Exponentiation rapide

1. Montrer que les deux algorithmes suivants calculent le même résultat avec la même complexité. Décrire les résultats intérmediares dans les deux cas.

```
def exp_mod_L(a,k,n):
    # Données d'entree:
    # entiers a, k, n > 0
    # Sortie: a^k mod n
    b = 1
    k_seq = k.bits()
    for t in k_seq:
        if t == 1:
            b = (b * a) % n
        a = (a**2) % n
    return b
```

`exp_mod_L`$(3, 101, 101) == 3$

```
def exp_mod_R(a,k,n):
    # Données d'entree:
    # entiers a, k, n > 0
    # Sortie: a^k mod n
    b = 1
    k_seq = k.bits(); k_seq.reverse()
    for t in k_seq:
        b = (b**2) % n
        if t == 1:
            b = (b * a) % n
    return b
```

`exp_mod_R`$(3, 101, 101) == 3$

Algorithmique d'Euclide

1. Pour $m = 7$ et $n = 17$, déterminer la suite des valeurs (m, n) dans la version soustractive et la version classique de `pgcd(m,n)`.

2. Écrire les représentations binaires de la suite des valeurs (m, n) de l'exercice précédent.

3. Pour $m = 16 \cdot 7$ et $n = 8 \cdot 17$, déterminer la suite des valeurs (m, n) dans la version classique de `pgcd(m,n)`.

4. Supposer que la complexité de l'algorithme pour la division euclidienne (`m%n`) est dans $O(\log(n)\log(m/n))$. Comparer la complexité des algorithmes soustractif et classique pour le `pgcd`.

5. Pour $m = 7$ et $n = 17$, déterminer les répresentations binaires de m et n et calculer la suite des valeurs de (m, n) dans la version binaire de `pgcd(m,n)`.

6. Répéter l'exercice précédent avec $m = 16 \cdot 7$ et $n = 8 \cdot 17$.

7. Si l'algorithme pour décalage binaire (`n >> k` ou `shift(n,k)`) a complexité dans $O(\log(n) - k)$, déterminer la complexité de la version binaire de l'algorithme `pgcd`.

Algorithme d'Euclide étendu

1. Dans l'algorithme `pgcde`, vérifier que $r_i = u_i m + v_i n$, pour chaque $i = 0$ et 1, à chaque itération de la boucle `while` et que cette identité reste valide. En particulier, vérifier que dans chaque itération de la boucle `while`, on a

$$\begin{pmatrix} r_0 & u_0 & v_0 \\ r_1 & u_1 & v_1 \end{pmatrix} \longleftarrow \begin{pmatrix} 0 & 1 \\ 1 & -q_1 \end{pmatrix} \begin{pmatrix} r_0 & u_0 & v_0 \\ r_1 & u_1 & v_1 \end{pmatrix} = \begin{pmatrix} r_1 & u_1 & v_1 \\ r_2 & u_2 & v_2 \end{pmatrix},$$

et

$$\begin{pmatrix} r_0 & u_0 & v_0 \\ r_1 & u_1 & v_1 \end{pmatrix} \begin{bmatrix} -1 \\ m \\ n \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

2. Déterminer la suite des tuples (r_0, u_0, v_0) dans le calcul du `pgcde`$(28, 34)$.

3. Soit n et a des entier premiers entre eux. Montrer comment utiliser le `pgcde` pour trouver l'inverse de la classe de a dans $\mathbb{Z}/n\mathbb{Z}$.

Indication. Utiliser la reduction de l'expression $1 = ua + vn$ modulo n.

4. Trouver l'inverse de 7 modulo 17.

Les théorèmes chinois et de Fermat

1. Si $1 = up + vq$, montrer que $x_1 + up(x_2 - x_1) = x_2 + vq(x_1 - x_2)$, et donner une demonstration de la version algorithmique du théoréme chinois.

2. Trouver x tel que $(x \bmod 3, x \bmod 5, x \bmod 7) = (1, 2, 3)$.

3. Soit $\varphi(n)$ le cardinal de $\mathbb{Z}/n\mathbb{Z}^*$. Pour un entier $n = pq$, avec p et q premier entre eux, montrer que $\varphi(n) = \varphi(p)\varphi(q)$.

4. Dans l'anneau $\mathbb{F}_p[x]$, démontrer que

$$\prod_{a=0}^{p-1}(x - a) = x^p - x,$$

et en déduire un isomorphisme $\dfrac{\mathbb{F}_p[x]}{(x^p - x)} \longrightarrow \prod_{a=0}^{p-1} \mathbb{F}_p$. donné par $g(x) \longmapsto (g(a))$.

5. Trouver la factorisation de $x^4 - x$ dans $\mathbb{F}_2[x]$.

6. Soit K un corps fini à $q\,(= p^r)$ éléments. Démontrer que

a. $\alpha^{q-1} = 1$ pour tout $\alpha \in K^*$, et

b. dans $K[x]$, on a l'identité $\prod_{\alpha \in K}(x - \alpha) = x^q - x$.