TP3 : Opérations élémentaires

Bibliographie : *Cours de calcul formel*, Saux-Picard.

But du TP : évaluer le coût des opérations élémentaires (addition, multiplication, évaluation...) sur les entiers et les polynômes.

1 Coût des opérations sur les polynômes

Rappelons qu'un polynôme à une indéterminée à coefficients dans un anneau A est une liste finie d'éléments de A. Si `Pol_A` désigne l'anneau des polynômes en X sur A :

— `list(P)` renvoie la liste des coefficients de `P`, par ordre croissant de degré.

— `Pol_A(L)` renvoie le polynôme de `Pol_A` dont la liste des coefficients est `L`.

1.1 Addition et multiplication naïve

1. Ecrire une fonction `addition_pol` qui prend en entrée la liste des coefficients de deux polynômes P et Q, et retourne la liste des coefficients de $P+Q$. Quelle est la complexité de l'algorithme ?
2. En vous inspirant des multiplications telles qu'on a l'habitude de les "poser" à la main, écrire une fonction `multiplication_pol` qui, étant donnée la liste des coefficients de deux polynômes P et Q, retourne la liste des coefficients du produit PQ. Quelle est la complexité de l'algorithme ?
3. L'algorithme est-il commutatif du point de vue des temps de calcul ? Sinon, améliorez-le.

1.2 Multiplication de Karatsuba

On considère deux polynômes P et Q de degré $n-1$, où n est un entier pair. Nous écrivons :

$$P = P_H X^{n/2} + P_L \qquad\qquad Q = Q_H X^{n/2} + Q_L$$

et développons le produit :

$$PQ = P_H Q_H X^n + (P_H Q_L + P_L Q_H) X^{n/2} + P_L Q_L$$

Cette expression contient quatre multiplications de polynômes de degré $n/2-1$, c'est-à-dire de polynômes dont la liste des coefficients est deux fois moins longue.

4. Quelle est la complexité de l'algorithme qui calcule PQ en utilisant ces décompositions récursivement ?
5. Trouver une façon de calculer $P_H Q_H, (P_H Q_L + P_L Q_H)$ et $P_L Q_L$ en n'effectuant en tout que *trois* produits (plus d'éventuelles additions) de polynômes de degré au plus $n/2-1$.
6. En déduire un algorithme récursif `karastuba_pol` qui calcule la liste des coefficients de PQ. Calculer sa complexité.
7. Implémenter votre algorithme, et comparer ses performances avec `multiplication_pol`. Comment expliquez-vous ces résultats ?

1.3 Evaluation naïve et schéma de Horner

8. Compter le nombre d'additions et de multiplications sur A lors de l'évaluation "naïve" du polynôme $P := a_nX^n + a_{n-1}X^{n-1} + ... + a_1X + a_0$ en x. Qu'économiseriez-vous en utilisant l'exponentiation rapide ? Et en calculant d'abord chacune des puissances de X, avant de les multiplier par les coefficients ?

La méthode de Horner consiste à écrire le polynôme comme :

$$P = ((...((a_nX + a_{n-1})X + a_{n-2})...)X + a_1)X + a_0.$$

9. Compter alors le nombre d'additions et de multiplications requises pour calculer $P(x)$. [1]

2 Coût des opérations sur les entiers

De la même façon que nous avons vu les polynômes comme des listes de scalaires, nous allons travailler sur les entiers en tant que listes de bits.

10. Montrer que le nombre de bits nécessaires pour coder un entier non nul n est $log_2(n) + 1$.
11. Ecrire deux fonctions `decimalTObinary` et `binaryTOdecimal` qui transforment un entier naturel en la liste minimale des bits de son écriture binaire, et vice versa. On fera le choix de placer les bits de poids fort à droite, et de choisir la liste vide comme plus court représentant de 0.
    Ex : `decimalTObinary(14) = [0,1,1,1]` ; `decimalTObinary(0) = []`
    Vous pourrez comparer votre fonction `decimalTObinary` avec la méthode `.digits(2)` de sage.
12. Ecrire une fonction `addition` qui, étant données deux listes de bits (telles que spécifiées ci-dessus), renvoie la liste des bits de la somme. Quelle différence majeure y a-t-il avec l'addition de deux polynômes ? Quelle est la complexité de l'algorithme ?
13. Ecrire une fonction `multiplication` qui, étant donnés deux listes de bits d'entiers naturels, renvoie la liste des bits du produit. Quelle est la complexité de l'algorithme ? L'algorithme est-il commutatif du point de vue des temps de calcul ? Sinon, améliorez-le.

♠ Multiplication de Karatsuba pour les entiers

Comme pour les polynômes, si l'on considère deux entiers A et B à n bits, où n est un entier pair, nous pouvons écrire :

$$A = a_H2^{n/2} + a_L \qquad\qquad B = b_H2^{n/2} + b_L$$

et développer le produit :

$$AB = a_Hb_H2^n + (a_Hb_L + a_Lb_H)2^{n/2} + a_Lb_L$$

Cette expression contient quatre multiplications de nombres deux fois plus courts.

14. ♠ Trouver une façon de calculer $a_Hb_H, (a_Hb_L + a_Lb_H)$ et a_Lb_L en n'effectuant en tout que trois produits *de nombres deux fois plus courts.*
15. Quelle est la complexité théorique de l'algorithme récursif qui permettrait de calculer le produit AB ?
16. ♠ Implémenter cet algorithme, et comparer ses performances avec votre algorithme de multiplication précédent. Comment expliquez-vous ces résultats ? Comment corriger ce défaut ?

1. Le schéma de Horner est de plus optimal : on ne peut pas évaluer un polynôme quelconque en moins de n multiplications/divisions et n additions/soustractions.