Methods of Binary Search Tree Traversion

1. InOrder Traversal

An *inorder* traversal of tree t is a recursive algorithm that follows the left subtree; once there are no more left subtrees to process, we process the right subtree. The elements are processed in *left-root-right* order. The basic algorithm is as follows:

inOrder(t) {

if(t is not empty) {

inOrder( left subtree of t )

process t's root element

inOrder( right subtree of t )

}

}

1. PostOrder Traversal

A *postorder* traversal of tree t is a recursive algorithm that follows the left and right subtrees before processing the root element. The elements are processed in *left-right-root* order. The basic algorithm is as follows:

postOrder(t) {

if(t is not empty) {

postOrder( left subtree of t )

postOrder( right subtree of t )

process t's root element

}

}

1. PreOrder Traversal (DFS)

A *preorder* traversal of tree t is a recursive algorithm that processes the root and then performs preorder traversals of the left and right subtrees. The elements are processed *root-left-right* order. The basic algorithm is as follows:

preOrder(t) {

if(t is not empty) {

process t's root element

preOrder( left subtree of t )

preOrder( right subtree of t )

}

}

1. Level-Order Traversal (BFS)

A *level-order* traversal of tree t is a recursive algorithm that processes the root, followed by the children of the root (from left to right), followed by the grandchildren of the root (from left to right), etc. The basic algorithm shown below uses a queue of references to binary trees to keep track of the subtrees at each level:

levelOrder(BinaryTree t) {

if(t is not empty) {

// enqueue current root

queue.enqueue(t)

// while there are nodes to process

while( queue is not empty ) {

// dequeue next node

BinaryTree tree = queue.dequeue();

process tree's root;

// enqueue child elements from next level in order

if( tree has non-empty left subtree ) {

queue.enqueue( left subtree of t )

}

if( tree has non-empty right subtree ) {

queue.enqueue( right subtree of t )

}

}

}

}