Langages de programmation Python et C

Une librairie de grands entiers

Master Calcul Haute Performance et Simulation — Université de perpignan Via Domitia

1 La librairie

Le but de cette première partie est de mettre au point une librairie travaillant sur de très grands entiers (plusieurs milliers de chiffres décimaux) non signés. Le principe est de simplement stocker un `bigint` (très grand entier) dans un tableau d'entiers de type `unsigned int`. La taille des bigints utilisés étant dynamique (à priori non connue à la compilation), ces tableaux seront alloués dynamiquement, et donc les tailles des bigints devront aussi être stockées. D'où le type bigint suivant :

```
typedef struct bigint {
  unsigned int size;   /* taille */
  unsigned int *value; /* valeur */
} bigint;
```

Un problème inhérent à ce type de librairie est le passage de la représentation en binaire des nombres à leur représentation en décimal. En effet, tous les chiffres binaires interviennent dans la valeur d'un des chiffres décimaux. Il est donc préférable de travailler en une base puissance de 10, et on choisira la plus grande puissance de 10 pouvant être contenue dans un élément du tableau (soit 109 pour un `unsigned int`). Le stockage se fera du poids faible (indice 0) au poids fort (dernier élément du tableau).

Écrire les fonctions arithmétiques suivantes :

```
/* Operators */
int cmp(bigint a, bigint b);        /* Comparaison   */
bigint add(bigint a, bigint b);     /* Addition      */
bigint sub(bigint a, bigint b);     /* Soustraction   (a >= b)*/
bigint product(bigint a, bigint b); /* Produit       */
                                    /* Division      */
void intdiv(bigint a, bigint b, bigint *quotient, bigint *modulo);
bigint pow2n(unsigned int n);       /* 2 puissance n */

/* Input / Output */
void printbigint(bigint n);   /* Affichage sortie standard     */
char *biginttostr(bigint n);  /* Conversion bigint vers chaine */
bigint *strtobigint(char *s); /* Conversion chaine vers bigint */
                              /* Conversion bigint vers chaine */
                              /* Decimales entre first et last */
char *biginttosubstr(bigint n, int first, int last);
```

`cmp` compare a et b, et retourne la position du premier `unsigned int` où a et b diffèrent. Le résultat est positif si $a > b$, négatif si $a < b$ et nul si $a = b$. Tous les calculs se feront élément de tableau par élément de tableau, et devront ainsi être faits en base 10^9. La taille des résultats est calculable à un élément près, et on supprimera celui-ci par réallocation si il est nul.

2 Application : Nombres premiers de Mersenne

Les nombres de Mersenne M_n sont des nombres de la forme $2n - 1$ où n est un nombre premier. On sait (Test de Lucas-Lehmer) de plus qu'un nombre de la forme $2p - 1$ est premier s'il divise $S(p - 1)$ avec $S(1) = 4$ et $S(k + 1) = S(k)2 - 2$ pour $k > 1$. Un nombre premier de Mersenne (voir `http://www.mersenne.org`) est donc un nombre de Mersenne vérifiant le test de Lucas-Lehmer. Écrire un programme qui calcule la valeur du nombre de Mersenne M_n et teste si il est premier. Notez qu'il suffit de calculer $S(k+1)\%M_n$ pendant le test de Lucas-Lehmer et de tester si le résultat final $S(n - 1)\%M_n$ est nul.

Pour réaliser des tests, voici un tableau présentant quelques nombres premiers de Mersenne :

Nombre	Nombre de chiffres	Année	Auteur ou Ordinateur
M_{17}	6	1588	Cataldi
M_{19}	6	1588	Cataldi
M_{31}	10	1772	Euler
M_{127}	39	1876	Lucas
M_{521}	157	1952	SWAC
M_{607}	183	1952	SWAC
M_{4253}	1281	1961	IBM 7090
M_{21701}	6533	1978	Cyber 174
M_{44497}	13395	1979	Cray 1
M_{756839}	227832	1992	Cray 2
$M_{8972593}$	2098960	1999	Pentium II

Le prochain record à battre est un nombre premier de Mersenne à plus de 10 millions de chiffres décimaux : 100000 dollars à la clé. Bonne chance !