華中科技大學

课 程 设 计 报 告

题目： 基于 AVL 树表示的集合 ADT 实现与应用

课程名称：数据结构

专业班级：CS1303

学　　号：U201314842

姓　　名：宗枫博

指导教师：李剑军

报告日期：2015/09/23

计算机科学与技术学院

目录

任务书

设计目的

平衡二叉树(AVL)作为一种重要的查找表结构，能有效地支持数据的并行处理。本设计使学生牢固掌握 AVL 树及其实现方法，并应用该结构实现集合抽象数据类型，提升学生对数据结构与数据抽象的认识，提高学生的综合实践与应用能力。

设计内容

本设计分为三个层次：（1）以二叉链表为存储结构，设计与实现 AVL 树-动态查找表的 6 种基本运算；（2）以 AVL 树表示集合，实现集合抽象数据类型；（3）以集合表示个人微博或社交网络中好友集、粉丝集、关注人集，实现共同关注、共同喜好、二度好友等查询功能。

设计要求

（1）参考有关文献，实现 AVL 树的删除操作，维护其动态平衡，这可能是设计中较为复杂的算法；要求提供关键算法的时间与空间复杂度分析。

（2）实现集合的运算包括：初始化 set_init，销毁 set_destroy，插入 set_insert，删除 set_remove，交 set_intersection，并 set_union，差 set_diffrence，判断元素是否为集合的成员的查找 set_is_member，判断是否为子集 set_is_subset，判断集合是否相等 set_is_equal。

（3）要求从互联网上获取测试数据集或随机生成测试数据集，数据集的大小具有一定规模；数据与结果以文件保存。

（4）对复杂算法有改进与创新或者界面友好可适当加分。

1 引言

1.1 课题背景与意义

平衡二叉树（AVL 树）是数据结构中的重要知识点，在实际应用中多用于在内存中组织数据，对于平衡二叉树最大的应用就是来查找数据，因为它的查找的时间效率为 logn，因此就存在要创建平衡二叉树、对其进行插入、删除这三种基本操作。同时因它在动态查找表中的查找效率非常高，在地理信息处理、医学 模型处理以及快速成形等技术中都有广泛应用。

1.2 国内外研究现状

当前对平衡二叉树的研究都基本上是对其的扩展和改进，像比较复杂的 B-树、B+树、键树、2-3 树等。

1.3 任务与分析

任务：实现平衡二叉树的创建，并且用二叉树树表示集合，并能对集合内的二叉树进行插入删除以及其他一些简单的操作。

分析：平衡二叉树又称 AVL 树。它或者是一棵空树，或者是具 有下列性质的二叉树：它的左子树和右子树都是平衡 二叉树，且二叉树上的所有结点的平衡因子绝对值不 超过 1。在平衡二叉树上插入或删除结点后，可能使树 失去平衡，因此，需要对失去平衡的树进行平衡化调 整。平衡一个二叉树方法：分别针对不同失衡结构采用 左转、右转、先左转后右转、先右转后左转 4 种。而其他的操作都是建立在如何保证实现平衡，所以这也是整个课设的难点及重点。

2 系统需求分析与总体设计

2.1 系统需求分析

以二叉树链表为储存结构，存储数据集并进行基本运算，能够随机生成大量数据，保存和读取数据，模拟实际中微博的人际关系，并实现常用功能。

2.2 系统总体设计

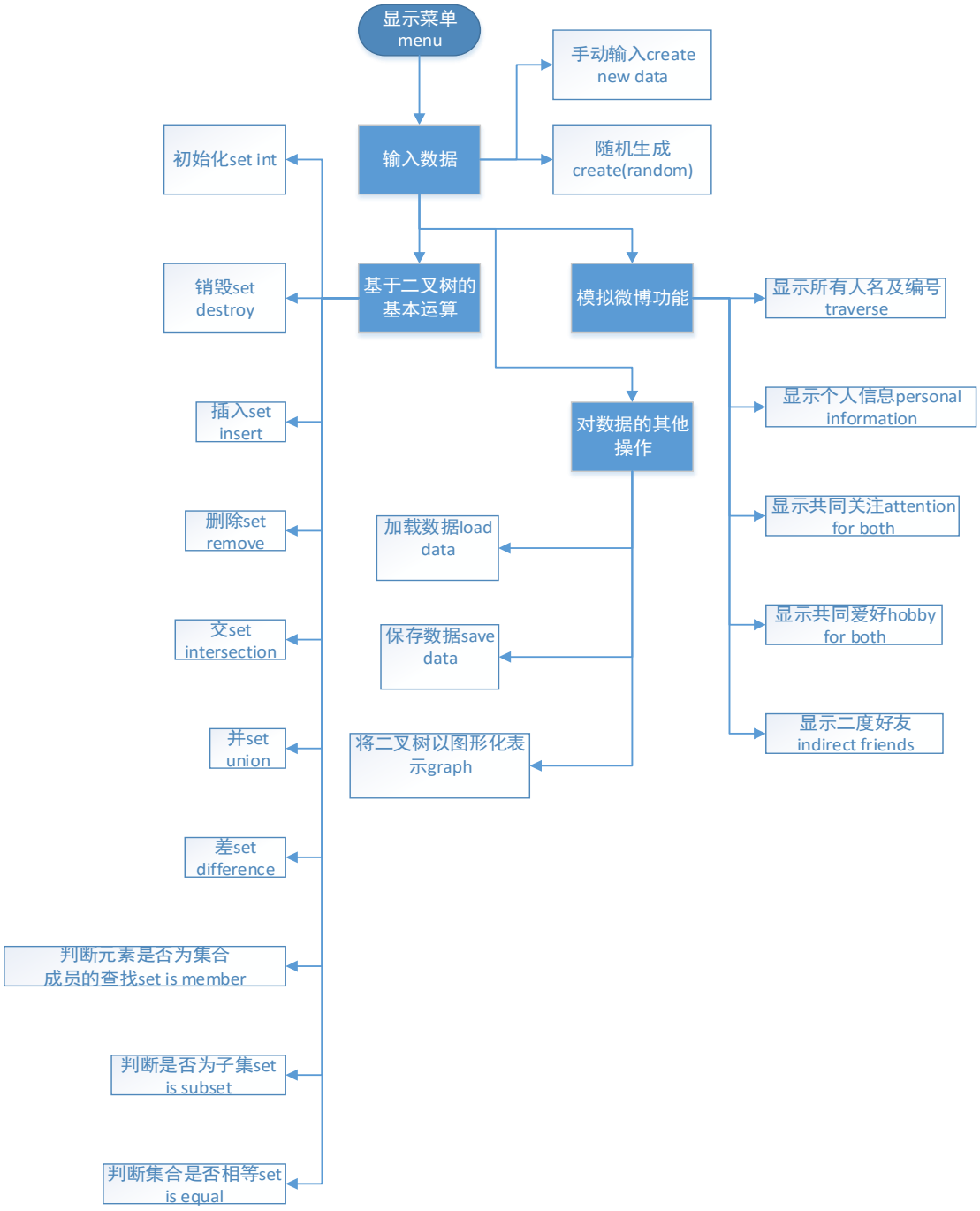


图 2-1 系统总体设计流程图

3 系统详细设计

3.1 有关数据结构的定义

表 3-1 数据项及类型

数据项	数据类型	数据项	数据类型
存储结构BSTNode	BSTNode	姓lastname	char[]
存储信息data	Info	名givenname	char[]
平衡因子bf	int	朋友friends	Bstree
深度h	int	粉丝fans	Bstree
左孩子lchild	BSTree	关注attention	Bstree
右孩子rchild	BSTree	兴趣hobby	Bstree
编号id	int		

