Correction des exercices de TD

Table des matières

1 Calculabilité

1.1 Divers

Exercice 1 - Paradoxe

Montrer que les problèmes suivants engendrent un paradoxe.

1. Le conseil municipal d'un village vote un arrêté municipal qui enjoint à son barbier (masculin) de raser tous les habitants masculins du village qui ne se rasent pas eux-même et seulement ceux-ci.
2. Un crocodile s'empare d'un bébé et dit à la mère : « si tu devines ce que je vais faire, je te rends le bébé, sinon je le dévore. ». En supposant que le crocodile tienne parole, que doit dire la mère pour que le crocodile rende l'enfant à sa mère ? Une réponse usuelle de la mère est : « Tu vas le dévorer ! »

1. Le barbier, s'il se rase, se rase lui-même, mais il est aussi rasé par le barbier. S'il ne se rase pas, il doit être rasé par le barbier, c'est à dire lui-même, donc il se rase. L'énoncé est faux (impossible).
2. Si le crocodile rend le bébé à la mère avec cette réponse, c'est qu'il comptait le dévorer. Seulement, s'il a l'intention de rendre le bébé, c'est qu'il n'a pas l'intention de le dévorer, donc il le dévorera.

Exercice 2 - Une preuve incorrecte

Nous considérons la fonction suivante donnée par l'algorithme 1 :

Algorithm 1: La fonction de Collatz

```
begin
    while n ≠ 1 do
        if n mod 2 = 0 then
            n := n/2
        else
            n := 3 × n + 1
```

Actuellement nous ne savons pas si cette fonction termine $\forall n$. Êtes-vous d'accord avec la preuve suivante ? « Si le problème de l'arrêt était décidable, il suffirait de l'appliquer à ce programme pour savoir si son exécution s'arrête. Or, on ne sait pas si son exécution s'arrête. D'où la contradiction. »

Le raisonnement est faux, la preuve est donc incorrecte. En effet, « on ne sait pas si son exécution s'arrête » signifie que soit elle s'arrête, soit elle ne s'arrête pas, on répète la question. Il n'y a pas de lien logique entre la première et la seconde phrase.

1.2 Variations sur le codage

Exercice 3 - Codage de couples d'entiers

Soit $Rang : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ tel que $Rang(x, y) = \frac{(x+y)(x+y+1)}{2} + y$.

1. Donner une version récursive de la fonction $Rang$.
2. Donner la fonction inverse.
3. Calculer $Rang(4, 5)$. Donner le couple pour lequel la valeur du codage est 8.

1. $RangRec = \begin{cases} 0 & \text{si } x = 0 \text{ et } y = 0 \\ RangRec(0, x-1) + 1 & \text{si } y = 0 \\ RangRec(x+1, y-1) + 1 & \text{sinon} \end{cases}$.
2. On pose la fonction inverse $RangInv(n) : \mathbb{N} \to \mathbb{N} \times \mathbb{N}$. On cherche d'abord $x + y$, on prend donc $x + y = max\{m \mid \frac{m(m+1)}{2} \leq n\}$. On pose $t = x + y$. Ainsi, comme $n = \frac{t(t+1)}{2} + y$, on a $y = n - \frac{t(t+1)}{2}$. De plus, comme $t = x + y$, pour retrouver x, il suffit de prendre $x = t - y$.
3. $Rang(4, 5) = \frac{(4+5)(4+5+1)}{2} + 5 = \frac{9 \times 10}{2} + 5 = 50$. Pour $n = 8$, on cherche d'abord t. On a $t = max\{1, 2, 3\} = 3$, et $\frac{t(t+1)}{2} = 6$. On a donc $y = 8 - 6 = 2$ et $x = 3 - 2 = 1$. Le couple codé par $n = 8$ est $(1, 2)$.

Exercice 4 - Codage de triplets

Soit c la fonction de codage pour les couples d'entiers vue dans l'exercice précédent.

1. Soit h la fonction de codage pour les triplets définie par $h(x, y, z) = c(c(x, y), z)$. Quel est le doublet codé par 67 ? Quel est le triplet codé par 67 ?
2. Le couple (z, t) succède au couple (x, y) si $c(z, t) = c(x, y) + 1$. Écrire la fonction successeur qui prend en paramètre un couple et retourne le couple successeur.

