# 数据结构课程设计实验报告

姓名： 侯越 学号： 16051615 专业：计科 周 三 班

## 题目说明：

给定一个二叉树，对于这个二叉树结点的值，同一个边上的值不能同时选取(即两个结点的关系为双亲和孩子的关系, 则两个结点在同一个边上, 那么这两个结点至多只能选取一个)，将选取的结点的值相加, 求这个二叉树所能选取出的结点值的和的最大值。输入的严格按照二叉树的先序遍历构造二叉树。

如：输入序列3 2 0 3 0 0 3 0 1 0 0 ，（0代表空节点）画出如下二叉树，最大值为3+3+1=7。

3

/ \

2 3

\ \

3 1

3

/ \

4 5

/ \ \

1 3 1

二叉树节点参考定义为如下形式。

\* struct TreeNode {

\* int val;

\* struct TreeNode \*left;

\* struct TreeNode \*right;

\* };

\*/

输入：

3 2 0 3 0 0 3 0 1 0 0

3 4 1 0 0 3 0 0 5 0 1 0 0

输出：

7

9

需求规格说明表

|  |  |
| --- | --- |
| 功能需求 | 详细说明 |
| 用户操作菜单 | 用户可以参照菜单输入操作编号来选择对结构的特定操作，若输入<0或者大于3则输出Please input number 0~3 |
| 创建二叉树 | 用户按规定输入二叉树，一边创建二叉树，一边递推以该节点为根的子树的最大和 |
| 显示该二叉树所能选取出的结点值的和的最大值 | 显示结果 |
| 删除二叉树 | 递归删除树的结点 |

功能模块结构说明表

|  |  |  |  |
| --- | --- | --- | --- |
| 函数名 | 参数说明 | 返回值说明 | 操作行为说明 |
| NewNode | 无参数 | 返回一个新结点 | 创建一个新的结点并赋初始值 |
| Pushup | 参数1：传入一个树的结点的指针 | 无返回值 | 在创建完该节点的左右孩子结点后，更新以该结点为根的树的最大和  （   1. 该节点值 2. 左右孩子结点的最大和的和 3. 左右孩子结点的左右孩子的最大和的和）   3者最大值 |
| Build | 参数1：传入树的根节点的指针 | 返回逻辑值 | 先给该节点赋值，再递归创建左右孩子节点（先序遍历顺序） |
| Destroy | 参数1：传入树的根节点指针 | 返回逻辑值 | 递归删除创建的树结点 |

测试计划表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试项目 | 测试步骤 | 期望结果 | 测试目的 |
| 创建测试 | 1. 正确输入二叉树 2. 列出表信息，观察结果 | 元素正确创建在制定位置，并记录了以该节点为根的树的最大和  创建成功显示：Yes. Build successfully!  失败显示：Sorry. Build failed! | 正面、负面测试，测试创建二叉树 |
| 显示测试 | 显示已创建的二叉树 | 显示成功：The max sum of the tree is 结果  显示失败：Sorry. No tree! | 正面、负面测试，测试显示结果是否正确 |
| 删除测试 | 销毁二叉树 | 销毁成功显示：Yes. Destroy successfully!  销毁失败显示：Sorry. Destroy failed | 正面、负面测试，测试是否成功删除二叉树 |

缺陷记录表

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 程序缺陷说明 | 修复情况 | 备注 |
| 1 | 使用空指针 | 已修复 | 严重bug必须修复 |
| 2 | 用指针作为参数无法做到形参实参行为一致 | 已修复 | 先给实参创建空间，再调用函数 |

代码：

#include <stdio.h>

#include <string.h>

#include <ctype.h>

#include <math.h>

#include <stdlib.h>

typedef long long ll;

typedef unsigned long long ull;

#define foR for

#define for9 for

#define retunr return

#define reutrn return

#define reutnr return

const int inf = (1 << 31) - 1;

const ll INF = (1ll << 63) - 1;

const int maxn = 1000000 + 100;

int N;

struct TreeNode

{

int val;

int sum;

struct TreeNode\* left;

struct TreeNode\* right;

};

TreeNode\* NewNode()

{

TreeNode\* root = (TreeNode\*)malloc(sizeof(TreeNode));

if (!root)

return 0;

root->left = NULL;

root->right = NULL;

root->val = 0;

root->sum = 0;

return root;

}

void Pushup(struct TreeNode\* root)

{

int Max = (root)->val;

int LeftSum = (root)->left->sum;

int RightSum = (root)->right->sum;

int TempSum = (root)->val;

if (Max < LeftSum + RightSum)

Max = LeftSum + RightSum;

if ((root)->left->val)

{

TempSum += (root)->left->left->sum;

TempSum += (root)->left->right->sum;

}

if ((root)->right->val)

{

TempSum += (root)->right->left->sum;

TempSum += (root)->right->right->sum;

}

if (Max < TempSum)

Max = TempSum;

(root)->sum = Max;

}

int Build(struct TreeNode\* root)

{

if (!root)

return 0;

scanf("%d", &((root)->val));

if ((root)->val)

{

(root)->left = NewNode();

if (!Build(((root)->left)))

return 0;

(root)->right = NewNode();

if (!Build(((root)->right)))

return 0;

Pushup(root);

}

else

(root)->sum = (root)->val;

return 1;

}

int Destroy(TreeNode\* root)

{

if (!root)

return 0;

if (root->left)

Destroy(root->left);

if (root->right)

Destroy(root->right);

free(root);

return 1;

}

int main()

{

//freopen("IN.txt", "r", stdin);

int i, j, n, m;

struct TreeNode\* root = NULL;

printf(" \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf(" \* Menu \*\n");

printf(" \* 0.Build \*\n");

printf(" \* 1.Show \*\n");

printf(" \* 2.Destroy \*\n");

printf(" \* 3.Exit \*\n");

printf(" \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

while (scanf("%d", &n) != EOF && n != 3)

{

switch (n)

{

case 0:

{

root = NewNode();

if (Build(root))

{

printf("Yes. Build successfully!\n");

}

else

printf("Sorry. Build failed!\n");

break;

}

case 1:

{

if (root)

printf("The max sum of the tree is %d!\n", root->sum);

else

printf("Sorry. No tree!\n");

break;

}

case 2:

{

if (Destroy(root))

printf("Yes. Destroy successfully!\n");

else

printf("Sorry. Destroy failed\n");

root = NULL;

break;

}

default:

printf("Please input intger number 0~3.\n");

}

}

return 0;

}