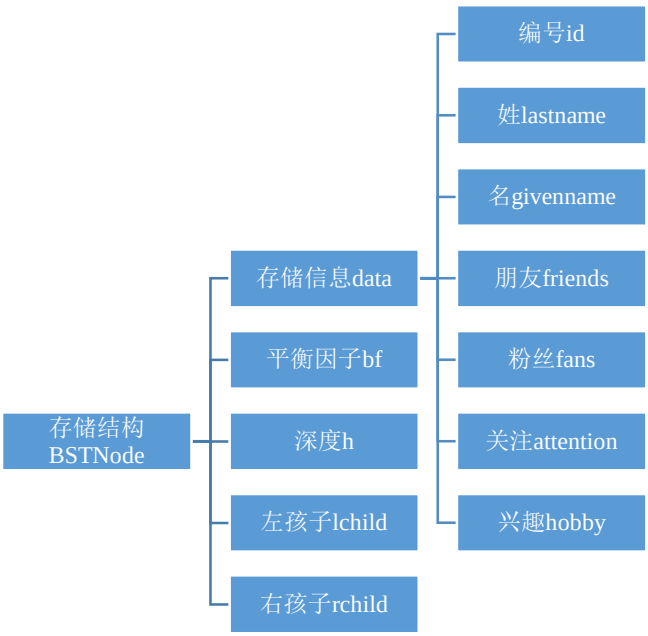


图 3-1 数据间的关系

3. 2主要算法设计

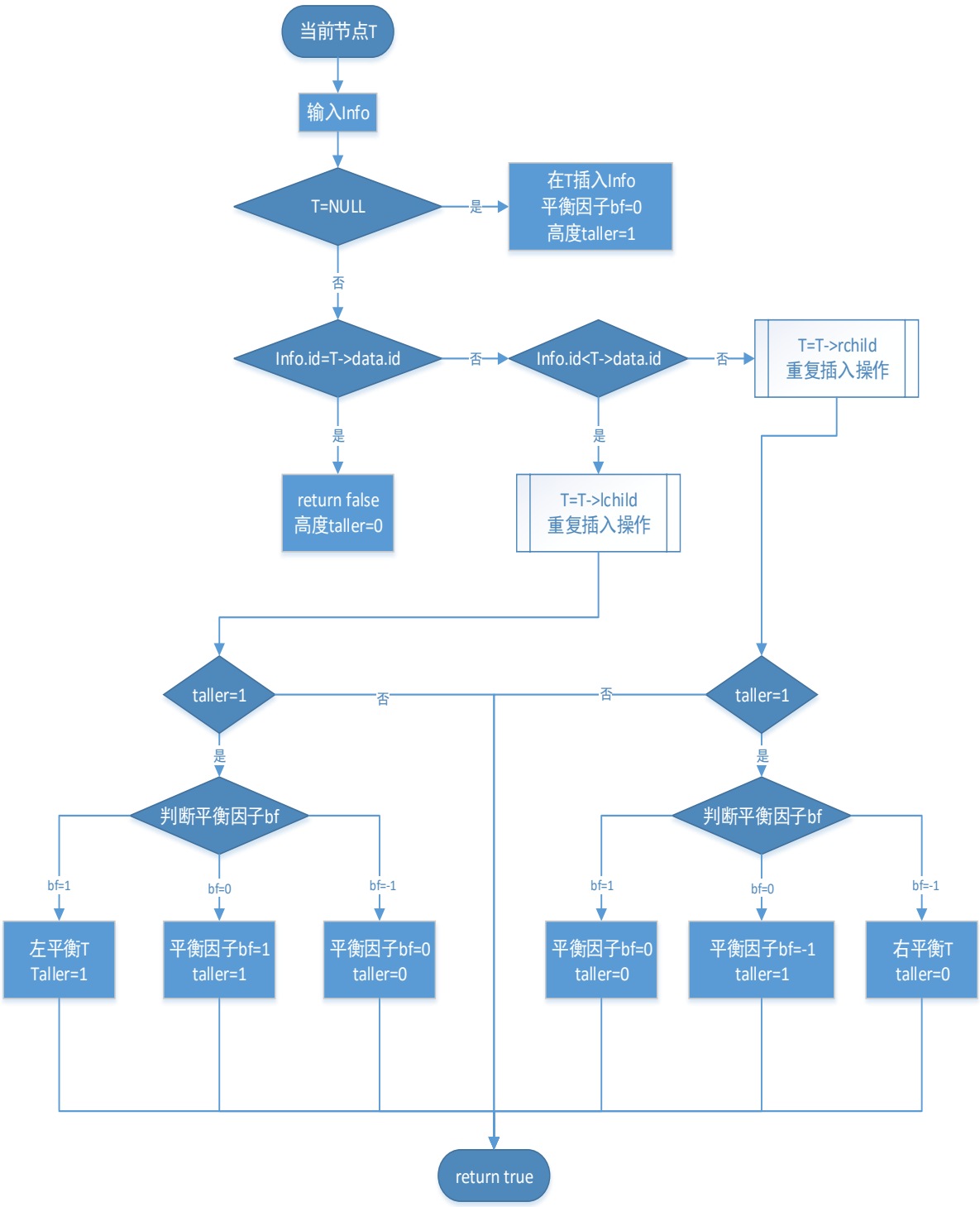


图 3-2 插入函数流程图

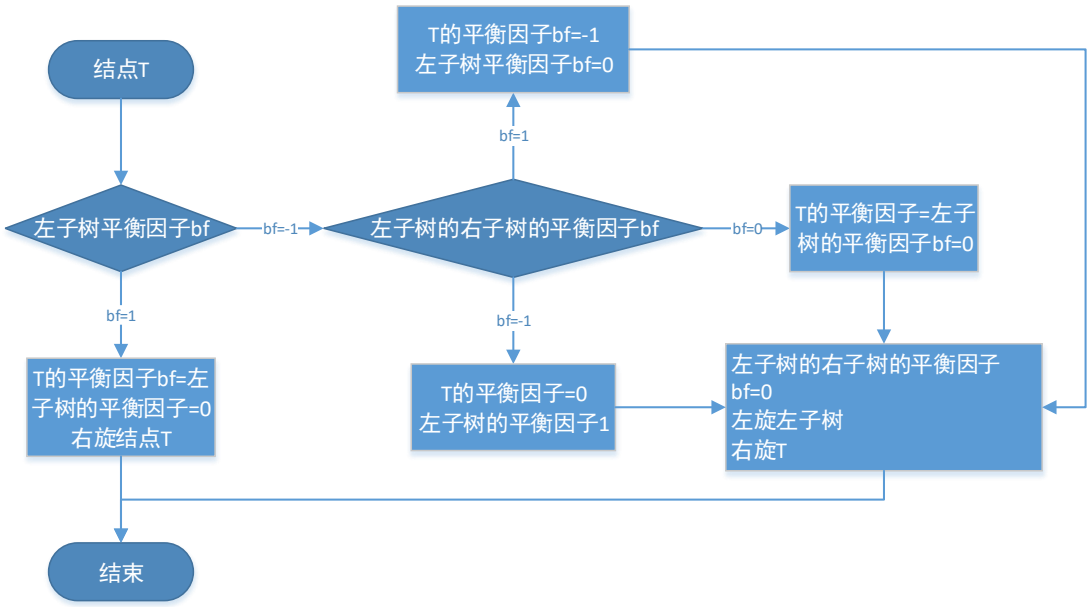


图 3-3 左平衡函数流程图

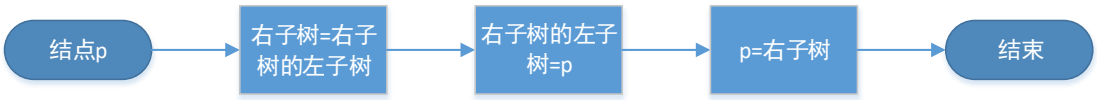


图 3-4 左旋函数流程图

4 系统实现与测试

4.1 系统实现

程序能够实现对平衡二叉树的基本操作，并在此基础上数显对微博用户集合的常用操作。

编译环境为 windows，编译软件为 codeblocks。

所包含的函数及对应功能为：

void set_init(BSTree *T);初始化二叉树；

void set_destory(BSTree *T);销毁二叉树；

bool set_insert(BSTree *T,Info e,bool *taller);在二叉树中插入信息为 e 的结点；

void LeftBalance(BSTree *T);左平衡函数；

void RightBalance(BSTree *T);右平衡函数

void L_Rotate(BSTree *p);左旋函数；

void R_Rotate(BSTree *p);右旋函数；

bool set_remove(BSTree *T,int key,bool *shorter);在二叉树中移除编号为 key 的结点；

void set_intersection(BSTree T,BSTree T1,BSTree *T0);求二叉树 T 和 T1 的交集，并赋值给给 T0；

void set_union(BSTree *T,BSTree T1);求二叉树 T 和 T1 的并集，并赋值给 T；

void set_difference(BSTree *T,BSTree T1);求二叉树 T 和 T1 的差，并赋值给 T；

bool set_is_member(BSTree T,int key,BSTree *Ts);判断编号为 key 的节点是否为二叉树 T 的元素，如果是返回 true，并赋值给 Ts；反之返回 false；

bool set_is_subset(BSTree TS,BSTree T1);判断二叉树 T1 是否为二叉树 TS 的子集，如果是返回 true；反之返回 false；

bool set_is_equal(BSTree T,BSTree T1);判断二叉树 T 和 T1 是否相等，如果是返回 true；反之返回 false；

bool load_data(BSTree *T,char *filename);加载名为 filename 的文件，并赋值给 T，如果成功返回 T；反之返回 false；

bool save_data(BSTree T,FILE *fp);保存二叉树到 fp 指向的文件里，如果成功返回 true；反之返回 false；

void input_data(Info *data);输入函数；

void create(BSTree *T);手动创建一个新的二叉树；

void height(BSTree T,int i);计算二叉树所有节点的深度并赋值；

void graph(BSTree T,int x,int y,visit fp);将二叉树图形化表是；

void operate_id(BSTree T);输出所有成员的 id；

void operate_relation(BSTree T1,BSTree T);输出所有成员的人际关系；

void operate_hobby(BSTree T1,BSTree H);输出所有成员的爱好；

void gotoxy(int x,int y);移动光标；

void menu();显示菜单；

bool person_input(BSTree *T,int p_gross);生成一个数量为 p_gross 的二叉树；

bool hobby_set_input(BSTree *H,int *h_gross);生成一个数量为 h_gross 的二叉树；

void relation_input(BSTree *T,int p_gross,int h_gross);为二叉树的每个节点随机生成人际关系和兴趣爱好；

void id_input(BSTree *T,int gross,int max,int min);随机生成 id 在 0~gross，数量为 min~max 的二叉树；

void PreOrderTraverse(BSTree T);遍历输出每个节点；

void indirect_friends_traverse(BSTree T,BSTree Tf,BSTree *T0);遍历输出二度好友；

void indirect_friends_insert(BSTree T,BSTree *T0);将二度好友赋值给 T0;

void complete_traverse(BSTree T,BSTree T0);完善二叉树；

void complete_friends(BSTree T,BSTree T1,BSTree Tf);完善二叉树，防止朋友的朋友集没有自己的情况出现；

void complete_fans(BSTree T,BSTree T1,BSTree Tf); 完善二叉树，防止关注人的粉丝集没有自己的情况出现；

void complete_sttention(BSTree T,BSTree T1,BSTree Tf); 完善二叉树，防止粉丝集的关注人集没有自己的情况出现；

