实验-6-递归与分治-参考程序

王 新 宇
计算机科学与技术系

```
//实验题第1题
//LinkList.h
template <class DataType>
Node<DataType> * LinkList<DataType>::Reverse(Node<DataType> *h)    //辅助函数
{
    Node<DataType> *t;
    if(!h || h->next == NULL)   return h;
    t = Reverse(h->next);
    h->next->next = h;
    h->next = NULL;
    return t;
}
template <class DataType>
void LinkList<DataType>::Reverse()          //对外的公有成员函数
{
    head = Reverse(head);                   //调用辅助函数
}
```

```
//实验题第2题
//BinaryTree.h
//判断r1和r2指向的两棵二叉树是否同构
template <class DataType>
bool BinaryTree<DataType>::Isomorphism(BTNode<DataType> *r1, BTNode<DataType> *r2)
const       //辅助函数
{
    if(r1 == NULL && r2 == NULL)    return true;
    if((r1 != NULL && r2 == NULL) || (r1 == NULL && r2 != NULL))
        return false;
    if(Isomorphism(r1->lChild, r2->lChild) && Isomorphism(r1->rChild, r2->rChild))
        return true;
    return false;
}
//对外的公有成员函数，判断与二叉树btree是否同构
template <class DataType>
bool BinaryTree<DataType>::Isomorphism(BinaryTree<DataType> & btree) const
{
    return Isomorphism(root, btree.GetRoot());                    //调用辅助函数
}
```

```
//实验题第3题
int Sum(int * t, int low, int high)  {
    int sum = 0;
    for (int i = low; i <= high; i++)    sum += t[i];
    return sum;
}
int Solve(int * weight, int low, int high)  {
    int mid, sum1, sum2;
    if(low == high)    return low;              //只剩一枚硬币
    if(low == high - 1)    return weight[low] < weight[high] ? low : high;   //剩余两枚硬币
    mid = (low + high) / 2;
    if((high - low + 1) % 2 == 0)//硬币数目为偶数，两边硬币数目相等，故左半计算[low, mid]
        sum1 = Sum(weight, low, mid);
    else//硬币数目为奇数，左半硬币数目多1，左半要少计算一个,故左半计算[low, mid-1]
        sum1 = Sum(weight, low, mid - 1);
    sum2 = Sum(weight, mid + 1, high);//计算右半重量
```

```
    if(sum1 == sum2)        //针对硬币数目为奇数的情况
        return mid;
    if(sum1 < sum2)         //假币在左半区间
    {
        if((high - low + 1) % 2 == 0)//硬币数目为偶数
            return Solve(weight, low, mid);
        else        //硬币数目为奇数
            return Solve(weight, low, mid - 1);
    }
    else            //假币在右半区间
        return Solve(weight, mid + 1, high);
}
```

```
//实验题第4题
int GetMode(int a[], int low, int high, int &max) {
    if(low <= high) {              //a[low..high]序列至少有1个元素
        int mid, count, left, right;
        int mode, modeLeft, countLeft, modeRight, countRight;
        mid = (low + high) / 2;
        //在a[low..high]中统计与a[mid]相等的元素数目，并确定这段区间[left,right]
        for(left = mid - 1; left >= low; left--)
            if(a[left] != a[mid])    break;
        left++;
        for(right = mid + 1; right <= high; right++)
            if(a[right] != a[mid])    break;
        right--;
        count = right - left + 1;              //求出a[mid]元素的重数
        countLeft = countRight = 0;
        modeLeft = GetMode(a, low, left - 1, countLeft);                    //寻找左序列的众数
        modeRight = GetMode(a, right + 1, high, countRight);                //寻找右序列的众数
        max = count; mode = a[mid];
        if(max < countLeft) {    max = countLeft;    mode = modeLeft;    }
        if(max < countRight) {    max = countRight;    mode = modeRight;    }
        return mode;
    }
}
```