Option Informatique
Arbres binaires équilibrés

Sujet

30 novembre 2007

Partie II : Algorithmique et programmation en CaML

Cette partie doit être traitée par les étudiants qui ont utilisé le langage CaML dans le cadre des enseignements d'informatique. Les fonctions écrites devront être récursives ou faire appel à des fonctions auxiliaires récursives. Elles ne devront pas utiliser d'instructions itératives (`for`, `while`, ...) ni de références.

Format de description d'une fonction : La description d'une fonction, lorsque celle-ci est demandée, c'est-à-dire à la questions II.25, doit au moins contenir :

1. L'objectif général ;
2. Le rôle des paramètres de la fonction ;
3. Les contraintes sur les valeurs des paramètres ;
4. Les caractéristiques du résultat renvoyé par la fonction ;
5. Le rôle des variables locales à la fonction ;
6. Le principe de l'algorithme ;
7. Des arguments de terminaison du calcul pour toutes les valeurs des paramètres qui vérifient les contraintes présentées au point 3.

La structure d'ensemble d'entiers est utilisée dans de nombreux algorithmes (par exemple, pour la déterminisation ou la minimisation d'automates). L'objectif de ce problème est l'étude d'une réalisation particulière de cette structure à base d'arbres binaires de recherche équilibrés nommés arbres AVL (du nom de leurs créateurs Adelson-Velskii et Landis). L'objectif de cette réalisation est d'optimiser à la fois l'occupation mémoire et la durée de recherche d'un élément dans un ensemble.

2 Réalisation à base d'arbres binaires de recherche

L'utilisation de la structure d'arbre binaire de recherche permet de réduire la complexité en temps de calcul pour les opérations d'insertion et de recherche.

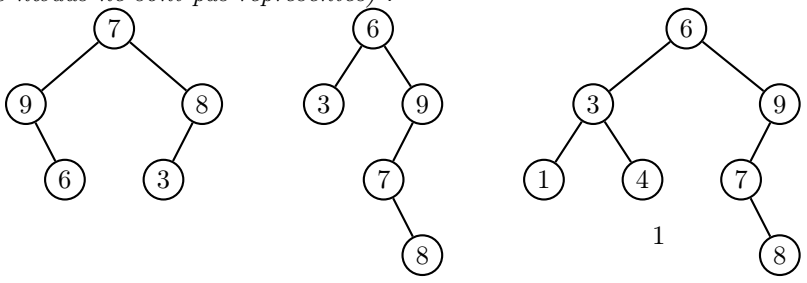
2.1 Arbres binaires d'entiers

Un ensemble d'entiers peut être réalisé par un arbre binaire en utilisant les étiquettes des nœuds pour représenter les éléments contenus dans l'ensemble.

2.1.1 Définition

Déf. II.1 (Arbre binaire d'entiers) *Un arbre binaire d'entiers a est une structure qui peut soit être vide (notée $\emptyset$), soit être un nœud qui contient une étiquette entière (notée $\mathcal{E}(a)$), un sous-arbre gauche (noté $\mathcal{G}(a)$) et un sous-arbre droit (noté $\mathcal{D}(a)$) qui sont tous deux des arbres binaires d'entiers. L'ensemble des étiquettes de tous les nœuds de l'arbre a est noté $\mathcal{C}(a)$.*

Exemple II.1 (Arbres binaires d'entiers) *Voici trois exemples d'arbres binaires étiquetés par des entiers (les sous-arbres vides des nœuds ne sont pas représentés)* :



2.1.2 Profondeur d'un arbre

Déf. II.2 (Profondeur d'un arbre) *Les branches d'un arbre relient la racine aux feuilles. La profondeur d'un arbre A est égale au nombre de nœuds de la branche la plus longue. Nous la noterons $|A|$.*

Exemple II.2 (Profondeurs) *Les profondeurs des trois arbres binaires de l'exemple II.1 sont respectivement 3, 4 et 4.*

Question II.5 *Donner une définition de la profondeur d'un arbre a en fonction de $\emptyset$, $\mathcal{G}(a)$ et $\mathcal{D}(a)$.*

Question II.6 *Considérons un arbre binaire d'entiers représentant un ensemble contenant n éléments.*

Quelle est la forme de l'arbre dont la profondeur est maximale ? Quelle est la forme de l'arbre dont la profondeur est minimale ? Calculer la profondeur de l'arbre en fonction de n dans ces deux cas. Vous justifierez vos réponses.