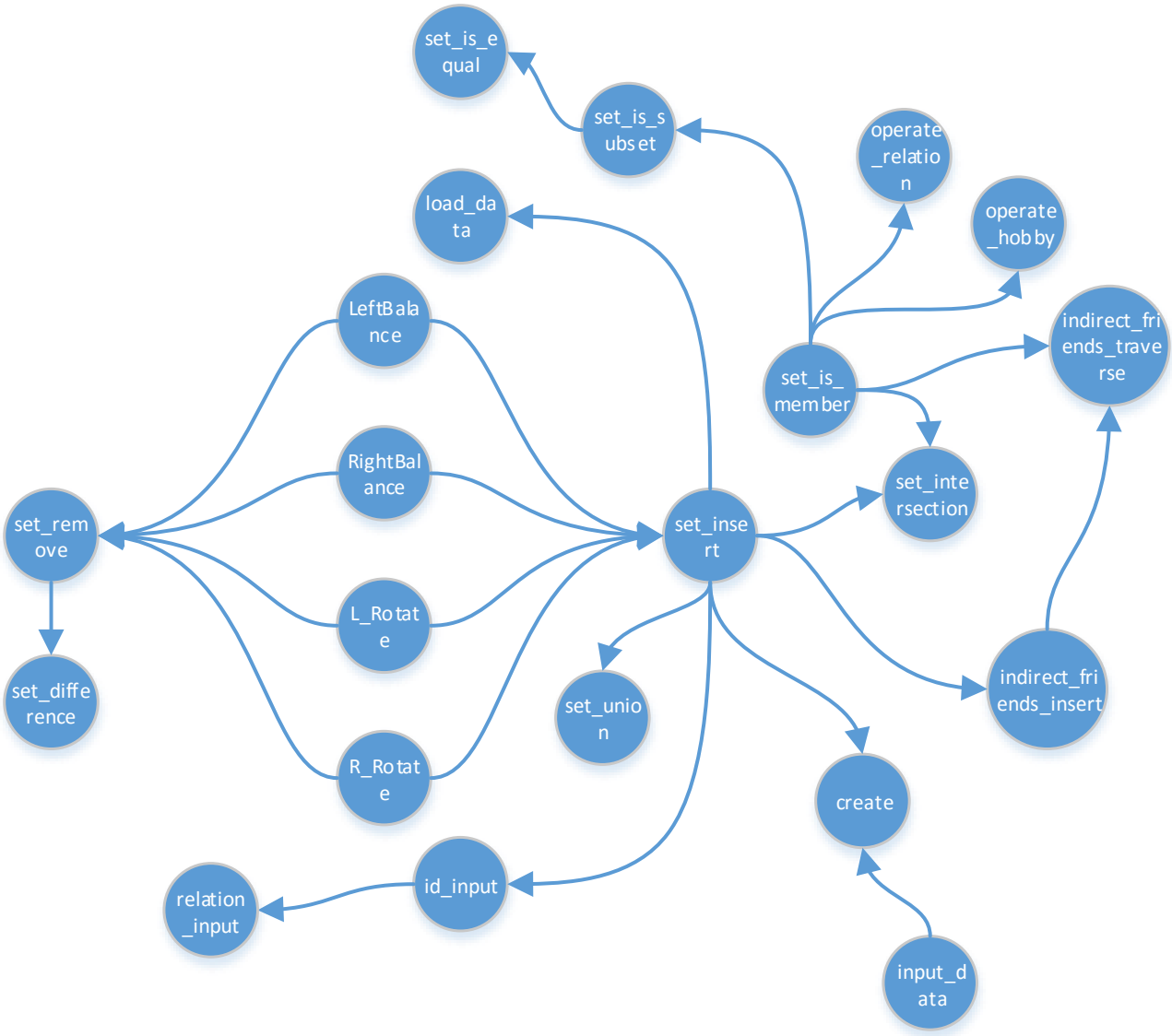


图 4-1 函数关系图

4.2 系统测试

首先对小数据集进行测试，利用 create 函数创建两个 AVL 树，然后分别测试基本功能和对微博数据集的生成和处理能力。

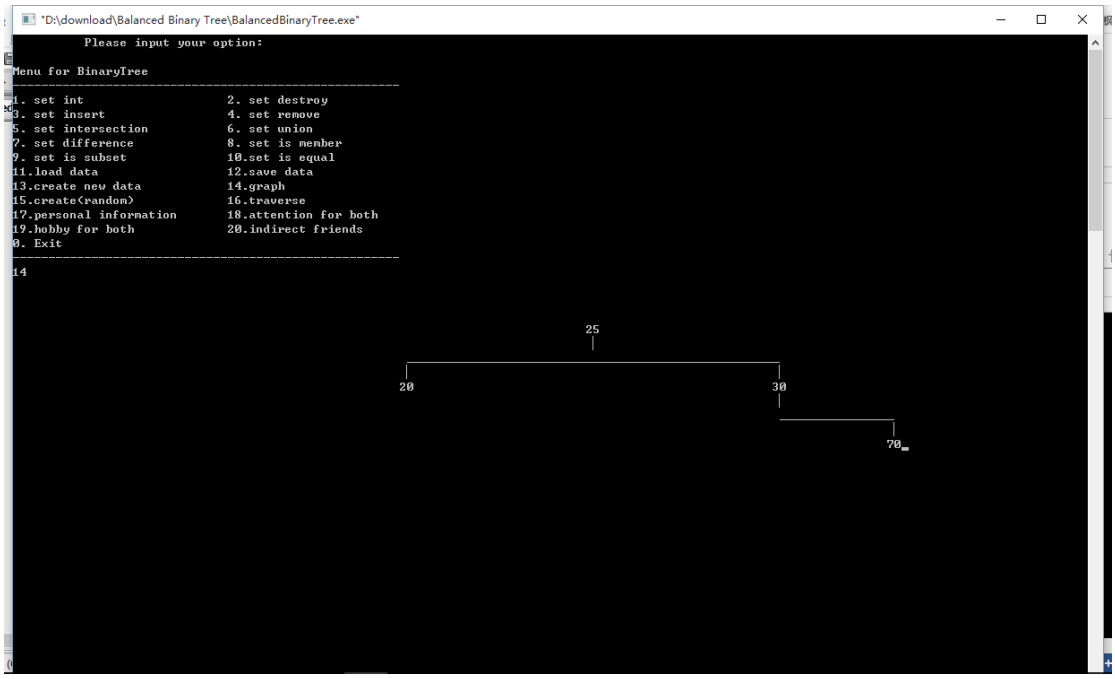


图 4-1 二叉树 a

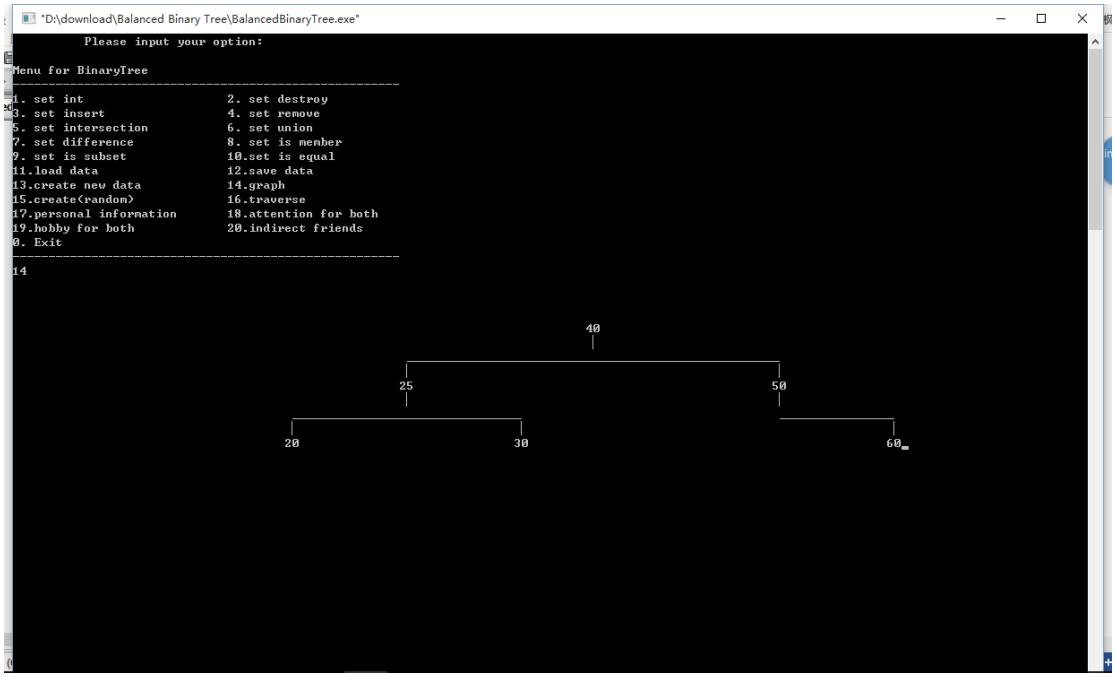


图 4-2-2 二叉树 b

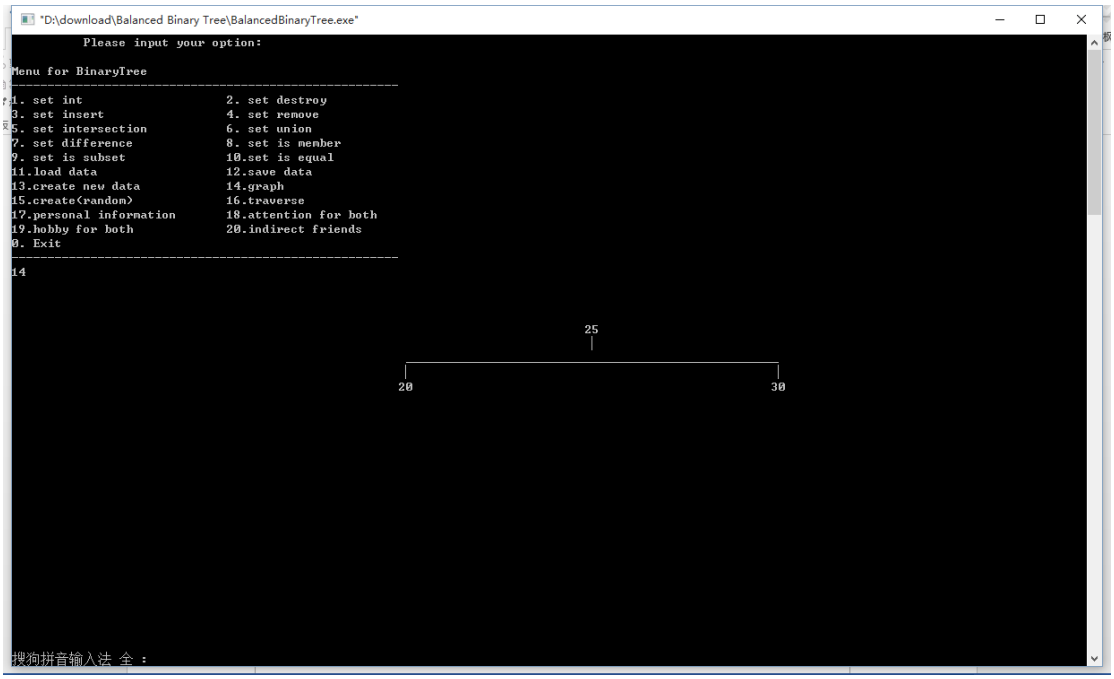


图 4-3 求两个树的交集

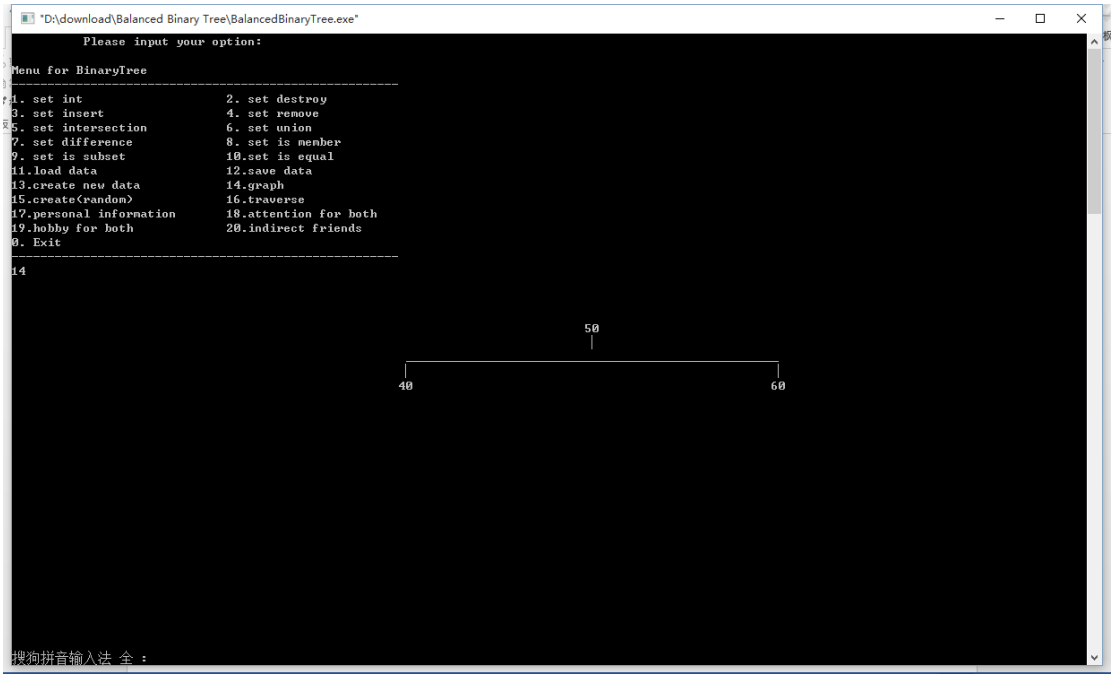


图 4-4 求两个树的差

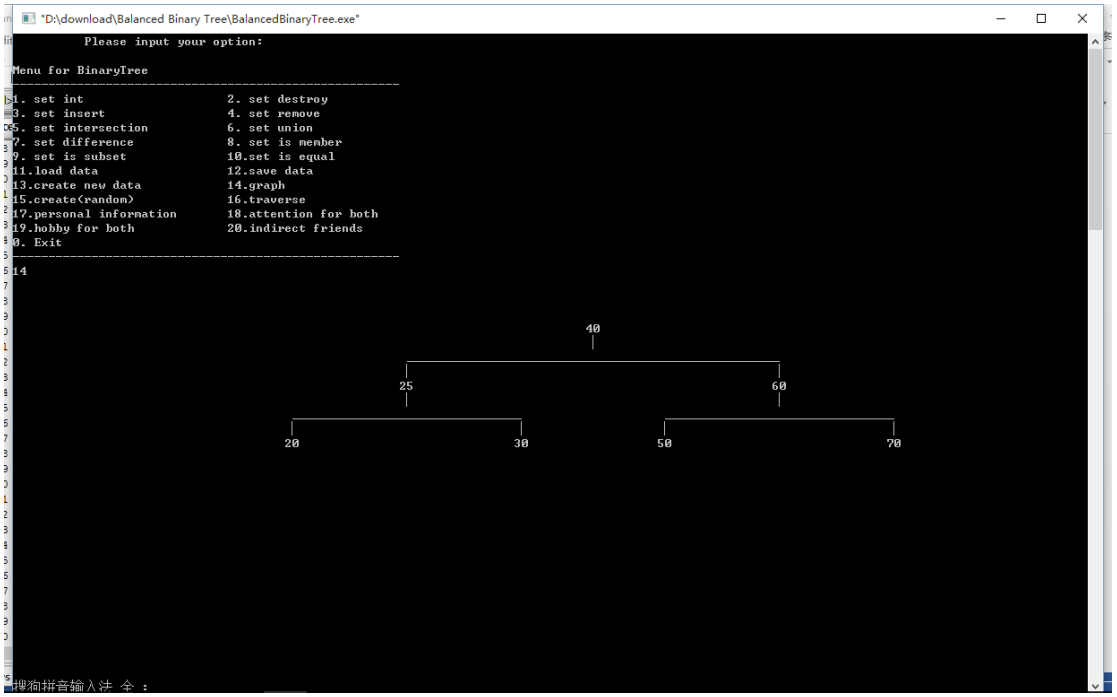
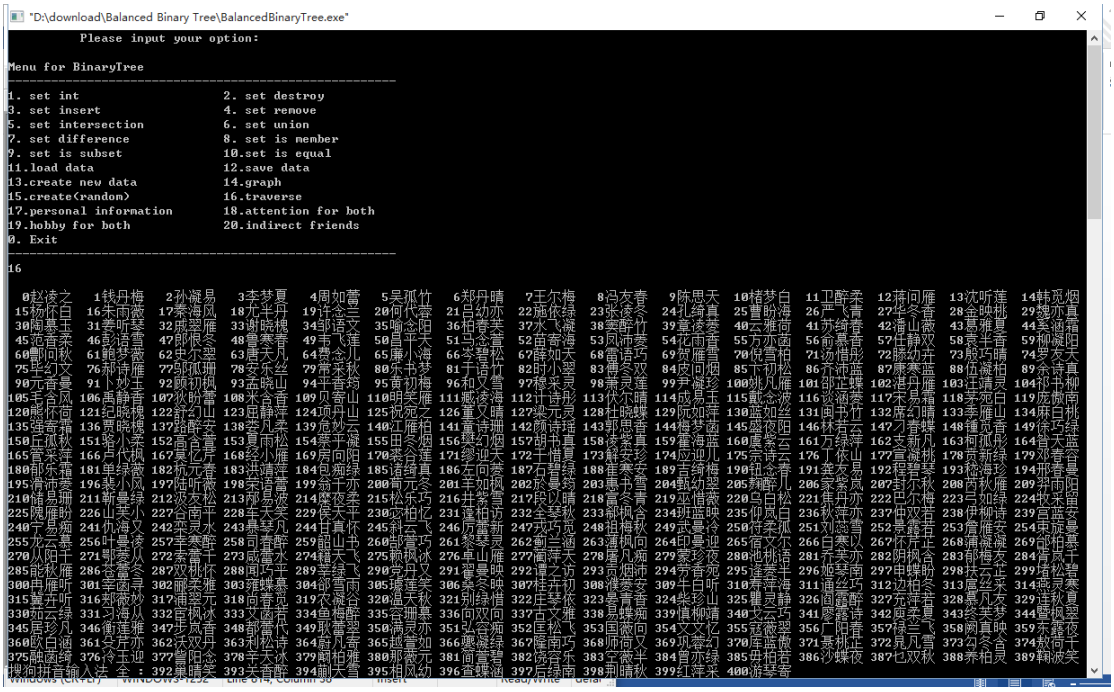

"D:\download\Balanced Binary Tree\BalancedBinaryTree.exe"
Please input your option:
Menu for BinaryTree
1. set int
2. set destroy
3. set insert
4. set remove
5. set intersection
6. set union
7. set difference
8. set is member
9. set is subset
10.set is equal
11.load data
12.save data
13.create new data
14.graph
15.create(random)
16.traverse
17.personal information
18.attention for both
19.hobby for both
20.indirect friends
0. Exit
14
40
25
60
20
30
50
70
搜狗拼音输入法 全 ：

图 4-5 求两个树的并集


"D:\download\Balanced Binary Tree\BalancedBinaryTree.exe"
Please input your option:
Menu for BinaryTree
1. set int
2. set destroy
3. set insert
4. set remove
5. set intersection
6. set union
7. set difference
8. set is member
9. set is subset
10.set is equal
11.load data
12.save data
13.create new data
14.graph
15.create(random)
16.traverse
17.personal information
18.attention for both
19.hobby for both
20.indirect friends
0. Exit
16
搜狗拼音输入法 全 ：

