UNIVERSITÉ DE MONTRÉAL
Département IRO

Été 2016
Neil Stewart

IFT 2015 E16

Devoir 2.

10/10, soit 10% de la note finale.

Les 10 points "Partie pratique" pour le cours E16 seront distribués de la façon suivante : Devoir #1 (1 point), Devoir #2 (3 points), Devoir #3 (6 points).

1 Partie Pratique (3 points)

Dans ce devoir nous allons implanter et utiliser les parcours "pre-order" et "post-order" dans un Arbre Binaire de Recherche, dont les clefs sont des entiers.

Il peut être démontré qu'un noeud x est l'ancêtre d'un noeud y si et seulement si x précède y dans le parcours "pre-order" et x succède y dans le parcours "post-order". On va utiliser ceci pour vérifier si un noeud est l'ancêtre d'un autre dans un arbre binaire. Vous devez utiliser le squelette mis à votre disposition.

NOTE : Ne modifiez pas le squelette.

Tâches :

1. (1 point) Implanter la fonction *preorder(BinaryTreeNode currentNode)* dans la classe BinaryTree.
2. (1 point) Implanter la fonction *postorder(BinaryTreeNode currentNode)* dans la classe BinaryTree.
3. (1 point) Compléter la fonction *verifyAncesters(int[] pre, int[] post, int n, String filename)* de la classe Devoir2. Plus précisemment, trouver les positions de node1 et node2 dans les parcours "pre-order" et "post-order".

Vous pouvez tester votre code en utilisant les fichiers data.in et data2.in disponibles sur le site *StudiUM*.

Modalités de remise : envoyer une archive <nom1-prenom1_nom2-prenom2.zip> à l'adresse teodora.dan@umontreal.ca.

2 Partie Théorique (7 points)

1. (2 points)

   On peut utiliser un monceau *min_heap* pour le problème de trouver le k'ième élément le plus grand (voir Weiss, page 1 et page 239). Il s'agit de garder une queue de priorité (implantée par monceau) avec les k éléments les plus grands.

   On commence par créer un monceau avec les k premiers éléments qui arrivent. Ensuite, chaque fois qu'un nouvel élément arrive, on le compare avec S_k = l'élément le plus petit dans le monceau (c'est le k'ième élément le plus grand pour les données vues jusqu'ici). Si le nouvel élément est plus grand que S_k, on remplace S_k par le nouvel élément. Maintenant le monceau a un nouvel élément S_k qui est le plus petit : cet élément est possiblement l'élément qu'on vient d'ajouter.

   Quand tous les éléments on été traités, on sort la valeur finale de S_k comme réponse.

   Démontrez que ce processus est $O(N \log k)$.

2. (2 points)

   (a) Weiss Question 4.4, page 161. (Les "children" correspondent au cas d'échec dans le cas d'un arbre binaire de recherche.)

   (b) Nous avons défini la quantité I (Longueur de chemin interne) parce que cela a un intérêt évident dans le contexte de la recherche dans un arbre binaire de recherche (cas où la clef est présente). Nous pouvons aussi définir la quantité E (Longueur de chemin externe) : c'est la somme des longueurs de tous les chemins depuis la racine vers les "children". (On peut aussi remplacer les pointeurs nuls par des "noeuds d'échec" : c'est une autre façon de s'exprimer. Cela a un intérêt évident dans le contexte de la recherche dans un arbre binaire de recherche : cas où la clef n'est pas présente.) Quelle est la relation entre I et E ?

   (c) Démontrez la relation que vous avez énoncé dans la partie 2b.

3. (2 points)

   Nous avons démontré que

$$D(N) = 2\left[\frac{1}{N}\sum_{j=0}^{N-1} D(j)\right] + N - 1$$

   où $D(N)$ est la valeur espérée de $I(\tau_N)$, τ_N un arbre particulier avec N noeuds.

   (a) Dessinez les six arbres qui correspondent aux $3! = 6$ permuations sur trois éléments.

   (b) Démontrez que selon la formule $D(3) = 8/3$, et en utilisant les six arbres, que la valeur moyenne de I dans le cas $N = 3$ est bel et bien égal à $D(3)$.

4. (1 point)

   Dans la preuve que la profondeur d'un arbre AVL est $O(\log N)$, nous avons dit, à un moment donné, que $T_D \in P_{d-1}$ (sinon remplacez-le par un arbre AVL avec moins de noeuds, pour avoir un arbre original "pire" que T, *i.e.*, avec moins de noeuds, et avec hauteur ègal à d.) Très bien, c'est clair.

   À la prochaine étape nous avons dit que la hauteur de T_G doit être $d-1$ ou $d-2$ (cela vient de la définition d'un arbre AVL), que cela doit être $d-2$, et qu'on doit avoir $T_G \in P_{d-2}$ ("sinon remplacez-le par ...", encore une fois, c'est clair).

   Intuitivement il est clair que "ça doit être $d-2$". Mais comment démontrer cela ? Sommes nous certains qu'il n'y a pas un arbre T_G svec hauteur $d-1$ qui a moins de noeuds ?

À réaliser en équipes de 1 ou 2. À remettre le 22 juin, 2016, avant `9:00`. Les solutions seront affichées le 22 juin. Les devoirs en retard ne seront pas acceptés.