1. Pour $n = 67$, on cherche d'abord t. On a $t = max\{m|m \leq 11\} = 11$. En effet, $\frac{t(t+1)}{2} = \frac{11\times 12}{2} = \frac{132}{2} = 66$. On a donc $y = 67 - 66 = 1$ et $x = 11 - 1 = 10$. Comme 67 code $(10, 1)$, pour avoir le triplet, on veut le couple codé par $n = 10$. De nouveau, on cherche $t = max\{1, 2, 3, 4\} = 4$. En effet, $\frac{4\times 5}{2} = 10$. Donc $y = 10 - 10 = 0$ et $x = 4 - 0 = 4$. Le couple codé par $n = 10$ est $(4, 0)$ et ainsi le triplet codé par $n = 67$ est $(4, 0, 1)$.
2. La fonction c fait augmenter les couples comme une diagonale. Prenons les premiers couples :

$c(x,y)$	0	1	2	3	4	5	6	7	8	9
(x,y)	$(0,0)$	$(1,0)$	$(0,1)$	$(2,0)$	$(1,1)$	$(0,2)$	$(3,0)$	$(2,1)$	$(1,2)$	$(0,3)$

On remarque que le successeur de (x, y) est $(x - 1, y + 1)$ (qui est d'ailleurs l'inverse de ce qu'on avait fait à l'exercice précédent pour $RangRec$), et que si $x = 0$, le successeur est $(y+1, 0)$. Cela nous donne la fonction suivante : $successeur(x, y) = \begin{cases} (y+1, 0) & \text{si } x = 0 \\ (x-1, y+1) & \text{sinon} \end{cases}$

Exercice 5 - Étude d'une équation fonctionnelle dans $\mathbb{N}$

Soit f une application de $\mathbb{N}$ dans $\mathbb{N}$ telle que : $\forall (m, n) \in \mathbb{N}, f(m^2+n^2) = f(m)^2+f(n)^2$. Nous voulons montrer que f est :

— l'application nulle, donnée par : $\forall n \in \mathbb{N}, f(n) = 0$,

— l'application identité, donnée par : $\forall n \in \mathbb{N}, f(n) = n$.

Nous supposerons que a est l'entier naturel $f(1)$.

1. Montrer que $f(0) = 0$. En déduire que $\forall n \in \mathbb{N}$, on a $f(n^2) = f(n)^2$.
2. Montrer alors que $a^2 = a$, donc que a est égal à 0 ou à 1.
3. Vérifier successivement les égalités $f(2) = 2a$, $f(4) = 4a$ et $f(5) = 5a$.
4. Utiliser les valeurs $f(4)$ et $f(5)$ pour montrer que $f(3) = 3a$.
5. Utiliser les valeurs de $f(1)$ et de $f(5)$ pour montrer que $f(7) = 7a$.
6. Montrer que $f(8) = 8a$, $f(9) = 9a$, $f(10) = 10a$ et $f(6) = 6a$.
7. Observer que

$$\forall m, \text{ on a } \begin{cases} (2k)^2 + (k-5)^2 = (2k-4)^2 + (k+3)^2 \\ (2k+1)^2 + (k-2)^2 = (2k-1)^2 + (k+2)^2 \end{cases}$$

   Montrer que $\forall n$, on a $f(n) = an$.
8. Conclure.

1. On a $m^2 + n^2 = 0 \Rightarrow m = 0$ et $n = 0$:

$$\begin{aligned} & f(m^2+n^2) = f(m)^2 + f(n)^2 \\ & \Rightarrow f(0) = f(0)^2 + f(0)^2 \\ & \Rightarrow f(0) = 2f(0)^2 \\ & \Rightarrow f(0) - 2f(0)^2 = 0 \\ & \Rightarrow f(0)(1 - 2f(0)) = 0 \end{aligned}$$

Il y a deux cas. Soit $f(0) = 0$, soit $1 - 2f(0) = 0 \Rightarrow f(0) = \frac{1}{2}$. Or, f est une fonction de $\mathbb{N}$ dans $\mathbb{N}$, donc le seul résultat possible ici est $f(0) = 0$. On en déduit que $f(n^2) = f(0^2+n^2) = f(0)^2+f(n)^2 = f(n)^2$.

