TD10: Arbres de recherche, rotations d'arbres

MP2I Lycée Pierre de Fermat

Exercice 1. *ABR*

On utilise pour cet exercice la procédure de suppression consistant à remplacer le noeud à remplacer par le plus grand élément de son sous-arbre gauche.

Q1. En partant de l'arbre vide, effectuer les opérations suivantes, et dessiner l'arbre aux endroits indiqués par (A), (B), etc... :

```
ajouter 8; ajouter 4; ajouter 9; ajouter 16;
ajouter 10; ajouter 2; ajouter 6; ajouter 5;
(A)
supprimer 8;
(B)
ajouter 24; ajouter 11; ajouter 15;
ajouter 13; ajouter 14; ajouter 12;
(C)
supprimer 15;
(D)
```

Q2. Dessiner un ABR contenant les mêmes étiquettes que l'arbre obtenu en (D), mais de hauteur 3.

Exercice 2. *Rééquilibrage*

On considère un ABR $A = N(z, G, d)$ avec :

— $G = N(x, a, D)$

— $D = N(y, b, c)$

où x, y, z sont des étiquettes, et a, b, c, d des arbres.
On suppose que :

— $h(G) > h(d) + 1$

— $h(D) > h(a)$

Montrer qu'en deux rotations, on peut transformer A en un arbre A' tel que $h(A') = h(A) - 1$.

Exercice 3. *Arbre Rouge-Noir*

Dessiner trois ARN valides contenant :

1. Les entiers de 1 à 7
2. Les entiers de 1 à 12
3. Les entiers de 1 à 20

Exercice 4. *Correction de l'insertion des ARN*

On s'intéresse à l'insertion dans les arbres rouge-noir. On rappelle que dans un ARN, une insertion ne peut pas faire apparaître d'anomalie du point de vue de la hauteur noire, mais peut causer l'apparition d'une succession de deux noeuds rouges.

La procédure d'insertion repose donc sur l'utilisation d'une fonction de correction de ces anomalies. Plus précisément, on étudie la situation suivante : On dispose d'un arbre A tel que :

— A a une racine noire ;

— A a un enfant rouge X ayant lui même un enfant rouge ;

— Tous les sous-arbres de A sont des ARN relaxés valides, à l'exception de A et de X, et la hauteur noire de A est bien définie.

On souhaite décrire une procédure permettant de corriger un tel arbre A, en obtenant un arbre A' tel que :

— A' est un ARN relaxé ;

— A' contient les mêmes étiquettes que A ;

— $bh(A') = bh(A)$.

Q1. On suppose que le petit-enfant rouge de A causant problème est le petit-enfant gauche-gauche. Dessiner la situation, et montrer qu'en faisant une rotation droite puis des recolorations, on obtient un arbre A' adéquat. On montrera soigneusement que les conditions d'ARN sont bien vérifiées.

Q2. On suppose maintenant que le petit-enfant rouge à problème est le gauche-droite. Montrer qu'alors une rotation **gauche-droite** suivie d'une bonne recoloration permet d'obtenir un arbre A' adéquat.

Q3. Écrire une fonction OCaml `correctionARN: 'a arn -> 'a arn` telle que pour A un arbre vérifiant les préconditions plus haut, `correctionARN A` est l'arbre A' décrit plus haut.

Exercice 5. *Correction de la suppression des ARN*

On s'intéresse à la suppression dans les arbres rouge-noir. On rappelle que dans un ARN, une suppression ne peut pas faire apparaître d'anomalie de type "nœuds rouges successifs", mais peut faire apparaître un **défaut de hauteur noire**, i.e. une situation où un nœud $N(c, x, g, d)$ a ses deux sous-arbres directs g, d étant de hauteurs noires différentes.

On considère un arbre $A = N(c, x, g, d)$ tel que g et d sont des ARN avec $bh(g) = bh(d) - 1$. A n'est donc pas un ARN, on définit néanmoins sa hauteur noire comme étant le nombre maximal de nœuds noirs sur le chemin entre sa racine et une feuille. On a donc $bh(A) = bh(d)$ si la racine de A est rouge, et $bh(A) = bh(d) + 1$ si la racine de A est noire. Comme $bh(d) > 0$, alors d n'est pas une feuille, et est donc de la forme $N(c', y, g', d')$.

On souhaite décrire une procédure permettant de corriger un tel arbre A. L'objectif est d'obtenir un arbre A' contenant les mêmes étiquettes que A, et tel que :

(i) A' est un ARN relaxé ;

(ii) $bh(A') = bh(A)$ ou $bh(A) - 1$. De cette façon, le défaut de hauteur noire est soit effacé, soit décalé d'un cran vers la racine ;

(iii) Si la racine de A est noire, alors la racine de A' l'est aussi. De cette façon, on ne peut pas introduire de nouvelle anomalie si le parent de A était rouge.

Afin de garantir que les arbres que l'on manipule restent toujours des arbres binaires de recherche, on ne s'autorise qu'à effectuer des rotations sur les différents nœuds de A, et des recolorations de nœuds. On raisonne par disjonction de cas, selon les couleurs des racines de A, d, g' et d'.

