DM2 : Représentation des entiers

À rendre le 20 novembre 2023

Vous pouvez rendre ce DM en version imprimée.

L'objectif de ce DM est de comprendre une manière dont on peut représenter les entiers en binaire.

1 Entiers naturels sur un tableau de taille fixe

On décide dans un premier temps de s'intéresser aux entiers naturels.

Question 1. On cherche à montrer que, pour tout $n \in \mathbb{N}*$, il existe un unique $k \in \mathbb{N}$, et un unique $k+1$-uplet d'entiers a_0, a_1, ..., a_k tels que :

$$\forall j \leqslant k, a_j \in \{0,1\}$$
$$a_k = 1$$
$$\sum_{j=0}^{k} a_j 2^j = n$$

1. Par récurrence forte, montrer l'existence d'une telle expression.
   Pour l'hérédité, on pourra s'intéresser à la vision euclidienne de n par 2.
2. Montrer que pour tout b_0, b_1, ..., b_l dans $\{0,1\}$, on a :
$$\sum_{j=0}^{l} b_j 2^j < 2^{l+1}$$
   On pourra procéder par récurrence.
3. On suppose qu'on a une première décomposition k, a_0, ... a_k, et une deuxième k', a'_0, ..., $a'_{k'}$ qui vérifient ces propriétés. Montrer que ces deux décomposition sont égales.
   On pourra s'intéresser au plus grand j tel que $a'_j \neq a_j$ et utiliser la propriété précédente sur la différence des deux solutions.
4. Conclure.

Correction :

1. On procède par récurrence sur n en remarquant que 1 s'écrit de manière unique avec $k = 0$ et a_0, et en remarquant que si $n = 2p + r > 1$, et qu'on peut écrire $p = \sum_{j=0}^{k} a_j 2^j$, alors $n = 2\sum_{j=0}^{k}(a_j 2^j) + r$ et donc $n = \sum_{j=1}^{k} a_j 2^{j+1} + r$.
2. $\sum_{j=0}^{l} b_j 2^j \leqslant \sum_{j=0}^{l} 2^j$ or $\sum_{j=0}^{l} 2^j = 2^{l+1} - 1$ donc $\sum_{j=0}^{l} b_j 2^j < 2^{l+1}$.
3. Quitte à rajouter des zéro au début de l'une des décomposition, on peut supposer que $k = k'$.
   Soit j le plus grand indice tel que $a'_j \neq a_j$. Quitte à échanger le rôle des deux décompositions, sans perte de généralité on suppose que $a' = 0$ et $a = 1$.

On a que :

$$\begin{aligned} 0 &= \sum_{i=0}^{k} a_i 2^i - \sum_{i=0}^{k} a'_i 2^i && \text{car il s'agit de deux décompositions.} \\ &= 2^j + \sum_{i=0}^{j-1} (a_i - a'_i) 2^i \\ &\geqslant 2^j - \sum_{i=0}^{j-1} 2^i && \text{par inégalité triangulaire et en remarquant que } |a_i - a'_i| \leqslant 1. \\ &> 0 \text{ d'après la question précédente.} \end{aligned}$$

4. On a montré l'unicité et l'existence d'une telle décomposition.

Anisi, pour tout nombre strictement positifs $n > 0$, il existe un unique $m \in \mathbb{N}$ et une unique suite finie a_0, a_1, ..., a_m d'entiers dans $\{0, 1\}$ tels que

$$n = \sum_{k=0}^{m} a_k^k$$

On peut alors écrire $n = \overline{a_k a_{k-1} ... a_1 a_0}^2$.

Il s'agit de la représentation en base 2 de n, et on peut se servir de cette représentation pour encoder un nombre en binaire sur un tableau de taille fixe.

