Identité de Bezout (Algorithme)

<u>Algo. d'Euclide</u> : PGCD$(a,b) = ?$ avec $a > b$

$a = q_1 b + r_1$ $\quad\quad$ $b := r_0$, $a := r_{-1}$

$b = q_2 r_1 + r_2$

$r_1 = q_3 r_2 + r_3$

....

$r_n = q_{n+2}\, r_{n+1} + r_{n+2}$ avec $r_{n+2} = 0$ donc $r_{n+1} = $ Pgcd

<u>Remontée</u> : $\text{PGD}(a,b) = r_{n+1} = r_{n-1} - q_{n+1} r_n$

$= r_{n-1} - q_{n+1}(r_{n-2} - q_n r_{n-1})$

$= (1 - q_{n+1} q_n) r_{n-1} - q_{n+1} r_{n-2}$

....

$= \bullet\, b + \bullet\, a$

Pb : pas pratique à programmer mais utilisable pour calcul à la main.

<u>Un algorithme donnant l'identité de Bezout</u> :

Le procédé précédant prouve l'existence de u_n et v_n tels que :

$$u_n r_n + v_n r_{n-1} = \text{PGCD}(a,b)$$

$$u_n r_n + v_n r_{n-1} = u_n (r_{n-2} - q_n r_{n-1}) + v_n r_{n-1}$$
$$= (-q_n u_n + v_n)\, r_{n-1} + u_n r_{n-2}$$

On choisit $u_{n-1} = -q_n u_n + v_n$ et $v_{n-1} = u_n$

Absolument pas utilisable en programme,
car il faut descendre <u>PUIS</u> remonter.

Une autre voie :

Partant du début d'Euclide, nous avons u_n, v_n tels que :

$$x_n = u_n a + v_n b$$

En effet,

$$x_{n+2} = x_n - q_{n+2} x_{n+1}$$

$$= u_n a + v_n b - q_{n+2} u_{n+1} a - q_{n+2} v_{n+1} b$$

$$= \underbrace{[u_n - q_{n+2} u_{n+1}]}_{u_{n+2}} a + \underbrace{[v_n - q_{n+2} v_{n+1}]}_{v_{n+2}} b$$

Plus intéressant car on ne fait <u>QUE</u> descendre, donc pas de stockage de données à faire.

On choisit $u_0 = 0$ $v_0 = 1$ car $x_0 = b$

$u_{-1} = 1$ $v_{-1} = 0$ car $x_{-1} = a$

<u>Applica°</u> : Résolu° de $ax \equiv b\ [n]$ $a, b \in [0; n-1]$

<u>1er cas : Pgcd $(a; n) = 1$</u>

On calcule a^{-1} (vu ce qui précède) puis $x = a^{-1}.b$.

<u>2ème cas : Pgcd $(a; n) \neq 1$</u>

• <u>Sous-cas 1 :</u> n premier

$\Longleftrightarrow a \equiv 0\ [n]$, donc soit

x qcq / $b \equiv 0$, soit x n'existe pas / $b \not\equiv 0$

• <u>Sous-cas 2 :</u> n composé, $b = 0$

On remplace a par $\frac{a}{\text{pgcd}(a,n)}$ et n par $\frac{n}{\text{pgcd}(a,n)}$ puis on cherche solutions de $\tilde{a}\tilde{x} \equiv 0\ [\tilde{n}]$ où $\tilde{n} < n$.

Car pgcd $(\tilde{a}, \tilde{n}) = 1$, on a $\tilde{x} \equiv 0\ [\tilde{n}]$ d'où $x = 0, \tilde{n}, 2\tilde{n}, \ldots$

$\ldots [\text{pgcd}(a,n) - 1] \cdot \tilde{n}$

Solutions "oubliées"

Schéma de programmation pour résoudre $ax \equiv b\ [n]$

• Sous-cas 3 : u composé, $b \neq 0$

$ax = b + ku \Rightarrow \operatorname{Pgcd}(a,u) / b$

$\Rightarrow \operatorname{Pgcd}(b ; \operatorname{Pgcd}(a,u)) = b$

Donc, si $\operatorname{Pgcd}(b ; \operatorname{Pgcd}(a,u)) \neq b$, pas de solution

Sinon, on remplace a par $\frac{a}{b}$, u par $\frac{u}{b}$, d'où $\tilde{a}\tilde{x} \equiv 1\ [\tilde{u}]$ puis $\underbrace{\tilde{x} = \tilde{a}^{-1}}_{\text{Inverse Mod } \tilde{u}}$, et $x = \underbrace{\tilde{a}^{-1}, \tilde{a}^{-1} + \tilde{u}, \dots, \tilde{a}^{-1} + (b-1)\tilde{u}}_{\text{Solutions cachées}}$

Problème : Comment calculer $a^{-1} \bmod u$ si $a \wedge u = 1$.

↪ Utilisa° de Bezout

↪ Si u premier calcul de a^{p-1} via $p-1 = \sum_i \varepsilon_i 2^i$, $\varepsilon_i = 0,1$