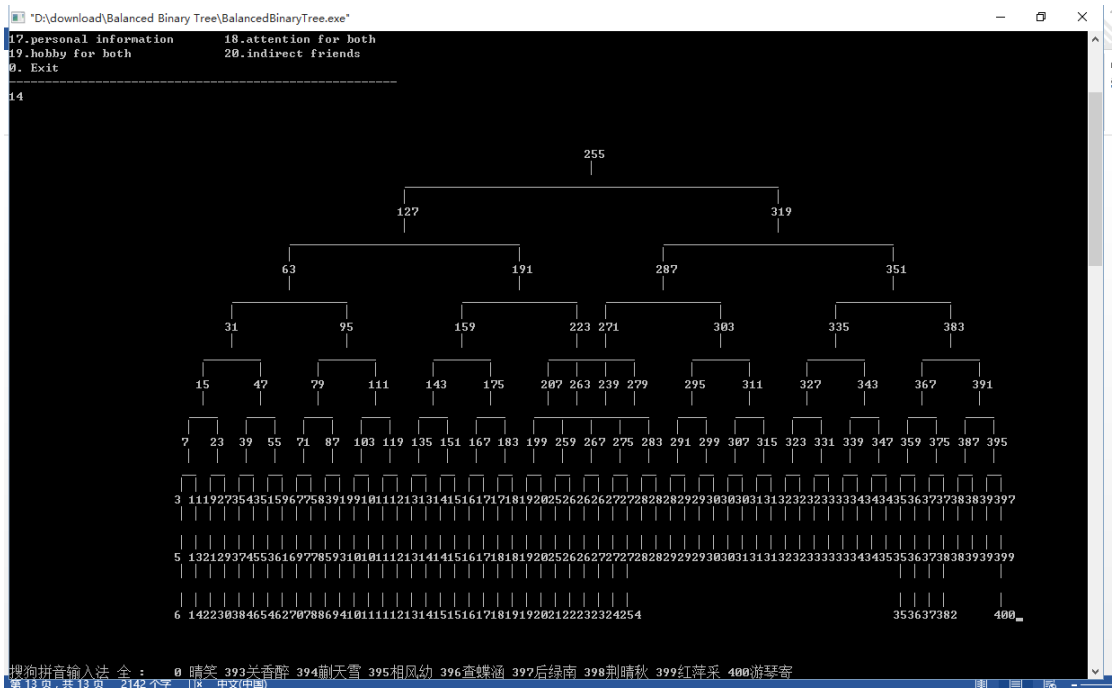


图 4-6-2 随机生成大数据集并遍历显示

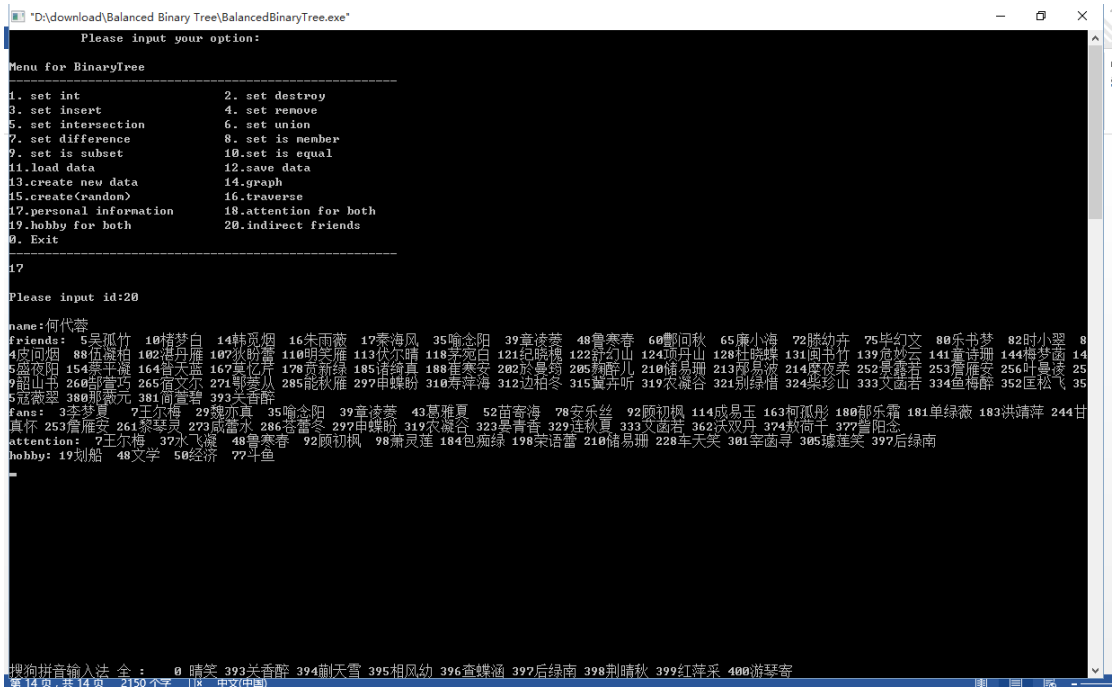


图 4-7 输出某个人的信息

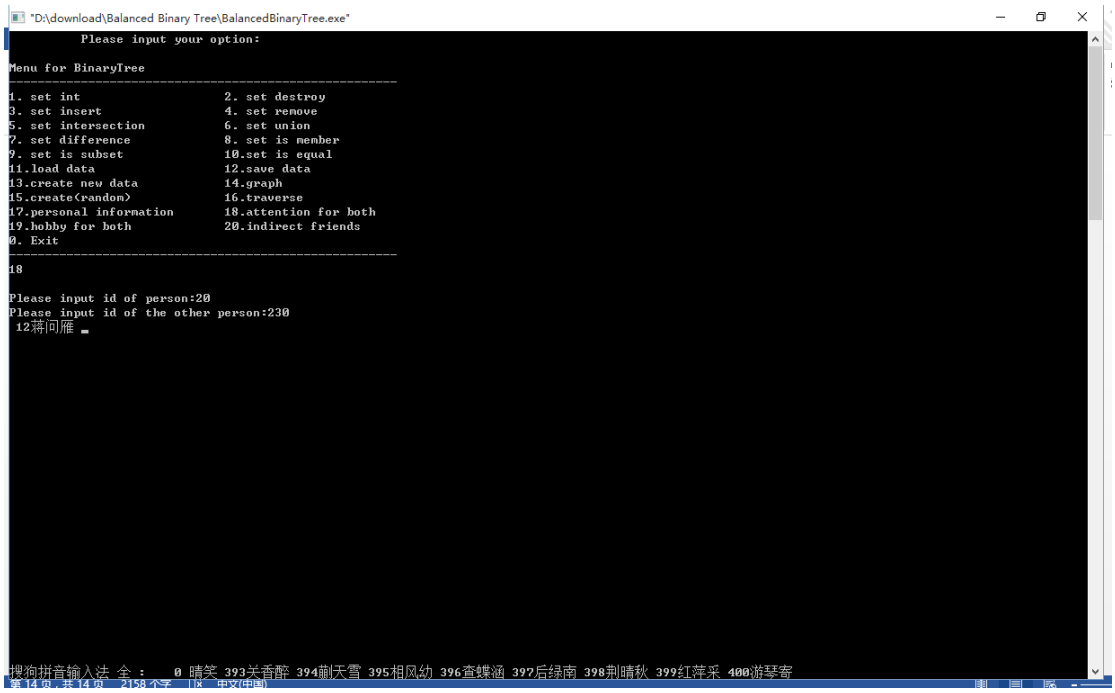


图 4-8 两个人的共同关注

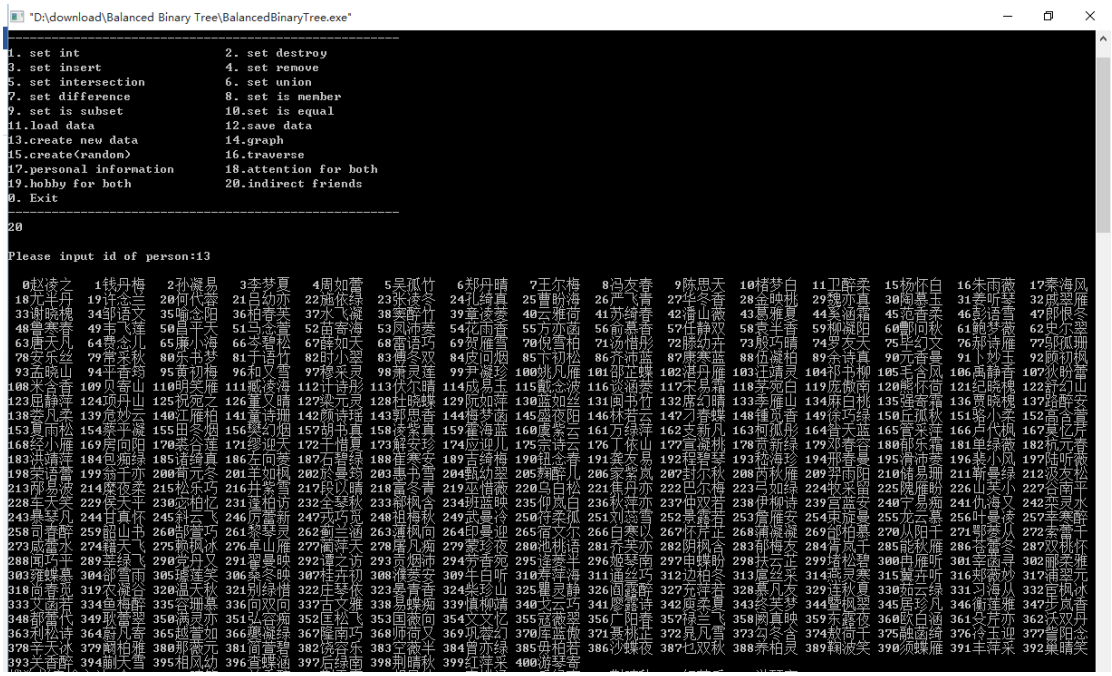


图 4-9 某个人的二度好友

结果与预期相同，程序运行无误，模块设计目标达成。

5 总结与展望

5.1 全文总结

1) 深刻理解了平二叉树的构建算法
2) 进一步熟悉了递归算法的运用
3) 开始熟悉从产品的角度，以实用性为要务来构建算法和程序

5.2 工作展望

在今后的研究中，围绕着如下几个方面开展工作

1) 寻求更高效的算法
2) 将界面更加人性化
3) 简化操作步骤，优先考虑使用者体验

6 体会

本次课程设计，使我对《数据结构》这门课程有了更深入的理解。《数据结构》是一门实践性较强的课程，为了学好这门课程，必须在掌握理论知识的同时，加强上机实践。

我的课程设计题目是平衡二叉树的运算。刚开始做这个程序的时候，感到完全无从下手，于是开始查阅各种资料以及参考文献，之后便开始着手写程序，写完运行时有很多问题。特别是实现平衡二叉树的删除运算时很多情况没有考虑周全，经常运行出现错误，但通过仔细阅读书本及资料最终解决问题。

