第二次编程作业$Report$

罗华坤 软件02 2019011799

第一题

实现方法

首先采用归并排序的方式，将输入数据从小到大进行排序。再采用二分查找的方式查找到对应的端点，求出端点的长度便可。

一点思考

由于本题有元素互异的条件，直接使用边界的查找，最后再用一个$if\ else$就可得出正确答案。

本题也可采用查找时将下界减一，再用二分查找中寻找右侧边界的方式，求出长度再减1。

```
int binarySearch(int value,int l,int r){
    int mid = 0;
    while(l <= r){
        mid = (l+r) >> 1;
        if(arr[mid] > value) r = mid - 1;
        else l = mid + 1;
    }
    return --l;
}
```

也就是说，当题目改成元素可重复时，简单的 $if\ else$ 无法得出正确的答案。

只能采用第二种方式，或者采用下界求左侧边界，上界求右侧边界的方式得出答案。

时间复杂度

由于输入的个数为 n ，查找次数为 m ，归并排序时间复杂度为 $O(n\log n)$

二分查找 m 次，时间复杂度为 $O(m\log n)$，总时间复杂度为$O(n\log n+m\log n)$

第二题

实现方法

首先不难发现前序与后序左右孩子的位置关系，因此将后序遍历倒序输入。

树的结点保留数据、左右孩子指针，创建时输入数据，左右孩子默认为空。

采用递归的方式，先创建根节点，再划分前序与后序遍历序列，递归调用，再返回根节点的方式。

```
Tree* construct(int preS,int preE,int postS,int postE){//在前序、后序序列中的位置
    if(preS > preE) return nullptr;//若为空，返回空指针
    Tree* root = new Tree(pre[preS]);//创建节点
    int mid1 = findL(preS,preE,post[postS + 1]);//找左子树的位置；
    int mid2 = findR(postS,postE,pre[preS + 1]);//找右子树的位置
    if(mid1 != -1) root -> left = construct(preS+1,mid1-1,mid2,postE);//递归
    else root -> left = nullptr;
    if(mid2 != -1) root -> right = construct(mid1,preE,postS+1,mid2-1);//递归
    else root -> right = nullptr;
    return root;//返回该节点
}
```

查找采用顺序查找的方式，查找到返回位置，否则为1。

最后中序遍历同样可以采用递归的方式实现。

```
void preOrder(Tree* head){
    if(head == nullptr) return;
    preOrder(head -> left);
    printf("%d ",head -> val);
    preOrder(head -> right);
}
```

第三题

实现方法

首先，令 a 等于 $'A'$ 的个数，b 为 $'B'$ 的个数，c 为 $'C'$ 的个数

从左到右扫描，时刻保证 $a+c \geq b$；从右到左保证 $b+c \geq a$；

若两侧均能保证，则输出 $True$，否则输出 $False$

正确性

正确的情况：

对于序列中任意一个 c，其左侧一定满足$a+c \geq b$，对于其右侧有$b+c \geq a$。

则两侧多余的a，b之差必然小于等于1，否则对于全序列不可能是正确的，矛盾！

当为0时，c取空字符；当左侧a的数目更多时，取b；反之取a。

错误的情况：

不妨以从左到右扫描为例，反之同理。

即存在一点$a+c<b$，由于 b 对应的 a 必须在左侧，所以就算 c 全为 a 也无法使 b 全部对应。

时间复杂度

由于从左到右，从右到左扫描都只用线性时间，因此复杂度为$O(n)$