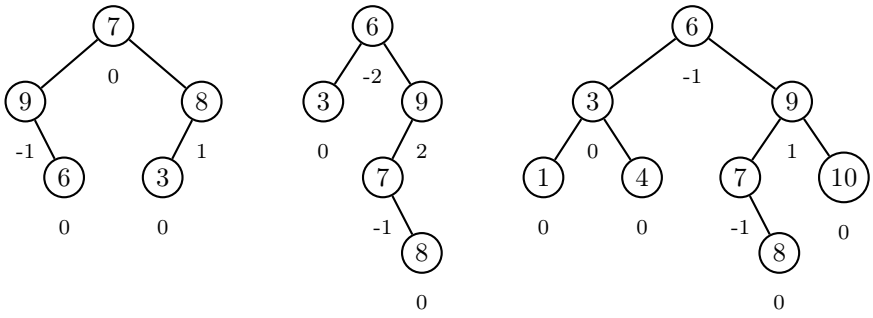
2.1.3 Déséquilibre d'un arbre

Déf. II.3 (Déséquilibre d'un nœud dans un arbre binaire) *Le déséquilibre $\Delta(a)$ d'un nœud a dans un arbre binaire est égal à la différence entre la profondeur de son fils gauche et la profondeur de son fils droit, c'est-à-dire :*

$$(\emptyset) = 0$$
$$a \neq \emptyset \Rightarrow \Delta(a) = |G(a)| - |D(a)|$$

Nous considérerons par la suite des arbres binaires dont chaque nœud est décoré par son déséquilibre.

Exemple II.3 (Arbres binaires d'entiers décorés) *Voici les arbres de l'exemple II.1 décorés par le déséquilibre de chaque nœud :*



2.1.4 Représentation des arbres binaires en CaML

Un arbre binaire d'entiers dont les nœuds sont décorés par leurs déséquilibres est représenté par le type CaML :

```
type arbre = Vide | Noeud of int * arbre * int * arbre ;;
```

Dans l'appel `Noeud( d, fg, v, fd)`, les paramètres `d`, `fg`, `v` et `fd` sont respectivement le déséquilibre, le fils gauche, l'étiquette et le fils droit de la racine de l'arbre créé.

Exemple II.4 *Le terme suivant*

```
Noeud( 0,
  Noeud( -1,
    Vide,
    9,
    Noeud( 0, Vide, 6, Vide)),
  7,
  Noeud( 1,
    Noeud( 0, Vide, 3, Vide),
    8,
    Vide))
```

est alors associé au premier arbre binaire représenté graphiquement dans l'exemple II.3.

2.1.5 Profondeur d'un arbre binaire d'entiers

Question II.7 *Écrire en CaML une fonction* `profondeur` *de type* `arbre` $\rightarrow$ `int` *telle que l'appel* (`profondeur a`) *renvoie la profondeur de l'arbre binaire d'entiers a. Cette fonction devra être récursive ou faire appel à des fonctions auxiliaires récursives.*

2.1.6 Validation d'un arbre binaire d'entiers décoré

Une première opération consiste à déterminer si les valeurs des déséquilibres des nœuds d'un arbre binaire sont correctes.