在本课程设计中，我明白了理论与实际应用相结合的重要性，并提高了自己组织数据及编写大型程序的能力。培养了基本的、良好的程序设计技能以及合作能力。这次课程设计同样提高了我的综合运用所学知识的能力。《数据结构》是一门实践性很强的课程，上机实习是对学生全面综合素质进行训练的一种最基本的方法，是与课堂听讲、自学和练习相辅相成的、必不可少的一个教学环节。上机实习一方面能活用书本上的知识，起到深化理解和灵活掌握教学内容的目的；另一方面，上机实习是对学生软件设计的综合能力的训练，包括问题分析，总体结构设计，程序设计基本技能和技巧的训练。此外，还有更重要的一点是：机器是比任何教师更严厉的检查者。因此，在数据结构的学习过程中，必须严格按照老师的要求，主动地、积极地、认真地做好每一个实验，以不断提高自己的编程能力与专业素质。

通过这段时间的课程设计，我认识到数据结构是一门比较难的课程。需要多花时间上机练习。这次的程序训练培养了我实际分析问题、编程和动手能力，使我掌握了程序设计的基本技能，提高了我适应实际，实践编程的能力。

总的来说，这次课程设计让我获益匪浅，对数据结构也有了进一步的理解和认识。

参考文献

1. 谭浩强等 编著. C++面向对象程序设计. 北京:清华大学出版社，2006
2. 陈清华 朱红主编. Visual C++课程设计案例精选与编程指导. 南京:东南大学出版社，2003.06
3. 徐孝凯 等著，《数据结构（C 语言描述）》，清华大学出版社，2004
4. 严蔚敏, 吴伟民. 数据结构（C 语言版）. 北京: 清华大学出版社,1997
5. 严蔚敏, 吴伟民, 米宁. 数据结构题集（C 语言版）. 北京: 清华大学出版社,1999
6. Lin Chen. O(1) space complexity deletion for AVL trees, Information Processing Letters, 1986, 22(3)：147-149
7. S.H. Zweben, M. A. McDonald. **An optimal method for deletion in one-sided height-balanced trees,** Communications of the ACM, 1978, 21(6): 441-445
8. Guy Blelloch. Principles of Parallel Algorithms and Programming, CMU, 2014
9. CSDNhttp://blog.csdn.net/sysu_arui/article/details/7897017
10. 中国知网论文《平衡二叉树可视化演示系统的设计与实现》
11. 论文《平衡排序二叉树的 C++算法实现》
12. C 博客 LeetCode: Balanced Binary Tree
13. C Primer Plus(英文版)　Stephen Prata 著
14. C Primer Plus 中文第五版
15. http://pages.cs.wisc.edu/~ealexand/cs367/NOTES/AVL-Trees/index.htm
16. https://www.cs.usfca.edu/~galles/visualization/AVLtree.html

附录

头文件 header.h

```
#ifndef HEADER_H_INCLUDED
#define HEADER_H_INCLUDED
#include<stddef.h>
#include<stdbool.h>
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#include<windows.h>
#include<time.h>
#define LH 1
#define EH 0
#define RH -1
typedef struct Info{
    int id;
    char lastname[3];
    char givenname[5];
    struct BSTNode *friends;
    struct BSTNode *fans;
    struct BSTNode *attention;
    struct BSTNode *hobby;
}Info;
typedef struct BSTNode{
    Info data;
    int bf;
    int h;
    struct BSTNode *lchild,*rchild;
}BSTNode,*BSTree;
```

```
typedef void (*visit)(BSTree T);
void set_init(BSTree *T);
void set_destory(BSTree *T);
bool set_insert(BSTree *T,Info e,bool *taller);
void LeftBalance(BSTree *T);
void RightBalance(BSTree *T);
void L_Rotate(BSTree *p);
void R_Rotate(BSTree *p);
bool set_remove(BSTree *T,int key,bool *shorter);
void set_intersection(BSTree T,BSTree T1,BSTree *T0);
void set_union(BSTree *T,BSTree T1);
void set_difference(BSTree *T,BSTree T1);
bool set_is_member(BSTree T,int key,BSTree *Ts);
bool set_is_subset(BSTree TS,BSTree T1);
bool set_is_equal(BSTree T,BSTree T1);
bool load_data(BSTree *T,char *filename);
bool save_data(BSTree T,FILE *fp);
void input_data(Info *data);
void create(BSTree *T);
void height(BSTree T,int i);
void graph(BSTree T,int x,int y,visit fp);
void operate_id(BSTree T);
void operate_relation(BSTree T1,BSTree T);
void operate_hobby(BSTree T1,BSTree H);
void gotoxy(int x,int y);
void menu();
bool person_input(BSTree *T,int p_gross);
bool hobby_set_input(BSTree *H,int *h_gross);
void relation_input(BSTree *T,int p_gross,int h_gross);
void id_input(BSTree *T,int gross,int max,int min);
```

```
    void PreOrderTraverse(BSTree T);
    void indirect_friends_traverse(BSTree T,BSTree Tf,BSTree *T0);
    void indirect_friends_insert(BSTree T,BSTree *T0);
    void adjust_traverse(BSTree T);
    void complete_traverse(BSTree T,BSTree T0);
    void complete_friends(BSTree T,BSTree T1,BSTree Tf);
    void complete_fans(BSTree T,BSTree T1,BSTree Tf);
    void complete_sttention(BSTree T,BSTree T1,BSTree Tf);
    bool taller,shorter;
    int x0=80,y0=20;
    BSTree Ts=NULL;
#endif // HEADER_H_INCLUDED
```

