Algorithme d'Euclide et applications

Exercice 1 *Euclide*

1. Implanter une fonction `div_eucl(a,b)` de division euclidienne pour les entiers (on pourra se restreindre, au moins dans un premier temps, à un couple d'entiers *positifs* a, b). Comparer ensuite avec les opérations // et %.
2. Implanter l'algorithme d'Euclide pour les entiers (on ne demande pas pour l'instant que l'algorithme renvoie des coefficients de Bézout) sous forme d'une fonction `euclide(a,b)`. Comparer avec `gcd`.
3. Écrire une fonction `fibo` permettant de calculer le n-ième nombre de Fibonacci F_n. On rappelle que
$$F_1 = F_2 = 1\,, \qquad F_{n+2} = F_n + F_{n+1}\,.$$
Vérifier que votre procédure donne un résultat rapide pour des valeurs de n "raisonnablement grandes" (disons dans le cas où n a 3 chiffres). Si ce n'est pas le cas, programmer (F_n) autrement... (chercher par exemple un algorithme calculant F_n en $O(n)$ opérations). Si vous avez implanté un calcul récursif des termes de (F_n), testez l'effet de la commande `@cached_function` en première ligne de votre code.
4. Vérifier sur quelques exemples que le nombre d'étapes dans l'algorithme d'Euclide pour le couple (F_{n+2}, F_{n+1}) est n.
5. Bonus : en utilisant l'algorithme d'exponentiation rapide, justifier l'existence d'un algorithme permettant de calculer F_n en $O(\log_2(n))$ opérations élémentaires.

Exercice 2 *Euclide étendu*

1. Implanter l'algorithme d'Euclide étendu pour les entiers (sans nécessairement tenir compte des questions de normalisation) sous forme d'une fonction `euclide_etendu(a,b)`.
2. La fonction de la question précédente est-elle encore valable pour les polynômes à coefficients rationnels ? (Dans le cas contraire, adapter la fonction pour englober le cas des polynômes.)
On notera que la commande `X=polygen(QQ,'X')` permet de définir `X` comme générateur de l'anneau de polynômes $\mathbf{Q}[X]$.
3. Comparer avec `xgcd`.
4. Écrire une fonction `sans_carre(P)` prenant en entrée un polynôme $P \in \mathbf{Q}[X]$ et renvoyant en sortie la partie sans facteur carré de P c'est-à-dire le produit des irréductibles de $\mathbf{Q}[X]$ deux à deux distincts divisant P.
5. Donner une commande `Sage` d'une ligne permettant d'inverser 38 modulo 101.
6. Écrire une fonction `inv_mod(a,m)` prenant en entrée un entier a et un entier $m \geqslant 2$ et renvoyant, s'il existe, un entier x vérifiant $ax \equiv 1 (\mathrm{mod}\, m)$ ou qui lève une exception
`raise ValueError("Le nombre %s n'est pas inversible modulo %s" %(a,m))`
dans le cas contraire. Comparer avec la réaction de `Sage` en cas d'erreur.

Exercice 3 *Premières applications*

1. Soit ζ_{15} une racine primitive 15-ième de 1 dans $\mathbf{C}$ et $\mathbf{Q}(\zeta_{15})$ le plus petit sous-corps de $\mathbf{C}$ contenant $\mathbf{Q}$ et ζ_{15}. Donner, sous forme d'un polynôme à coefficients rationnels en ζ_{15}, l'expression de l'inverse de $1+\zeta_{15}$ dans $\mathbf{Q}(\zeta_{15})$.
2. On souhaite écrire une procédure alternative à celle de l'exercice précédent d'inversion modulaire dans le cas où l'on réduit modulo un nombre premier p. On procède en deux étapes :
   - Écrire une fonction `exp_mod(a,e,m)` permettant de calculer rapidement $a^e \bmod m$.
   - Utilser le petit théorème de Fermat combiné à la fonction `exp_mod(a,e,m)` pour écrire une procédure permettant le calcul rapide de l'inverse de a modulo p où p est un nombre premier ne divisant pas a.

Exercice 4 *Fractions continues*

1. Soit r un rationnel mis sous forme irréductible r_0/r_1. Les premières étapes de l'algorithme d'Euclide pour (r_0, r_1) donnent :

$$\frac{r_0}{r_1} = \frac{q_1 r_1 + r_2}{r_1} = q_1 + \frac{r_2}{r_1} = q_1 + \frac{1}{\frac{r_1}{r_2}} = q_1 + \frac{1}{\frac{q_2 r_2 + r_3}{r_2}} = q_1 + \frac{1}{q_2 + \frac{r_3}{r_2}} = \cdots$$

   Plus généralement, étant donné un réel a, on appelle *développement en fractions continues* de a la suite $[a_0; a_1, a_2, \ldots]$ obtenue de la manière suivante : a_0 est la partie entière de a et $t_0 = a - a_0$. Pour $n \geqslant 0$, on pose, tant que $t_n \neq 0$,

$$a_{n+1} = \lfloor 1/t_n \rfloor\,, \qquad t_{n+1} = \frac{1}{t_n} - a_{n+1}\,.$$

   On a donc

$$a = a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{a_3 + \cdots}}}\,,$$

   où la fraction se poursuit "indéfiniment" à moins qu'un t_n ne soit nul pour un certain n auquel cas le processus s'arrête.
2. Écrire une procédure `cont_frac(a,N)` prenant en entrée un réel positif a et un entier naturel N et renvoyant la suite $[a_0; a_1, \ldots, a_N]$ comme ci-dessus ou bien $[a_0; a_1, \ldots, a_n]$ si $n \leqslant N$ est le plus petit indice tel que $t_n = 0$.
3. Tester votre procédure pour les réels $8/3$, $\sqrt{2}$, π, $(1+\sqrt{5})/2$, e, $\sqrt[3]{7}$.
4. Que dire si le réel a de départ est rationnel ?
5. Énoncer une conjecture caractérisant les réels a dont le développement en fractions continues est infini et périodique.
6. Écrire une procédure récursive `conv(L)` prenant en entrée une liste d'entiers $L = [a_0, a_1, \ldots a_n]$ et renvoyant le rationnel (sous forme réduite)

$$r = a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{a_3 + \ldots + \frac{1}{a_n}}}}\,.$$

   On pourra utiliser la fonction `float` (permettant de donner des valeurs approchées de réels) pour vérifier que la procédure `conv` fonctionne.
7. D'où provient l'approximation rationnelle classique $22/7$ pour π ?