2. On sait que $f(n^2) = f(n)^2$. De plus, $1^2 = 1$, donc $f(1) = f(1)^2 \Rightarrow a = a^2$. Dans les entiers, seulement 0 et 1 vérifient cette égalité.
3. $f(2) = f(1^2 + 1^2) = f(1)^2 + f(1)^2 = a^2 + a^2 = 2a^2 = 2a$.
   $f(4) = f(0^2 + 2^2) = f(0)^2 + f(2)^2 = (2a)^2 = 4a^2 = 4a$.
   $f(5) = f(1^2 + 2^2) = f(1)^2 + f(2)^2 = a^2 + 4a^2 = 5a^2 = 5a$.
4. $f(5^2) = f(25) = f(3^2 + 4^2) = f(3)^2 + f(4)^2 = f(3)^2 + 16a^2$. De plus, $f(5^2) = f(5)^2 = (5a)^2 = 25a^2$. On a donc $f(3)^2 = 25a^2 - 16a^2 = 9a^2$. Ainsi, $f(3) = \sqrt{9a^2} = 3a$.
5. $f(5^2+5^2) = f(1^2+7^2) = f(50)$. On a $f(50) = f(5)^2+f(5)^2 = 50a^2$ et $f(50) = f(1)^2+f(7)^2 = a^2+f(7)^2$. On a donc $f(7)^2 = 50a^2 - a^2 = 49a^2$. Ainsi, $f(7) = \sqrt{49a^2} = 7a$.
6. $f(8) = f(2^2 + 2^2) = f(2)^2 + f(2)^2 = 4a^2 + 4a^2 = 8a^2 = 8a$.
   $f(9) = f(3^2 + 0^2) = f(3)^2 + f(0)^2 = 9a^2 = 9a$.
   $f(10) = f(3^2 + 1^2) = f(3)^2 + f(1)^2 = 9a^2 + a^2 = 10a^2 = 10a$.
   $f(10^2) = f(8^2 + 6^2) = f(100)$. On a $f(100) = f(10^2) = f(10)^2 = 100a^2$ et $f(100) = f(8^2 + 6^2) = f(8)^2 + f(6)^2 = 64a^2 + f(6)^2$. On a donc $f(6)^2 = 100a^2 - 64a^2 = 36a^2$. Ainsi, $f(6) = \sqrt{36a^2} = 6a$.
7. On veut prouver l'hypothèse $H(n) : \forall n, f(n) = an$. On prouve ça par induction :
   Base On a prouvé précédemment tous les cas pour $n \leq 10$.
   Induction On suppose $\forall i < n, H(i)$. Montrons $H(n)$. Il y a deux cas : n pair, c'est à dire qu'il existe k tel que $n = 2k$ ou bien n impair, c'est à dire qu'il existe k tel que $n = 2k + 1$.
   - n est pair. On sait que $f((2k)^2 + (k-5)^2) = f((2k-4)^2 + (k+3)^2)$:
$$
\begin{aligned}
& f((2k)^2 + (k-5)^2) = f((2k-4)^2 + (k+3)^2) \\
& \Rightarrow f(2k)^2 + f(k-5)^2 = f(2k-4)^2 + f(k+3)^2 \\
& \Rightarrow f(2k)^2 = f(2k-4)^2 + f(k+3)^2 - f(k-5)^2 \\
& \Rightarrow f(2k)^2 = a^2(2k-4)^2 + a^2(k+3)^2 - a^2(k-5)^2 \\
& \Rightarrow f(2k)^2 = a^2(4k^2 - 16k + 16) + a^2(k^2 + 6k + 9) - a^2(k^2 - 10k + 25) \\
& \Rightarrow f(2k)^2 = a^2(4k^2) \\
& \Rightarrow f(2k) = \sqrt{4a^2k^2} \\
& \Rightarrow f(2k) = a2k
\end{aligned}
$$
   - n est impair. On sait que $f((2k+1)^2 + (k-2)^2) = f((2k-1)^2 + (k-2)^2)$:
$$
\begin{aligned}
& f((2k+1)^2 + (k-2)^2) = f((2k-1)^2 + (k+2)^2) \\
& \Rightarrow f(2k+1)^2 + f(k-2)^2 = f(2k-1)^2 + f(k+2)^2 \\
& \Rightarrow f(2k+1)^2 = f(2k-1)^2 + f(k+2)^2 - f(k-2) \\
& \Rightarrow f(2k+1)^2 = a^2(2k-1)^2 + a^2(k+2)^2 - a^2(k-2)^2 \\
& \Rightarrow f(2k+1)^2 = a^2(4k^2 - 4k + 1) + a^2(k^2 + 4k + 4) - a^2(k^2 - 4k + 4) \\
& \Rightarrow f(2k+1)^2 = a^2(4k^2 + 4k + 1) \\
& \Rightarrow f(2k+1) = \sqrt{a^2(4k^2 + 4k + 1)} \\
& \Rightarrow f(2k+1) = a(2k+1)
\end{aligned}
$$
   Conclusion On a prouvé que $\forall i \leq 10, H(i)$ et $\forall i < n, H(i) \Rightarrow H(n)$, on a donc bien $\forall n, f(n) = an$.
8. On a prouvé que $\forall n, f(n) = an$. De plus, on sait que $a = 0$ ou $a = 1$. Ainsi, il y a deux cas :
   - $\forall n, f(n) = 0 \times n = 0$ (application nulle),
   - $\forall n, f(n) = a \times n = n$ (application identité).

   On a prouvé que les deux seules applications de $\mathbb{N} \to \mathbb{N}$ telles que $\forall (m, n) \in \mathbb{N}, f(m^2 + n^2) = f(m)^2 + f(n)^2$ sont l'application nulle et l'application identité.

Exercice 6 - Codage rationnels

Proposer un codage pour les nombres rationnels.

On peut proposer un codage naïf : toute fraction rationnelle $\frac{a}{b}$ se réduit en fraction $\frac{p}{q}$ avec p, q premiers. Pour coder les rationnels, on pourrait prendre $c(p, q)$. Cependant, ce codage est très inefficace, car il y a énormément de couples non premiers entre eux.

