

TP 5 : Arbres Binaires

Objectif du TP

L'objectif de cette séance est de pratiquer la programmation d'arbres.

Recommandations

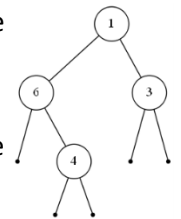
Immédiatement après chaque séance de TD/TP, chaque étudiant ou groupe d'étudiants enverra un mail à l'enseignant responsable du groupe un compte rendu de l'activité réalisée pendant la séance au format PDF dans lequel les réponses aux questions auront été saisies. Il est fortement conseillé de faire des copies d'écran de vos programmes pour aller plus vite.

Le sujet du mail devra obligatoirement respecter le format suivant : **[S3T ou S3D ou S3A][M313][Gx][Seance_y] Nom1 / Nom2** ou x représente le numéro de groupe et y le numéro de la séance (par exemple : [mailto:denis.pallez@unice.fr?subject=\[S3A\]\[M313\]\[G1\]\[Seance2\]Turing](mailto:denis.pallez@unice.fr?subject=[S3A][M313][G1][Seance2]Turing)). Le fichier PDF devra également être renommé de la façon suivante : **<NomEtudiant1_NomEtudiant2>_TP<N° du TP>_<date de la séance de TP en anglais>.pdf** (par ex : Pallez_TP1_20140613.pdf).

Vous ne pouvez pas faire plus de 2 séances avec le même binôme et vous ne pouvez pas faire plus de la moitié des séances seul. Tout manquement à ces règles pourra entraîner des retraits de points sur la note de contrôle continu.

Exercice 1 Arbre Binaire

- En imitant l'implémentation des listes chaînées, implémenter la classe `ABin` qui représente les arbres binaires.
- Dans la méthode `main`, programmer la construction de l'arbre ci-contre.
- On souhaite construire un arbre à partir de données stockées dans un tableau de taille n de la façon suivante :
 - $T[0]$ représente le nœud racine
 - si $T[i]$ représente un nœud v , alors v possède un fils gauche stocké à $T[2i+1]$ (avec $2i+1 < n$) et un fils droit stocké à $T[2i+2]$ (avec $2i+2 < n$). Ecrire la méthode qui permet de construire cet arbre.
- Ecrire les méthodes `hauteur` et `taille`.



Exercice 2 Dessiner graphiquement un arbre

On souhaite visualiser graphiquement l'arbre contenu en mémoire. Pour cela, nous allons utiliser le logiciel [GraphViz](#) qui propose un langage de programmation pour définir un graphe ; donc un arbre. Pour éviter de perdre du temps à apprendre la syntaxe de GraphViz, vous avez à votre disposition la classe java `GraphViz` qui servira d'interface entre votre programme java et ce logiciel. Cette classe permet de déclarer le graphe (`declareGraph`), un nœud avec un numéro identifiant et une chaîne de caractères (`declareNode`), un nœud vide représenté graphiquement par un point (`declareNullNode`), une arête entre 2 nœuds identifiés par leur numéro (`declareEdge`), et de finir la définition de l'arbre (`endGraph`).

- Dans la classe `ABin`, écrire une méthode `toGraphviz(String filename)` qui écrira la définition de l'arbre dans le format de Graphviz dans un fichier texte nommé `filename.gv`. Indication : vous testerez votre fonction avec un graphe d'entiers contenant *que* des entiers différents.


- b) On souhaite générer automatiquement une image de l'arbre. Pour cela, ajouter le répertoire `GraphViz\bin` dans votre `Path` sous Windows. Ensuite, copier les fichiers `Trans2Tree.gv` et `Trans2Tree.bat` dans le répertoire qui contiendra les fichiers `gv` générés nommés `workdir` (ça risque d'être la racine du projet Eclipse). Créer la méthode `toImage(String filename)` dans la class `ABin`.


Exercice 3 Parcours

On souhaite parcourir l'arbre suivant les 3 parcours d'arbres (préfixé, infixé, postfixé).

- Créez une méthode récursive qui parcourt l'arbre de manière infixé et affiche chaque nœud.
- Déduisez en trois méthodes qui parcourent l'arbre de manière infixé, postfixé et préfixé et renvoi une liste des nœuds parcourus. L'intérêt de cette solution est d'être indépendant du traitement à effectuer sur chaque nœud.
- Créez la méthode `public boolean equals(Object)` qui vérifie que deux arbres sont égaux. Pensez à créer la méthode de test correspondante.

Exercice 4 Liste d'arbres

On souhaite construire un programme récursif qui construit la liste de TOUS les arbres à n nœuds (n étant un paramètre). Par exemple, si on souhaite l'ensemble des arbres possibles de taille 2, on obtient  et à 3

nœuds, on obtient 

- Identifiez à la main, tous les arbres à 4 nœuds.
- Identifiez la récurrence.
- Programmer la méthode `arbresDeNNoeud(int n)` qui renvoie une liste d'arbres de n nœuds.



Pour aller plus loin

Exercice 5 Itérateurs d'arbres

Comme pour les listes, on souhaite créer des itérateurs sur des arbres. Néanmoins, il existe plusieurs parcours possibles pour un même arbre. Proposez plusieurs solutions pour réaliser cet objectif.

Sources pour ce TP

[Claire David](#), Structures de données et objets Java 2005-2006.

Visualizing binary trees with Graphviz, <http://eli.thegreenplace.net/2009/11/23/visualizing-binary-trees-with-graphviz/>, <http://rafal.io/posts/binary-trees.html>