TD 2 : 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. *Algorithme de* PGCD *binaire*

1. Soit u et v deux entiers.
   i. Exprimer $\gcd(2u, 2v)$ en fonction de $\gcd(u, v)$.
   ii. Si v est impair, exprimer $\gcd(2u, v)$ en fonction de $\gcd(u, v)$.
   iii. Si u et v sont impairs, montrer que $\gcd(u, v) = \gcd(|u - v|/2, \min(u, v))$.
2. Déduire de la question précédente un algorithme de calcul de PGCD qui n'utilise comme opérations de base que la soustraction et la multiplication et la division par 2.
3. Prouver la correction de votre algorithme, et analyser sa complexité binaire pour deux entiers de n bits en entrée.
4. Qu'est-ce qui rend cet algorithme efficace en pratique ?

Exercice 3. *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 4. *Exponentiation*

L'algorithme d'exponentiation vu en cours est dit *binaire* car il considère l'exposant écrit en base 2. Il considère les chiffres de l'exposant en commençant par le chiffre de poids faible (puisqu'à chaque étape, on teste si l'exposant est pair, puis on le divise par 2).

Dans cet exercice, on s'intéresse à des algorithmes de calcul de g^n où g est élément d'un groupe multiplicatif quelconque, et n est entier positif. Pour une base β, on note $n = \overline{n_{\ell-1} \cdots n_0}^\beta$ les chiffres de n en base β. Les complexités doivent être exprimées de manière exacte, pas avec des $O(\cdot)$.

1. Écrire une version modifiée de l'exponentiation binaire, qui considère les chiffres de la gauche (poids fort) vers la droite : la boucle principale devra être « Pour $i = \ell - 1$ à 0 : ». Analyser sa complexité.
2. Adapter l'algorithme précédent au cas où les chiffres de l'exposant sont donné en base β quelconque. *Attention : l'algorithme nécessite des pré-calculs, qu'il faut spécifier.*
3. Adapter l'algorithme au cas de la base $\beta = 2^k$, en ne s'autorisant que des élévations au carré et des multiplications dans le groupe. Analyser la complexité de l'algorithme obtenu en fonction de la longueur de n en base 2.
4. Comment minimiser la mémoire et les pré-calculs requis par l'algorithme précédent, sans augmenter la complexité des itérations ? Analyser la complexité de l'algorithme obtenu en fonction de la longueur de n en base 2.
5. Quelle est la valeur de k qui minimise la complexité pour un exposant de 64 bits ? Même question pour un exposant de 1024 bits ?