Travaux dirigés n^o 8
Algorithmes combinatoires (suite)

Exercice 8.1 — Ensembles et listes chaînées

On se propose de continuer à écrire les primitives manipulant des listes chaînées pour gérer les ensembles. L'univers $\mathcal{U}$ duquel sont issus les éléments doit être un ensemble totalement ordonné.

1. Commencez par écrire la fonction `ajouteEnTête(d,L)` prenant en entrée un entier d à ajouter au début de la liste chaînée L. Cette fonction a déjà été utilisée dans la fonction `Union(L1,L2)` vue en cours.
2. Ecrivez une fonction `Intersection(L1,L2)` qui bâtit l'intersection de deux ensembles $E1$ et $E2$ passés en paramètres sous forme de listes chaînées **triées**.
3. *(facultatif)*
   Pour bien faire, il resterait à écrire le reste des primitives pour gérer les ensembles. Il y a de quoi s'entraîner jusqu'à ce que la manipulation des listes chaînées devienne fluide, en itératif comme en récursif.

Exercice 8.2 — Ensembles et tableaux

On considère toujours un ensemble-univers $\mathcal{U}$ totalement ordonné.

> **Entrée :** deux ensembles finis A et B inclus dans $\mathcal{U}$ et représentés sous forme de tableaux.
> **Problème :** trouver des algorithmes efficaces pour :
> — les primitives pour calculer l'union $A \cup B$, l'intersection $A \cap B$ et la différence $A \setminus B$
> — décider si A est inclus dans B, décider si A égale B.

1. Evoquez rapidement sans les écrire des algorithmes naïfs et sûrement quadratiques pour ce problème.
2. Commencez par écrire un algorithme `Ensembles(A,B)` prenant en entrées les deux ensembles A et B. Il doit réordonner A et retourner comme seul résultat un indice j situé entre 1 et n, la longueur du tableau A. On interprète alors cet indice j comme le fait que les cases $A[1..j]$ de A contiennent les éléments de l'ensemble $A \setminus B$ tandis que les cases $A[j+1, n]$ contiennent les éléments de l'intersection $A \cap B$.
   (Indication : on pourra utiliser l'algorithme de recherche dichotomique).
3. Ecrivez les fonctions `Intersection(A,B)`, `Différence(A,B)` et `Union(A,B)` qui se fonde sur la fonction `Ensembles(A,B)` précédente.
4. Déduisez-en les tests `Inclusion(A,B)` et `Egalité(A,B)`.

Exercice 8.3 — Problème des 2D-maxima

On se place dans le plan et on considère les points de coordonnées entières. Un point (x_1, y_1) **domine** un autre point (x_2, y_2) si :

$$x_1 \geq x_2 \quad \textbf{et} \quad y_1 \geq y_2$$

Si un point n'est dominé par absolument aucun autre, ce point est un **maximum**.

> **Entrée :** un ensemble fini de points S de coordonnées positives
> **Problème :** trouver tous les points maxima de l'ensemble S

1. Ecrivez le pseudo-code d'un algorithme `2D_maxima(points T[]` prenant en entrée le tableau des points et qui résout ce problème efficacement.
   (Indication : on pourra commencer par trier les points comme vu en cours).
2. Quelle est la complexité de cet algorithme ?