重庆大学本科学生实验项目任务书

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 实验题目 | 二叉树与二叉查找树实践 | | | |
| 实验时间 |  | 实验地点 | |  |
| 实验性质 | □验证性 √设计性 □综合性 | | | |
| 实验目的  1. 掌握二叉树与二叉查找树的基本原理  2. 训练使用二叉树基本操作，通过编程解决不同难度问题的实践能力 | | | | |
| 实验内容   1. 二叉树与二叉查找树算法基本原理及时空复杂度分析方法 2. 在线编程解决3道不同难度的编程题，共40分   3． 在线编程时间（答题时长）150分钟，从开始答题时刻起算  4． 在线程序测评平台采用PTA (https://pintia.cn/)  5． 每人必须独立完成编程，可查阅教科书、PPT等资料，不得相互抄袭以及抄袭网上已有的程序  6． 实验课结束后，会对所有程序进行查重，如检测出有抄袭的程序，成绩计零分处理  注意：最后提交完整的实验报告，包括对每道题的算法思路、代码描述、复杂度分析等内容并回答思考题。 | | | | |
| 参考资料：   * Data Structures and Algorithm Analysis (C++ Version) Clifford A. Shaffer   + Introduction to Algorithms, 3rd Edition, MIT Press, T.H. Cormen, et al.   + 《数据结构（ C 语言版）》，严蔚敏，吴伟民编著，清华大学出版社 | | | | |
| 任务下达日期 年 月 日 | | | 完成日期 年 月 日 | |

