

数据结构作业第六章

庄震丰 22920182204393

Oct. 25th, 2019

6-33

题目要求：假定用两个一维数组作为有 n 个结点的二叉树的存储结构， $L[i]$ 和 $R[i]$ 分别指示结点 $i(i=1,2,\dots,n)$ 的左孩子和右孩子，0 表示空，试写出一个算法判断 u 是否为 v 的子孙。

算法分析：对于给定的 u 和 v ，从 v 开始向下递归搜索每个结点看是否为 u ，并用全局变量进行标记，如果 v 的子孙没有则会全部遍历，若找到则直接输出判断结果。

时间复杂度为 $O(N)$ ，空间复杂度 $O(N)$ 。

6-33.cpp

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  #define maxn 1000
4  int L[maxn+1]={0},R[maxn+1]={0}; //建立L,数组储存节点左右儿子R
5  bool flag=false;
6  void getfa(int x,int target) //从开始往下递归搜索儿子节点v
7  {
8      if (x==target) {flag=true;return;}
9      if (L[x]!=0) getfa(L[x],target);
10     if (R[x]!=0) getfa(R[x],target);
11 }
12 int main()
13 {
14     int n;
15     int v,u;
16     cin>>n;
17     for (int i=1;i<=n;i++)
18         cin>>L[i]>>R[i];
19     cin>>v>>u;
20     getfa(v,u);
21     cout<<flag;
22     return 0;
23 }
```

样例输入 1

```

12
2 3
6 7
8 9
10 11
12 0
0 0
0 0
0 0
0 0
0 0
```

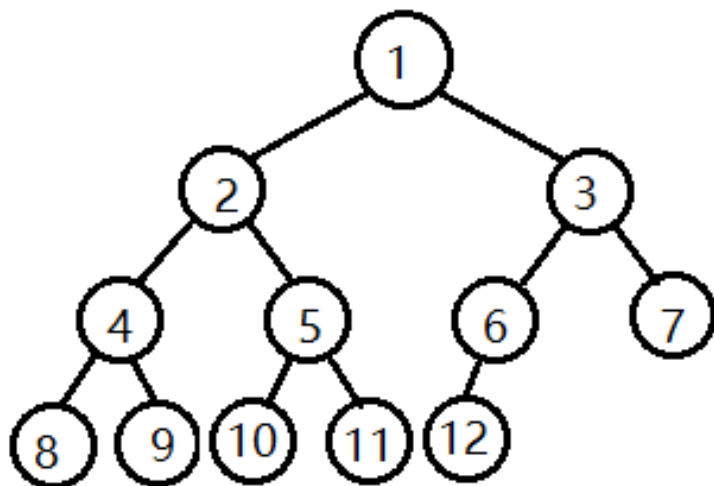
0 0

2 10

样例输出 1

1

说明



输入判断 2 是否为 10 的祖先，由图可知为真，输出 1。

6-62

题目要求：试编写算法，求一棵以孩子-兄弟表示的树，编写其深度的算法。

算法分析：输入一棵双亲表示的树，运用机构提结点将其转换成孩子-兄弟表示法，再将其进行深搜，每从左结点递归则层数加一，最后统计最大值即可。

时间复杂度 $O(n)$ ，空间复杂度 $O(n)$ 。

6-62.cpp

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  #define maxn 1000
4  #define inf 0x3f3f3f3f
5  struct node
6  {
7      char name;
8      node * lson;
9      node * bro;
10 };//孩子兄弟结构结点定义
11 struct parentnode
12 {
13     int fa;
14     char c;
15     node * point;
16 };//双亲数组结构结点定义
17 parentnode A[maxn];
18 int n;
```

```

19  int cnt=1;
20  node* trans()//将双亲数组结构转变成孩子兄弟树形结构
21  {
22      node * hp,*p1,*p2;
23      hp=(node*) malloc( sizeof( node));
24      hp->name=A[1].c;
25      hp->bro=NULL;
26      A[1].point=hp;//根节点单独判断
27      for (int i=2;i<=n;i++)
28          {
29              if (A[i].fa!=A[i-1].fa)
30                  {
31                      p1=(node*) malloc( sizeof( node));
32                      p1->bro=NULL;
33                      p1->lson=NULL;
34                      p1->name=A[i].c;
35                      A[i].point=p1;
36                      A[A[i].fa].point->lson=p1;
37                      //当前结点是新一层
38                  }
39              else
40                  {
41                      p2=(node *) malloc( sizeof( node));
42                      p2->bro=NULL;
43                      p2->lson=NULL;
44                      p2->name=A[i].c;
45                      A[i].point=p2;
46                      p1->bro=p2;
47                      p1=p2指针移动到兄弟结点;
48                      //当前结点层数不变
49                  }
50          }
51      return hp;
52  }
53  void finddeep(node * p,int deep)//按照孩子兄弟结构的指针进行遍历即可-
54  {
55      cout<<p->name<<" ";
56      if (p->bro==NULL&&p->lson==NULL)
57          {
58              cnt=max(cnt,deep);
59
60              return;
61          }
62      if (p->lson!=NULL) {finddeep(p->lson,deep+1);cout<<p->name<<" ";}
63      if (p->bro!=NULL) {finddeep(p->bro,deep);cout<<p->name<<" ";}
64      return;
65  }
66  bool cmp(parentnode a, parentnode b)
67  {
68      if (a.fa<b.fa) return true;
69      else return false;
70  }