Une méthode moins naïve de faire serait de poser la fonction σ qui prend $(p + q)$ et ordonne par p (numérateur) croissant lors de l'égalité :

$\sigma(p/q)$	1	2	3	4	5	6
$\frac{p}{q}$	$\frac{0}{1}$	$\frac{1}{1}$	$\frac{1}{2}$	$\frac{2}{1}$	$\frac{1}{3}$	$\frac{3}{1}$

On remarque qu'on saute $\frac{2}{2}$, car on peut réduire la fraction a $\frac{1}{1}$.

Exercice 7 - Codage des listes d'entiers

Pour coder les listes d'entiers, peut-on :

1. Faire la somme des entiers de la liste, et à somme égale prendre l'ordre lexicographique ?
2. Faire comme pour les mots : prendre les listes les plus courtes d'abord et à égalité de longueur l'ordre lexicographique ?

1. Non, car dans ce cas, on n'aurait que les listes qui contiennent des 0 : $(0), (0, 0), (0, 0, 0), \ldots$.
2. Non, car dans ce cas, on n'aurait que les listes de longueur 1 : $(0), (1), (2), \ldots$.

Exercice 8 - Codage de listes d'entiers

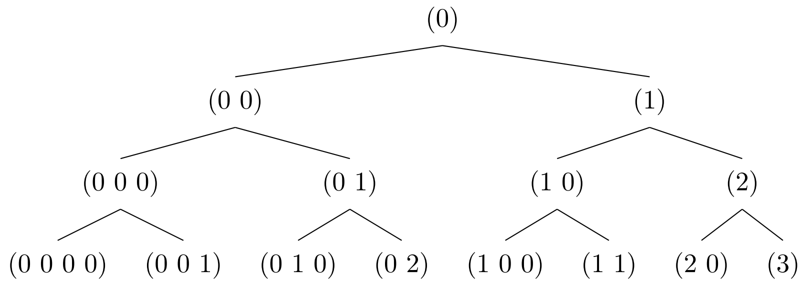
On ordonne les listes de la façon suivante :

$$\sigma(l) = \text{somme des entiers de la liste} + \text{longueur de la liste}$$

Puis à valeur de σ égale on ordonne dans l'ordre lexicographique.
On note U_k l'ensemble des listes l telles que $\sigma(l) = k$ et $u_k = |U_k|$.

1. Donner les ensemble $U_i, i = 0, \ldots, 4$.
2. Montrer que $u_k = 2^{k-1}, \forall k \geq 1$.
3. Quelle est la première liste de $U_k, \forall k \in \mathbb{N}^*$ et la dernière ?
4. Donner la fonction de codage en version itérative et récursive (resp. décodage).

1. $U_0 = \{()\}$ la liste vide.
   $U_1 = \{(0)\}$.
   $U_2 = \{(0, 0), (1)\}$.
   $U_3 = \{(0, 0, 0), (1, 0), (0, 1), (2)\}$.
   $U_4 = \{(0, 0, 0, 0), (1, 0, 0), (0, 1, 0), (0, 0, 1), (2, 0), (1, 1), (0, 2), (3)\}$.
2. Pour tout $k \geq 1$, on peut exhiber la méthode de construction suivante : pour chaque élément l de U_{k-1}, le fils gauche est l à laquelle est ajoutée 0 à la fin, et le fils droit est l en ajoutant 1 à son dernier élément :



Arbre de construction des listes d'entiers

On peut ainsi faire une preuve par induction :

Base On a prouvé le cas pour $k \leq 4$.

Induction On suppose $\forall i \leq n$, $u_i = 2^{i-1}$. Prouvons le pour u_{n+1}. Comme $u_n = 2^{n-1}$ et que la construction est inductive sous forme d'arbre, il y a 2 fils pour chaque élément de U_n donc la taille est $2 \times u_n = 2 \times 2^{n-1} = 2^n$.

Conclusion On a montré la base, et pour tout $k \geq 1$, on a bien $u_k = 2^{k-1} \Rightarrow u_{k+1} = 2^k$.

3. La première liste de U_k est la liste $(0, \ldots, 0)$ composée de k 0, et la dernière est le singleton $(k-1)$.
4. On peut donner la version récursive suivante, qui se déduit pratiquement immédiatement de l'arbre de construction des listes :

Algorithm 2: Codage d'une liste d'entier

begin
- **if** $n = 1$ *et* $e_0 = 0$ **then**
  - **return** 1
- **if** $e_n = 0$ **then**
  - **return** $2 \times codage(e_0, \ldots, e_{n-1})$
- **return** $2 \times codage(e_0, \ldots, e_n - 1) + 1$

Le décodage est ainsi intuitif : on prend un compteur, si $n \mod 2 = 0$, on ajoute 0 à la liste, sinon, tant que $n \mod 2 = 1$, on ajoute 1 au compteur :

Algorithm 3: Décodage en liste d'entier

begin
- **if** $n = 1$ **then**
  - **return** (0)
- **if** $n \mod 2 = 0$ **then**
  - **return** $cons(decodage(n/2), 0)$
- $L := decodage(\lfloor n/2 \rfloor)$;
- $L_{[-1]} := L_{[-1]} + 1$;
- **return** L

$L_{[-1]}$ correspond au dernier élément de la liste.

