# 实验报告

|  |  |
| --- | --- |
| 实验名称： | * 给出后序遍历序列###ca##ji####spom, 构建二叉树。 * 上述二叉树实际是一颗二叉排序树，请实现程序   + 查找c节点，输出从树根到c节点的路径。   + 删除其中的m节点，使得删除后仍为二叉排序树。 |
| 试验时间： | 2018.12 |
| 实验目的： | 树的各种操作 |
| 实验原理： | 二叉树 二叉排序树 |
| 实验步骤： | 我啥都写了……  BTree(); ~BTree();//用remove来析构，自己写的delete没法析构，因为如果左右都没的情况不能删节点只能设为特殊字符$在print的时候不计而已 **void** print\_pre(TreeNode \*&aroot);前序遍历输出 **void** print\_mid(TreeNode \*&aroot);中序遍历输出，为排序树排序正序输出服务 **int** height(TreeNode \*&aroot);树高 **void** build\_pre(TreeNode \*&aroot, string &astring, **int** &index);前序遍历构建 **void** build\_post(TreeNode \*&aroot, string &astring, **int** &index);后序遍历构建 **void** build\_BFS(string &astring);BFS构建 **void** build\_premid(TreeNode \*&aroot, string &pre, string &mid, **int** &index, **int** pstart, **int** pend);前序+中序构建 **void** build\_midpost(TreeNode \*&aroot, string &post,string &mid , **int** &index, **int** pstart, **int** pend);中序+后序构建 **void** find\_path(string c);找到到某节点的路径 **void** insert(string c);插入一个节点 **void** build\_bintree(string c);*//二叉查找树,a way of sort* **void** delete\_node(TreeNode \*&aroot,string c);删除节点 **void** remove(string m, TreeNode\* &root); TreeNode\* findMax(TreeNode \*root) **const**;  删除节点官方版本  TreeNode\* findMax(TreeNode \*root) **const**;  找到左子树最右 |