Question II.8 *Écrire en CaML une fonction* `valider_decore` *de type* `arbre` $\rightarrow$ `bool` *telle que l'appel* (`valider_decore A`) *renvoie la valeur* `true` *si l'arbre binaire A est vide ou si l'arbre binaire A n'est pas vide et si les valeurs des déséquilibres des nœuds de l'arbre binaire A sont correctes et la valeur* `false` *sinon. L'algorithme utilisé ne devra parcourir qu'une seule fois l'arbre.*

Cette fonction devra être récursive ou faire appel à des fonctions auxiliaires récursives.

Question II.9 *Calculer une estimation de la complexité de la fonction* `valider_decore` *en fonction du nombre de nœuds de l'arbre A. Cette estimation ne prendra en compte que le nombre d'appels récursifs effectués.*

2.2 Arbres binaires de recherche

Déf. II.4 (Arbre binaire de recherche) *Un arbre binaire de recherche est un arbre binaire d'entiers dont les étiquettes de tous les nœuds composant le fils gauche de la racine sont strictement inférieures à l'étiquette de la racine et les étiquettes de tous les nœuds composant le fils droit de la racine sont strictement supérieures à l'étiquette de la racine. Cette contrainte s'exprime sous la forme :*

$$a \neq \emptyset \Rightarrow (\forall v \in \mathcal{C}(\mathcal{G}(a)),\ v < \mathcal{E}(a)) \wedge (\forall \in \mathcal{C}(\mathcal{D}(a)),\ \mathcal{E}(a) < v)$$

Exemple II.5 (Arbres binaires de recherche) *Le deux-*

ième et le troisième arbre de l'exemple II.3 sont des arbres binaires de recherche.

> Notons qu'un arbre binaire de recherche ne peut contenir qu'un seul exemplaire d'une valeur donnée.

> Nous considérerons par la suite des arbres de recherche dont chaque nœud est décoré par son déséquilibre.

2.2.1 Validation d'un arbre binaire de recherche

Une première opération consiste à déterminer si un arbre binaire d'entiers est un arbre binaire de recherche.

Question II.10 *Écrire en CaML une fonction* `valider_abr` *de type* `arbre` $\rightarrow$ `bool` *telle que l'appel* `valider_abr A` *renvoie la valeur* `true` *si l'arbre binaire d'entiers A est un arbre binaire de recherche et la valeur* `false` *sinon. L'algorithme utilisé ne devra parcourir qu'une seule fois l'arbre. Cette fonction devra être récursive ou faire appel à des fonctions auxiliaires récursives.*

2.2.2 Recherche dans un arbre binaire de recherche

Une deuxième opération consiste à déterminer si un arbre binaire contient une étiquette particulière.

Question II.11 *Écrire en CaML une fonction* `rechercher_abr` *de type* `int` $\rightarrow$ `arbre` $\rightarrow$ `bool` *telle que l'appel* (`rechercher_abr v A`) *renvoie la valeur* `true` *si l'un des nœuds de l'arbre binaire de recherche A contient l'étiquette v et la valeur* `false` *sinon.*

Cette fonction devra être récursive ou faire appel à des fonctions auxiliaires récursives.

Question II.12 *Donner des exemples de valeurs des paramètres v et A de la fonction* `rechercher_abr` *qui correspondent au meilleur et pire cas en nombre d'appels récursifs effectués.*

Calculer une estimation de la complexité dans les meilleur et pire cas de la fonction `rechercher_abr` *en fonction de la profondeur de l'arbre A. Cette estimation ne prendra en compte que le nombre d'appels récursifs effectués.*

2.2.3 Insertion dans un arbre binaire de recherche

Une deuxième opération consiste à insérer un élément dans un arbre binaire de recherche en préservant cette structure. Pour cela, l'insertion se fait au niveau des feuilles de l'arbre (ou des nœuds ne contenant qu'un seul fils en lieu et place du fils absent).

