Une page recto verso manuscrite autorisée

Préalables

Vous pourrez utiliser les primitives suivantes, toutes dans $\theta(1)$:

primitives traitant les couples

- *Couple (x,y)* renvoie le couple dont le premier élément est x et le deuxième y
- *Prem(C)* renvoie le premier élément du couple C
- *Deuz(C)* renvoie le deuxième élément du couple C

primitives sur les arborescences binaires

- *ArboVide()* renvoie une arborescence binaire vide
- *CreerArbo(v)* renvoie une arborescence binaire réduite à une feuille contenant la valeur v
- *VideP(A)* indique si l'arborescence A est vide ou non ;

  les primitives suivantes ne sont définies que si A n'est pas vide :
- *SAG(A)* et *SAD(A)* qui renvoient respectivement les sous arborescences (éventuellement vide) gauche et droite de A
- *Racine(A)* qui renvoie la valeur à la racine de A
- les deux primitives *GrefG(A,B)* et *GrefD(A,B)* qui greffent B comme sous arborescence gauche (resp. droite) de A sont définies dés que A n'est pas vide.

Algorithme récursif *Decoupe*

l' algorithme découpe en deux un ABR :

Données:
- un ABR A dont toutes les valeurs sont distinctes
- un entier v (présent ou non dans A)

Résultat: un couple de deux ABR
- la première contenant toutes les valeurs de A inférieures à v et
- la deuxième contenant toutes les valeurs de A supérieures à v

Complexité réclamée
La complexité de l'algorithme doit être dans $O(h)$
(où h est la hauteur de l'ABR à découper).

Question 1 (1 point)
Trouver deux points d'arrêt de l'algorithme.
Pour chacun de ces points d'arrêt, préciser ce qui est renvoyé.

Question 2 (4 points)
Ecrire l'algorithme **Decoupe**.

Question 3 (2 points)
Justifier que la complexité de **Decoupe** est dans $O(h)$
(où h est la hauteur de l'ABR à découper).

Utilisation : l'algorithme InsèreRacine

Données:
- un ABR A dont toutes les valeurs sont distinctes
- un entier v (qui n'apparait pas dans A)

Résultat: un ABR
- qui contient v, tous les éléments de A, aucune autre valeur et
- dont la racine a pour valeur v

Question 4 (2 points)
Utiliser **Decoupe** pour écrire l'algorithme **InsèreRacine**
Justifier que la complexité de **InsèreRacine** est dans $O(h)$ (où h est la hauteur de A).

IntersectionABR

Données: deux ABR A et B dont toutes les valeurs sont distinctes,
mais pouvant avoir des valeurs communes

Résultat: Le nombre de valeurs communes à A et B.

On supposera égales les hauteurs des deux ABR et on appellera h cette hauteur commune. On appelle n_A (resp. n_B) le nombre de sommets de A (resp. B) et on supposera que n_B est beaucoup plus grand que n_A.

Question 5 (3 points)
Exprimer en fonction de n_B la valeur minimum de n_A. Justifier.

Vous allez écrire deux versions de **IntersectionABR**, la première avec une complexité dans $O(n_A \times h)$, la deuxième avec une complexité dans $O(2^h)$.

Question 6 (1 point)
Dans quel cas la deuxième version est-elle meilleure que la première ?

Une première version de IntersectionABR de complexité dans $\mathbf{O(n_A \times h)}$

Vous utiliserez un algorithme de traversée en profondeur d'une arborescence binaire.

Question 7 (1 point)
Quelle est la complexité de cet algorithme de traversée en profondeur ?

Question 8 (1 point)
Ecrire la première version de **IntersectionABR**(A,B). Justifier la complexité.
Indication : vous avez le droit d'utiliser la répétitive
Pour chacun des sommets S obtenus lors de la traversée de ...

La deuxième version de IntersectionABR utilise la procédure **Decoupe**.
Pour simplifier le calcul de complexité on supposera (ce qui n'est pas vrai) que les deux arborescences renvoyées par le découpage d'une arborescence de hauteur h sont toutes deux de hauteur $h - 1$.

Question 9 (1 point)
Montrez que cette supposition peut être fausse

Question 10 (4 points)
Ecrire un algorithme qui est en $O(2^h)$ quand on admet la supposition.
Justifier alors cette complexité.