{a}{n}\right)\left(\frac{a}{m}\right)$
    - $\left(\frac{m}{n}\right) = \left(\frac{n}{m}\right)(-1)^{\frac{m-1}{2}\frac{n-1}{2}}$ donc
      $\left(\frac{n}{m}\right)$ si l'un est congru à 1 mod 4 et
      $-\left(\frac{n}{m}\right)$ si les deux sont congrus à 3 mod 4
    - $\left(\frac{-1}{n}\right) = (-1)^{\frac{n-1}{2}}$
    - $\left(\frac{2}{n}\right) = (-1)^{\frac{n^2-1}{8}}$ soit 1 si n congru à 1 ou 7 mod8 et -1 si n congru à 3 ou 5 mod8

    Le test verifie : si $\left(\frac{a}{n}\right) = a^{\frac{n-1}{2}}$, a est un témoin pour n

Pour calculer $S = \left(\frac{n}{m}\right)$

```
 N := n; M := m; S := 1;
 si M < 0 alors
 M := -M;
 S=(-1)^((N-1)/2)
    finnsi
 Tant que M ≥ 2 faire
 si M pair alors
M := M/2;
 S := S * (-1)^((N^2-1)/8)
 sinon
 S := S * (-1)^((N-1)(M-1)/4)
 Aux:=N%M
 N:=M.
 finsi
fintq
 si M = 0 alors S = 0 ;
```