Exercice 9 - Codage d'entiers

Soit la fonction f suivante de $\mathbb{N}^* \to \mathbb{N}$:

$$f(n) = k \text{ si } n = 2^k$$
$$f(n) = f(n/2) \text{ si } n \text{ est pair et n'est pas une puissance de 2}$$
$$f(n) = f(3n+1) \text{ sinon}$$

Nous appelons $A_i = \{x \mid f(x) = i\}$.

1. Donner quelques éléments de A_i, $\forall i \in \{1, 2, 3, 4, 5, 6\}$.
2. Donner un algorithme qui prend i en paramètre et qui affiche tous les éléments de A_i.
3. Donner un algorithme qui affiche $A_1 \cup A_2$.
4. Donner un algorithme qui affiche $A_4 \cup A_6$.

1. $A_1 = \{2\}$
   $A_2 = \{4\}$
   $A_3 = \{8\}$
   $A_4 = \{16, 3, 5, 10, 24, 48, 20, 40, \dots\}$
   $A_5 = \{32, \dots\}$
   $A_6 = \{64, 21, 42, 84, \dots\}$
2. Intuitivement, on voudrait faire l'algorithme suivant :

   Algorithm 4: Afficher A_i

   ```
   begin
       k := 0;
       while true do
           if f(k) == i then
               afficher(k)
           k := k + 1
   ```

   Seulement, on ne sait pas si $f(k)$ est calculable. Par exemple, on pourrait avoir $f(10^{38}) = i$, f qui boucle pour $k = 10^{39}$, et $f(10^{40}) = i$. Dans ce cas, 10^{40} ne sera jamais affiché alors qu'il fait bel et bien partie de A_i. Cet algorithme est donc faux.

Ainsi, c'est l'algorithme suivant qui affiche A_i :

Algorithm 5: Afficher A_i

```
begin
    afficher(2^i);
    L := ∅;
    if (2^i − 1) mod 3 = 0 then
        L = {(2^i − 1)/3}
    while L ≠ ∅ do
        n := tete(L);
        afficher(n);
        L := queue(L);
        L := ajouter(2n, L);
        if (n−1)/3 est impair then
            L := ajouter((n − 1)/3, L)
```

3. Comme A_1 et A_2 sont finis ($(2^i - 1) \mod 3 \neq 0$), il suffit d'afficher 2^1 et 2^2 (et 1 si on considère que A_0 n'existe pas).
4. Comme A_4 et A_6 sont infinis, on ne peut pas afficher A_4 puis A_6, car on n'afficherait que l'un ou l'autre, et pas les deux. Ainsi, il faut alterner entre élément de A_4 et élément de A_6.

1.3 Diagonalisation

La diagonalisation est un procédé assez intuitif une fois qu'il est compris. On utilise cette technique pour prouver qu'il n'existe pas d'énumération d'ensembles infinis. La procédure est simple : il suffit de poser un tableau qui énumère tous les ensembles e_i possibles, puis de prendre un ensemble contradictoire : celui qui prend la valeur v_i de l'ensemble e_i. Comme notre tableau énumère tous les ensembles possibles, cela veut dire que l'ensemble contradictoire est dans le tableau. Or, si c'est le k-ème, on aura v_k de $e_k = v_k$ et v_k de $e_k \neq v_k$, ce qui amène à une contradiction !

Exercice 10 - Diagonalisation

1. Montrer que l'ensemble des parties d'un ensemble E infini dénombrable n'est pas dénombrable.
2. Que peut-on conclure sur la cardinalité de l'ensemble des fonctions ? Et de l'ensemble des programmes ?
3. Préciser le cas où E est un ensemble fini (donc dénombrable) ?