```

```

71  int main()
72  {
73      node * Heap;
74      cin>>n;
75      A[0].fa=-inf;
76      for (int i=1;i<=n;i++)
77      {
78          scanf("%d %c",&A[i].fa,&A[i].c); //输入每个结点的父亲信息和该点的标号
79      }
80      sort(A+1,A+n+1,cmp);
81      Heap=trans(); //返回根节点指针
82      finddeep(Heap,1);
83      cout<<endl<<cnt;
84      return 0;
85  }

```

样例输入一

```

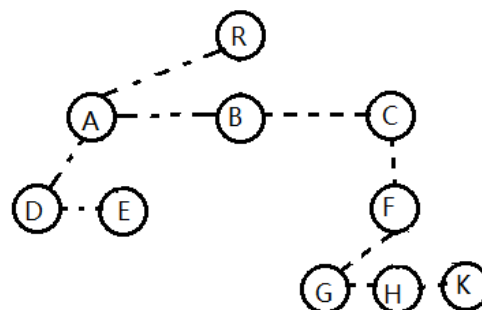
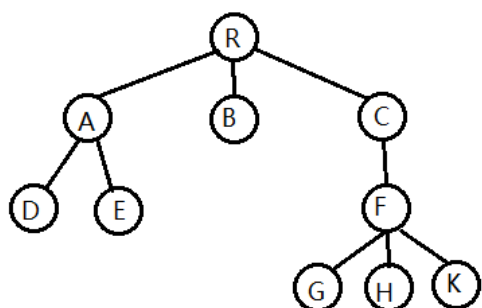
10
R -1
A 1
B 1
C 1
D 2
E 2
F 4
G 7
H 7
K 7

```

样例输出一

4

样例说明



样例描述的树和孩子-兄弟表示法如图所示，层数为 4。

6-65

题目描述：已知一棵二叉树的前序和中序序列分别储存在两个一维数组中，试编写算法建立改树的二叉链表。

算法分析：先求树的结构，将其储存在一个数组中，在通过这个数组将其转变为链表形式，最后用后序遍历验证输出。

时间复杂度 $O(n)$, 空间复杂度 $O(n)$ 。

6-65.cpp

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  #define maxn 1000
4  #define inf 0x3f3f3f3f//设置无穷大
5  struct node
6  {
7      char c;
8      node *lson;
9      node *rson;
10 };//二叉链表结构体数组
11 int n;
12 node * hp;
13 char Tree[maxn]={0};
14 void work(string ptree,string mtree,int cnt)//将二叉树两种遍历转变成数组存储
15 {
16     char root=ptree[0];
17     Tree[cnt]=root;
18     if (ptree.length()!=1&&mtree.length()!=1)
19     {
20         int pos=mtree.find(root);
21         work(ptree.substr(1,pos),mtree.substr(0,pos),cnt*2);
22         work(ptree.substr(ptree.length()-pos,pos),mtree.substr(mtree.length()-pos,pos),cnt
23             *2+1);
24     }
25     else
26     {
27         Tree[cnt]=root;
28     }
29 }
30 void trans(int x,node * last,int size)//将数组变成二叉链表存储
31 {
32     node *p1;
33     if (x==1)
34     {
35         hp=(node *) malloc(sizeof(node));
36         hp->lson=NULL;
37         hp->rson=NULL;
38         hp->c=Tree[x];
39         if (Tree[x*2]) trans(x*2, hp, 1);
40         if (Tree[x*2+1]) trans(x*2+1, hp, 2);
41     }
42     else
43     {
44         p1=(node *) malloc(sizeof(node));

```

```

45         p1->lson=NULL;
46         p1->rson=NULL;
47         p1->c=Tree[x];
48         if (Tree[x*2]) trans(x*2,p1,1);
49         if (Tree[x*2+1]) trans(x*2+1,p1,2);
50         if (size==1) last->lson=p1;
51         else last->rson=p1;
52     }
53 }
54 void printlast(node * p)//后序遍历输出
55 {
56
57     if (p->lson==NULL && p->rson==NULL) return;
58     if (p->lson!=NULL) printpre(p->lson);
59     if (p->rson!=NULL) printpre(p->rson);
60     cout<<p->c<<" ";
61     return;
62 }
63 int main()
64 {
65     cin>>n;
66     string pre,mid;
67     cin>>pre;
68     cin>>mid;
69     //初始化
70     work(pre,mid,1);
71     trans(1,NULL,0);
72     printlast(hp);
73     return 0;
74 }

```

样例输入 1

ABDECFG

DBEAFCG

样例输出 1

DEBFGCA

(标准的三层二叉树)