**22275:二叉搜索树的遍历**

def cs22275():  
 ans = []  
  
 def post(m, p):  
 try:  
 if m:  
 id = m.index(p[0])  
 ans.append(p[0])  
 post(m[id + 1:], p[1 + id:])  
 post(m[:id], p[1:1 + id])  
 except:  
 pass  
  
 post([i + 1 for i in range(int(input()))], list(map(int, input().split())))  
 ans.reverse()  
 for i in ans[:-1]:  
 print(i, end=" ")  
 print(ans[-1])



## 05455:二叉搜索树的层次遍历

def cs05455():  
 t = tuple(map(int, input().split()))  
 tree = {t[0]: [-1, -1]}  
 ans = {0: [t[0]]}  
 mx = 0  
 for i in t:  
 j = t[0]  
 ly = 0  
 while True:  
 flg = True  
 ly += 1  
 if i < j:  
 if tree[j][0] == -1:  
 tree[j][0] = i  
 tree[i] = [-1, -1]  
 break  
 else:  
 j = tree[j][0]  
 elif i > j:  
 if tree[j][1] == -1:  
 tree[j][1] = i  
 tree[i] = [-1, -1]  
 break  
 else:  
 j = tree[j][1]  
 elif i == j:  
 flg = False  
 break  
 if flg:  
 if ly > mx:  
 mx += 1  
 ans[ly] = []  
 ans[ly].append(i)  
 for i in range(mx + 1):  
 ans[i].sort()  
 for j in ans[i]:  
 if i != mx or j != ans[i][-1]:  
 print(j, end=" ")  
 else:  
 print(j)



## 04078:实现堆结构

def cs04078():  
 a = []  
 for i in range(int(input())):  
 t = input().split()  
 if t[0] == "1":  
 a.append(int(t[1]))  
 a.sort(reverse=True)  
 else:  
 print(a.pop())



## 22161:哈夫曼编码树

def cs22161(n):  
 tmp, ans = [], []  
 for i in range(n):  
 a, b = input().split()  
 lst = [[a], int(b), -1, -1, i]  
 tmp.append(lst)  
 ans.append(lst)  
 while len(ans[-1][0]) < n:  
 tmp.sort(key=lambda x: (x[1], x[0]), reverse=True)  
 a = tmp.pop()  
 b = tmp.pop()  
 st = a[0] + b[0]  
 st.sort()  
 fth = [st, a[1] + b[1], a[-1], b[-1], len(ans)]  
 tmp.append(fth)  
 ans.append(fth)  
 while True:  
 try:  
 ipt = input()  
 pr = ""  
 if ipt[0] in "01":  
 flag = -1  
 for i in ipt:  
 if i == "0":  
 flag = ans[flag][2]  
 else:  
 flag = ans[flag][3]  
 if ans[flag][2] == ans[flag][3] == -1:  
 pr += ans[flag][0][0]  
 flag = -1  
 else:  
 for i in ipt:  
 flag = -1  
 while True:  
 if i in ans[ans[flag][2]][0]:  
 pr += "0"  
 flag = ans[flag][2]  
 else:  
 pr += "1"  
 flag = ans[flag][3]  
 if ans[flag][2] == ans[flag][3] == -1:  
 break  
 print(pr)  
 except:  
 exit()



## 02524:宗教信仰

def cs02524():  
 case = 0  
 while True:  
 nm = input().split()  
 n, m = int(nm[0]), int(nm[1])  
 case += 1  
 if n == m == 0:  
 break  
 lst = [-1 for i in range(n)]  
 for i in range(m):  
 ab = input().split()  
 a = int(ab[0]) - 1  
 b = int(ab[1]) - 1  
 fa, fb = a, b  
 while lst[fa] + 1:  
 fa = lst[fa]  
 while lst[fb] + 1:  
 fb = lst[fb]  
 if fa != fb:  
 lst[fb] = fa  
 print("Case", str(case) + ":", lst.count(-1))



晴问9.5.平衡二叉树

def sw95(n):  
 ans = [""]  
  
 def right(ik):  
 fik = avl[ik][2]  
 ffik = avl[fik][2]  
 if ffik + 1:  
 if avl[ffik][0] == fik:  
 avl[ffik][0] = ik  
 else:  
 avl[ffik][1] = ik  
 else:  
 a[0] = ik  
 avl[ik][2] = ffik  
 avl[fik][0] = avl[ik][1]  
 if avl[ik][1] + 1:  
 avl[avl[ik][1]][2] = fik  
 avl[ik][1] = fik  
 avl[fik][2] = ik  
 avl[fik][3] = 1 + max(avl[avl[fik][0]][3], avl[avl[fik][1]][3])  
 avl[ik][3] = 1 + max(avl[avl[ik][0]][3], avl[avl[ik][1]][3])  
 renew(avl[fik][2])  
  
 def left(ik):  
 fik = avl[ik][2]  
 ffik = avl[fik][2]  
 if ffik + 1:  
 if avl[ffik][0] == fik:  
 avl[ffik][0] = ik  
 else:  
 avl[ffik][1] = ik  
 else:  
 a[0] = ik  
 avl[ik][2] = ffik  
 avl[fik][1] = avl[ik][0]  
 if avl[ik][0] + 1:  
 avl[avl[ik][0]][2] = fik  
 avl[ik][0] = fik  
 avl[fik][2] = ik  
 avl[fik][3] = 1 + max(avl[avl[fik][0]][3], avl[avl[fik][1]][3])  
 avl[ik][3] = 1 + max(avl[avl[ik][0]][3], avl[avl[ik][1]][3])  
 renew(avl[fik][2])  
  
 def renew(ik):  
 fik = avl[ik][2]  
 if fik + 1:  
 lft = avl[avl[fik][0]]  
 rt = avl[avl[fik][1]]  
 avl[fik][3] = 1 + max(lft[3], rt[3])  
 if lft[3] - rt[3] == 2:  
 if avl[avl[ik][0]][3] > avl[avl[ik][1]][3]:  
 right(ik)  
 else:  
 tik = avl[ik][1]  
 left(tik)  
 elif lft[3] - rt[3] == -2:  
 if avl[avl[ik][1]][3] > avl[avl[ik][0]][3]:  
 left(ik)  
 else:  
 tik = avl[ik][0]  
 right(tik)  
 else:  
 renew(fik)  
  
 ipt = list(map(int, input().split()))  
 avl = [[-1, -1, -1, 0] for i in range(n + 1)]  
 a = [0]  
 for i in range(1, n):  
 flag = a[0]  
 while True:  
 if ipt[i] < ipt[flag]:  
 if avl[flag][0] + 1:  
 flag = avl[flag][0]  
 else:  
 avl[flag][0] = i  
 avl[i][2] = flag  
 avl[i][3] = 1  
 renew(i)  
 break  
 else:  
 if avl[flag][1] + 1:  
 flag = avl[flag][1]  
 else:  
 avl[flag][1] = i  
 avl[i][2] = flag  
 avl[i][3] = 1  
 renew(i)  
 break  
  
 def out(flg=a[0]):  
 ans[0] += str(ipt[flg]) + " "  
 if avl[flg][0] + 1:  
 out(avl[flg][0])  
 if avl[flg][1] + 1:  
 out(avl[flg][1])  
  
 out()  
 print(ans[0][:-1])  
  
  
sw95(int(input()))



总结：难度大。前4题搞清楚概念能一遍过，第5题卡了，研究了并查集的路径优化，也过了，平衡二叉树真的花了很久才AC，指标和操作顺序是写了好几次才对，递归的时候函数互相调用还死循环了。。。继续加油吧