Aix-Marseille Université
Préparation à l'agrégation: Option C
`http://iml.univ-mrs.fr/~kohel/tch/Agreg/`

Algorithme d'Euclide

Algorithme pgcd – version soustractive

La version soustractive est effective et suffisante pour la démonstration de l'unicité du pgcd, mais elle n'est pas efficace.

```
def pgcd(m,n):
    # Entrees: entiers m, n > 0
    # Sorties: pgcd(m,n)
    # Version soustractive
    while n > 0:
        while m >= n:
            m = m - n
        (m,n) = (n,m)
    return m
```

Algorithme pgcd – version classique

La version "classique" de cet algorithme utilise un algorithme plus efficace pour la division euclidienne.

```
def pgcd(m,n):
    # Entrees: entiers m, n > 0
    # Sorties: pgcd(m,n)
    # Version classique
    while n > 0:
        (m,n) = (n,m%n)
    return m
```

Algorithme pgcd – version binaire

Les ordinateurs ont souvent des implémentations très efficaces des opérations de décalages binaires $n \mapsto \lfloor n/2^k \rfloor$ (`n >> k` ou `shift(n,k)`) ou ($n \mapsto 2^k n$ (`n << k` ou `shift(n,-k)`). L'algorithme `pgcd` classique considère les plus grands bits. En utilisant les décalages binaires l'algorithme binaire suivant opère sur les plus petits bits de m et n.

```
def pgcd(m,n):
    # Entrees: entiers m, n > 0
    # Sorties: pgcd(m,n)
    # Version binaire
    if m == 0: return n
    if n == 0: return m
    i = 0
    while m%2==0:
        m = m >> 1; i += 1
    j = 0
    while n%2==0:
        n = n >> 1; j += 1
    k = min(i,j)
    # m et n sont impairs
    if m < n:
        n = (n-m) >> 1
    else:
        m = (m-n) >> 1
    return pgcd(m,n)<<k
```

Exercices

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

Le `pgcd` étendu de deux entiers peut être calculé avec la fonction suivante (où la sortie de `quo_rem`(m,n) est le tuple $(q,r) = (\lfloor m/n \rfloor, m \bmod n)$ tel que $m = r + qn$) :

```
def pgcde(m,n):
    # Entrees: entiers m, n > 0
    # Sorties (r,u,v) tel que
    # r = pgcd(m,n) et que
    # r = u m + v n.
    if m < n:
        (r0, u0, v0) = (n, 0, 1)
        (r1, u1, v1) = (m, 1, 0)
    else:
        (r0, u0, v0) = (m, 1, 0)
        (r1, u1, v1) = (n, 0, 1)
    while r1 > 0:
        (q1, r2) = r0.quo_rem(r1)
        (u2, v2) = (u0 - u1*q1, v0 - v1*q1)
        (r0, u0, v0) = (r1, u1, v1)
        (r1, u1, v1) = (r2, u2, v2)
    return (r0, u0, v0)
```

Exercices

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

Théorème (Théorème chinois.)**.** *Soient p et q des entiers premiers entre eux et n leur produit. Pour tout couple (x_1, x_2) il existe un entier x, uniquement determiné modulo n, tel que $x \equiv x_1 \bmod p$ et $x \equiv x_2 \bmod q$.*

Autrement dit, l'application

$$\mathbb{Z}/n\mathbb{Z} \longrightarrow \mathbb{Z}/p\mathbb{Z} \times \mathbb{Z}/q\mathbb{Z}$$

donnée par $x \mapsto (x \bmod p, x \bmod q)$ est une bijection. Comme elle est un homomorphisme d'anneaux,

- elle est un isomorphisme d'anneaux ;
- elle est un isomorphisme de groupes additifs ;
- elle induit un isomorphisme de groupes multiplicatifs $\mathbb{Z}/n\mathbb{Z}^* \longrightarrow \mathbb{Z}/p\mathbb{Z}^* \times \mathbb{Z}/q\mathbb{Z}^*$.

L'algorithme pour l'inverse de l'homomorphisme $\mathbb{Z}/n\mathbb{Z} \to \mathbb{Z}/p\mathbb{Z} \times \mathbb{Z}/q\mathbb{Z}$ est un conséquence de l'algorithme `pgcde`.

Théorème (Algorithme CRT)**.** *Soient p et q des entiers premier entre eux, n leur produit, et u et v des entiers tel que $1 = up+vq$. Pour tout couple (x_1, x_2) l'entier $x = x_1+up(x_2-x_1)$ satisfait $x \equiv x_1 \bmod p$ et $x \equiv x_2 \bmod q$.*

Théorème (Fermat)**.** *Soit p un nombre premier. On a $a^p \equiv a \bmod p$ pour tout entier a, et $a^{p-1} \equiv 1 \bmod p$ pour tout entier a premier à p.*

Dans un language plus moderne, $\mathbb{Z}/p\mathbb{Z}^*$ est un groupe cyclique d'ordre $p-1$.

Notation. Si p est premier, on écrit $\mathbb{F}_p$ pour le corps $\mathbb{Z}/p\mathbb{Z}$.

Exercices

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
$$\prod_{a=0}^{p-1}(x - a) = x^p - x.$$
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 $\prod_{\alpha \in K}(x - \alpha) = x^q - x$.