Séquence 9 - TD1 - Nouveaux outils théoriques pour les preuves d'algorithmes

Exercice 1 (Propriétés de base.).

1. Soit $(X, <)$ un ensemble ordonné et soit $Y \subseteq X$. Montrer que Y possède au plus un plus petit élément (et donc au plus un plus grand élément), ce qui justifie le fait de parler DU plus petit élément (resp. plus grand élément) de Y .
2. Montrer que dans un ensemble totalement ordonné, tout élément minimal est le plus petit élément (et donc tout élément maximal est le plus grand élément).
3. Donner un exemple fini d'ensemble bien ordonné. Donner un exemple infini d'ensemble bien ordonné.
4. Donner un exemple d'ensemble bien ordonné qui n'est pas totalement ordonné.

Exercice 2.

Dans chacun des cas suivants donner un exemple d'un ensemble ordonné ayant la propriété demandée :

1. un ensemble totalement ordonné infini ayant à la fois un plus petit élément et un plus grand élément,
2. un ensemble totalement ordonné infini ayant un plus grand élément mais pas de plus petit élément,
3. un ensemble infini muni d'un ordre qui n'est pas total et qui a un plus grand élément et un plus petit élément,
4. un ensemble infini muni d'un ordre qui admet des éléments minimaux mais qui n'a pas de plus petit élément,
5. un ensemble infini bien ordonné qui admet un plus grand élément,
6. un exemple d'ensemble totalement ordonné mais non bien ordonné. Pouvez-vous trouver un exemple fini ?

Exercice 3.

L'ordre de Sarkovski $\succ$ sur $\mathbb{N}^*$ est défini par :

$$3 \succ 5 \succ 7 \succ 9 \ldots \succ 2 \times 3 \succ 2 \times 5 \ldots\ldots \succ 2^3 \times 3 \succ 2^3 \times 5 \succ \ldots \succ 2^3 \succ 2^2 \succ 2 \succ 1$$

1. Comparer 1988, 1989 et 1990 pour cet ordre
2. L'ordre de Sarkovski est-il un ordre bien fondé sur N^* ?

Exercice 4 (Terminaison de la fusion des listes).

1. Réécrire un code OCaml `merge` réalisant la fusion de deux listes triées par ordre croissant en une nouvelle liste triée.
2. Prouver sa terminaison

Exercice 5 (Fonction d'Ackermann).
Prouver la terminaison de la fonction d'Ackermann $A : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$: définie par :

$$\begin{cases} A(0,m) & = m+1 \\ A(n,0) & = A(n-1,1) \\ A(n,m) & = A(n-1, A(n,m-1)) \end{cases}$$

Exercice 6 (Coefficients binomiaux).

1. Écrire en OCaml une fonction `comb` qui calcule un coefficient binomial $\begin{pmatrix} n \\ p \end{pmatrix}$, n et p étant donnés.
2. Prouver la terminaison de cette fonction
3. Pour les plus rapides : redémontrer la formule utilisée de deux manières différentes : par le calcul et par une stratégie de dénombrement.
4. A la maison, pour retravailler OCaml : écrire une fonction itérative `triangle_pascal` qui retourne les lignes 0 à n du triangle de Pascal sous forme d'un tableau de tableaux d'entiers :

```
# triangle_pascal 3;;
- : int array array = [|[|1|]; [|1; 1|]; [|1; 2; 1|]; [|1; 3; 3; 1|]|]
```

Exercice 7 (Tri par épuisement des inversions).
On propose l'algorithme de tri suivant pour trier un tableau t de n entiers : tant que le tableau n'est pas trié, on sélectionne au hasard deux indices tels que $i < j$ et $t[i] > t[j]$ et on échange les valeurs de $t[i]$ et $t[j]$.
Prouver que cet algorithme étonnant termine !

Exercice 8 (Fonction de Sudan).
La fonction de Sudan est la fonction $F : \mathbb{N}^3 \to \mathbb{N}$ définie par :

$$\begin{cases} F(0,x,y) & = x+y \\ F(n,x,0) & = x \\ F(n+1,x,y+1) & = F(n, F(n+1,x,y), F(n+1,x,y)+y+1) \end{cases}$$

1. Écrire un code OCaml implémentant cette fonction
2. Prouver sa terminaison

Exercice 9 (Fonction f_{91} de McCarthy).
La fonction $f_{91} : \mathbb{Z} \to \mathbb{Z}$ de McCarthy est définie de la manière suivante :

$$\begin{cases} f_{91}(n) = f(f(n+11)) & \text{si } n \leq 100 \\ f_{91}(n) = n-10 & \text{sinon} \end{cases}$$

Jouer avec la fonction pour comprendre son fonctionnement, donner une expression explicite de cette fonction et en déduire sa terminaison.