Arithmétique en OCaml

Florian Bourse

1 Écriture décimale

Dans cette section, on pourra s'aider de la division euclidienne d'un nombre par 10. Si $n = 10 \times d + u$, u est son chiffre des unités et d son nombre de dizaines. En OCaml, d est `n / 10` et u est `n mod 10`.

Question 1. Écrire une fonction `inverser : int -> int = <fun>` qui prend en entrée un entier n et qui renvoie le nombre n' tel que les écritures décimales de n et n' sont inversées.
Exemple : `inverser 123` renvoie 321
Indication : *On pourra créer une fonction récursive auxiliaire qui serait appelée sur les entrées suivantes par exemple :*

123	0
12	3
1	32
0	321

Question 2. Écrire une fonction `palindrome : int -> bool = <fun>` qui prend en entrée un entier n et qui renvoie `true` si et seulement si n est un palindrome, c'est-à-dire qui se lit de la même manière de gauche à droite et de droite à gauche.

Question 3. Écrire une fonction `solve : int -> int = <fun>` qui prend en entrée un entier n et qui renvoie le plus petit $k \geq n$ tel que k multiplié par son écriture inversée (en base 10) donne un palindrome.
Exemple : `solve 2018` renvoie 2021 car $2021 \times 1202 = 2429242$

Question 4. Écrire une fonction `combiendefois42 : int -> int = <fun>` qui prend en entrée un entier n et qui renvoie le nombre de fois que 42 apparaît dans sa représentation décimale.
Exemple : `combiendefois42 4256423` renvoie 2

Question 5. Écrire une fonction `combiendefois : int -> int -> int = <fun>` qui généralise la fonction précédente et prend en entrée un entier k et un entier n et qui renvoie le nombre de fois que k apparaît dans la représentation décimale de n.
Exemple : `combiendefois 42` doit être la même fonction que `combiendefois42`
Attention : *Le cas $k = 0$ est à traiter à part. On peut dans un premier temps, coder la fonction en supposant $k > 0$.*

2 Logarithme en base 2

On rappelle que $\log_2(x)$ est le nombre tel que $2^{\log_2(x)} = x$, c'est-à-dire $\log_2(x) = \frac{\ln(x)}{\ln(2)}$.

Question 6. Écrire une fonction `log2 : int -> int = <fun>` qui prend en entrée un entier n et qui renvoie $\lceil \log_2(n) \rceil$.
Exemple : `log2 123` renvoie 7

Le logarithme itéré $\log_2^*$ est le nombre de fois qu'il faut appliquer $\log_2$ à un nombre avant d'arriver à un nombre inférieur ou égal à 1.

Question 7. Écrire une fonction `logstar2 : int -> int = <fun>` qui prend en entrée un entier n et qui renvoie $\log_2^*(n)$.
Exemple : `logstar2 2020` renvoie 4, `logstar2 1000000` renvoie 5

3 Couplage de Cantor

Le couplage de Cantor exhibe une bijection entre $\mathbb{N} \times \mathbb{N}$ et $\mathbb{N}$, en numérotant la grille des éléments de $\mathbb{N} \times \mathbb{N}$ comme illustré sur la figure page 3

Question 8. Écrire une fonction `nro2point : int -> int * int = <fun>` qui prend en entrée un entier n et qui renvoie les coordonnées entières (x, y) du point numéroté par n.
Exemple : `nro2point 42` renvoie $(2, 6)$

Question 9. Écrire une fonction `point2nro : int * int -> int = <fun>` qui prend en entrée un point (x, y) et qui renvoie son numéro n.
Exemple : `nro2point (2,6)` renvoie 42

Question 10. On définit la fonction suivante :

```
let rec iterate n =
  if n = 1 then
    true
  else if n = 0 then
    false
  else
    let (x,y) = nro2point n in
    iterate (x+y);;
```

Étudier cette fonction.

