TD 3 : Pgcd et inversion modulaire

Exercice 1. *Complexité binaire de l'algorithme d'Euclide*

Soit A et B deux entiers tels que $A \geq B$. On considère l'exécution de l'algorithme d'Euclide sur l'entrée (A, B). On note $R_0 = A$, $R_1 = B$ et $R_{i+2} = R_i \bmod R_{i+1}$ pour $i \geq 0$. On note ℓ l'indice du dernier R_i non nul.

1. Exprimer le nombre de divisions euclidiennes effectuées en fonction de ℓ.
2. Montrer que le nombre de divisions euclidiennes est borné par $1 + \log_\phi(A)$ et par $2 + \log_\phi(B)$. Quelle borne est la meilleure ?
3. Pour $i \geq 0$, on note n_i le nombre de chiffres (en une base β quelconque) de R_i.
   - i. Exprimer la complexité du calcul de $R_i \bmod R_{i+1}$ en fonction de n_i et n_{i+1}.
   - ii. En déduire la complexité binaire de l'algorithme d'Euclide, en fonction des n_i.
   - iii. En déduire que la complexité binaire est bornée par $O(\log A \log B)$.

Exercice 2. *Inversion par le petit théorème de Fermat*

1. Montrer que le petit théorème de Fermat permet d'obtenir un algorithme d'inversion modulaire dans $\mathbb{Z}/p\mathbb{Z}$ (pour p premier), et analyser sa complexité.
2. Que se passe-t-il si on essaie d'utiliser l'algorithme dans $\mathbb{Z}/N\mathbb{Z}$, avec N non premier ?

Exercice 3. *Programmation*

Il est impératif de tester chaque fonction codée !

1. - i. Écrire l'algorithme d'Euclide étendu.
   - ii. Écrire une version *itérative* de l'algorithme d'Euclide étendu.
   - iii. Comparer les temps de calcul [1] des deux versions.
2. Écrire une fonction d'inversion modulaire : étant donné a et N, cette fonction calcule l'inverse de a *modulo* N. Si a n'est pas inversible *modulo* N, la fonction doit soulever une exception. *Par exemple, on peut utiliser :* `raise ValueError(f'{a} n'est pas inversible modulo {N}')` *pour cela.*

1. On pourra utiliser `%time` : `https://doc.sagemath.org/html/en/thematic_tutorials/profiling.html`.