Algorithmique TD 3: structures de données récursives

Exercice 1. Algorithmes sur des listes

Pour chaque problème suivant, donner un algorithme le résolvant, puis le prouver et donner son coût (en précisant le modèle de coût choisi).

a. Calculer la longueur d'une liste

b. Calculer l'élément maximum d'une liste d'entiers

c. Supprimer un élément d'une liste, s'il est présent

d. Concaténer deux listes

e. Calculer la liste miroir d'une liste (dont les éléments apparaissent dans l'ordre inverse). Proposer deux versions : l'une récursive et l'autre itérative de cet algorithme.

Exercice 2. Interfaces : listes, piles, files, etc

a. implémenter un pile à partir d'une liste

b. implémenter un file à partir d'une liste

c. implémenter une file à partir d'un tableau (on gérera le débordement de capacité)

d. implémenter une file à partir de deux piles

e. proposer une implémentation d'une double pile à partir d'un seul tableau, de sorte aucun débordement de pile n'ait lieu avant que tous les éléments du tableau ne soient occupés.

Exercice 3. Listes ou tableaux

a. Dresser une liste des avantages et inconvénients pour les deux structures de données : listes et tableaux (on étudiera en particulier les coûts des opérations de base telles que la recherche, l'insertion, la suppression).

b. Pour chacun des objets suivants, dire (et justifier) quelle structure de donnée vous préférerez utiliser (liste, tableau, pile, file, ou autre) :

— un polynôme dense à coefficients entier
— un polynôme creux à coefficients entiers (seuls quelques monômes sont non-nuls)
— un polygône dans le plan
— l'historique des actions effectuées dans un logiciel, en vue de les annuler (fonction `undo`)
— un ensemble de tâche à faire effectuer par une machine lente (téléchargement de fichiers lourds sur internet par exemple)
— une matrice dense
— une matrice creuse (seuls quelques coefficients sont non nuls)

Exercice 4. Parcours d'arbres

Écrire des algorithmes itératifs effectuant le parcours d'un arbre

a. en profondeur prefixe

b. en largeur.

On pourra utiliser des structures de donnée dynamiques comme les piles, files, etc.

Exercice 5. Parcours infixe, préfixe, postfixe

Pour chaque problème suivant, quel parcours en profondeur vous semble le plus adapté :

a. former un tableau trié contenant les éléments d'un arbre binaire de recherche,

b. copier efficacement un arbre binaire de recherche,

c. organiser une grande consultation nationale (on supposera que la démocratie participative idéale est basé sur une hiérarchie de niveaux de représentativité connectée en arbre).

Exercice 6. Arbres binaires de recherche

a. Écrire les algorithmes de recherche dans un arbre binaire de recherche

1. d'un élément donné,
2. du minimum, du maximum,
3. du successeur d'un élément donné, de son prédécesseur,
4. de l'élément médian.

b. Écrire l'algorithme d'insertion d'un élément dans une feuille d'un arbre binaire de recherche.

c. Proposer un algorithme de tri sur un tableau, qui utilise en interne un arbre binaire de recherche.

d. Écrire un algorithme `Coupure` qui, à partir d'un élément x, découpe un arbre binaire de recherche en deux arbres binaires de recherche contenant des éléments inférieurs (respectivement supérieurs) à x. En déduire un algorithme d'insertion à la racine

e. Écrire un algorithme effectuant la suppression d'un élément d'un arbre binaire de recherche s'il existe. On distinguera les trois cas, où l'élément à supprimer possède 0, 1 ou 2 descendants.

Exercice 7. Tri par tas

a. Ré-écrire l'algorithme de tri par tas quand le tas est représenté par un tableau où pour tout noeud $T[i]$, son fils gauche est l'élément $T[2i+1]$ et son fils droit l'élément $T[2i+2]$.

b. Tracer l'exécution de cet algorithme sur le tableau $[5, 8, 6, 3, 7, 10, 2, 1, 12, 4, 11, 9]$

Exercice 8. File à priorité

a. Implémenter une file à priorité à partir d'un tas. On utilisera la représentation en tableau d'un tas, et on supposera disposer d'une fonction `Priorité` qui retourne pour chaque élément un entier positif indiquant son niveau de priorité.