数据结构 A
作业 10 参考答案

作业情况

教材：数据结构教程（C++ 语言描述）李春葆等
题目范围：树
邮箱：wjyyy1@126.com
授课教师：彭蓉 教授
助教：王骏峣

习题 1 单选题

一颗完全二叉树上有 1001 个结点，其中叶子结点的个数是（　　）
A.250
B.501
C.254
D.505

答案：B

解析：注意完全二叉树指的是前 $h-1$ 层全满，第 h 层从左到右依次填满的二叉树。所以第 h 层的叶子节点个数为 2^{h-1}，第 $h-1$ 层的叶子节点个数为 2^{h-2}，以此类推。

那么 1001 是介于 $2^9(512)$ 和 $2^{10}(1024)$ 之间的，所以二叉树停止在第 10 层。前 9 层的节点个数为 $\sum_{i=1}^{9} 2^{i-1} = 511$，因此第 10 层叶子节点个数为 $1001 - 511 = 490$，那么第 9 层有 $490/2 = 245$ 个节点不是叶子，剩余 $256 - 245 = 11$ 个叶子节点。

因此共有 $490 + 11 = 501$ 个叶子节点。

（更详细的过程可参考第九次作业）

习题 2 单选题

如果将一棵有序树 T 转换为二叉树 B，那么 T 中结点的层次序列对应 B 的（　　）序列
A. 先序遍历
B. 中序遍历
C. 层次遍历
D. 以上都不对

答案：D

解析：参考教材 7.8.1。此题可以用排除法进行判断

二叉树无论先序遍历、中序、后序，都会分离左右两棵子树。而有序树是有层次的，左右子树可能是混合连接的，所以要先排除 A、B 两项。

当一个节点有很多孩子时，它的孩子会顺着当前的右子树**逐层加深**排布，因此也无法满足层次遍历。所以选 D。

习题 3 单选题

某二叉树的先序序列和后序序列正好相反，则该二叉树一定是（　　）

A. 空或者只有一个结点

B. 完全二叉树

C. 二叉排序树

D. 高度等于其结点数

答案：D

解析：先序序列的遍历顺序是:【当前、左子树、右子树】；后序序列的遍历顺序是:【左子树、右子树、当前】。

为了完全倒转过来，我们需要让【当前、左子树、右子树】变成【右子树倒序、左子树倒序、当前】，并和【左子树、右子树、当前】保持一致。

所以只要保证左右子树只有其中一棵，就可以了，此时除了叶子节点，所有点的度数都为 1，高度等于其结点数。

习题 4 单选题

设有 13 个值，用它们组成一棵哈夫曼树，则该哈夫曼树共有（　　）个结点

A.13

B.12

C.26

D.25

答案：D

解析：若有 13 个值，说明有 13 个叶子节点。每次合并两个叶子节点会**增加一个节点并减少一个联通块**。一开始是 13 个联通块，需要合并 12 次到 1 个联通块。