Question II.13 *Calculer* $\delta(v, a)$ *l'accroissement de la profondeur d'un arbre binaire de recherche a dans lequel l'étiquette v est insérée en fonction de* $\emptyset$*, de la relation entre v et* $\mathcal{E}(a)$*, de* $\Delta(a)$*, de* $\delta(v, \mathcal{G}(a))$ *ou de* $\delta(v, \mathcal{D}(a))$*.*

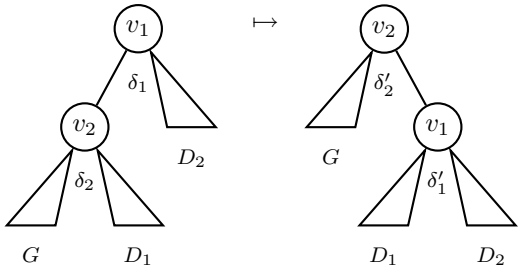
Question II.14 *Écrire en CaML une fonction* `inserer_abr` de type `int` $\rightarrow$ `arbre` $\rightarrow$ `arbre` telle que l'appel (`inserer_abr v A`) *renvoie un arbre binaire de recherche contenant les mêmes étiquettes que l'arbre binaire de recherche A ainsi que l'étiquette v s'il ne la contenait pas déjà. Cette fonction doit également calculer les valeurs des déséquilibres de chaque nœud de l'arbre résultat.*

L'algorithme utilisé ne devra parcourir qu'une seule fois l'arbre. Cette fonction devra être récursive ou faire appel à des fonctions auxiliaires récursives.

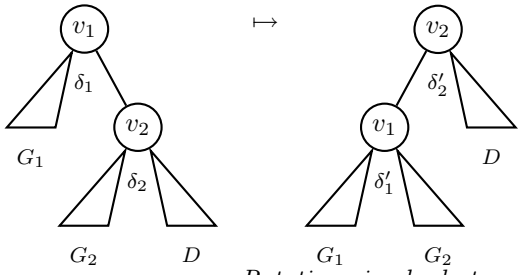
Question II.15 *Donner une estimation de la complexité dans les meilleur et pire cas de la fonction* `inserer_abr` *en fonction de la profondeur de l'arbre A. Cette estimation ne prendra en compte que le nombre d'appels récursifs effectués.*

Déf. II.5 *Rotation simple*

Une rotation simple concerne 2 nœuds imbriqués de l'arbre. Un arbre qui ne possède pas la structure adéquate (partie gauche de la règle) ne sera pas transformé.



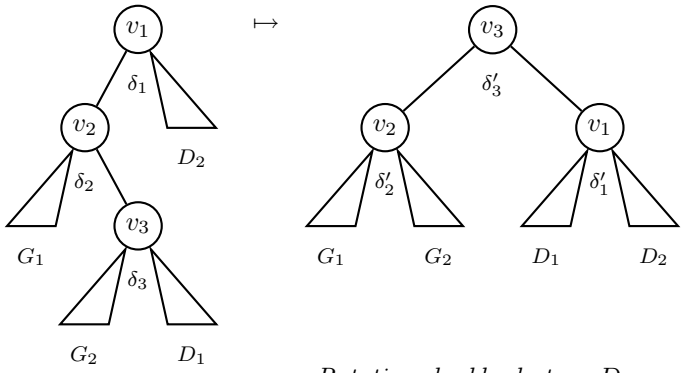
Rotation simple de type D



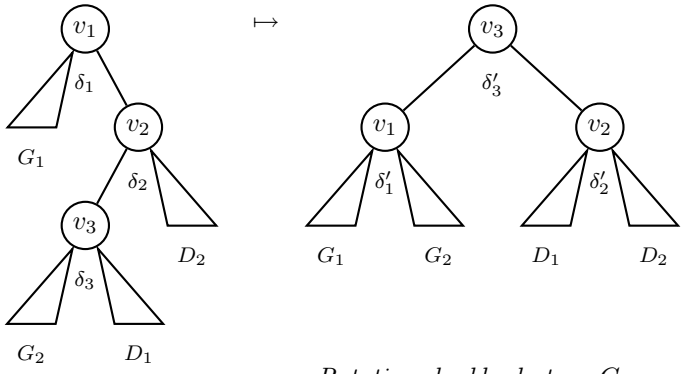
Rotation simple de type G

Déf. II.6 *Rotation double*

Une rotation double concerne 3 nœuds imbriqués de l'arbre. Un arbre qui ne possède pas la structure adéquate (partie gauche de la règle) ne sera pas transformé.