On pose de manière globale :

```
let taille_entier = 8
```

Il s'agira de la taille de tableaux qu'on utilise pour représenter les entiers. Si un nombre n peut être représenté par $\overline{a_k a_{k-1} ... a_1 a_0}^2$ avec $k < 8$, quitte à rajouter des 0 avant sa représentation binaire, on pourra donc le représenter dans un tableau de cette taille. Un nombre $\overline{a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0}^2$ pourra être représenté par le tableau suivant (ici, a_7 peut être nul) :

a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0

Ainsi, le nombre 11, peut s'écrire $11 = 2^3 + 2^1 + 2^0$, et donc, dans un tableau de taille 8, on peut représenter 11 par le tableau suivant :

0	0	0	0	1	0	1	1
				2^3		2^1	2^0

Question 2. Quel est le plus grand nombre qui peut être représenté de cette manière avec cette taille de tableau ? Que se passe-t-il si `taille_entier` est égal à 16 ?

Correction : L'entier le plus grnad correspond au tableau rempli de zéro égal à $\sum_{k=0}^{7} 2^k = 255$.
Si la taille du tableau est égale à 16, on obtiens $\sum_{k=0}^{1} 52^k = 65535$.

Question 3.

1. Proposer une fonction `vers_binaire` de signature `int -> int array` qui renvoie la représentation binaire d'un entier en argument.

2. Proposer une fonction `depuis_binaire` de signature `int array -> int` qui renvoie l'entier correspondant à une représentation binaire en argument.

Correction :

1.
2. On réalise un parcourt.

```
let depuis_binaire entier =
```

Question 4. L'objectif de cette question est d'implémenter quelques fonctions qui permentent de modifier un tableau qui représente un entier. On veillera à bien modifier le tableau concerné.

Attention !

Dans les fonctions suivantes, on prendra soin de ne pas repasser par les entiers OCaml : l'objectif est de manipuler le tableau d'entier directement.

Ainsi, il n'est PAS possible d'écrire le code suivant :

```
let incrementer t = vers_binaire (1 + depuis_binaire t)
```

On ne lèvera pas d'erreur en cas de dépassement, au lieu de ça, on ignorera la retenue. Par exemple, en appliquant `incrementer` à `[|1; 1; 1; 1; 1; 1; 1; 1|]`, on obtient `[|0; 0; 0; 0; 0; 0; 0; 0|]`.

1. Proposer une fonction `incrementer` de signature `int array -> unit` qui modifie une représentation binaire d'un entier n pour la transformer en la représentation binaire de $n+1$.
2. Proposer une fonction `decrementer` de signature `int array -> unit` qui modifie une représentation binaire d'un entier n pour la transformer en la représentation binaire de $n-1$.
3. Proposer une fonction `ajouter` de signature `int array -> int array -> unit` qui modifie le premier tableau en entrée, pour y ajouter la valeur du second tableau en entrée.
4. Proposer une fonction `plus_grand` de signature `int array -> int array -> bool` qui renvoie `true` si le nombre représenté par le premier tableau en argument est plus grand que le nombre représenté par le second tableau en argument.
5. Proposer une fonction `soustraire` de signature `int array -> int array -> bool` qui modifie le premier tableau en entrée pour y soustraire la valeur du second tableau en entrée.

Question 5. Pour une question de place en mémoire, étant donné que les cases dans le tableau ne peuvent prendre que la valeur 0 ou 1, on peut à la place utiliser un tableau de booléens (`false` pour 0, `true` pour 1). Comment modifier les fonctions précédentes pour s'adapter à cette représentation ? Quel définition de type utiliser ?

Il n'est pas nécessaire de modifier explicitement les fonctions, il est possible de présenter les différences principales dans le code avec l'implémentation proposées dans les questions précédentes.

Question 6. On peut décider d'utiliser une base autre que 2. Par exemple utiliser une base b (les a_i doivent alors être dans l'ensemble $\{0, 1, ..., b-1\}$). Comment modifier les fonctions précédentes pour s'adapter à cette base ?

Il n'est pas nécessaire de modifier explicitement les fonctions, il est possible de présenter les différences algorithmiques principales.

