Algorithmique et structures de données : Mission 2 (produit)

Groupe 1.2: Ivan Ahad - Jérôme Bertaux - Rodolphe Cambier
Baptiste Degryse - Wojciech Grynczel - Charles Jaquet

07 octobre 2014

Question 1 La profondeur est le nombre de parents qu'un noeud comporte. La racine est donc de profondeur 0, et ses enfants sont de profondeur 1. La hauteur est le nombre maximum de générations en dessous du noeud. Une feuille a une hauteur de 0, et les autre noeuds ont la hauteur de leur enfant le plus haut+1. Le niveau n est l'ensemble des nœuds de profondeur n. Ces notions ne dépendent pas du style d'arbre, elles s'appliquent aux arbres en général car il suffit d'avoir la notion de racine, feuille, enfant et parent pour pouvoir appliquer ces définitions. Ces notions ne dépendent pas de la structure de données utilisée car la représentation ne dépend pas de l'implémentation.

Question 2

Question 3

Question 4

Question 5 Dans la majorité des cas de parcours, cela ne pose pas de problème, car on parcours l'arbre depuis la racine jusqu'aux feuilles. Cela peut cependant être handicapant dans certaines situations. Dans le cas où l'on voudrait, par exemple, parcourir tous les noeuds de l'arbre situés à un même niveau, ne pas avoir la possibilité de remonter peut poser problème et ralentir fortement le processus.

Pour réaliser la méthode parent, on peut faire comme suit: Garder le pointeur sur le noeud dont on veut le parent. Vérifier si ce noeud n'est pas le root, auquel cas on retourne null. Parcourir l'arbre noeud par noeud, en vérifiant pour chacun si un de ses fils n'est pas le noeud dont on cherche le parent. On finit donc par trouver le parent du noeud de départ.Puisqu'il faut, au pire, parcourir tout l'arbre pour trouver le père, on a une complexité en O(n).

Question 6

Question 7

Question 8 C'est le parcours "inorder traversal" d'un arbre binaire qui permet de parcourir l'arbre dans le bon sens. Voir figure 1

```
1 public String toString(){
2 if ( T.isExternal(v) )
3   return v.getElement().toString();
4 else
5   return "("+ v.left() + v.getElement() + v.right() +")";
6 }
```

Question 9

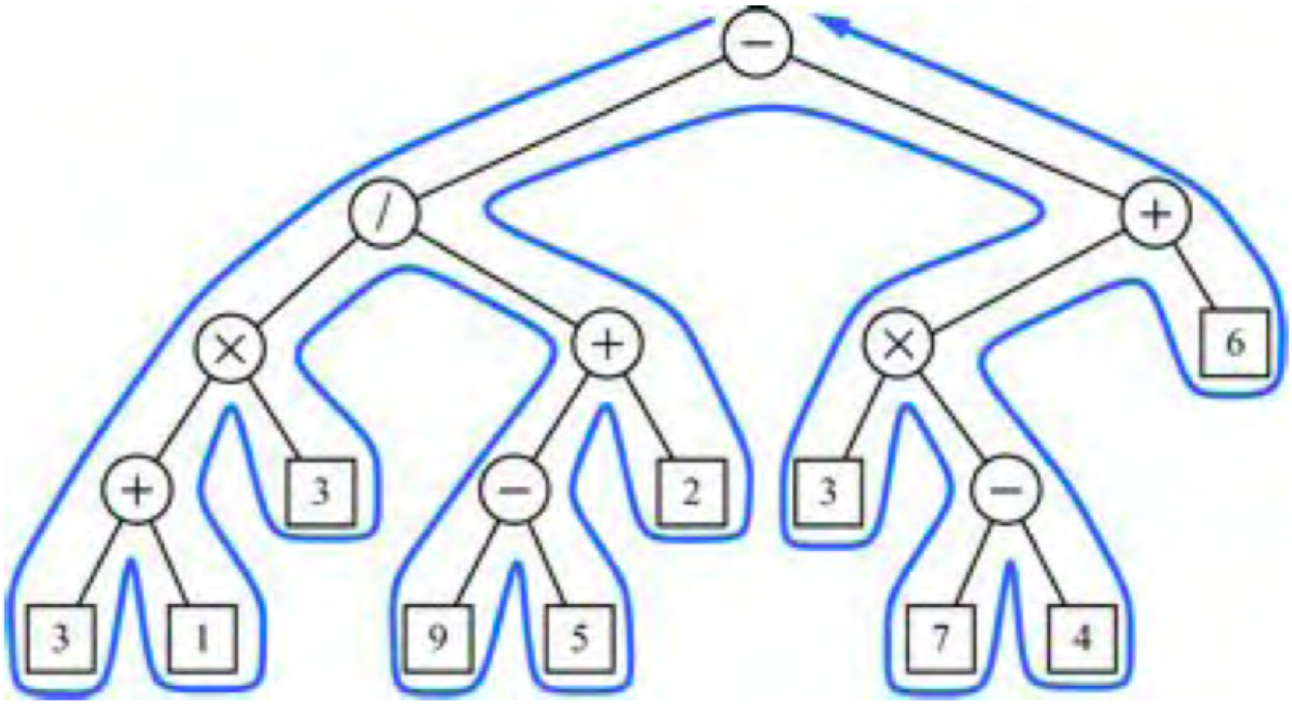

Figure 1: inorder traversal path (from Data Structure And Algorithms in Java p 424)