Rotation double de type D



Rotation double de type G

2.3 Rotations d'un arbre binaire de recherche

Les rotations sont des transformations d'un arbre binaire de recherche qui sont utilisées pour équilibrer la structure de l'arbre, c'est-à-dire donner une valeur proche de la longueur de chaque branche. Il s'agit donc de réduire le déséquilibre entre les différentes branches qui sont de longueurs différentes.

Question II.16 *Exprimer les rotations doubles comme des combinaisons de rotations simples.*

Question II.17 *Montrer que les rotations ne perturbent pas la structure d'arbre binaire de recherche.*

Question II.18 *Pour chaque rotation possible, calculer les nouvelles valeurs δ'_i des déséquilibres des 2 ou 3 nœuds concernés de l'arbre après la rotation en fonction des valeurs δ_i avant la rotation.*

2.3.1 Rotation simple de type D

Une première opération consiste à effectuer une rotation simple de type D dans un arbre binaire de recherche pour équilibrer celui-ci.

Question II.19 *Écrire en CaML une fonction* `rotationSD` *de type* `arbre` → `arbre` *telle que l'appel* (`rotationSD A`) *sur un arbre A dont la structure permet l'application d'une rotation simple de type D sur la racine renvoie l'arbre binaire de recherche A sur lequel une rotation simple de type D a été appliquée à la racine.*

2.3.2 Rotation simple de type G

Une deuxième opération consiste à effectuer une rotation simple de type G dans un arbre binaire de recherche pour équilibrer celui-ci.

Nous supposons prédéfinie la fonction `rotationSG` de type `arbre` → `arbre` telle que l'appel (`rotationSG A`) sur un arbre A dont la structure permet l'application d'une rotation simple de type G à la racine renvoie l'arbre binaire de recherche A sur lequel une rotation simple de type G a été appliquée à la racine. Son calcul se termine quelles que soient les valeurs de son paramètre. Elle pourra éventuellement être utilisée dans les réponses aux questions.

2.3.3 Rotation double de type D

Une troisième opération consiste à effectuer une rotation double de type D dans un arbre binaire de recherche pour équilibrer celui-ci.

Question II.20 *Écrire en CaML une fonction* `rotationDD` *de type* `arbre` → `arbre` *telle que l'appel* (`rotationDD A`) *sur un arbre A dont la structure permet l'application d'une rotation double de type D à la racine renvoie l'arbre binaire de recherche A sur lequel une rotation double de type D a été appliquée à la racine.*

2.3.4 Rotation double de type G

Une quatrième opération consiste à effectuer une rotation double de type G dans un arbre binaire de recherche pour équilibrer celui-ci.

Nous supposons prédéfinie la fonction `rotationDG` de type `arbre` → `arbre` telle que l'appel (`rotationDG A`) sur un arbre A dont la structure permet l'application d'une rotation double de type G à la racine renvoie l'arbre binaire de recherche A sur lequel une rotation double de type G a été appliquée à la racine. Son calcul se termine quelles que soient les valeurs de son paramètre. Elle pourra éventuellement être utilisée dans les réponses aux questions.

3 Arbres équilibrés

Pour réduire la complexité des opérations d'insertion et de recherche, il est nécessaire que les longueurs de toutes les branches de l'arbre aient des valeurs semblables. L'arbre a alors une structure dite équilibrée.

3.1 Définitions

Déf. II.7 (Arbres binaires équilibrés) *Un arbre binaire est dit équilibré si le déséquilibre de chacun de ses nœuds appartient à* $\{-1, 0, 1\}$. *Nous appellerons nœuds déséquilibrés les nœuds dont le déséquilibre n'appartient pas à* $\{-1, 0, 1\}$.

