M2 CCI – Algorithmique – Devoir surveillé

Durée 1h, sans documents **janvier 2025**
NE PAS RECOPIER les énoncés des questions. Ne pas perdre de temps à un soin excessif de la présentation. Les questions sont indépendantes. Le barème est indicatif.

1. Compactage d'une séquence d'entiers

Considérons une séquence d'entiers, par exemple :

X = [**3,3,3,3**,0,0,**1**,2,2,2,2,2,**7,7**,9,9,9,**5**,1,1,1,1,1,1,**4,4**]

Chacun des groupes de cette séquence peut être représenté par un couple comportant sa longueur et la valeur répétée. La séquence initiale est alors représentée par une séquence de tels couples. Dans notre exemple, on obtient ainsi :

Y = [<4,3>, <2,0>, <1,1>, <5,2>, <2,7>, <3,9>, <1,5>, <6,1>, <2,4>].

On appelle la séquence de couples d'entiers ainsi obtenue la *séquence compactée* de la séquence initiale.

Q1 [11 points]

(i) Décompactage

On spécifie la fonction suivante :

Décompacter : fonction (Y : séquence de couple d'entier) → séquence d'entier

{Construit, à partir de la séquence de couples d'entiers Y, la séquence décompactée d'entiers correspondante. Sur l'exemple ci-dessus, Décompacter(Y) = X.}

— Donner les **équations de récurrence** définissant la fonction **Décompacter**. On conseille de définir (donner les équations) et utiliser une fonction intermédiaire **CreerListe** qui construit une séquence de **n** entiers **e** identiques.

(ii) Compactage version 1

On spécifie la fonction suivante :

Compacter : fonction (X : séquence d'entier) → séquence de couple d'entier

{Construit, à partir de la séquence d'entiers X, la séquence compactée de couples d'entiers correspondante. Sur l'exemple ci-dessus, Compacter(X) = Y.}

Pour cela, on spécifie deux fonctions intermédiaires :

NbPrem : fonction (n : entier, X : séquence d'entier) → entier

{Renvoie le nombre d'occurrences de l'entier n au début de la séquence X.
Sur l'exemple ci-dessus, NbPrem(3, X) = 4; NbPrem(1, X) = 0 .}

Reste : fonction (n : entier, X : séquence d'entier) → séquence d'entier

*{Construit la séquence constituée de la séquence X sans tous les premiers éléments de valeur n. Sur l'exemple ci-dessus, Reste(3,X) = [0,0,**1**,2,2,2,2,2,7,**7**,9,9,9,**5**,1,1,1,1,1,1,**4,4**]*
Reste(1,X) = X .}

— Donner les **équations de récurrence** définissant les fonctions **NbPrem** et **Reste**.
— Donner les **équations de récurrence** définissant la fonction **Compacter** en utilisant les fonctions **NbPrem** et **Reste**.

(iii) Compactage version 2

On spécifie maintenant une nouvelle fonction :

NbPremEtReste : fonction (n : entier, X : séquence d'entier) → <entier, séquence d'entier>

{Construit le couple <NbPrem(n,X), Reste(n,X)>.}

— Donner des **équations de récurrence** définissant la fonction **NbPremEtReste (sans utiliser les fonctions NbPrem et Reste !)**.
— Donner de **nouvelles équations de récurrence** définissant la fonction **Compacter** en utilisant la fonction **NbPremEtReste** (mais pas les fonctions **NbPrem** ni **Reste !**).

2. À propos d'arbres binaires équilibrés

Q2 [4 points]

On définit ici le *poids* d'un arbre binaire d'entiers strictement positifs comme étant la somme de ses éléments et on convient que le poids d'un arbre vide est 0.

Par ailleurs, on dit qu'un arbre binaire d'entiers strictement positifs est *équilibré* si, **pour tout nœud de l'arbre**, le poids de son sous-arbre gauche est égal au poids de son sous-arbre droit. On convient qu'un arbre vide est équilibré.