因此需要增加 $13-1=12$ 个节点，所以哈夫曼树共有 $13+12=25$ 个节点。

习题 5 单选题

设森林 F 对应的二叉树为 B，B 有 m 个结点，B 的根结点为 p，p 的右子树结点个数为 n，森林 F 中第一棵树的结点个数是（　　）

A. $m-n$

B. $m-n-1$

C. $n+1$

D. 条件不足，无法确定

答案：A

解析：参考教材 7.8.2，**转换后二叉树的根节点**是森林中第一棵树的根节点，因为在将森林从多棵二叉树拼接成一棵二叉树之前，所有的根结点会连成兄弟。

那么第一棵树的根结点就是整个二叉树的根结点，所以第一棵树的结点个数就是整个二叉树的结点个数减去右子树的结点个数，即 $m-n$。

习题 6 多选题

已知一棵二叉树的前序序列为 BACDEGHF，中序序列为 CADBHGEF，则其后序序列为（　　）

A. CAHGDFEB

B. CDABGHFE

C. CDAHGFEB

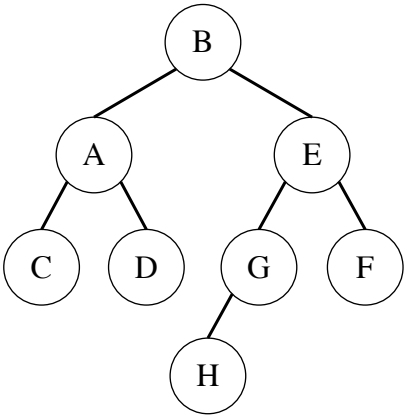
D. HGFEBCDA

答案：C

解析：前序遍历的顺序是：【当前、左子树、右子树】；中序遍历的顺序是：【左子树、当前、右子树】；后序遍历的顺序是：【左子树、右子树、当前】。

所以可以利用**前序遍历的第一个节点**确定根节点，再用这个节点在中序遍历中区分左右子树，具体过程如下。

- 前序：BACDEGHF，中序：CADBHGEF，推出根节点为 B
  - 左子树前序 ACD，右子树前序 EGHF
  - 左子树中序 CAD，右子树中序 HGEF
- 左子树根节点推出为 A，右子树根节点推出为 E
  - 左子树的左子树就是 C，右子树是 D
  - 右子树的左子树为 HG，右子树为 F
    - 容易看出 H 是 G 的左孩子，构造出如下的二叉树：



注：前序和后序是等效的，有前序中序可以推后序，有后序中序也可以推前序。但有前序后序时，推出的中序不唯一。

习题 7 畅通工程问题

【问题描述】

某省调查城镇交通状况，得到现有城镇道路统计表，表中列出了每条道路直接连通的城镇。省政府“畅通工程”的目标是使全省任何两个城镇间都可以实现交通（但不一定有直接的道路相连，只要互相间接通过道路可达即可）。问最少还需要建设多少条道路?

【输入形式】

每个测试用例的第 1 行给出两个正整数，分别是城镇数目 n（$n < 1000$）和道路数目 m，随后的 m 行对应 m 条道路，每行给出一对正整数，分别是该条道路直接连通的两个城镇的编号。为简单起见，城镇从 1 到 n 编号。注意两个城市之间可以有多条道路相通，也就是说:

```
3 3
1 2
1 2
2 1
```

这种输入也是合法的。

【输出形式】

对每个测试用例，在一行里输出最少还需要建设的道路数目。

【样例输入】

```
3 3
1 2
1 3
2 3
```

【样例输出】
0
【样例说明】
测试数据的文件名为 in.txt。
【评分标准】
该题目有 10 个测试用例，每通过一个测试用例，得 10 分。

答案：

```cpp
#include<iostream>
#include<fstream>
using namespace std;
const int MAXN=1005;
int parent[MAXN]; //定义并查集数组
int rnk[MAXN];  //存储并查集的秩，方便按秩合并降低复杂度
int n,m;
void Init()      //初始化并查集，所有人都是自己的根
{   for (int i=1;i<=n;i++)
    {   parent[i]=i;
        rnk[i]=0;
    }
}
int Find(int x)                     //并查集查找根节点
{   if (x!=parent[x])
        parent[x]=Find(parent[x]);  //查找时进行路径压缩
    return parent[x];
}
void Union(int x,int y)             //合并x和y所在的连通块
{   int rx=Find(x);
    int ry=Find(y);
    if (rx==ry)                     //如果两个在同一连通块就不做操作
        return;
    if (rnk[rx]<rnk[ry])
        parent[rx]=ry;              //rx秩更低，应该并给ry
    else
    {   if (rnk[rx]==rnk[ry])       //秩相等，合并后会加1
            rnk[rx]++;
        parent[ry]=rx;              //否则ry合并给rx
    }
}
int main()
{   ifstream txtfile;
    txtfile.open("in.txt");
    if(!txtfile){
        cout << "error open in.txt!" << endl;
```

```
        return 1;
    }
    txtfile>>n;
    txtfile>>m;
    Init();                         //并查集需要初始化
    for (int i=1;i<=m;i++)          //共有m条边
    {   int a,b;
        txtfile>>a;
        txtfile>>b;
        Union(a,b);
    }
    int ans=0;
    for (int i=1;i<=n;i++)          //连通块个数统计为ans
        if (parent[i]==i)           //一旦发现一个根节点
            ans++;                  //就说明有一个独立连通块
    cout<<ans-1;                    //需要添加的道路为ans-1条
    return 0;
}
```