1. Soit $\mathcal{P}(E)$ l'ensemble des parties de E. Supposons que tout élément de $\mathcal{P}(E)$ est représentable par $(P_i)_{i \in \mathbb{N}}$. Soit $\mathcal{X}_E$ la fonction caractéristique de E, $\mathcal{X}_E : E \to \{0, 1\}$. On peut construire le tableau suivant :

	e_0	e_1	e_2	$\dots$	e_i	$\dots$
P_0	$\boxed{0}$	1	0	$\dots$	1	$\dots$
P_1	1	$\boxed{1}$	0	$\dots$	1	$\dots$
P_2	0	0	$\boxed{0}$	$\dots$	0	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
P_i	1	0	1	$\dots$	$\boxed{0}$	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

On prend l'ensemble $P \in \mathcal{P}(E)$ tel que $P = \{i \mid e_i \notin P_i\}$. On suppose que $\exists i.P = P_i$, car $\forall k \in \mathbb{N}.P_k \in \mathcal{P}(E)$. Si $P = P_i$, alors si $e_i \in P_i$, $e_i \notin P$ par construction, donc $P \neq P_i$. De même, si $e_i \notin P_i$, par construction, $e_i \in P$, donc $P \neq P_i$. On ne peut pas trouver i tel que $P = P_i$, donc le tableau ne contient pas tous les éléments de $\mathcal{P}(E)$, donc l'ensemble des parties d'un ensemble E infini dénombrable n'est pas dénombrable ($|\mathcal{X}_E| > |\mathbb{N}|$).

2. L'ensemble des fonctions caractéristiques est un sous-ensemble de l'ensemble des fonctions : $\mathcal{X} \subset \mathcal{F}$ car $\mathcal{X} : \mathbb{N} \to \{0,1\}$ et $\mathcal{F} : \mathbb{N} \to \mathbb{N}$. Comme $|\mathcal{X}| > \mathbb{N}$ et $|\mathcal{F}| > |\mathcal{X}|$ (car $\mathcal{X} \subset \mathcal{F}$), on a $|\mathcal{F}| > |\mathbb{N}|$, donc l'ensemble des fonctions n'est pas dénombrable.

   L'ensemble des programmes, lui, est dénombrable. On peut par exemple prendre le taille puis l'ordre lexicographique à taille égale. Vu qu'on travaille sur un alphabet fini, cet ensemble sera fini.
3. Si E est fini, $\mathcal{P}(E)$ est dénombrable. En effet, on ne peut pas construire de tableau infini, on ne peut donc pas appliquer la technique de diagonalisation et on conclut pratiquement immédiatement que c'est dénombrable.

Exercice 11 - Diagonalisation

Montrer que l'ensemble des sous-ensembles d'un ensemble dénombrable n'est pas dénombrable. Pour cela, considérer un ensemble dénombrable $A = \{a_0, a_1, a_2, \ldots\}$, et S l'ensemble de ses sous-ensembles.

L'ensemble des sous-ensembles d'un ensemble est l'ensemble des parties d'un ensemble $\mathcal{P}(A)$. On vient de démontrer cette exacte propriété dans l'exercice précédent.

Exercice 12 - Diagonalisation

1. Soit une suite quelconque d'ensembles $E_i \subset \mathbb{N}$. Construire un ensemble qui n'appartient pas à cette suite (en vous inspirant de la diagonalisation).
2. Que pouvons nous conclure sur l'ensemble des sous-ensembles de $\mathbb{N}$?

1. On prend l'ensemble $E = \{e_i \mid \mathcal{X}_{E_i}(e_i) = 0\}$. Cet ensemble est différent de tous les ensembles de E_i, et n'est donc pas dans la suite.
2. L'ensemble des sous-ensembles de $\mathbb{N}$ n'est pas dénombrable.

Exercice 13 - Diagonalisation

Montrer que $[0, 1[$ n'est pas dénombrable.

On travaille en binaire dans la suite de cet exercice. On pose la fonction $d_i(x)$ qui renvoie le i-ème décimal après la virgule de x. Si $[0, 1[$ est dénombrable, on peut construire un tableau qui contient tous les éléments de $[0, 1[$. On prend tous les éléments qui ont des chiffres après la virgule ($x_i = \sum_{k=0}^{\infty} d_k(x_i) \times 2^{-k-1}$) :

	d_0	d_1	d_2	$\ldots$	d_i	$\ldots$
x_0	$\boxed{1}$	1	0	$\ldots$	1	$\ldots$
x_1	1	$\boxed{1}$	0	$\ldots$	1	$\ldots$
x_2	0	0	$\boxed{0}$	$\ldots$	0	$\ldots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
x_i	1	0	1	$\ldots$	$\boxed{1}$	$\ldots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Soit le réel x suivant : $x = \sum_{k=0}^{\infty}((-d_k(x_k) + 1) \times 2^{-k-1})$, autrement dit, $d_k(x) = 0$ si $d_k(x_k) = 1$ et $d_k(x) = 1$ sinon. Comme le tableau contient tous les éléments de $[0, 1[$, $\exists i. x = x_i$. Or, $d_i(x_i) \neq d_i(x)$ par construction, donc $x \neq x_i$. Contradiction, il n'y a pas de i tel que $x = x_i$, et dans ce cas, le tableau n'énumère pas tous les éléments de $[0, 1[$, donc $[0, 1[$ n'est pas dénombrable.

