

Arithmétique multiprécision : projet 2010-2011

Dans les ordinateurs actuels (qui sont des machines de von Neumann), les registres contenant les types entiers font 16, 32 ou 64 bits (quelques rares architectures ont des entiers 128 bits). Ici, on notera B la taille d'un entier machine en bit. Mais la mémoire ne peut être adressée que par "paquets" de 8 bits (un octet). Un entier machine est donc un paquet de 2, 4 ou 8 octets. On notera b la taille d'un entier machine en octet. On a donc $B = 8b$. L'objectif de ce sujet est d'étudier une arithmétique multiprécision permettant de traiter des entiers arbitrairement grands. Il faudra faire un ensemble d'algorithmes pour la somme et le produit des entiers multiprécisions. J'adopterai la convention de la programmation en C. Les opérations que nous estimerons élémentaires sont les opérations machines (sur les types fournis par le langage C). Ainsi, la structure pour un entier multiprécision est :

```
typedef struct INTEGER {
    unsigned int *val;
    unsigned int size;
} integer;
```

La norme C vous garantit que les opérations sur les entiers machines non signés (`unsigned int`) est l'arithmétique dans $\mathbb{Z}/2^B\mathbb{Z}$. Autrement dit, si vous faites ma sommes de deux entiers non signé $a + b$ ou leur produit $a \times b$ vous avez comme résultat le reste de la division par 2^B de la vraie somme ou du vrai produit. Les B premiers bits du résultat de l'opération machine sont les B premiers bits de l'opération exacte. En particulier, si le résultat est plus petit que 2^B , le résultat de l'opération machine est le résultat exacte. C'est cette idée que nous allons exploiter pour implanter l'arithmétique multiprécision.

1 Quelques opérations de base

Voici une fonction permettant d'initialiser un entier (allocation) :

```
void InitInteger(integer *N, unsigned int s) {
    N->val=(integer*)malloc(s*sizeof(int));
    N->size = s;
}
```

Ecrire une fonction qui permet de dupliquer un entier dont la signature est la suivante :

```
void duplication(integer *source, integer *but)
```

copiant la valeur et la taille de `source` dans `but` (attention tout est passé par adresse).

Ecrire une fonction permettant de récupérer les $B/2$ premiers bits d'un entier, dont la signature est la suivante :

```
unsigned int bas(unsigned int a)
```

La partie des $B/2$ bits de poids faible d'un entier machine est appelée la partie basse. Si N est un entier, je noterai \hat{N} sa partie basse.

Ecrire une fonction permettant de récupérer les $B/2$ bits de poids fort d'un entier. La signature doit être la suivante :

```
unsigned int haut(unsigned int a)
```

La partie contenant les $B/2$ bits de poids fort d'un entier machine est appelée partie haute et si l'entier est noté N sa partie haute est notée \overline{N} .

Exercice 1. Montrer que $N = \hat{N} + 2^{B/2} \overline{N}$.

Ecrire une fonction permettant, étant donné un entier machine `source` $< 2^{B/2}$, d'affecter les $B/2$ bits de poids faible de `but` avec les $B/2$ bits de poids faible de `source`. Autrement, on duplique la partie basse de `source` dans `but`. Le prototype de la fonction est le suivant :

```
void set_bas(unsigned int *source, unsigned int *but)
```

De la même façon, on souhaite pouvoir modifier la partie haute avec la fonction dont le prototype est le suivant :

```
void set_haut(unsigned int *source, unsigned int *but)
```

2 Addition

Exercice 2. Si $N < 2^n$ et $M < 2^m$, montrer que $N + M < 2^n + 2^m$. Donner deux entiers de tailles 2 et 3 tels que la somme soit la plus grande possible pour des entiers de cette taille.

Soit $N = n_0 + n_1 2^B + \dots + n_k 2^{kB}$ avec $n_i < 2^B$ pour tout $i \in \{0, \dots, k\}$ et $M = m_0 + m_1 2^B + \dots + m_l 2^{lB}$ avec $m_i < 2^B$ pour tout $i \in \{0, \dots, l\}$.

