Algorithme d'Euclide

Exercice 1 (Algorithme d'Euclide)**.**

1. Implanter une fonction `div_eucl(a,b)`de division euclidienne pour les entiers (on pourra se restreindre au cas d'un couple d'entiers *positifs* (a, b)). Comparer ensuite avec les commandes `//` et %.
2. Implanter l'algorithme d'Euclide pour les entiers sous forme d'une fonction `euclide(a,b)` (on ne demande pas pour l'instant que l'algorithme renvoie des coefficients de Bézout mais simplement le pgcd du couple d'entiers donné en entrée). Comparer avec `gcd`.
3. Écrire une fonction `fib(n)` qui à l'entrée $n \geq 1$ associe le n-ème nombre de Fibonacci F_n. On rappelle que la suite (F_n) est définie par :

   $$F_1 = F_2 = 1 \text{ et } F_{n+2} = F_{n+1} + F_n \text{ pour } n \geq 1\,.$$

   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 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. En modifiant légèrement la fonction `euclide(a,b)` pour qu'apparaisse en sortie de nombre de divisions euclidiennes nécessaires au calcul du pgcd, vérifier que le calcul de $\text{pgcd}(F_{n+1}, F_n)$ nécessite n divisions euclidiennes, pour $1 \leq n \leq 100$.

Exercice 2 (Algorithme d'Euclide étendu et applications)**.**

1. Écrire l'algorithme d'Euclide étendu pour les entiers (sans nécessairement tenir compte des problèmes de normalisation) sous forme d'une fonction `euclide_etendu(a,b)`.
2. Vérifier que la procédure ci-dessus fonctionne toujours dans $\mathbb{Q}[X]$.
   La commande `X=polygen(QQ,'X')` définit `X` comme étant le générateur de l'anneau de polynômes $\mathbb{Q}[X]$. On peut aussi utiliser la commande `R.<X>=PolynomialRing(QQ, 'X')` qui définit `R` comme $\mathbb{Q}[X]$ et `X` comme l'indéterminée associée.
3. Comparer avec `xgcd`.
4. Écrire une procédure `inverse_mod(a,m)` permettant de calculer l'inverse d'un entier a premier avec un entier $m \geq 2$ donné. Si a n'est pas premier avec m, on utilisera la commande

   ```
   raise ValueError("Le nombre {} n'est pas inversible modulo {}".format(a,m))
   ```

   où la méthode `format` appliquée à une chaîne de caractères remplace les `{}` par la valeur des expressions données comme arguments (ici a et m). On comparera la réponse à la commande `inverse_mod(2,4)` avec la réponse habituelle de `Sage` en cas d'erreur.
5. Écire une procédure de résolution des équations diophantiennes $ax + by = c$ où a, b et c sont des entiers.

Exercice 3 (Décomposition sans carré)**.**

Soit $P \in \mathbb{Q}[X]$. On note $P = P_1^{r_1} \dots P_k^{r_k}$ sa décomposition en facteurs irréductibles. La *partie sans carré* de P est $P_1 \dots P_k$. On dit que P est *sans carré* s'il est égal à sa partie sans carré.

1. Expliquer comment trouver la partie sans carré d'un polynôme sans le factoriser.
2. Écrire une procédure calculant la partie sans carré d'un polynôme.
3. Montrer que tout polynôme $P \in \mathbb{Q}[X]$ peut s'écrire $P = Q_1 Q_2^2 \dots Q_n^n$ où les $Q_i \in \mathbb{Q}[X]$ sont sans carré et premiers entre eux. On parle de *décomposition sans carré*.
4. Écrire une procédure calculant la décomposition sans carré d'un polynôme sans le factoriser.

Exercice 4 (Fractions continues)**.** 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 $[a_0; a_1, a_2, \dots]$ obtenue de la manière suivante : a_0 est la partie entière de a et $t_0 = a - a_0$. Pour $n \geq 0$, 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

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

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.

1. Écrire une procédure `cont_frac(a,N)` prenant en entrée un réel positif a et un entier naturel N et renvoyant la suite $[a_0; a_1, \dots, a_N]$ comme ci-dessus ou bien $[a_0; a_1, \dots, a_n]$ si $n \leq N$ est le plus petit indice tel que $t_n = 0$.
2. Tester votre procédure pour les réels $8/3$, $\sqrt{2}$, π, $(1+\sqrt{5})/2$, e, $\sqrt[3]{7}$.
3. Que dire si le réel a de départ est rationnel ?
4. Écrire une procédure récursive `conv(L)` prenant en entrée une liste d'entiers $L = [a_0, a_1, \dots a_n]$ et renvoyant le rationnel (sous forme réduite)

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

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