Université Paris 7 — Éléments d'Algorithmique
L2 — Année 2010-2011

TD 6
Listes

On considère des listes : on notera $l = a :: q$ pour signifier que a est la tête de la liste et q sa queue. On notera [] la liste vide et $[a]$ la liste réduite à l'élément a. La liste étant une structure récursive, on privilégiera des algorithmes récursifs. On notera $|l|$ la taille de la liste l (i.e. son nombre d'éléments). Pour chaque algorithme, on analysera sa complexité en temps.

Exercice 1. Algorithmes élémentaires sur les listes.

1. Écrire un algorithme `map` (f, l) qui applique la fonction f à tous les éléments de la liste l et qui renvoie la liste des résultats.
2. Écrire un algorithme `somme` (l) qui renvoie la somme des éléments de l.
3. Écrire un algorithme `concat` (l_1, l_2) qui renvoie la concaténation de l_1 et l_2.
4. Écrire un algorithme `min` (l) qui renvoie le plus petit élément de l.
5. Écrire un algorithme `dedans` (e, l) qui renvoie 1 si $e \in l$ et 0 sinon.
6. Écrire un algorithme `triée` (l) qui renvoie 1 si l est triée par ordre croissant, et 0 sinon.
7. Écrire un algorithme `insère` (e, l) qui insère e dans l.
8. Écrire un algorithme `insère_ordre` (e, l) qui insère e à la bonne place dans l supposée triée.
9. Écrire un algorithme `supprime` (e, l) qui supprime e de l (en supposant les éléments de l distincts 2 à 2).
10. Écrire un algorithme `supprime_ordre` (e, l) qui supprime e de l supposée triée (en supposant les éléments de l distincts 2 à 2).
11. Écrire un algorithme `miroir` (l) qui renverse l. Faites-le naïvement puis en $O(|l|)$ (indication : utiliser une variante de `concat`).
12. Écrire un algorithme `id` (l_1, l_2) qui teste si l_1 et l_2 sont identiques.
13. Écrire un algorithme `palindrome` (l) qui teste si l est un palindrome.

Exercice 2. Sous-listes.

On dit que l est une sous-liste de m ssi l s'obtient en supprimant certains éléments de m mais en conservant l'ordre entre les éléments restants.

1. Écrire un algorithme `sous_liste` (l, m) qui teste si l est une sous-liste de m. Quelle est sa complexité ?
2. Soit $S(l)$ l'ensemble des sous-listes de l. Supposons que $l = a :: q$. Exprimer $S(l)$ en fonction de a et de $S(q)$. En déduire un algorithme `sous_listes` (l) qui renvoie la liste des sous-listes de l. Combien l admet-elle de sous-listes, en supposant ses éléments 2 à 2 distincts ?

Exercice 3. Opérations ensemblistes sur les listes.

1. Écrire un algorithme `simplifie` (l) qui remplace les éléments consécutifs identiques de l par un seul d'entre eux. Quelle est sa complexité ?
2. Écrire un algorithme `union` (l_1, l_2) qui renvoie l'union de l_1 et de l_2. Quelle est sa complexité ?
3. Écrire un algorithme `intersection` (l_1, l_2) qui renvoie l'intersection de l_1 et l_2. Quelle est sa complexité ?
4. Écrire un algorithme `différence` (l_1, l_2) qui renvoie la différence de l_1 avec l_2, c'est-à-dire la liste des éléments qui sont dans l_1 mais pas dans l_2. Quelle est sa complexité ?