解析：本题问还需要多少条道路能将所有城镇连接起来。若城镇目前不相连，说明整张图有超过一个连通块。易于理解的是，**每当两个连通块之间添加一条边时**，整张图的连通块个数就会减少 1。

因此本题要做的是把连通块个数统计出来，最后输出连通块个数 -1。

统计连通块个数有两种方法：一是使用并查集计算有多少节点是并查集树的根，这就是连通块个数；二是先把图建出来，计算需要多少次 BFS/DFS 才能将图遍历完，总共进入 BFS/DFS 的次数就是连通块个数。

上述代码里使用了**并查集**处理，但考虑到并查集可能是非必学内容（我不知道 !! 以老师通知为准），下给出图搜索判断连通块个数的方法：

（以 DFS 为例，仅作参考，无文件输入输出；如要改为 BFS，使用队列即可）

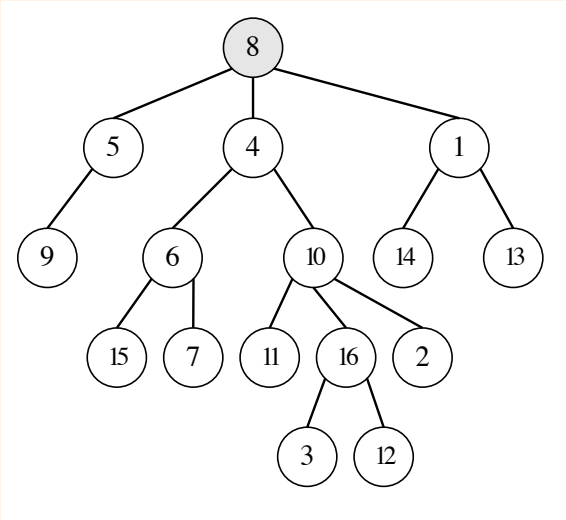
```
#include<iostream>
#include<vector>
using namespace std;
vector<int> v[1050];
int vis[1050];
void dfs(int x)
{
    for(int i=0;i<v[x].size();i++)
        if(!vis[v[x][i]])
        {
            vis[v[x][i]]=1;
            dfs(v[x][i]);
        }
}
int main()
```

```
{
    int n,m,u,w;
    cin>>n>>m;
    for(int i=1;i<=m;i++)
    {
        cin>>u>>w;//说明u,w相连
        v[u].push_back(w);
        v[w].push_back(u);//向邻接表添加边
    }
    int cnt=0;//统计连通块个数
    for(int i=1;i<=n;i++)
        if(!vis[i])
        {
            dfs(i);
            ++cnt;
        }
    cout<<cnt-1;
    return 0;
}
```

习题 8 求树中两个结点的最近公共祖先（LCA）

【问题描述】

如图所示是一棵有根树，图中每个结点用 1 ∼ 16 的整数标识，结点 8 是树根。如果结点 x 位于根结点到结点 y 之间的路径中，则结点 x 是结点 y 的祖先。如果结点 x 是结点 y 和结点 z 的祖先，则结点 x 称为两个不同结点 y 和 z 的公共祖先。如果 x 是 y 和 z 的共同祖先并且在所有共同祖先中最接近 y 和 z，则结点 x 被称为结点 y 和 z 的最近公共祖先，如果 y 是 z 的祖先，那么 y 和 z 的最近共同祖先是 y。例如结点 16 和 7 的最近公共祖先是结点 4，结点 2 和 3 的最近公共祖先是结点 10，结点 4 和 12 的最近公共祖先是结点 4。编写一个程序，找到树中两个不同结点的最近公共祖先。



【输入形式】

每个测试用例的第一行为树中结点数 n（$2 \leq n \leq 10000$），所有结点用整数 $1 \sim n$ 标识，接下来的 $n-1$ 行中的每一行包含一对表示边的整数，第一个整数是第二个整数的父结点。请注意，具有 n 个结点的树具有恰好 $n-1$ 个边。每个测试用例的最后一行为两个不同整数，需要计算它们的最近公共祖先。

