

MIAS 1°A, UE SM22 ; SM 2°A, module H7 ; STPI 2°A/IUP1 Miage

Informatique – Examen, 1^{ère} session (4 Mai 2000) : des exemples de solutions**1. A propos d'ensembles****Q1i)**

Version 1. La définition de l'opérateur de différence suggère d'introduire une fonction nommée **Ap** :

Ap : une séquence d'entiers, un entier \rightarrow un booléen

$\{Ap(S, e) \text{ est vrai} \Leftrightarrow e \text{ appartient à } S\}$

Pour définir la fonction **Dif**, il suffit de faire porter la récurrence sur le premier paramètre avec un découpage selon un élément :

(1) $Dif([], Y) = []$

(2) $Dif(e_0X, Y) = \text{si } Ap(Y, e) \text{ alors } Dif(X, Y) \text{ sinon } e_0Dif(X, Y)$

(3) $Ap([], e) = \text{faux}$

(4) $Ap(e_1_0S, e_2) = (e_1=e_2) \text{ ou alors } Ap(S, e_2)$

Version 2. On peut aussi introduire une fonction **Moins** :

Moins : unEnsEnt, un entier \rightarrow un booléen

$\{Moins(S, e) \text{ est l'ensemble déduit de } S \text{ en enlevant l'élément } e. \text{ Si } e \text{ n'appartient pas à } S,$

$Moins(S, e) = S\}$

(1) $Dif([], []) = []$

$\{on \text{ peut regrouper les cas 1 et 2}\}$

(2) $Dif([], e_0Y) = []$

(3) $Dif(e_0X, []) = e_0X$

(4) $Dif(e_1_0X, e_2_0Y) = Dif(Moins(e_1_0X, e_2), Y)$

(5) $Moins([], e) = []$

(6) $Moins(e_1_0S, e_2) = \text{si } e_1=e_2 \text{ alors } S \text{ sinon } e_0Moins(S, e_2)$

Q1ii) On applique un principe d'interclassement

(1) $Dif([], []) = []$

$\{on \text{ peut regrouper les cas 1 et 2}\}$

(2) $Dif([], e_0Y) = []$

(3) $Dif(e_0X, []) = e_0X$

(4) $Dif(e_1_0X, e_2_0Y) =$

selon e_1, e_2

$e_1 < e_2 : e_1_0Dif(X, e_2_0Y)$

$\{e_1 \notin e_2_0Y\}$

$e_1 = e_2 : Dif(X, Y)$

$e_1 > e_2 : Dif(e_1_0X, Y) \{e_2 \notin e_1_0Y\}$

2. Eléments de rang multiple de R

Q2i) LesRangsImpairs (découpage selon deux éléments)

(1) $LesRangsImpairs([]) = []$ (2) $LesRangsImpairs([e_1]) = [e_1]$

(3) $LesRangsImpairs([e_1, e_2] \& S) = e_1_0LesRangsImpairs(S)$

Q2i) LesRangsPairs : réalisation en termes de la fonction **LesRangsImpairs** :

$LesRangsPairs(S) : \text{si } EstVide?(S) \text{ alors } [] \text{ sinon } LesRangsImpairs(\text{fin}(S))$

ou $LesRangsPairs(S) : \text{fin}(LesRangsImpairs(e_0S)) \quad \{où e \text{ est un élément de valeur quelconque}\}$

Q2ii)

(1) $LesDerniers(0, []) = []$

$\{on \text{ peut regrouper les cas 1 et 2}\}$

(2) $LesDerniers(1+n, []) = []$

(3) $LesDerniers(0, e_0S) = e_0S$

(4) $LesDerniers(1+n, e_0S) = LesDerniers(n, S)$

$\{n \geq 0\}$

(5) $ElémentsDeRang(X, S) =$

soit $D = LesDerniers(X-1, S)$

dans si $EstVide?(D)$ alors $[]$

sinon $\text{premier}(D) \circ ElémentsDeRang(X, \text{fin}(D))$

3. Des motifs réguliers

Q3 On spécifie la fonction nommée **MotifSimple** de la manière suivante :

MotifSimple : un entier impair > 0 , deux entiers \rightarrow une séquence d'entiers

{MotifSimple(n, e1, e2) est le motif simple de longueur n et constitué d'une alternance des entiers e1 et e2, commençant par e1. Par exemple MotifSimple(5, 8, 9) = [8, 9, 8, 9, 8]}

(1) MotifSimple(1, x, y) = [x]

(2) MotifSimple(n+2, x, y) = x_0 MotifSimple(n, y, x)•x *{n impair > 0}*

ou bien

(2') MotifSimple(n+2, x, y) = [x,y]&MotifSimple (n, x, y) *{n impair > 0}*

(3) MotifDouble(1, x, y) = [[x]]

(4) MotifDouble(n+2, x, y) = *{n impair > 0}*

soit M = MotifSimple(n+2, x, y) dans M_0 MotifDouble(n, y, x)•M

4. Arbres binaires partiellement ordonnés

Q4i). Propriétés des arbres partiellement ordonnés (arbres PO)

— Les sous-arbres d'un arbre PO sont eux-mêmes des arbres PO.

— Pour tout chemin d'un arbre PO, la valeur du chemin est une séquence en ordre décroissant (non strict).

— La racine d'un arbre PO a la valeur maximum des éléments de l'arbre.

Q4ii). Un arbre est-il partiellement ordonné

On peut d'abord définir une fonction intermédiaire nommée **EstSup** :

EstSup : unArbBinE non vide \rightarrow un booléen

{EstSup(A) est vrai \Leftrightarrow la racine de A est supérieure ou égale à ses fils s'ils existent.}

EstSup(/G, r, D\) = (EstVide?(G) ou alors $r \geq$ Racine(G))
et puis (EstVide?(D) ou alors $r \geq$ Racine(D))

(1) EstPO(\wedge) = vrai

(2) EstPO(/G, r, D\) = EstSup(/G, r, D\) et puis EstPO(G) et puis EstPO(D)

Q5. Construction d'un arbre partiellement ordonné

(1) UnArbrePO([]) = \wedge (2) UnArbrePO(e₀S) = Plus(UnArbrePO(S),e)

Q6 Suppression de la racine

(1) SaufRac(/r\) = \wedge

(2) SaufRac(/G, r\) = G

(3) SaufRac(/r, D\) = D

(4) SaufRac(/G, r, D\) =

soit $rg =$ Racine(G), $rd =$ Racine(D)

dans si $rg \geq rd$ alors /SaufRac(G), rg, D\ sinon /G, rd, SaufRac(D)\

Q7. Un principe de tri

Pour appliquer le principe de tri énoncé, on dispose de la fonction **UnArbrePO** (question Q5) et on spécifie une fonction **SéqTriée** :

SéqTriée : un AbPO \rightarrow une séquence triée d'entiers

{SéqTriée(A) est la séquence en ordre croissant formée de tous éléments de A et seulement eux. SéqTriée(\wedge) = []}

On a alors :

TriParArbrePO(S) : SéqTriée(UnArbrePO(S))

Pour définir la fonction **SéqTriée**, on observe que la racine d'un arbre partiellement ordonné est la valeur maximum de l'arbre. On utilise ainsi la fonction **SaufRac** (question Q6) :

(1) SéqTriée(\wedge) = []

(2) SéqTriée(/G, r, D\) = SéqTriée(SaufRac(/G, r, D\))•r