主程序 BalancedBinaryTree.c

```
    #include"header.h"
    main()
    {
        int i;
        int p_gross=400,h_gross;
        BSTree T=NULL;
        BSTree H=NULL;
        srand((unsigned) time(NULL));
        system("mode con: cols=150 lines=160");
        do{
            printf("                Please input your option:");
            menu();
            scanf("%d",&i);getchar();
            printf("\n");
            switch(i){
                case 0:break;
```

```
case 1:set_init(&T);break;
case 2:{
    set_destory(&T);
    printf("Successed to destory!\n");
    break;
}
case 3:{
    Info sdata;
    input_data(&sdata);
    if(set_insert(&T,sdata,&taller)){
        printf("Successed to insert!\n");
        height(T,0);
    }
    else
        printf("Failed to insert!\n");
        break;
}
case 4:{
    int key;
    printf("Please input id:");
    scanf("%d",&key);getchar();
    printf("\n");
    if(set_remove(&T,key,&shorter)){
        printf("Successed to remove!\n");
        height(T,0);
    }
    else
        printf("Failed to remove!\n");
        break;
}
```

```
case 5:{
        char filename[20];
        BSTree T1=NULL;
        BSTree T0=NULL;
        printf("Please input filename:");
        scanf("%s",filename);getchar();
        printf("\n");
        if(load_data(&T1,filename)){
            printf("Successed to load!\n");
            set_intersection(T,T1,&T0);
            set_init(&T);
            T=T0;
            height(T,0);
            printf("Successed to take the intersection!\n");
        }
        else
            printf("Failed to load!\n");
        break;
}
case 6:{
        char filename[20];
        BSTree T1=NULL;
        printf("Please input filename:");
        scanf("%s",filename);getchar();
        printf("\n");
        if(load_data(&T1,filename)){
            printf("Successed to load!\n");
            set_union(&T,T1);
            height(T,0);
            printf("Successed to union!\n");
```

```
        }
        else
            printf("Failed to load!\n");
        break;
    }
    case 7:{
        char filename[20];
        BSTree T1=NULL;
        printf("Please input filename:");
        scanf("%s",filename);getchar();
        printf("\n");
        if(load_data(&T1,filename)){
            printf("Successed to load!\n");
            set_difference(&T,T1);
            height(T,0);
            printf("Successed to union!\n");
        }
        else
            printf("Failed to load!\n");
        break;
    }
    case 8:{
        int key;
        printf("Please input id:");
        scanf("%d",&key);getchar();
        printf("\n");
        if(set_is_member(T,key,&Ts))
            printf("Yes\n");
        else
            printf("No\n");
```

```
        break;
    }
    case 9:{
        char filename[20];
        BSTree T1=NULL;
        printf("Please input filename:");
        scanf("%s",filename);getchar();
        printf("\n");
        if(load_data(&T1,filename)){
            printf("Successed to load!\n");
            if(set_is_subset(T,T1))
                printf("Yes\n");
            else
                printf("No\n");
        }
        else
            printf("Failed to load!\n");
        break;
    }
    case 10:{
        char filename[20];
        BSTree T1=NULL;
        printf("Please input filename:");
        scanf("%s",filename);getchar();
        printf("\n");
        if(load_data(&T1,filename)){
            if(set_is_equal(T,T1))
                printf("Yes\n");
            else
                printf("No\n");
```

```
            }
            else
                printf("Failed to load!\n");
            break;
        }
        case 11:{
            char filename[20];
            printf("Please input filename:");
            scanf("%s",filename);getchar();
            printf("\n");
            if(load_data(&T,filename))
                printf("Successed to load!\n");
            break;
        }
        case 12:{
            FILE *fp;
            char filename[20];
            printf("Please input filename:");
            scanf("%s",filename);getchar();
            printf("\n");
            if((fp=fopen(filename,"wb"))==NULL){
                printf("Failed to open file!\n");
                break;
            }
            if(save_data(T,fp))
                printf("Successed to save!\n");
            else
                printf("Failed to save!\n");
            fclose(fp);
            break;
```

```
}
case 13:create(&T);break;
case 14:{
    visit fp=&operate_id;
    graph(T,0,0,fp);
    break;
}
case 15:{
    BSTree T0;
    if(person_input(&T,p_gross))
        printf("Successed to input person information!\n");
    else{
        printf("Failed to input person information!\n");
        break;
    }
    if(hobby_set_input(&H,&h_gross))
        printf("Successed to input hobby information!\n");
    else{
        printf("Failed to input person information!\n");
        break;
    }
    relation_input(&T,p_gross,h_gross);
    T0=T;
    complete_traverse(T,T0);
    break;
}
case 16:{
    PreOrderTraverse(T);
    break;
}
```

```
                case 17:{
                    int key;
                    printf("Please input id:");
                    scanf("%d",&key);getchar();
                    if(!set_is_member(T,key,&Ts)){
                        printf("No such person!\n");
                        break;
                    }
                    printf("\n");

printf("name:%s%s\n",Ts->data.lastname,Ts->data.givenname);
                    printf("friends:");
                    operate_relation(Ts->data.friends,T);printf("\n");
                    printf("fans:");
                    operate_relation(Ts->data.fans,T);printf("\n");
                    printf("attention:");
                    operate_relation(Ts->data.attention,T);printf("\n");
                    printf("hobby:");
                    operate_hobby(Ts->data.hobby,H);printf("\n");
                    break;
                }
                case 18:{
                    int key1,key2;
                    BSTree T1=NULL,T2=NULL,T0=NULL;
                    printf("Please input id of person:");
                    scanf("%d",&key1);getchar();
                    printf("Please input id of the other person:");
                    scanf("%d",&key2);getchar();
                    if(!set_is_member(T,key1,&T1)){
                        printf("No such person!\n");
```

```
            break;
        }
        if(!set_is_member(T,key2,&T2)){
            printf("No such person!\n");
            break;
        }
        set_intersection(T1->data.attention,T2->data.attention,&T0);
        if(T0){
            operate_relation(T0,T);
            break;
        }
        else{
            printf("No attention for both!\n");
            break;
        }
    }
    case 19:{
        int key1,key2;
        BSTree T1=NULL,T2=NULL,T0=NULL;
        printf("Please input id of person:");
        scanf("%d",&key1);getchar();
        printf("Please input id of the other person:");
        scanf("%d",&key2);getchar();
        if(!set_is_member(T,key1,&T1)){
            printf("No such person!\n");
            break;
        }
        if(!set_is_member(T,key2,&T2)){
            printf("No such person!\n");
            break;
```

```
                }
                set_intersection(T1->data.hobby,T2->data.hobby,&T0);
                if(T0){
                    operate_hobby(T0,H);
                    break;
                }
                else{
                    printf("No hobby for both!\n");
                    break;
                }
            }
            case 20:{
                int key;
                BSTree T1=NULL,T0=NULL;
                printf("Please input id of person:");
                scanf("%d",&key);getchar();
                printf("\n");
                if(!set_is_member(T,key,&T1)){
                    printf("No such person!\n");
                    break;
                }
                indirect_friends_traverse(T,T1->data.friends,&T0);
                set_difference(&T0,T1);
                operate_relation(T0,T);
                break;
            }
            default:printf("Input error!\n");
        }
        getchar();
        system("cls");
```

```
    }while(i);
    printf("\n--------------------Welcome again!--------------------\n");
}
void set_init(BSTree *T)
{
    *T=NULL;
    printf("Successed to init!\n");
}
void set_destory(BSTree *T)
{
    if((*T)!=NULL){
        set_destory(&((*T)->lchild));
        set_destory(&((*T)->rchild));
        free(*T);
    }
}
bool set_insert(BSTree *T,Info e,bool *taller)
{
    if(!(*T)){
        (*T)=(BSTree)malloc(sizeof(BSTNode));
        (*T)->data=e;
        (*T)->lchild=(*T)->rchild=NULL;
        (*T)->bf=EH;
        *taller=true;
    }
    else{
        if(e.id==(*T)->data.id){
            *taller=false;
            return false;
        }
```

```
        if(e.id<(*T)->data.id){
            if(!set_insert(&((*T)->lchild),e,taller))return false;
            if(*taller)
                switch((*T)->bf){
                    case LH:
                        LeftBalance(T);*taller=false;break;
                    case EH:
                        (*T)->bf=LH;*taller=true;break;
                    case RH:
                        (*T)->bf=EH;*taller=false;break;
                }
        }
        else{
            if(!set_insert(&((*T)->rchild),e,taller))return false;
            if(*taller)
                switch((*T)->bf){
                    case LH:
                        (*T)->bf=EH;*taller=false;break;
                    case EH:
                        (*T)->bf=RH;*taller=true;break;
                    case RH:
                        RightBalance(T);*taller=false;break;
                }
        }
    }
    return true;
}
void LeftBalance(BSTree *T)
{
    BSTree lc,rd;
```

```
    lc=(*T)->lchild;
    switch(lc->bf){
        case LH:
            (*T)->bf=lc->bf=EH;
            R_Rotate(T);break;
        case RH:
            rd=lc->rchild;
            switch(rd->bf){
                case LH:(*T)->bf=RH;lc->bf=EH;break;
                case EH:(*T)->bf=lc->bf=EH;break;
                case RH:(*T)->bf=EH;lc->bf=LH;break;
            }
            rd->bf=EH;
            L_Rotate(&((*T)->lchild));
            R_Rotate(T);
    }
}
void RightBalance(BSTree *T)
{
    BSTree rc,ld;
    rc=(*T)->rchild;
    switch(rc->bf){
        case RH:
            (*T)->bf=rc->bf=EH;
            L_Rotate(T);break;
        case LH:
            ld=rc->lchild;
            switch(ld->bf){
                case LH:(*T)->bf=EH;rc->bf=RH;break;
                case EH:(*T)->bf=rc->bf=EH;break;
```

```
                    case RH:(*T)->bf=LH;rc->bf=EH;break;
                }
                ld->bf=EH;
                R_Rotate(&((*T)->rchild));
                L_Rotate(T);
        }
}
void L_Rotate(BSTree *p)
{
        BSTree rc;
        rc=(*p)->rchild;
        (*p)->rchild=rc->lchild;
        rc->lchild=(*p);
        (*p)=rc;
}
void R_Rotate(BSTree *p)
{
        BSTree lc;
        lc=(*p)->lchild;
        (*p)->lchild=lc->rchild;
        lc->rchild=(*p);
        (*p)=lc;
}
bool set_remove(BSTree *T,int key,bool *shorter)
{
        if((*T)==NULL)return false;
        else if(key==(*T)->data.id){
                BSTree p;
                if((*T)->lchild==NULL){
                        p=(*T);
```

```
            (*T)=(*T)->rchild;
            free(p);
            *shorter=true;
        }
        else if((*T)->rchild==NULL){
            p=(*T);
            (*T)=(*T)->lchild;
            free(p);
            *shorter=true;
        }
        else{
            p=(*T)->lchild;
