Master Sciences, Technologies, Santé
Mention Mathématiques, spécialité Enseignement des mathématiques

Algorithmique et graphes, thèmes du second degré

Feuille TD n° 4 – Exercices d'algorithmique
Éléments de correction

Exercice 1. Fusion de deux listes triées

Écrire un algorithme permettant, à partir de deux listes triées, de construire « l'union » triée de ces deux listes. À partir des listes [3, 6, 9] et [1, 6, 8, 12, 15], on obtiendra la liste [1, 3, 6, 6, 8, 9, 12, 15]. On supposera que l'utilisateur entre correctement les deux listes triées…

Réponse. *L'idée est la suivante : on parcourt en parallèle les éléments des deux listes d'entrée et, à chaque tour, on rajoute en fin de la liste résultat le plus petit des deux éléments courants, puis on progresse dans la liste dont l'élément a été retenu. À la fin de cette séquence de comparaisons (ou au début si l'une des deux listes est vide), il reste des éléments non traités dans l'une des deux listes. Il suffit alors de recopier cette fin de liste dans la liste résultat…*

L'algorithme est le suivant :

```
Fonction fusionListes
       ( liste1, liste2 : listes d'entiers ) : liste d'entiers
# cet algorithme permet de fusionner deux listes triées
variables   listeFusion : liste d'entiers
            n, i1, i2, nbEléments1, nbEléments2 : entiers naturels
début
       # initialisations pour fusion
  nbEléments1 ← nombreEléments ( liste1 )
  nbEléments2 ← nombreEléments ( liste2 )
  listeFusion ← [ ]
  i1 ← 0, i2 ← 0
       # boucle principale de fusion
  tantque ( i1 < nbEléments1 ) et ( i2 < nbEléments2 )
  faire
       si ( liste1 [ i1 ] < liste2 [ i2 ] )
       alors    listeFusion ← listeFusion + [ liste1 [ i1 ] ]
                i1 ← i1 + 1
       sinon    listeFusion ← listeFusion + [ liste2 [ i2 ] ]
                i2 ← i2 + 1
       fin_si
  fin_tantque
       # on recopie la fin de la liste non épuisée
  si ( i1 < nbEléments1)
  alors         # on recopie la fin de liste1
       listeFusion ← listeFusion + liste1 [ i1, nbEléments1 ]
  sinon         # on recopie la fin de liste2
       listeFusion ← listeFusion + liste2 [ i2, nbEléments2 ]
       # on retourne la liste résultat
  retourner (liste2)
fin
```

Exercice 2. Suppression des doublons

Écrire un algorithme permettant de supprimer les doublons (éléments déjà présents) dans une liste triée donnée. À partir de la liste [3, 3, 6, 9, 9, 9, 9, 11], on obtiendra la liste [3, 6, 9, 11].

Réponse. *L'idée est en fait très simple : on doit conserver le premier élément, ainsi que tous les éléments différents de leur prédécesseur… On peut proposer deux versions de cet algorithme. Dans la première, on construit une liste résultat distincte de la liste d'entrée. Dans la deuxième, on modifie la liste elle-même.*

Première version :

```
Action suppressionDoublonsDansListeVersion1
       ( E liste1 : liste d'entiers, S liste2 : liste d'entiers )
# cet algorithme permet de construire, à partir d'une liste triée,
# une liste sans doublons
variables   i, n1 : entiers naturels
début
       # initialisation liste2
  liste2 ← [ ]
  n1 ← nombreEléments ( liste1 )
       # si liste1 est vide, il n'y a rien à faire
  si ( n1 > 0 )
  alors
            # on recopie le premier élément
       liste2 ← [ liste1 [0] ]
            # parcours de liste1 en ne recopiant que les bons éléments
       pour i de 1 à nombreEléments ( liste1 ) - 1
       faire
            si ( liste1 [ i1 ] <> liste1 [i1-1] )
            alors    liste2 ← liste2 + [ liste1 [ i1 ] ]
            fin_si
       fin_pour
fin
```

Pour la deuxième version, il suffit de parcourir la liste lue et de supprimer tous les éléments égaux à leur prédécesseur (on démarrera donc au 2ème élément de la liste…) :

```
Action suppressionDoublonsDansListeVersion2 (ES liste : liste d'entiers)
# cet algorithme permet de supprimer les doublons dans une liste triée
variables   liste : listes d'entiers naturels
            i, n : entiers naturels
début
       # initialisation pour suppression des doublons
  i ← 0
  n ← nombreEléments ( liste )
       # si la liste a moins de deux éléments, il n'y a rien a faire
  si ( n >= 2 )
  alors début
            # boucle de suppression
       tantque ( i < n )
       faire    si ( liste [ i ] = liste [ i-1 ] )
                alors    # on supprime l'élément i, sans avancer i,
                         # et n diminue de 1
                     liste ← liste [ 0 : i ]
                               + liste [ i+1, nombreEléments (liste) ]
                     n ← n-1
                sinon    # on conserve l'élément, i avance
                     i ← i + 1
                fin_si
       fin_tantque
fin
```