【输出形式】

为每个测试用例输出一行，该行应包含最近公共祖先结点的编号。

【样例输入】

```
16
1 14
8 5
10 16
5 9
4 6
8 4
4 10
1 13
6 15
10 11
6 7
10 2
16 3
8 1
```

16 12

16 7

【样例输出】

4

【样例说明】

测试数据的文件名为 in.txt。

【评分标准】

该题目有 10 个测试用例，每通过一个测试用例，得 10 分。

答案：

```
#include<iostream>
#include<fstream>

using namespace std;
const int MAXN=10005;
int parent[MAXN];    //和并查集不同的是，parent只存储当前节点的直接父亲
int Level(int x)     //求取x的深度
{   int cnt=0;
    while(x!=-1)     //直到跳转到根节点才停下
    {   x=parent[x];//x跳转到其父亲
        cnt++;       //每跳一层，cnt+1
    }
    return cnt;
}
int solve(int x,int y)       //求x与y的最近公共祖先
{   int lx=Level(x);         //先判断x和y的深度
    int ly=Level(y);
    while (lx>ly)            //至少要跳到同级
    {   x=parent[x];
        lx--;
    }
    while (ly>lx)            //另一种深度相对情况
    {   y=parent[y];
        ly--;
    }
    while (x!=y)             //直到x和y遇到相等的位置就停下
    {   x=parent[x];         //这里就是他们的公共祖先（最近）
        y=parent[y];
    }
    return x;
}

int main()
```

```
{   int n,a,b,x,y;
    ifstream txtfile;
    txtfile.open("in.txt");
    if(!txtfile){
        cout << "error open in.txt!" << endl;
        return 1;
    }
    txtfile>>n;
    for (int i=0;i<=n;i++)          //初始化，没有被更新到的就是根节点
        parent[i]=-1;
    for (int i=1;i<n;i++)           //n个节点的树，共有n-1条边
        {   txtfile>>a;
            txtfile>>b;
            parent[b]=a;
        }
    txtfile>>x>>y;
    int ans=solve(x,y);
    cout<<ans;
}
```

解析：关于求取最近公共祖先，最简单的方法是找出两个点的所有祖先，**然后返回深度最大的一个**，但是这样的话显然复杂度太高，并且没有用到重复性信息。

当两个点深度不同时，它们的深度差距不会带来任何祖先相关的信息，因此需要先将深度调整到一致的水平，也就是让更深的（深度更大的）节点先往上调整，直到两个点深度一样。

此后，两个节点同时向上跳，也一定同时找到一个公共祖先，这时所找到的就是最近的那个公共祖先。因此使用一个 `while` 来控制两个节点的跳转，每次向上跳一层。一旦遇到相同的情况，就直接输出。

注意任意两个点一定有公共祖先，因为根节点是所有人的祖先（包括它自己）。

总结

题目：树 3

日期：2024 年 6 月 17 日

批改人：王骏峣

邮箱：wjyyy1@126.com

习题 1：考察完全二叉树的概念（注意与满二叉树进行区分）。

习题 2：考察层次遍历，这个概念较为冷门，建议多掌握这种不太常见但是可能考到的概念。

习题 3：考察先序遍历的特点，以及树的构建和遍历。如果一边构建一边遍历不是很熟练的话，可以先构建再遍历。

习题 4：考察哈夫曼树的概念、原理和构建。

习题 5：对于森林、多叉树、二叉树的互转，需要知道步骤（尤其是思路，代码可以暂缓理解）。注意兄弟之间的顺序和优先级区分。

习题 6：注意掌握前序、中序、后序的原理与特点，会利用其性质解决问题。

习题 7：判断连通块个数的多种方法可以尽量掌握，但主要是需要看出这道题需要大家处理连通块并统计个数。相信代码部分并不是特别有难度，但是还是需要多注意细节。

习题 8：最近公共祖先的朴素算法，注意向上跳的细节，要先调整深度到一致。

各位同学如有问题欢迎及时在群里提出，或者通过邮件/QQ 联系我。