Exercice 14 - Diagonalisation

On considère l'ensemble U des suites $(u_n)_{n\in\mathbb{N}}$ à la valeur dans $\{0,1\}$, c'est à dire $\forall n \in \mathbb{N}$. Montrer que U n'est pas dénombrable.

	u_0	u_1	u_2	$\dots$	u_i	$\dots$
U_0	$\boxed{0}$	1	0	$\dots$	0	$\dots$
U_1	0	$\boxed{1}$	1	$\dots$	1	$\dots$
U_2	1	1	$\boxed{1}$	$\dots$	1	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
U_i	0	1	0	$\dots$	$\boxed{1}$	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Soit la suite V suivante : $V(i) = 0$ si $U_i(i) = 1$ et $V(i) = 1$ sinon. On suppose $\exists i.V = U_i$. Or, $U_i(i) \neq V(i)$ par construction. Donc, $V \neq U_i$, et ainsi l'ensemble U n'est pas dénombrable.

1.4 Dénombrabilité

Exercice 15 - Ensemble fini/infini

Un ensemble est fini si on ne peut pas le mettre en bijection avec une partie stricte de lui-même. Il est infini sinon.
Montrer que l'ensemble des entiers est infini.

On peut mettre l'ensemble des entiers en bijection avec une partie stricte de lui-même : soit la fonction $f : \mathbb{N} \to \mathbb{N}^*$ où $f(n) = n + 1$. f est en bijection avec une partie stricte de $\mathbb{N}$. On peut exhiber la fonction inverse $q : \mathbb{N}^* \to \mathbb{N}$ avec $q(n) = n - 1$. L'ensemble des entiers est donc bien infini.

Exercice 16 - Taille des ensembles

Soit E un ensemble, et soit $\mathcal{P}(E)$ l'ensemble des parties de E. Montrer que $|E| < |\mathcal{P}(E)|$.

On sait que $|E| \neq |\mathcal{P}(E)|$. On veut montrer que $|E| \leq |\mathcal{P}(E)|$. Pour ce faire, il suffit de trouver $\varphi : E \to \mathcal{P}(E)$ injective, ou bien $\varphi : \mathcal{P}(E) \to E$ surjective. On prend par exemple $\varphi : E \to \mathcal{P}(E)$ avec $\varphi(e) = \{e\}$.

Exercice 17 - Dénombrabilité

1. Donner les bijections :
   (a) de $\mathbb{N}$ sur $\mathbb{N} - \{0\}$.
   (b) de $\mathbb{N}$ sur $2\mathbb{N}$.
   (c) de $\mathbb{N}$ sur $\mathbb{Z}$.
2. Est-ce que la fonction $f(n) = (-1)^n \lceil \frac{n}{2} \rceil$ est une bijection de $\mathbb{N}$ sur $\mathbb{Z}$?
3. Montrer que tout sous-ensemble $X \subset \mathbb{N}$ est dénombrable.
4. Il existe une application $f : X \to \mathbb{N}$ qui est injective si et seulement si X est dénombrable.
5. Un produit fini d'ensembles dénombrables est dénombrable.
6. Il existe une application $f : \mathbb{N} \to X$ qui est surjective si et seulement si X est dénombrable.
7. Soit E un ensemble dénombrable infini. Alors il existe une bijection de $\mathbb{N}$ sur E. Autrement dit, on peut numéroter les éléments de E, i.e. écrire $E = \{e_0, e_1, \ldots, e_n, \ldots\}$.
8. Montrer que $\mathbb{Q}$ est dénombrable.
9. Soit $(E_n)_{n\in\mathbb{N}}$ une famille dénombrable de sous-ensembles dénombrables d'un ensemble E. Montrer que la réunion $\bigcup_{n\in\mathbb{N}} E_n$ est dénombrable.
10. Soit $A = \mathbb{Q} \cap [0, 1]$ et $B = \mathbb{Q} \cap]0, 1[$. Existe-t'il une bijection de A vers B ?

1. (a) $\varphi : \mathbb{N} \to \mathbb{N} - \{0\}$ avec $\varphi(n) = n + 1$.
   (b) $\varphi : \mathbb{N} \to 2\mathbb{N}$ avec $\varphi(n) = 2n$.
   (c) $\varphi : \mathbb{N} \to \mathbb{Z}$ avec $\varphi(n) = \frac{-(n+1)}{2}$ si n est impair et $\varphi(n) = \frac{n}{2}$ sinon.
2. Oui, cette fonction est une bijection de $\mathbb{N}$ sur $\mathbb{Z}$, c'est celle que j'ai décrit en 1.(c).
3.

1.5 Fonctions (non)-calculables

Exercice 18 - Calculabilité

Soit $f : \mathbb{N} \to \{0, 1\}$ une fonction totale non calculable.

1. Rappeler la définition d'une fonction totale et d'une fonction non calculable.
2. Construire une fonction $g : \mathbb{N} \to \mathbb{N}$ totale, croissante et non calculable à partir de f.