Q1. Supposons que A, d, g' et d' sont noirs (i.e. de racines noires). Montrer qu'il suffit de colorer d en rouge pour obtenir un arbre A' convenable, en montrant que toutes les conditions d'ARN relaxé sont toutes vérifiées. Comment la hauteur noire de l'arbre évolue t-elle ?

Q2. Supposons que d' est rouge et que les autres sont noirs. Montrer qu'en considérant $\mathbf{rotG}(A)$ et en colorant d' en noir, on obtient un arbre A' convenable. Comment la hauteur noire évolue t-elle ?

Q3. Si g' est rouge et que A, d et d' sont noirs, justifier que l'on peut effectuer une rotation droite sur d, et montrer qu'en effectuant une rotation droite sur d puis une rotation gauche sur A, on obtient un arbre A' convenable après recoloration. Comment la hauteur noire évolue t-elle ?

Q4. Déterminer une solution convenable lorsque A et g' sont rouges et que d et d' sont noirs, et étudier l'évolution de la hauteur noire.

Q5. Écrire le code d'une fonction OCaml `suppressionARN` en présupposant l'existence d'une fonction `correction_supprARN` implémentant la correction d'anomalie étudiée dans les questions précédentes. La fonction `suppressionARN` devra renvoyer un ARN relaxé valide, de même hauteur noire que son entrée, où l'élément à supprimer l'a bien été.

Q6. Écrire le début de `correction_supprARN` : son type, et les cas du match-with correspondant au travail effectué dans les questions 1-4.

Exercice 6. *Décomposition de rotations*

On note $\mathcal{A}$ l'ensemble des arbres binaires de recherche sur les entiers. On note **rotG** et **rotD** les rotations gauche et droite. Ce sont toutes deux des fonctions de $\mathcal{A}$ dans $\mathcal{A}$.

On définit également les rotations sur des noeuds de l'arbre, inductivement. On rappelle que formellement, un noeud d'un arbre binaire est une liste de booléens indiquant un chemin dans l'arbre : **true** indique que l'on va vers l'enfant droit, et **false** vers l'enfant gauche. On note $\mathbb{B}$ l'ensemble des booléens et $\mathbb{B}^*$ l'ensemble des mots sur $\mathbb{B}$.

On définit inductivement $\mathbf{rotG}^u(A)$ l'arbre obtenu par rotation gauche sur le noeud u de A pour $u \in \mathbb{B}^*$ et $A \in \mathcal{A}$ comme suit (on rappelle que ε est le mot vide) :

— $\mathbf{rotG}^\varepsilon(A) = \mathbf{rotG}(A)$ pour $A \in \mathcal{A}$

— $\mathbf{rotG}^{1.v}(N(x, g, d)) = N(x, g, \mathbf{rotG}(d))$ pour $x \in \mathbb{N}$ et $g, d, \in \mathcal{A}$

— $\mathbf{rotG}^{0.v}(N(x, g, d)) = N(x, \mathbf{rotG}(g), d)$ pour $x \in \mathbb{N}$ et $g, d, \in \mathcal{A}$

On définit de même $\mathbf{rotD}^u(A)$. Ces deux fonctions permettent simplement d'effectuer des rotations à l'intérieur d'un arbre, et pas seulement à la racine. Le but de cette partie est de montrer le théorème suivant :

Théorème 1. Toute rotation $\mathbf{rotX}^u$ peut se décomposer en suites de rotations de la forme $\mathbf{rotY}^\varepsilon$ et $\mathbf{rotZ}^1$, où $\mathbf{rotX}, \mathbf{rotY}, \mathbf{rotZ}$ désignent des rotations gauche ou droite.

En d'autre termes, toutes les rotations peuvent s'écrire comme des suites de rotations sur la racine et son enfant droit.

Q1. Appliquer la suite de rotations suivantes sur quelques arbres. Déterminer à quelle condition elle est bien définie, et à quelle rotation elle est équivalente :

$$\mathbf{rotD}^\varepsilon; \mathbf{rotD}^\varepsilon; \mathbf{rotG}^1; \mathbf{rotG}^\varepsilon$$

Q2. Si L est une liste de rotations, on note $r_L : \mathcal{A} \to \mathcal{A}$ la fonction obtenue en composant chacune des rotations de L dans l'ordre. Étant donné L une liste de rotations, comment obtient-on une liste L' telle que $r_{L'} = r_L^{-1}$?

Q3. Soit $u = 11 \in \mathbb{B}^*$. Exprimer $\mathbf{rotD}^u$ en fonction de $\mathbf{rotD}^\varepsilon, \mathbf{rotG}^\varepsilon$ et $\mathbf{rotD}^1$

Q4. Donner de même une expression de $\mathbf{rotD}^{01}$ en fonction de rotations sur ε et 0

Q5. En déduire une fonction inductive permettant de simuler n'importe quelle rotation à partir de rotations sur la racine et son enfant droit.

Q6. (**Bonus**) Implémentez et vérifiez vos résultats en OCaml ! Attention, pour simuler les rotations, on déconstruit les chemins en partant de la fin et pas en partant du début. Il pourra donc être plus aisé de stocker les noeuds à l'envers.