Exemple II.6 (Arbres binaires équilibrés) *Le premier et le troisième arbres de l'exemple II.3 sont équilibrés. Le deuxième arbre est déséquilibré. Il possède 2 nœuds déséquilibrés qui contiennent les étiquettes 6 et 9.*

3.2 Principe de l'équilibrage

Question II.21 *L'insertion d'une nouvelle étiquette dans un arbre binaire de recherche équilibré peut produire des nœuds déséquilibrés. Donner les différentes formes possibles pour le nœud déséquilibré le plus profond produit lors de l'insertion d'une nouvelle étiquette dans un arbre binaire équilibré en précisant la valeur du déséquilibre de ce nœud.*

Question II.22 *Montrer qu'une rotation suffit à rééquilibrer le nœud déséquilibré le plus profond qui est apparu lors de l'insertion d'une nouvelle étiquette dans un arbre binaire équilibré. Donner la rotation qui doit être utilisée pour rééquilibrer ce nœud pour chacune des formes étudiées dans la question précédente.*

3.3 Équilibrage d'un nœud

Une première opération consiste à équilibrer les nœuds qui viennent d'être déséquilibrés par l'opération d'insertion.

Question II.23 *Écrire en CaML une fonction* `equilibrage` *de type* `arbre` → `arbre` *telle que l'appel* (`equilibrage A`) *avec A un arbre binaire de recherche dont la racine est déséquilibrée et dont les fils gauche et droit sont équilibrés renvoie un arbre binaire de recherche équilibré contenant les mêmes étiquettes que A.*

3.4 Insertion équilibrée

Une seconde opération consiste à insérer une nouvelle étiquette dans un arbre de recherche équilibré en préservant la structure équilibrée de l'arbre.

Question II.24 *Modifier la fonction* `inserer_abr` *de type* `int` → `arbre` → `arbre` *écrite en CaML à la question II.14 telle que si A est un arbre binaire de recherche équilibré alors l'appel (*`inserer_abr v A`*) renverra un arbre binaire de recherche équilibré.*

Question II.25 *Expliquer la fonction* `inserer_abr` *définie à la question précédente.*

3.5 Complexité des opérations dans un arbre équilibré

La complexité calculée aux questions II.15 et II.12 pour les opérations d'insertion et de recherche dépend de la profondeur de l'arbre. Pour comparer celle-ci avec celle d'une réalisation à base de liste, il faut estimer la profondeur de l'arbre en fonction du nombre d'éléments contenus dans l'ensemble. Nous avons calculé celle-ci dans le meilleur des cas pour un arbre équilibré à la question II.6. Il faut maintenant évaluer celle-ci dans le pire des cas.

Question II.26 *Quelles sont les formes de l'arbre équilibré a dans le meilleur cas et le pire cas pour l'insertion et la recherche d'une étiquette v dans a ?*

Question II.27 *Soit* $\mathcal{N}(h)$ *le nombre de nœuds dans un arbre dans le meilleur des cas, exprimer* $\mathcal{N}(h)$ *en fonction de h.*

Question II.28 *Soit* $\mathcal{N}(h)$ *le nombre de nœuds dans un arbre dans le pire des cas, exprimer* $\mathcal{N}(h)$ *sous la forme d'une relation de récurrence dépendant de* $h-1$ *et* $h-2$.

Question II.29 *Soit* $\mathcal{F}(h) = \mathcal{N}(h) - 1$, *résoudre l'équation linéaire à coefficients constants dérivée de la relation de récurrence précédente pour obtenir la valeur de* $\mathcal{F}(h)$ *en fonction de h.*

Question II.30 *Déduire des questions II.27 et II.29 un encadrement de la profondeur d'un arbre en fonction du nombre de nœuds qu'il contient. En déduire la complexité des opérations de recherche et d'insertion dans un arbre binaire de recherche équilibré.*