

DS–Autumn 2023 — Homework 3Solutions

王子隆, SID 2221411126

2023 年 10 月 24 日

摘要

二叉树实现及应用
面向过程面向对象
定义
二叉树遍历（九种方法递归非递归 * 2 * 3
preorder inorder postorder
复杂度时间空间
最坏情况最好情况最废空间最省空间
二叉树的主要性质
语法制导编辑器
修改数据结构（不借助外力 stack queue
三叉链表利用空指针（增加标识
哈夫曼树定义性质应用
二叉树分类
BST
ASL 平均查找长度

1. 二叉树实现

(a) 功能实现

(b) 应用

1.1 面向过程

Solutions:

1. 结构体 ArrayDeque.java

```
/** Array based list.  
 * @author zilong  
 */
```

1.2 面向对象

Solutions:

1. interface BinaryTree.java

```

1      #include <iostream>
2      #include "../hw2/SeqStack.cpp"
3      // #define NULL nullptr
4      #define cin std::cin
5      #define cout std::cout
6      #define endl std::endl
7
8
9      template <class T>
10     class BinaryTreeNode{
11     private:
12         T data;
13         BinaryTreeNode<T>* left;
14         BinaryTreeNode<T>* right;
15
16
17     public:
18         BinaryTreeNode();
19         BinaryTreeNode(const T& elem);
20         BinaryTreeNode(const T& elem, BinaryTreeNode<T>* l,
21             BinaryTreeNode<T>* r);
22         ~BinaryTreeNode(){};
23         T value() const;
24         BinaryTreeNode<T>* leftchild() const;
25         BinaryTreeNode<T>* rightchild() const;
26         void setLeftchild(BinaryTreeNode<T>*);
27         void setRightchild(BinaryTreeNode<T>*);
28         void setValue(const T& val);
29         bool isLeaf() const;
30     };
31
32
33     template<class T>
34     BinaryTreeNode<T>::BinaryTreeNode(){
35         data = nullptr;
36         left = nullptr;
37         right = nullptr;
38     }
39
40     template<class T>

```

```
41     BinaryTreeNode<T>::BinaryTreeNode(const T& elem){
42         data = elem;
43         left = nullptr;
44         right = nullptr;
45     }
46
47     template<class T>
48     BinaryTreeNode<T>::BinaryTreeNode(const T& elem, BinaryTreeNode<T>* l,
49         BinaryTreeNode<T>* r){
50         data = elem;
51         left = l;
52         right = r;
53     }
54
55     template<class T>
56     T BinaryTreeNode<T>::value() const{
57         return this->data;
58     }
59
60     template<class T>
61     BinaryTreeNode<T>* BinaryTreeNode<T>::leftchild() const{
62         return this->left;
63     }
64
65     template<class T>
66     BinaryTreeNode<T>* BinaryTreeNode<T>::rightchild() const{
67         return this->right;
68     }
69
70     template<class T>
71     void BinaryTreeNode<T>::setLeftchild(BinaryTreeNode<T>* Lchild){
72         this->left = Lchild;
73     }
74
75     template<class T>
76     void BinaryTreeNode<T>::setRightchild(BinaryTreeNode<T>* Rchild){
77         this->right = Rchild;
78     }
79
80     template<class T>
81     void BinaryTreeNode<T>::setValue(const T& Value){
82         this->data = Value;
83     }
84
85     template<class T>
```

```

86     bool BinaryTreeNode<T>::isLeaf() const{
87         return false;
88     }
89
90
91
92
93
94     template <class T>
95     class BinaryTree{
96     protected:
97         BinaryTreeNode<T>* root;
98     public:
99         BinaryTree() {root = nullptr;}
100        BinaryTree(BinaryTreeNode<T>* r) {root = r;}
101        ~BinaryTree() { DeleteBinaryTree(root); };
102        bool isEmpty() { return root==nullptr; };
103        void visit(BinaryTree<T>& curr){cout <<curr->data << "□";}
104        BinaryTreeNode<T>* Root() {return root;};
105        void CreateTree(const T& data, BinaryTreeNode<T>* lefttree,
106                        BinaryTreeNode<T>* righttree);
107        void CreateTree(BinaryTreeNode<T> *&r);
108        void DeleteBinaryTree(BinaryTreeNode<T>* root);
109
110        void PreOrder(BinaryTreeNode<T>* root);
111        void InOrder(BinaryTreeNode<T>* root);
112        void PostOrder(BinaryTreeNode<T>* root);
113        void PreOrderWithoutRecursion(BinaryTreeNode<T>* root);
114        void InOrderWithoutRecursion(BinaryTreeNode<T>* root);
115        void PostOrderWithoutRecursion(BinaryTreeNode<T>* root);
116        void LevelOrder(BinaryTreeNode<T>* root);
117
118
119     template<class T>
120     void BinaryTree<T>::CreateTree(const T& data, BinaryTreeNode<T>*
121                                   leftTree, BinaryTreeNode<T>* rightTree) {
122         root = BinaryTreeNode<T>(data, leftTree, rightTree);
123         BinaryTree(root);
124     }
125
126
127
128     template<class T>
129     void BinaryTree<T>::DeleteBinaryTree(BinaryTreeNode<T>* Root){