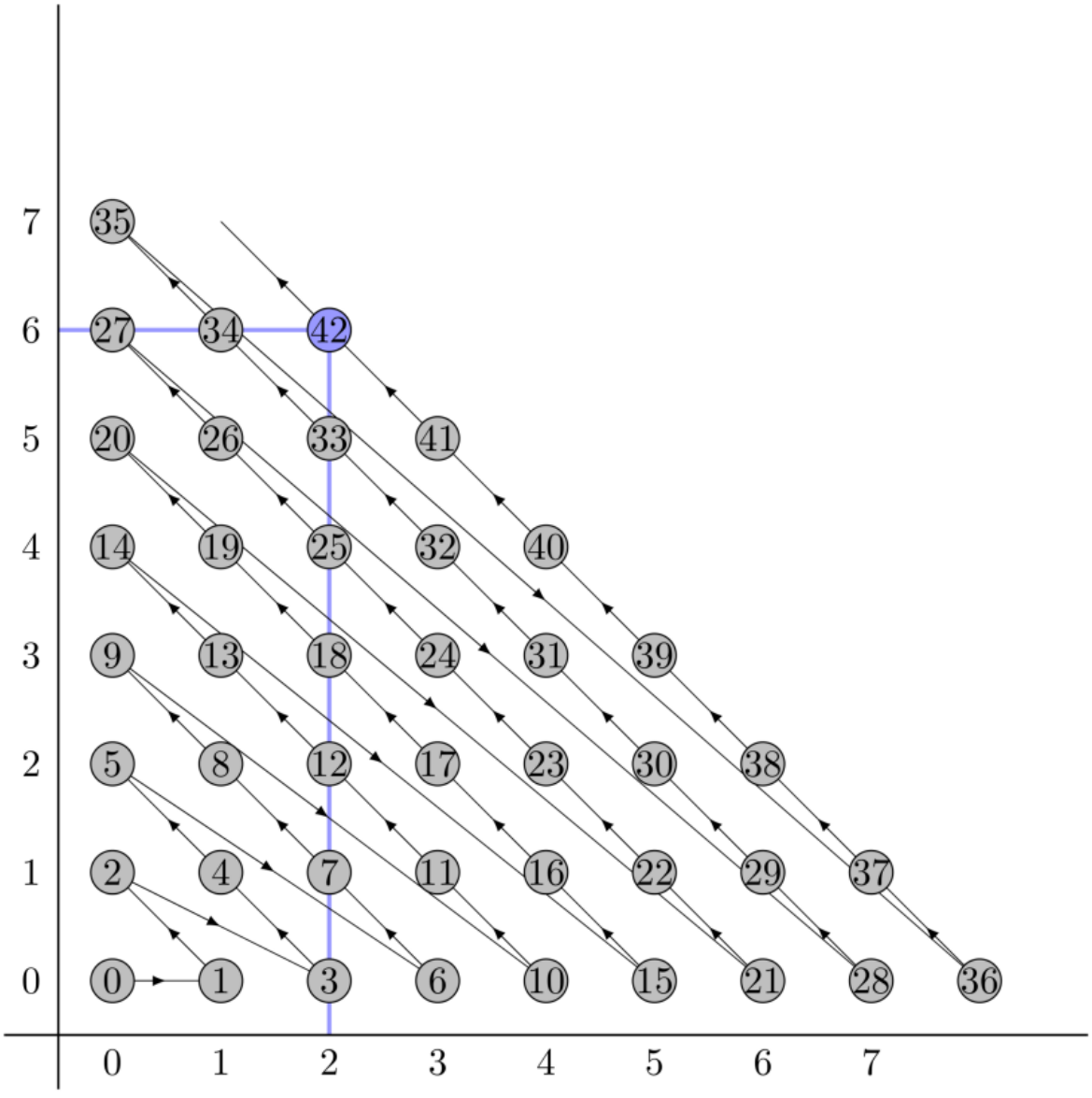

FIGURE 1 – Couplage de Cantor

4 Décompositions en carré parfaits

Question 11. Écrire une fonction `decomp : int -> int = <fun>` qui prend en entrée un entier n et qui renvoie le nombre de façons de décomposer n comme une somme de carrés parfaits distincts strictement positifs.
Exemple : `decomp 100` renvoie 3 car $100 = 1^2 + 3^2 + 4^2 + 5^2 + 7^2 = 6^2 + 8^2 = 10^2$
`decomp 4` renvoie 1 car $4 = 2^2$, et $4 = 1^2 + 1^2 + 1^2 + 1^2$ n'est pas compté car les termes ne sont pas distincts *Indication : On pourra définir une fonction auxiliaire récursive*

```
decomp_aux : int -> int -> int = <fun>
```

qui prend en entrée un entier n et un entier i et qui résoud le problème en ne considérant pas comme termes possibles les carrés plus petits que $i \times i$
Remarque : On ne cherche pas à trouver ces décompositions, juste à les dénombrer.

Question 12. Écrire une fonction `decomp_bis : int -> int = <fun>` qui prend en entrée un entier n et qui renvoie le nombre de façons de décomposer n comme une somme de carrés parfaits strictement positifs. On autorise les doublons.
Exemple : `decomp_bis 4` renvoie 2 car cette fois, on compte $4 = 1^2 + 1^2 + 1^2 + 1^2$
`decomp_bis 100` renvoie 1116.