SÉANCE 2. ALGORITHME D'EUCLIDE ET CONGRUENCES

Objectifs : écriture matricielle de l'algorithme d'Euclide, exponentiation rapide et théorème chinois.

Exercice 1 (Algorithme d'Euclide) — Soit $a, b \in \mathbf{N}$ deux nombres entiers.

1. Vérifier les identités matricielles
$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} b \\ a \end{pmatrix} \text{ et } \begin{pmatrix} 1 & -q \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} a - qb \\ b \end{pmatrix}$$
pour tout $q \in \mathbf{Z}$.
2. En déduire une matrice P de taille 2×2 telle que
$$P \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} b \\ a - qb \end{pmatrix}.$$
3. En déduire un algorithme `Euclide(a,b)` calculant $d = \text{pgcd}(a, b)$ et une matrice P de taille 2×2 à coefficients entiers telle que $\det(P) = \pm 1$ et
$$P \cdot \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} d \\ 0 \end{pmatrix}.$$
4. Que peut-on dire des coefficients de la première ligne de P ?
5. (Bonus) Implémenter une version récursive de l'algorithme d'Euclide.

Exercice 2 (Exponentiation rapide) — Soit $N \geqslant 2$ et $n \geqslant 1$ deux nombres entiers. Étant donné un nombre entier $a \in \mathbf{Z}$, nous allons voir que le calcul de a^n modulo N peut s'effectuer très rapidement.

1. Soit $N \geqslant 2$, $n \geqslant 0$ et $a \in \mathbf{Z}$ des nombres entiers. Écrire un algorithme `ExpNaive(a,n,N)` qui calcule a^n modulo N en effectuant $n-1$ multiplication dans $\mathbf{Z}/N\mathbf{Z}$.
2. (Bonus) Implémenter une version récursive de l'algorithme précédent.
3. Fixons un nombre entier $b \geqslant 2$.
   (i) Démontrer que tout nombre entier $n \geqslant 1$ s'écrit d'une manière et d'une seule sous la forme
   $$n = x_0 b^0 + x_1 b^1 + \ldots + x_k b^k$$
   avec $x_0, \ldots, x_k \in \{0, \ldots, b-1\}$ et $x_k \neq 0$. Les x_i sont les *chiffres* de n en base b.
   (ii) Vérifier en outre que l'on a
   $$k = \left[\frac{\log n}{\log b} \right].$$
   (iii) Écrire un algorithme `Base(n,b)` qui renvoie la liste $[x_0, x_1, \ldots, x_k]$ des chiffres de n en base b. Combien de divisions euclidiennes utilisez-vous ?
   (iv) (Bonus) Ecrire une version récursive de l'algorithme précédent.

4. Notons $\varepsilon_0, \ldots, \varepsilon_k$ les chiffres de n en base 2. Pour tout entier $a \in \mathbf{Z}$,

$$a^n = a^{\varepsilon_0 + 2\varepsilon_1 + \ldots + 2^k \varepsilon_k} = \prod_{i=0}^{k} \left(a^{2^i}\right)^{\varepsilon_i} = \prod_{\substack{0 \leqslant i \leqslant k \\ \varepsilon_i \neq 0}} a^{2^i}.$$

Pour calculer a^n modulo N, il est donc suffisant de calculer les carrés itérés

$$a^2, a^4 = (a^2)^2, \ldots, a^{2^k} = (a^{2^{k-1}})^2$$

modulo N, puis de faire le produit (modulo N) des a^{2^i} correspondant aux indices i tels que $\varepsilon_i \neq 0$.

(i) Écrire un algorithme `ExpRapide(a,n,N)` qui calcule a^n modulo N de la façon que l'on vient de décrire.

(ii) Combien d'opérations (divisions euclidiennes et multiplications) cet algorithme requiert-il ?

(iii) Comparer le temps de calcul des algorithmes `ExpNaive`(a,n,N) et `ExpRapide`(a,n,N).

(iv) (Bonus) Écrire une version récursive de l'algorithme `ExpRapide(a,n,N)`.

5. Déterminer les quatre dernières décimales de 12^{1000}.

6. On rappelle que le n-ième nombre de Fermat est défini par $F_n = 2^{2^n} + 1$.

(i) Calculer $3^{F_5 - 1}$ modulo F_5.

(ii) En déduire que F_5 n'est pas premier.

Pour aller plus loin (préparatoin du prochain TP python)

Exercice 3 (Théorème des restes chinois) — Rappelons le *théorème des restes chinois* :

étant donné un nombre entier $m > 1$ et une factorisation $m = m_1 m_2 \cdots m_r$ en produit d'entiers $m_i > 1$ deux à deux premiers entre eux*, l'application naturelle*

$$f : \mathbf{Z}/m\mathbf{Z} \longrightarrow \mathbf{Z}/m_1\mathbf{Z} \times \mathbf{Z}/m_2\mathbf{Z} \times \cdots \times \mathbf{Z}/m_r\mathbf{Z}$$

$$x \bmod m \longmapsto (x \bmod m_1, x \bmod m_2, \ldots, x \bmod m_r)$$

est un isomorphisme d'anneaux.

1. Démontrez que f est bien un isomorphisme d'anneaux.
2. Expliquez comme construire f^{-1}.