

OPÉRATIONS SUR LES GRANDS NOMBRES

Les entiers de CAML sont limités à 2^{31} , ce qui vaut environ 2 milliards. Au-delà, les calculs sont en fait effectués modulo 2^{32} . L'objet du TP est de construire des fonctions de calculs sur les entiers positifs arbitrairement grands. Pour cela, nous allons revenir à l'école primaire.

Vous n'hésitez pas à consulter l'aide-mémoire CAML pour trouver des fonctions prédéfinies bien utiles.

Comme CAML sait quand même faire des additions sur des « petits » entiers, nous allons nous en servir pour inventer la notion de grand entier. On représente un grand entier par une liste d'entiers de la façon suivante :

si n est un entier écrit en base 10 : $n = \overline{a_k a_{k-1} \dots a_0}$ où les symboles a_i sont les chiffres de n , on le découpe en tranches de 3 chiffres qu'on range dans une liste $\ell = (\overline{a_2 a_1 a_0}, \overline{a_5 a_4 a_3}, \dots)$ comme dans l'exemple suivant :

à $n = 12345678$ on associe la liste $\ell = [678; 345; 12]$

C'est en fait l'écriture en base 1000 du nombre n , mais écrite à l'envers.

Dans cette première partie, la locution « grand entier » signifie une liste d'entiers compris entre 0 et 1000, alors que « entier » (seul) signifie nombre entier de CAML.

De nombreuses fonctions seront sans doute récursives encapsulées : rappelons qu'on peut simuler des boucles avec des fonctions récursives.

1 Addition

Pour additionner deux grands entiers, on fait comme à l'école primaire : on additionne d'abord les « unités », on écrit ce qu'il faut, on met de côté la retenue puis on passe « aux dizaines », etc.

1) Écrivez deux fonctions `unite`, `dizaine` de type `int -> int` qui à tout entier classique compris entre 0 et 999 999 s'écrivant $\overline{a_5 a_4 a_3 a_2 a_1 a_0}$ associent respectivement les entiers s'écrivant $\overline{a_2 a_1 a_0}$, $\overline{a_5 a_4 a_3}$.

2) Écrivez une fonction `addition` de type `int list -> int list -> int list` qui additionne deux grands entiers.

2 Soustraction

3) Écrivez une fonction `comp` de type `int list -> int list -> bool` de paramètres t, u deux grands entiers qui donne la valeur de l'inégalité $t \leq u$.

4) Écrivez une fonction `soustraction` de type `int list -> int list -> int list` qui soustrait le deuxième grand entier au premier en vérifiant auparavant si le premier est bien plus grand que le second.

3 Multiplication

5) Écrivez une fonction `produit_unite` de type `int list -> int -> int list` de paramètres un grand entier t et un entier n compris entre 0 et 999 (une unité, donc) qui calcule le produit de t par n .

6) Écrivez une fonction `multiplication` de type `int list -> int list -> int list` qui multiplie deux grands entiers.

À titre d'application, calculez la factorielle de 150.

4 Une interface plus humaine

Découper les grands entiers en listes ou réassembler les listes en grands entiers est vite fastidieux, ce qui augmente le risque d'erreurs de saisie. Pour faciliter l'utilisation des grands nombres, nous souhaitons les écrire comme d'habitude dans une chaîne de caractères et nous laissons à l'ordinateur le soin de transformer cette chaîne en un tableau. Réciproquement, après calculs, l'ordinateur doit réécrire les tableaux en chaînes lisibles par un humain.

7) Écrivez une fonction `lire_nombre` de type `string -> int list` qui effectue cette transformation. Vous trouverez dans l'aide-mémoire CAML des fonctions qui permettent de transformer une chaîne de caractères numériques en entier et vice-versa, ainsi que celle qui permet d'extraire une sous-chaîne. `lire_nombre "99856"` donne comme résultat `[856; 99]`.

8) Écrivez de même une fonction `ecrire_nombre` de type `int list -> string` qui effectue la transformation inverse.

9) Pour finir, nous pouvons offrir une interface de calcul plus pratique. Nous pouvons créer des lois de composition interne qui traduisent les opérations d'addition, de soustraction et de multiplication des grands entiers grâce à la déclaration suivante : **let prefix** `+!` `ch_n ch_p` = Ceci définit un nouveau symbole opératoire `+!`, qu'on utilise sous la forme habituelle `a +! b`.

Avec cette définition, créez trois symboles opératoires `+!`, `-!`, `*!` qui opèrent sur les grands entiers écrits sous forme de chaîne et qui donnent la somme, la différence et le produit sous forme de chaîne, comme dans l'exemple suivant :

<code>"999" +! "999" ;; (* resultat "1998" *)</code>

Enfin, s'il vous reste du temps, vous pourrez affiner les fonctions pour supprimer les zéros non significatifs, ajouter dans la représentation en chaîne des espaces améliorant la lisibilité, et toute autre amélioration cosmétique.

En ajoutant à chaque liste de grand nombre une information supplémentaire contenant le signe (par exemple en travaillant avec des couples (signe, grand entier)), nous pourrions sans grande modification faire des calculs arbitrairement grands dans \mathbb{Z} .