说明：学院、专业、年级均填全称，如：计算机学院、计算机科学与技术、2021。

**《数据结构与算法》实验报告**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **年级、专业、班级** | |  | | | **姓名** |  |
| **实验题目** | 二叉树与二叉查找树实践 | | | | | |
| **实验时间** |  | | **实验地点** |  | | |
| **实验成绩** |  | | **实验性质** | **□验证性** √**设计性 □综合性** | | |
| 教师评价：  □算法/实验过程正确； □源程序/实验内容提交 □程序结构/实验步骤合理；  □实验结果正确； □语法、语义正确； □报告规范；  其他：  评价教师签名： | | | | | | |
| 实验目的  1. 掌握二叉树与二叉查找树的基本原理  2. 训练使用二叉树基本操作，通过编程解决不同难度问题的实践能力 | | | | | | |
| 二、实验项目内容  注：每道题按下面的格式分别描述  实验课题1：  题目内容：  **6-1 The Kth Largest in BST**  Given a binary search tree, you are supposed to find the node that contains the *K*-th largest key.  **Format of function:**  BinTree KthLargest ( BinTree T, int K );  where BinTree is defined as the following:  typedef struct TNode \*BinTree;struct TNode{int Key;BinTree Left;BinTree Right;};  The function KthLargest is supposed to return the pointer that points to the node that contains the K-th largest key in the binary search tree T.  Here T is not empty and all its keys are distinct positive integers. K is positive and is never more than the total number of nodes in the tree.  解题思路：  从根节点开始按照根右左的顺序访问树，首次访问至叶子节点（即二叉查找树的最大值节点）时全局变量cnt开始计数，并返回空指针，当cnt值取K时以当前节点为返回值  源代码（格式正确，对关键步骤加注释）：  int cnt=0;//记录当前节点在降序序列中的位置  BinTree KthLargest ( BinTree T, int K )  {  if(!T)return NULL;//当前节点为空，返回  BinTree res=NULL;//初始化返回值  res=KthLargest(T->Right,K);//在右子树查找目标节点  cnt++;//记录当前节点在降序序列中的位置  if(cnt==K)return T;//如果当前节点是目标节点，返回该节点  if(!res)res=KthLargest(T->Left,K);//如果当前节点及其右子树中不含目标节点，则在当前节点的左子树中查找目标节点  return res;//返回空节点或目标节点  }  时间与空间复杂度分析  时间复杂度：O（n）,最坏情况下，递归函数将依次遍历每个节点各1次。  空间复杂度：O（n）,最坏情况下，递归函数为每个节点申请1个返回值指针。  实验课题2：  题目内容：  **6-2 从下往上打印指定元素的所有祖先**  本题要求实现按从下往上打印二叉树中指定元素的祖先。  函数接口定义：  int PrintAncestors(BiTree T,char ch);  T是二叉树树根指针，ch为指定的元素值，如果ch存在，则PrintAncestors函数按从下到上输出二叉树的ch的所有祖先结点序列，格式为一个字符后面跟着一个空格，并返回1，否则返回0。  其中BinTree结构定义如下：  typedef char ElemType;  typedef struct BiTNode {  ElementType data;  struct BiTNode \*lchild;  struct BiTNode \*rchild;  }BiTNode, \*BiTree;  解题思路：  从根节点开始中序遍历整颗树，若当前节点的子树中含有目标节点，即递归访问子树的函数返回值为1时，打印当前节点值。  源代码（格式正确，对关键步骤加注释）：  int PrintAncestors(BiTree T,char ch)  {  if(!T)  return 0;  if(T->data==ch)  return 1;//当前节点是目标节点  if(PrintAncestors(T->lchild,ch)||PrintAncestors(T->rchild,ch))//如果左右子树中找到了目标节点  {  printf("%c ",T->data);//打印当前节点值  return 1;//当前节点是祖先之一  }  return 0;  }  时间与空间复杂度分析  时间复杂度：O（n）,最坏情况下，递归函数访问树的每个节点各一次。  空间复杂度：O（0）,递归函数不申请新变量。  实验课题3：  题目内容：  **7-1 构造二叉检索树**  本题目构造一棵二叉检索树。要求读入n个整数，以0结束。最后输出这棵树的先序序列。  解题思路：  读入当前非0节点值并通过递归函数插入二叉检索树中。  源代码（格式正确，对关键步骤加注释）：  #include<bits/stdc++.h>  using namespace std;  #define loop(i,start,end) for(int i=start;i<end;i++)  #define rep(i,start,end) for(int i=start;i<=end;i++)  #define rrep(i,end,start) for(int i=end;i>=start;i--)  typedef pair<int,int> P;  typedef long long ll;  typedef unsigned long long ull;  const int N=1e5+10,inf=0x3f3f3f3f,mod=1e9+7;  struct Node  {  int value;  Node \*left,\*right;  Node(int \_value=0,Node \*\_left=NULL,Node\*\_right=NULL):value(\_value),left(\_left),right(\_right){};//含默认值的构造函数  };//树节点的结构体定义  void insert(Node \*&root,int v)//插入函数  {  if(!root)  {  root=new Node(v);  return;  }//如果当前节点为空，则在当前位置申请新节点并初始化节点值为v  if(v<=root->value)insert(root->left,v);//v值大于等于当前节点值，则插入在当前节点的左子树中  else insert(root->right,v);//否则插入当前节点的右子树中  }  void dfs(Node \*root)//先序遍历打印二叉搜素树  {  if(!root)  return;  printf("%d ",root->value);  dfs(root->left);  dfs(root->right);  }  int main()  {  Node \*root=NULL;//初始化总根节点为空  int x;  while(scanf("%d",&x)&&x)//如果仍在读入节点值且节点值不为0  insert(root,x);//插入该节点值  dfs(root);//打印结果  return 0;  }  时间与空间复杂度分析  时间复杂度：O（nlogn）——O（n²）：对于每个节点，插入操作的复杂度介于理想情况下（平衡树）的O（logn）和最差情况下（顺序表）的O（n）之间，共进行n次插入操作，故时间复杂度为O（nlogn）——O（n²）。  空间复杂度：O（n），递归函数为每个新插入的节点申请一次空间。 | | | | | | |
| 三、思考题  遍历二叉树是按一定的顺序依次访问树中各结点并输出结点存放的数据。二叉树有四种遍历方式：前序、中序、后序遍历即层序遍历。假设二叉树各结点的数据互异。如果出现如下的情况，即对二叉树用两种不同的方式遍历，生成的结点序列却相同，则该二叉树具有什么特征？   1. 前序遍历与中序遍历相同   只有根结点的二叉树或非叶子结点只有右子树的二叉树。  （2）后序遍历与中序遍历相同  只有根结点的二叉树或非叶子结点只有左子树的二叉树。  （3）前序遍历与后序遍历相同  只含根节点的二叉树。   1. 前序遍历与层序遍历相同   每层仅有最右侧节点可能有子树的二叉树。 | | | | | | |