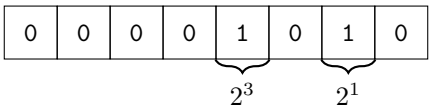
2 Représentation par complément à deux

On s'intéresse désormais à la représentation des entiers relatifss. Une solution serait de réserver une case dans le tableau pour le signe, et le reste du tableau pour la valeur absolue, mais ce n'est pas très pratique. À la place, la représentation des entiers négatifs se fait par **complément à deux**.

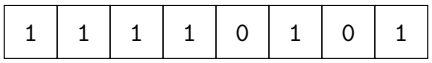
Les nombres positifs sont représentés usuellement, mais les nombres négatifs sont représentés d'une autre manière. Pour représenter $-n$, on procède de la manière suivante :

— On prend la représentation usuelle de n ;
— On transforme tous les 0 de la représentation en 1, et tous les 0 en 1 ;
— On rajoute 1 à cette représentation (en ignorant les dépassement).

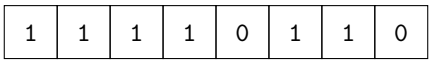
Ainsi, pour représenter −10, on calcule la représentation de 10 :



On inverse ensuite les 0 et les 1 :



Enfin, on ajoute 1 :



Pour distinguer entre les nombres négatifs et les nombres positifs à partir d'une représentation, on considère que si la représentation du nombre commence par un 0, c'est un nombre positif, et si la représentation du nombre commence par 1, c'est un nombre strictement négatif.

Ainsi, le tableau `[|1; 0; 0; 1; 0; 0; 1; 1|]` représente l'entier −109.

Question 7. Montrer qu'un nombre représenté par le tableau `[| a7; a6; a5; a4; a3; a2; a1; a0|]` est égal à :

$$-a_7 2^7 + a_6 2^6 + a_5 2^5 + a_4 2^4 + a_3 2^3 + a_2 2^2 + a_1 2^1 + a_0 2^0$$

Question 8. Quelle est le plus grand nombre qu'il est possible de représenter ? Quelle est le plus petit nombre qu'il est possible de représenter ?

Question 9.

1. Proposer une fonction `vers_binaire_signe` de signature `int -> int array` qui renvoie la représentation binaire en complément à deux d'un entier relatif en argument.
2. Proposer une fonction `depuis_binaire_signe` de signature `int array -> int` qui renvoie l'entier relatif correspondant à une représentation binaire en complément à deux en argument.

Question 10. Montrer que les fonctions `incrementer` et `ajouter` pour les entiers naturels sont compatibles avec les entiers en complément à deux (quitte à ignorer les dépassements).

Question 11. Que se passe-t-il lorsqu'on fait `incrementer` sur un tableau `[|0; 1; 1; 1; 1; 1; 1; 1|]` ? Que se passe-t-il du point de vue des entiers relatifs représentés ? Est-ce que ce comportement peut être observé avec les entiers OCaml ?

Question 12.

1. Proposer une fonction `plus_grand_signe` de signature `int array -> int array -> bool` qui détermine si l'entier représenté par le premier tableau en argument est plus grand que l'entier représenté par le deuxième tableau en argument.
2. Proposer une fonction `inverser` de signature `int array -> unit` qui modifie la représentation binaire en complément à deux de n pour qu'elle représente $-n$.

3 Représentation à taille variable

On désire représenter les entiers naturels à taille variable. Pour ce faire, on fait le choix de représenter les entiers par des listes de booléens, en commençant par leur bit de poid faible (c'est-à-dire le booléen qui représente le coefficient le plus faible de la puissance de 2).

Ainsi, 6 sera représenté par [false ; true; true].

Question 13. De quel type est cette représentations des entiers ?

Question 14. Quelle contrainte doit on imposer pour que la représentation soit unique ?

Question 15. Comment représenter 0 avec cette contrainte d'unicité ?

Question 16. Comment implémenter les fonctions incrementer, ajouter, decrementer, soustraire, multiplier ?
On pourra se contenter d'une description générale de la manière d'implémenter ces fonctions.