```

```
130         if (Root != NULL) {
131             DeleteBinaryTree(Root->left);
132             DeleteBinaryTree(Root->right);
133             delete Root;
134         }
135     }
136
137     template<class T>
138     void BinaryTree<T>::PreOrder(BinaryTreeNode<T>* root){
139         if (root == NULL) return;
140         visit(root->value());
141         PreOrder(root->leftchild());
142         PreOrder(root->rightchild());
143     }
144
145
146     template<class T>
147     void BinaryTree<T>::PreOrderWithoutRecursion(BinaryTreeNode <T> * root){
148         SeqStack<BinaryTreeNode<T>* > tStack;
149         BinaryTreeNode<T>* pointer = root;
150         while(!tStack.empty() || pointer){
151             if (pointer){
152                 visit(pointer->value());
153                 tStack.push(pointer);
154                 pointer = pointer->leftchild();
155             } else{
156                 pointer = tStack.top();
157                 tStack.pop();
158                 pointer = pointer->rightchild(); }
159         }
160     }
```

2. 二叉树遍历

(a) 二叉树遍历（九种方法递归非递归 * 2 * 3

preorder inorder postorder

复杂度时间空间

最坏情况最好情况最废空间最省空间

preorder

recursive

Solutions:

nonrecursive

1. recursivelylike

Solutions:

2. other way

Solutions:

inorder

recursive

Solutions:

nonrecursive

1. recursivelylike

Solutions:

2. other way

Solutions:

postorder

recursive

Solutions:

nonrecursive

1. recursivelike

Solutions:

2. other way

Solutions:

复杂度分析

1. 复杂度时间空间
2. 最坏情况最好情况

层次遍历

1. 复杂度时间空间
2. 借助外力 stack queue 逻辑

3. 二叉树的定义、主要性质、定理

- (a) 递归定义及基本术语
- (b) 分类顺序存储链式存储
- (c) 性质*5

definition

Solutions:

1. a recursive definition

category

Solutions:

1. sequential structure
2. list structure

properties quality character

Solutions:

1. $i \leq 2^{i-1}$ floor
2. $k \leq 2^k - 1$ all the tree
3. $n_0 = n_2 + 1$
4. $\text{height} = \lceil \log_2 n \rceil + 1$
5. structure of root, Lchild, Rchild

4.证明

YOUR ANSWER GOES HERE

something to prove

Solutions:

1. 给定tree 遍历序列唯一给定位置唯一存在（顺序唯一
2. 中序遍历猜想 + 先序猜想
3. 遍历时的性质*5
4. n node $2*n$ pointer $N-1$ in use

5.修改数据结构

YOUR ANSWER GOES HERE

- (a) 利用空指针（增加标识
- (b) 三叉链表

1.1 利用空指针

Solutions:

1. 结构体 ArrayDeque.java

```
/** Array based list .  
 *   @author zilong  
 */
```

1.2 三叉链表

Solutions:

1. interface List.java

```
public interface List<Item>
```

6.Huffman tree

- (a) 递归定义及基本术语
- (b) 分类顺序存储链式存储
- (c) 应用*4

definition

Solutions:

1. a recursive definition

category

Solutions:

1. sequential structure
2. list structure

properties quality character

Solutions:

1. $i \leq 2^{i-1}$ floor
2. $k \leq 2^k - 1$ all the tree
3. $n_0 = n_2 + 1$
4. $\text{height} = \lceil \log_2 n \rceil + 1$
5. structure of root, Lchild, Rchild

7.Binary searching/sorting Tree

8.BinaryTree categories

9.BinaryTree in java