TP – RSA, primalité, factorisation

Pour tester les différents programmes, on peut produire en exemple des nombres premiers de grande taille grâce aux commandes `next_prime` ou `previous_prime`.

Arithmétique modulaire

Exercice 1. (Racines carrées, Symbole de Legendre, de Jacobi)

`crt(a,b,m,n)` `is_square` `gcd` `Mod` `valuation(c,p)`

1. Écrire un programme qui prend en entrée p et q, un entier a, et donne les racines carrées de a mod pq si elles existent (et sort « non carré » sinon). Combien en a-t-on ?
2. Écrire un programme qui prend en entrée deux entiers m et n et renvoie la valeur de $\left(\frac{m}{n}\right)$, à l'aide des propriétés du symbole de Jacobi vues en cours. À noter que la commande `a.jacobi(p)` permet directement ce calcul.

RSA

Exercice 2. (RSA)

`power_mod` `xgcd`

1. Écrire un programme qui prend en entrée la clé publique (n, e) d'un cryptosystème RSA et un message en clair m, puis renvoie le message chiffré m^e mod n.
2. Écrire un programme de déchiffrement, sachant les facteurs premiers de n.
3. On peut utiliser le théorème des restes chinois pour accélérer le calcul de c^d mod n (où c est le message chiffré, d l'exposant de déchiffrement), en calculant $c^{d \bmod p-1}$ mod p et $c^{d \bmod q-1}$ mod q. Justifier cette approche, et l'implémenter si cela n'a pas été fait ci-dessus.
4. Comparer les temps d'exécution des différents programmes en faisant varier e (pour n fixé).

Exercice 3. (Différentes attaques)

1. Supposons que $n = pq$ où p et q sont « proches ».
   (a) Montrer qu'on peut écrire $n = r^2 - s^2$ où r est de l'ordre de grandeur de $\sqrt{n}$ (et donc s petit).
   (b) Comment déterminer facilement si un entier est le carré d'un entier ?
   (c) En déduire un programme qui prend en entrée n et renvoie (p, q) en temps rapide si p et q sont proches.
2. Écrire un programme permettant de recouvrer un message en clair m en prenant en entrée ses chiffrés pour deux clés publiques (n, e_1) et (n, e_2), où e_1 et e_2 sont premiers entre eux.
3. (a) En utilisant le résultat de l'exercice 2 du TD sur RSA, montrer que si $ed = 1 + k\varphi(n)$ et $d < \frac{1}{3}\sqrt[4]{n}$, alors $\frac{k}{d}$ est une des réduites du développement de $\frac{e}{n}$ en fraction continuée.

`continued_fraction`

   (b) En déduire un programme qui prend en entrée (n, e) et renvoie d en temps rapide si $d < \frac{1}{3}\sqrt[4]{n}$.
   (c) Tester le programme en remplaçant e par $e' = e + t \cdot \varphi(n)$ pour t grand (de sorte que $e' > n\sqrt{n}$), et comparer les temps d'exécution. Pourquoi cette différence, bien que d reste petit ?

`p.roots()` `solve`

Exercice 4. (Factoriser n à l'aide de $\varphi(n)$) Écrire un programme qui prend en entrée n et k, puis renvoie « Échec » si $k \neq \varphi(n)$ ou si n n'est pas le produit de deux nombres premiers, et renvoie les deux facteurs premiers de n sinon.

Exercice 5. (Factoriser n à l'aide de d) À l'aide de l'exercice 11 du TD sur RSA, écrire un algorithme probabiliste polynomial qui prend en entrée des entiers n, e et d (où e et d sont respectivement les exposants de chiffrement et de déchiffrement dans RSA), et renvoie un facteur premier de n.

Primalité

Exercice 6. (Crible d'Ératosthène) Écrire un programme qui prend en entrée un entier $n \geqslant 2$, et renvoie une liste de tous les nombres premiers inférieurs à n, en adaptant le pseudo-code suivant :

> **Entrée :** n un entier.
> **Sortie :** Les nombres premiers inférieurs à n.
> Initialiser un tableau $L[2] = \cdots = L[n] = 1$;
> Pour $k = 2, \ldots, \sqrt{n}$ faire
> Si $L[k] = 1$ alors
> Pour $m = 2, \ldots, n/k$ faire $L[mk] = 0$ fin pour fin si fin pour ;
> Renvoyer les k tels que $L[k] = 1$.

Pourquoi les nombres de la liste sortante sont-ils tous premiers ? Que dire de la complexité ?

Exercice 7. (Test de Lucas-Lehmer) À l'aide de l'exercice 6 du TD sur RSA, écrire un programme qui prend en entrée un entier n et renvoie « composé » si $2^n - 1$ est composé, et « premier » sinon.

randrange **Exercice 8. (Test de Solovay-Strassen)** À l'aide du test de Solovay-Strassen, écrire un programme qui prend en entrée un entier p, et renvoie « composé » si p est composé ou « premier » si p est premier, avec une probabilité d'erreur inférieure ou égale à $\frac{1}{2}$. Comment diminuer la probabilité d'erreur ?