1. Une fonction totale est une fonction définie pour tout n. Une fonction non-calculable est une fonction qui ne peut pas être calculée par une procédure automatique. Autrement dit, f est non calculable si et seulement si $\forall n. f(n) \downarrow, \nexists p \forall n. p(n) = f(n)$.
2. On peut construire la fonction g suivante : $g(n) = \sum_{i=0}^{n} f(i)$. Cette fonction est forcément croissante, car l'ensemble d'arrivée de f est $\{0, 1\}$ (donc $f(i) + f(i-1) \geq f(i-1)$), $\forall i. f(i) \downarrow$ par hypothèse, donc $(\sum_{i=0}^{n} f(i)) \downarrow$.

   Enfin, supposons que g soit calculable. On a $g(n-1) = \sum_{i=0}^{n-1} f(i)$ et $g(n) = \sum_{i=0}^{n} f(i)$, donc $f(n) = g(n) - g(n-1)$, si g est calculable, alors f est calculable. Or, f n'est pas calculable, donc g ne l'est pas non plus (contraposée).

Exercice 19 - Calculabilité

Montrer que l'inverse d'une fonction f calculable et bijective est calculable.

f calculable, cela signifie qu'il existe une procédure automatique p telle que $p(n) = f(n)$. Si on trouve une procédure automatique q qui calcule g, alors g est calculable. Soit la procédure suivante :

```
int q(int n) {
    for (int i = 0; ; ++i) {
        if (p(i) == n)
            return i;
    }
}
```

Comme f est bijective et totale, q termine forcément : pour n'importe quelle entrée n, $p(n)$ termine, et comporte un unique antécédent dans $\mathbb{N}$.

Dans le cas où f est injective, q ne termine pas forcément : la procédure p peut boucler pour certains n. Dans le cas où f est surjective, q renverra $min\{x \mid p(x) = n\}$ le minimum des antécédents de n par p.

Exercice 20 - Calculabilité

Montrer qu'une fonction f totale $\mathbb{N} \to \mathbb{N}$ est calculable si et seulement si son graphe

$$G = \{(x, f(x))|x \in \mathbb{N}\}$$

est décidable.

Si $E \subset \mathbb{N}$ est décidable, c'est qu'on peut écrire une procédure p qui calcule la fonction caractéristique $\mathcal{X}_E$ de cet ensemble : $\exists p \forall n \in \mathbb{N}. p(n) = \begin{cases} 1 & \text{si } n \in E \\ 0 & \text{sinon} \end{cases}$

$\Rightarrow$ On suppose que f totale est calculable. Montrons que le graphe G_f de f est décidable. La procédure p calcule f. Soit q la procédure qui calcule la fonction caractéristique $\mathcal{X}_{G_f}$:

```
int q(int x, int y) {
    return y == p(x);
}
```

Comme p est calculable, q l'est aussi. On a trouvé une procédure q qui calcule la fonction caractéristique $\mathcal{X}_{G_f}$, donc si f est calculable, alors G_f est décidable.

$\Leftarrow$ On suppose que G_f est décidable. Montrons que la procédure f du graphe G_f est calculable. La procédure q calcule la fonction caractéristique $\mathcal{X}_{G_f}$. Soit p la procédure qui calcule f :

```
int p(int x) {
    for (int i = 0; ; ++i) {
        if (q(x, i)) return i;
    }
}
```

Comme q est calculable, alors p l'est aussi. On a trouvé une procédure p qui calcule f, donc si G_f est décidable, alors f est calculable.

Exercice 21 - Calculabilité

Soit E un ensemble et ϕ une fonction telle que $\phi(n)$ est égale au nombre d'éléments de E strictement inférieurs à n. Montrer que ϕ totale est calculable si et seulement si E est décidable.

$\Rightarrow$ On suppose que ϕ est calculable. Montrons que E est décidable. La procédure p calcule ϕ. Soit q la procédure qui calcule la fonction caractéristique $\mathcal{X}_E$:

```
int q(int x) {
    return p(x) != p(x + 1); // On pourrait aussi dire p(x) == p(x + 1) - 1
}
```

p termine et définie pour tout x, donc q termine et définie pour tout x. On a trouvé une procédure q qui calcule la fonction caractéristique $\mathcal{X}_E$, donc si ϕ calculable, alors E est décidable.

$\Leftarrow$ On suppose que E est décidable, c'est à dire qu'il existe une procédure automatique q telle que q calcule $\mathcal{X}_E$. Montrons que ϕ est calculable. Soit p la procédure qui calcule ϕ :

```
int p(int x) {
    if (x == 0) return q(x);
    return q(x) + p(x - 1);
}
```

Comme q termine et définie pour tout $n \in \mathbb{N}$, p termine. On a trouvé une procédure p qui calcule ϕ, donc si E décidable, alors ϕ est calculable.

1.6 Problèmes indécidables

Exercice 22 - Variantes du problème de l'arrêt