            while(p->rchild)p=p->rchild;
            (*T)->data=p->data;
            set_remove(&((*T)->lchild),p->data.id,shorter);
        }
    }
    else if(key<(*T)->data.id){
        if(!set_remove(&((*T)->lchild),key,shorter))return false;
        if(*shorter){
            switch((*T)->bf){
                case LH:(*T)->bf=EH;*shorter=true;break;
                case EH:(*T)->bf=RH;*shorter=false;break;
                case RH:
                    if((*T)->rchild->bf==EH)
                        *shorter=false;
                    else
                        *shorter=true;
                    RightBalance(T);
                    break;
```

```
                }
            }
        }
        else{
            if(!set_remove(&((*T)->rchild),key,shorter))return false;
            if(*shorter){
                switch((*T)->bf){
                    case LH:
                        if((*T)->lchild->bf==EH)
                            *shorter=false;
                        else
                            *shorter=true;
                        LeftBalance(T);
                        break;
                    case EH:(*T)->bf=LH;*shorter=false;break;
                    case RH:(*T)->bf=EH;*shorter=true;break;
                }
            }
        }
        return true;
}
void set_intersection(BSTree T,BSTree T1,BSTree *T0)
{
        if(T1==NULL)return;
        if(set_is_member(T,T1->data.id,&Ts))
            set_insert(T0,T1->data,&taller);
        set_intersection(T,T1->lchild,T0);
        set_intersection(T,T1->rchild,T0);
}
void set_union(BSTree *T,BSTree T1)
```

```
{
    if(T1==NULL)return;
    set_insert(T,T1->data,&taller);
    set_union(T,T1->lchild);
    set_union(T,T1->rchild);
}
void set_difference(BSTree *T,BSTree T1)
{
    if(*T==NULL||T1==NULL)return;
    set_remove(T,T1->data.id,&shorter);
    set_difference(T,T1->lchild);
    set_difference(T,T1->rchild);
}
bool set_is_member(BSTree T,int key,BSTree *Ts)
{
    if(T==NULL)return false;
    if(T->data.id==key){
        *Ts=T;
        return true;
    }
    else if(key<T->data.id){
        if(set_is_member(T->lchild,key,Ts))return true;
    }
    else{
        if(set_is_member(T->rchild,key,Ts))return true;
    }
    return false;
}
bool set_is_subset(BSTree T,BSTree T1)
{
```

```
    if(T1==NULL)return true;
    if(set_is_member(T,T1->data.id,&Ts)){
        if(!set_is_subset(T,T1->lchild))
            return false;
        if(!set_is_subset(T,T1->rchild))
            return false;
        return true;
    }
    else return false;
}
bool set_is_equal(BSTree T,BSTree T1)
{
    if(!set_is_subset(T,T1))return false;
    if(!set_is_subset(T1,T))return false;
    return true;
}
bool load_data(BSTree *T,char *filename)
{
    BSTree p;
    FILE *fp;
    if((fp=fopen(filename,"rb"))==NULL){
        printf("Failed to open the file!\n");
        return false;
    }
    while(!feof(fp)){
        if((p=(BSTree)malloc(sizeof(BSTNode)))==NULL){
            printf("Failed to apply memory!\n");
            fclose(fp);
            return false;
        }
```

```
            if(fread(p,sizeof(BSTNode),1,fp)!=1){
                free(p);
                break;
            }
            set_insert(T,p->data,&taller);
        }
        fclose(fp);
        height(*T,0);
        return true;
}
bool save_data(BSTree T,FILE *fp)
{
        if(T){
            if(fwrite(T,sizeof(BSTNode),1,fp)!=1){
                printf("Failed to read-in the data!\n");
                fclose(fp);
                return false;
            }
        }
        else return true;
        if(!save_data(T->lchild,fp))return false;
        if(!save_data(T->rchild,fp))return false;
        return true;
}
void height(BSTree T,int i)
{
        if(T)T->h=i+1;
        else return;
        height(T->lchild,T->h);
        height(T->rchild,T->h);
```

```
}
void input_data(Info *data)
{
    printf("Please input id:");
    scanf("%d",&(data->id));getchar();
    printf("\n");
}
void create(BSTree *T)
{
    Info *q;
    char s;
    if((q=(Info *)malloc(sizeof(Info)))==NULL){
        printf("Failed to apply memory!\n");
        return;
    }
    input_data(q);
    if(set_insert(T,*q,&taller)){
        printf("Successed to create,do you want to continue?(y/n):");
        while(1){
            s=getchar();getchar();
            printf("\n");
            switch(s){
                case 'y':create(T);height(*T,0);return;
                case 'n':printf("Completed!\n");return;
                default:printf("Input error!Please input again:");
            }
        }
    }
    else
        printf("Failed to create!\n");
```

```
    return;
}
void graph(BSTree T,int x,int y,visit fp)
{
    if(T){
        gotoxy(x+x0,y+y0);
        fp(T);
    }
    else return;
    if(T->lchild||T->rchild){
        gotoxy(x+x0,y+y0+1);
        printf("© ¦");
        if(T->lchild){
            int i=80/(pow(2,T->h)+1);
            int j=i;
            gotoxy(x+x0-j,y+y0+3);
            printf("© ¦");
            gotoxy(x+x0-i+1,y+y0+2);
            while(i){
                printf("_");
                i--;
            }
            graph(T->lchild,x-j,y+4,fp);
        }
        if(T->rchild){
            int i=80/(pow(2,T->h)+1);
            int j=i;
            gotoxy(x+x0+1,y+y0+2);
            while(i){
                printf("_");
```

```
                    i--;
                }
                gotoxy(x+x0+j,y+y0+3);
                printf("© ");
                graph(T->rchild,x+j,y+4,fp);
            }
        }
    }
    void operate_id(BSTree T)
    {
        printf("%d",T->data.id);
    }
    void operate_relation(BSTree T1,BSTree T)
    {
        if(T1==NULL)return;
        operate_relation(T1->lchild,T);
        if(set_is_member(T,T1->data.id,&Ts))
            printf("%3d%s%s
",Ts->data.id,Ts->data.lastname,Ts->data.givenname);
        operate_relation(T1->rchild,T);
    }
    void operate_hobby(BSTree T1,BSTree H)
    {
        if(T1==NULL)return;
        operate_hobby(T1->lchild,H);
        if(set_is_member(H,T1->data.id,&Ts))
            printf("%3d%s ",Ts->data.id,Ts->data.givenname);
        operate_hobby(T1->rchild,H);
    }
    void menu()
```

```
{
    printf("\n\n");
    printf("Menu for BinaryTree \n");
    printf("-------------------------------------------------\n");
    printf("1. set int                    2. set destroy\n");
    printf("3. set insert                 4. set remove\n");
    printf("5. set intersection           6. set union\n");
    printf("7. set difference             8. set is member\n");
    printf("9. set is subset              10.set is equal\n");
    printf("11.load data                  12.save data\n");
    printf("13.create new data            14.graph\n");
    printf("15.create(random)             16.traverse\n");
    printf("17.personal information       18.attention for both\n");
    printf("19.hobby for both             20.indirect friends\n");
    printf("0. Exit\n");
    printf("-------------------------------------------------\n");
}
void gotoxy(int x,int y)
{
    COORD coord={x,y};

SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE),coord);
}
bool person_input(BSTree *T,int p_gross)
{
    FILE *fpl,*fpg;
    Info *q;
    int number;
    if((fpl=fopen("lastname.txt","r"))==NULL){
        printf("Failed to open the lastname!\n");
```

```
        return false;
    }
    if((fpg=fopen("givenname.txt","r"))==NULL){
        printf("Failed to open the givennname!\n");
        return false;
    }
    for(number=0;number<=p_gross;number++){
        if((q=(Info *)malloc(sizeof(Info)))==NULL){
            printf("Failed to apply memory!\n");
            fclose(fpl);fclose(fpg);
            return false;
        }
        if(feof(fpl)){
            fclose(fpl);
            if((fpl=fopen("lastname.txt","r"))==NULL){
                printf("Failed to open the lastname!\n");
                return false;
            }
        }
        else{
            if(fgets(q->lastname,3,fpl)==NULL){
            free(q);
            break;
            }
        }
        if(feof(fpg)){
            fclose(fpg);
            if((fpg=fopen("givenname.txt","r"))==NULL){
                printf("Failed to open the givennname!\n");
                return false;
```

```
                }
            }
            else{
                if(fgets(q->givenname,5,fpg)==NULL){
                free(q);
                break;
                }
            }
            q->id=number;
            q->friends=NULL;
            q->fans=NULL;
            q->attention=NULL;
            q->hobby=NULL;
            set_insert(T,*q,&taller);
            }
        fclose(fpl);fclose(fpg);
        height(*T,0);
        return true;
}
bool hobby_set_input(BSTree *H,int *h_gross)
{
        FILE *fp;
        Info *q;
        int number=0;
        if((fp=fopen("hobby.txt","r"))==NULL){
            printf("Failed to open the hobby!\n");
            return false;
        }
        while(!feof(fp)){
            if((q=(Info *)malloc(sizeof(Info)))==NULL){
```

```
                printf("Failed to apply memory!\n");
                fclose(fp);
                return false;
                }
            if(fgets(q->givenname,5,fp)==NULL){
                free(q);
                break;
            }
            q->id=number++;
            q->friends=NULL;
            q->fans=NULL;
            q->attention=NULL;
            q->hobby=NULL;
            set_insert(H,*q,&taller);
        }
        *h_gross=number;
        fclose(fp);
        height(*H,0);
        return true;
}
void relation_input(BSTree *T,int p_gross,int h_gross)
{
        if(*T==NULL)return;
        id_input(&((*T)->data.friends),p_gross,120,20);
        id_input(&((*T)->data.fans),p_gross,120,20);
        id_input(&((*T)->data.attention),p_gross,30,5);
        id_input(&((*T)->data.hobby),h_gross,10,2);
        relation_input(&((*T)->lchild),p_gross,h_gross);
        relation_input(&((*T)->rchild),p_gross,h_gross);
}
```

```
void id_input(BSTree *T,int gross,int max,int min)
{
    int i;
    Info *q;
    for(i=0;i<=rand()%max+min;i++){
        if((q=(Info *)malloc(sizeof(Info)))==NULL){
            printf("Failed to apply memory!\n");
            break;
        }
        q->id=rand()%gross;
        q->friends=NULL;
        q->fans=NULL;
        q->attention=NULL;
        q->hobby=NULL;
        set_insert(T,*q,&taller);
    }
}
void PreOrderTraverse(BSTree T)
{
    if(T==NULL)return;
    PreOrderTraverse(T->lchild);
    printf("%3d%s%s ",T->data.id,T->data.lastname,T->data.givenname);
    PreOrderTraverse(T->rchild);
}
void indirect_friends_traverse(BSTree T,BSTree Tf,BSTree *T0)
{
    if(Tf==NULL)return;
    set_is_member(T,Tf->data.id,&Ts);
    indirect_friends_insert(Ts->data.friends,T0);
    indirect_friends_traverse(T,Tf->lchild,T0);
```

```
        indirect_friends_traverse(T,Tf->rchild,T0);
    }
    void indirect_friends_insert(BSTree T,BSTree *T0)
    {
        if(T==NULL)return;
        set_insert(T0,T->data,&taller);
        indirect_friends_insert(T->lchild,T0);
        indirect_friends_insert(T->rchild,T0);
    }
    void complete_traverse(BSTree T,BSTree T0)
    {
        if(T0==NULL)return;
        complete_traverse(T,T0->lchild);
        complete_friends(T,T0,T0->data.friends);
        complete_traverse(T,T0->rchild);
    }
    void complete_friends(BSTree T,BSTree T1,BSTree Tf)
    {
        if(Tf==NULL)return;
        BSTree Ts0;
        set_is_member(T,Tf->data.id,&Ts);
        if(!set_is_member(Ts->data.friends,T1->data.id,&Ts0))
            set_insert(&Ts->data.friends,T1->data,&taller);
        complete_friends(T,T1,Tf->lchild);
        complete_friends(T,T1,Tf->rchild);
}
```