Université
Caen Normandie

Année 2018-2019

Portail Mathématiques, Informatique
Licence deuxième année (L2 Info, L2 maths, L2 MIASHS)

Algorithmique et structures de données
TD 3 – les listes simplement et doublement chaînées

Exercice 1. Liste d'entiers

Question 1. Rappelez la structure *nœud* et le type *liste* utilisés pour représenter une suite finie d'entiers par une liste simplement chaînée.

Question 2. Écrivez une procédure récursive ou itérative *copieListe(L : liste) : liste* qui prend en argument une liste L et retourne une copie de L.
Attention ! la copîe doit être physiquement distincte de de L, c'est-à-dire sans aucun nœud en commun.

Question 3. Écrivez une procédure récursive *supprimeListe(L : liste, x : entier) : liste* qui supprime toutes les occurrences de la valeur x dans une liste L. Cette procédure prend en entrée une liste L et un entier x et retourne la liste L privée de toutes les occurrences de x (la liste L est donc modifiée).

Question 4. * Donnez une version itérative de cette procédure.

Exercice 2. Liste ordonnées d'entiers
On s'intéresse ici à des listes d'entiers triés par ordre croissant (au sens large). On utilise des listes simplement chaînées (voir exercice précédent).

On souhaite fusionner deux listes triées L1 et L2 en une liste L. Par exemple,
- L1 contient les entiers $1, 3, 7, 7, 11$
- L2 contient les entiers $2, 3, 4, 8, 12$
- L contiendra les entiers $2, 3, 3, 4, 7, 7, 8, 11, 12$.

Question 1. Proposez un algorithme pour réaliser cette fusion et exécutez-le sur l'exemple précédent. Quelle est la complexité de l'algorithme ? Précisez les opérations que vous comptez.

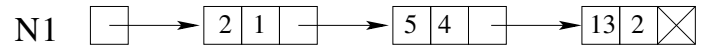
Question 2. Écrivez une procédure récursive *fusionListeTriees(L1, L2 : liste) : liste* qui prend en argument deux listes triées L1 et L2 et retourne une nouvelle liste fusion de ces deux listes.
Attention ! La liste fusionnée ne doit pas avoir de nœud en commun avec les listes L1 et L2.
Indication : vous distinguerez les cas suivants :

1. au moins l'une des listes est vide.
2. aucune des deux listes n'est vide.

Pour le cas 1, vous pourrez utiliser la procédure *copieListe(L : liste) : liste* de l'exrcice précédent.
Pour le cas 2, quel entier est ajouté à la liste fusionnée ?

Exercice 3. **Entiers représentés par leur décompositions en facteurs premiers**
Dans cet exercice, on code des entiers naturels strictement positifs comme des produits de puissances de facteurs premiers. Précisément, chaque entier est représenté par une structure de liste simplement chaînée où chaque nœud contient un facteur premier de cet entier et sa puissance. On supposera que la liste est ordonnée de façon croissante suivant les facteurs premiers.

L'entier 1 est ainsi représenté par la liste vide et l'entier $N1 = 211250 = 2 * 5^4 * 13^2$ est représenté par la liste suivante où les facteurs premiers des nœuds sont rangés par ordre croissant ($2 < 5 < 7$).



Question 1. Définir la structure nœudFacteur adéquate.

On définit le type suivant :
type listeFacteurs = pointeur sur nœudFacteur

Question 2. Écrivez une procédure *valeur(N : listeFacteurs) : entier* qui prend en entrée un entier codé par sa liste de facteurs et qui retourne la valeur de cet entier. Par exemple, valeur(N1) retournera 211250.

Question 3. Écrivez une procédure *plusGrandFacteur (N : listeFacteurs) : entier* qui retourne le plus grand facteur premier qui divise l'entier codé par N.

Question 4. Écrivez une procédure *produitEntiers(N1,N2 : listeFacteurs) : listefacteurs* qui retourne N3, la listeFacteurs du produit des deux entiers codés par N1 et N2. Autrement dit, N3 est la liste de facteurs vérifiant valeur(N1)*valeur(N2) = valeur(N3).

Exercice 3. **Listes doublement chaînées**

On définit une liste doublement chaînée à partir de la structure suivante :

Structure nœudDouble {
    valeur : entier
    precedent : pointeur sur nœudDouble
    suivant : pointeur sur nœudDouble
}
type listeDouble = pointeur sur nœudDouble

Question 1. Dessinez la représentation d'une liste doublement chaînée contenant les valeurs $5, 8, 3$.

Question 2. * Écrivez une procédure *concateneListes(L1, L2 : listeDouble) : listeDouble* qui effectue la concaténation de deux listes doublement chaînées L1 et L2.

Question 3. * Écrivez une procédure *insereListeDouble(L : listeDouble, x : entier) : listeDouble* qui insère un nouvel élément x dans une liste doublement chaînée triée (les valeurs sont rangées par ordre croissant au sens large).