On spécifie les fonctions suivantes :

LePoids : fonction (A : arbre binaire d'entier > 0) → entier ≥ 0
{Somme des éléments de A, ou 0 si l'arbre est vide}

EstEquilibré? : fonction (A : arbre binaire d'entier > 0) → booléen
{vrai ⇔ l'arbre est équilibré}

— Donner des **équations de récurrence** définissant la fonction **LePoids**.
— Donner des **équations de récurrence** définissant le prédicat **EstEquilibré?**, en termes de la fonction **LePoids**.

3. À propos des feuilles d'un arbre binaire

Q3 [5 points]

(i) Les feuilles d'un arbre binaire

On spécifie une fonction nommée **LesFeuilles** :

T : type
ArbreBin : type arbre binaire de T
Séq : type séquence de T

LesFeuilles : fonction (A : ArbreBin) → Séq
{Séquence formée des feuilles de A.}

— Donner des **équations de récurrence** définissant la fonction **LesFeuilles**.

(ii) Suppression des feuilles d'un arbre binaire

On spécifie une fonction nommée **MoinsF** :

MoinsF : fonction (A : ArbreBin) → ArbreBin, Séq
{Posons <B, SF> = MoinsF(A) : SF=LesFeuilles(A) et B est l'arbre obtenu à partir de A en supprimant ses feuilles.}

— Donner des **équations de récurrence** définissant la fonction **MoinsF** (sans utiliser la fonction **LesFeuilles**).

M2 CCI - Algorithmique

Des exemples de solutions **janvier 2025**

1. Compactage d'une séquence de bits

Q1

(i) **[4 points]** On a besoin d'une fonction intermédiaire :

CreerListe : fonction (n : entier, e : entier) → séquence d'entier
CreerListe(0,e) = []
CreerListe(n+1,e) = e_oCreerListe(n,e) *{n>0}*

Decompacter([]) = []
Decompacter(<n,e>$_o$S) = CreerListe(n,e) & Decompacter(S)

(ii) **[3 points]**

NbPrem(n,[]) = 0
NbPrem(n,e_oS) = si e=n alors 1 + NbPrem(n,S) sinon 0

Reste(n,[]) = []
Reste(n,e_oS) = si e=n alors Reste(n,S) sinon e_oS

Compacter([]) = []
Compacter(e_oS) = <NbPrem(e,e_oS),e>$_o$Compacter(Reste(e,e_oS))

(iii) **[4 points]**

NbPremEtReste(n,[]) = <0,[]>
NbPremEtReste(n,e_oS) = si e=n alors
soit <n1,r1> = NbPremEtReste(n,S) dans
<n1+1,r1>
sinon <0, e_oS>

Compacter([]) = []
Compacter(e_oS) = soit <n1,r1> = NbPremEtReste(e,e_oS) dans
<n1,e>$_o$Compacter(r1)

2. À propos d'arbres équilibrés

Q2 **[2 points]**

(1) LePoids (/\) = 0
(2) LePoids (/G, r, D\) = r + LePoids(G) + LePoids(D)

[2 points]

(1) EstEquilibré(/\) = vrai
(2) EstEquilibré(/G, r, D\) =
(LePoids(G) = LePoids(D)) et puis EstEquilibré(G) et puis EstEquilibré(D)

3. À propos des feuilles d'un arbre binaire

Q3

(i) Les feuilles d'un arbre binaire [2 points]

(1) LesFeuilles(/\) = []
(2) LesFeuilles(/G, r, D\) = si EstVide?(G) et EstVide?(D) alors [r]
sinon LesFeuilles(G)&LesFeuilles(D)

(ii) Suppression des feuilles d'un arbre binaire [3 points]

(1) MoinsF(/\) = </\, []>

(2) MoinsF(/G, r, D\) = si EstVide?(G) et EstVide ?(D) alors </\,[r]>
sinon soit <Bg, Sg>=MoinsF(G), <Bd, Sd>=MoinsF(D)
dans </Bg, r, Bd\, Sg&Sd>