Exercice 9. (Test de Rabin-Miller) Écrire un programme qui prend en entrée un entier impair p, et renvoie « composé » si p est composé ou « premier » si p est premier, avec une probabilité d'erreur inférieure ou égale à $\frac{1}{4}$, en adaptant le pseudo-code suivant :

> **Entrée :** p un entier impair.
> **Sortie :** "Premier" si p est premier, "Composé" sinon.
> $a :=$`rand`$(1, p-1)$;
> Écrire $p - 1 = 2^e q$, q impair ;
> Calculer $b := a^q$;
> Si $b = 1$ ou $b^{2^i} = -1$ pour un $i \in [\![0, e-1]\!]$ alors `return` "Premier" fin si ;
> `return` "Composé".

Factorisation

Exercice 10. (Algorithme ρ de Pollard)

1. À l'aide des résultats de l'exercice 12 du TD sur RSA, écrire un programme qui prend en entrée un entier n et renvoie un facteur premier p de n, avec au plus $\sqrt{p}$ calculs de pgcd d'entiers inférieurs à n. On produira une suite aléatoire avec le polynôme $P = X^2 + 1$.
2. Reproduire le programme en faisant d'autres choix de P (notamment de degré 1), et comparer le temps d'exécution.

prime_range **Exercice 11. (Algorithme $p-1$ de Pollard)** Écrire un programme qui prend en entrée deux entiers $n \geqslant 2$ et B (la « borne de *lissité* ») et renvoie « échec » ou un facteur non trivial de n, en adaptant le pseudo-code suivant :

Entrée : n, B deux entiers.
Sortie : Un facteur non trivial de n, ou "Échec".
Utiliser le crible d'Érastosthène pour produire tous les nombres premiers $p \leqslant B$;
$a :=$`rand`$(0, N-1)$;
$b := a$;
Pour $p \leqslant B$ faire
Soit k le plus grand entier tel que $p^k \leqslant B$;
$b := b^{p^k}$ fin pour ;
Si $\mathrm{pgcd}(b-1, N)$ est un facteur non trivial de N alors `return` $\mathrm{pgcd}(b-1, N)$
Sinon `return` "Échec" fin si.

Justifier cet algorithme. Pour quel choix de B, et pour quel a peut-on espérer que l'algorithme produise un facteur non trivial de n ?

Exercice 12. (Crible quadratique) Nous avons vu en cours comment, en factorisant $y_j = ([\sqrt{n}] + j)^2 - n$ modulo n sur une base de facteurs $\mathcal{B} = \{-1, p_1, \ldots, p_k\}$ (on dit alors que y_j est $\mathcal{B}$-lisse), on peut produire des congruences non triviales du type $x^2 \equiv y^2 \bmod n$ pour factoriser n par un calcul de pgcd. Le problème est qu'il est coûteux de tester, pour chaque $y_j \bmod n$, s'il est $\mathcal{B}$-lisse ou non. On propose une méthode de crible pour produire simultanément beaucoup de nombres $\mathcal{B}$-lisses.

1. (a) Soit $Q(x) = ([\sqrt{n}] + x)^2 - n \in \mathbb{Z}[x]$. Montrer comment, étant donnée une relation de congruence $Q(a_1) \equiv 0 \bmod p$, on peut produire des entiers a_k tels que $Q(a_k) \equiv 0 \bmod p^k$ pour tout $k \geqslant 1$ (lemme de Hensel).

   (b) Expliquer pourquoi l'algorithme ci-dessous (qu'on retranscrira sous Sage) produit une liste d'entiers $\mathcal{B}$-lisses.

Entrée : n entier, $\mathcal{B}$ base de facteurs, M entier.
Sortie : Une liste d'entiers $a \leqslant M$ tels que $Q(a)$ est $\mathcal{B}$-lisse.
Faire un tableau L, avec $L[a] := Q(a)$ pour $a \in [\![1, M]\!]$;
Si $2 \in \mathcal{B}$, remplacer $L[a]$ par son plus grand diviseur impair pour tout $a \leqslant M$;
Pour $p \in \mathcal{B}$ impair faire
Soit k le plus grand entier tel que $p^k \leqslant Q(M)$;
Produire a_k, b_k tels que $Q(a_k) \equiv Q(b_k) \equiv 0 \bmod p^k$;
Pour $i = k, k-1, \ldots, 1$ faire
Si $i < k$, alors $(a_i, b_i) := (a_{i+1}, b_{i+1}) \bmod p^i$ fin si ;
Pour $\lambda \leqslant M/p^i$ tel que p ne divise pas λ, faire
$L[a_i + \lambda p^i] := L[a_i + \lambda p^i]/p^i$;
$L[b_i + \lambda p^i] := L[b_i + \lambda p^i]/p^i$;
fin pour fin pour fin pour ;
Renvoyer les a tels que $L[a] = 1$.

2. En déduire un programme basé sur le crible quadratique qui prend en entrée $(n, \mathcal{B})$ et renvoie un facteur de n. On exclura préalablement de $\mathcal{B}$ tous les nombres premiers p tels que $\left(\frac{n}{p}\right) = -1$.
3. Faire varier $\mathcal{B}$; est-il utile ou pénalisant d'inclure les petits nombres premiers ?