Exercice 3. Montrer que $n_0 + m_0 = \widehat{n_0 + m_0} + \overline{\widehat{n_0 + m_0}} 2^B + \overline{\overline{\widehat{n_0 + m_0}}} 2^{2B}$. Donner une majoration pour $\overline{\widehat{n_0 + m_0}}$ que nous appellerons la première retenue.

Indication : on note $\alpha = 2^{B/2}$, alors on a $n_0 = \hat{n}_0 + \bar{n}_0 \alpha$ et $m_0 = \widehat{m}_0 + \overline{m}_0 \alpha$.

Exercice 4. Donner un algorithme permettant de faire la somme de deux entiers multiprécisions.

Pour les informaticiens : programmez cet algorithme et testez-le.

Pour les mathématiciens : faites la preuve que votre algorithme fait bien ce que vous prétendez.

Exercice 5. Donner la complexité de l'algorithme en fonction de la taille des deux entiers à sommer.

3 Multiplication

Exercice 6. Si $N < 2^n$ et $M < 2^m$, montrer que $N + M < 2^{n+m}$. Donner deux entiers de taille 2 et 3 tel que leur produit soit maximal pour des entiers de cette taille.

Soit $N = n_0 < 2^B$ et $M = m_0 < 2^B$. On pose $N = n_0 = \hat{n}_0 + \bar{n}_1 \alpha$ et $M = m_0 = \widehat{m}_0 + \overline{m}_1 \alpha$.

Exercice 7. Montrer que $NM = \widehat{\hat{n}_0 \widehat{m}_0} + (\widehat{\hat{n}_0 \overline{m}_1} + \overline{\hat{n}_1 \widehat{m}_0}) \alpha + \overline{\overline{\hat{n}_0 \overline{m}_1} + \overline{\hat{n}_1 \widehat{m}_0}} \alpha^2$.

On déduit de l'exercice précédent que $NM = \widehat{\hat{n}_0 \widehat{m}_0} + \overline{(\widehat{\hat{n}_0 \overline{m}_1} + \overline{\hat{n}_1 \widehat{m}_0})} \alpha + \overline{(\overline{\hat{n}_0 \overline{m}_1} + \overline{\hat{n}_1 \widehat{m}_0})} \alpha^2$. Remarquez que la somme utilisée est celle des entiers machines.

Exercice 8. Si $N = n_0 + n_1 2^B$ et $M = m_0 + m_1 2^B$, donner une formule analogue à celle de l'exercice précédent.

On pose maintenant $N = n_0 + n_1 2^B + \dots + n_k 2^{kB}$ avec $n_i < 2^B$ pour tout $i \in \{0, \dots, k\}$ et $M = m_0 + m_1 2^B + \dots + m_l 2^{lB}$ avec $m_i < 2^B$ pour tout $i \in \{0, \dots, l\}$.

Exercice 9. Donner un algorithme permettant de calculer le produit de deux entiers multiprécision en vous basant sur les deux exercices précédents.

Donner le nombre d'opérations machines utilisées.

4 Pour les informaticiens

Implantez ces algorithmes et vérifiez expérimentalement la complexité des algorithmes proposés.

5 Pour les mathématiciens

Reprenons $N = n_0 = \hat{n}_0 + \bar{n}_1 \alpha < 2^B$ et $M = m_0 = \hat{m}_0 + \bar{m}_1 \alpha < 2^B$. On pose $a = \hat{n}_0 \hat{m}_0$, $c = \bar{m}_1 \bar{n}_1$ et $b = (\hat{n}_0 + \bar{n}_1)(\hat{m}_0 + \bar{m}_1)$.

Exercice 10. Montrer que $NM = a + (b - c - a)\alpha + c\alpha^2$.

Exercice 11. Dédurre de l'exercice précédent un algorithme de type Karatsuba pour le produit d'entiers dont vous donnerai la complexité en termes d'opérations machines.