中山大学数据科学与计算机学院本科生实验报告

（2016 学年春季学期）

课程名称：Algorithm design　　任课教师：张子臻

年级	15	专业（方向）	软件工程(移动信息工程)
学号	15352408	姓名	张镓伟
电话	13531810182	Email	709075442@qq.com
开始日期		完成日期	

目录

1.实验题目 2

Soj 1140 国王的遗产 2

Soj 1034 Forest 3

Soj 1219 新红黑树 3

Soj 1310 Right-Heavy Tree 3

Soj 1083 Networking 3

Soj 1426 Phone List 3

Soj 7766 Dark roads 3

Soj 2002 Feeding Time 4

Soj 1170 countdown 4

Soj 1490 Tree Grafting 4

Soj 14181 Flying Safely 4

2.实验目的 4

3.程序设计 4

Soj 1140 国王的遗产 4

Soj 1034 Forest 7
Soj 1219 新红黑树 8
Soj 1310 Right-Heavy Tree 9
Soj 1083 Networking 11
Soj 1426 Phone List 12
Soj 7766 Dark roads 14
Soj 2002 Feeding Time 15
Soj 1170 countdown 16
Soj 1490 Tree Grafting 17
Soj 14181 Flying Safely 19
5.实验总结与心得 20
附录、提交文件清单 20

1.实验题目

图论专题

Soj 1140 国王的遗产

题意：给出一棵有 n 个点的树(点是金块)。K 个人按照年龄大小的顺序去瓜分。对于某个人，他可以在现有金块链中剪掉某个链条，获得不超过当前金块总数一般的那部分。一个人只能拿一次，拿了之后让下一个拿，最后一个人拿完剩下的金块。人们在剪的时候总是选择使自己获得的“金块组编号”最小的那一条链来剪。“金块组编号”大小定义为：对于长度相同为 L 的两个有序数组 A 和 B，A<B 当且仅当存在一个整数 i(0<i≤L)，使得 A[1]=B[1],…,A[i-1]=B[i-1]且 A[i]<B[i]。求在每个人获得尽量多的金块条件下，每个人获得几个金块，按照拿的顺序输出。
约束：1<=n<=30000　1<=k<=100

Soj 1034 Forest

题意：给出一个 n 个点，m 条有向边的森林，求森林的深度和宽度。深度定义为组成森林的树中，有最多层节点的那棵树的节点层数。宽度为森林最多点的那一层的点数，注意，森林某一层的点数是指组成森林的所有数的该层点数之和。如果给出的图不是森林(有环或者某个点有两条及以上的入边)，那么输出 INVALID。
约束： (1≤n≤100, 0≤m≤100, m≤n*n)

Soj 1219 新红黑树

题意：A 君和 B 君在玩一种叫做新红黑树的游戏，即在一棵由红枝和黑枝构成的树上轮流砍树枝，每次砍一枝，A 君每次只能砍红枝，B 君每次只能砍黑枝，当其中某人已经没有树枝砍的时候，由另外一人砍，直到砍完全部树枝。树枝是带权的，每个人的总分是他砍的树枝的权值之和，那些由于其他树枝被砍掉而与根失去联系的树枝会自动消失。每次由 A 君先砍，设“D=A 君的得分－B 君的得分”，A 君想让 D 最大，而 B 君想让 D 最小，A 君和 B 君都是极其聪明的人，他们始终以最优策略进行整个游戏，求最后的 D 值。
约束： n(1<=n<=20)条树枝,每条树枝 4 个值 a、b、c、w，分别表示树枝的两个端点，颜色和权值，端点编号 0-n，0 为根节点，c=1 表示红色，c=-1 表示黑色，1<=w<=1000

Soj 1310 Right-Heavy Tree

题意：Right-Heavy Tree 是这样一种树：它的每个结点的权值>=左子树所有点的权值且<右子树所有点的权值，它可以为空。给定 n 个数，建立一棵右重树，输出中序，前序，后序遍历。
约束： n<=200000

Soj 1083 Networking

题意：给出一个包含 n 个点，m 条边的无向连通图。用最少的边将这 n 个点连通，且边权和最小，求这个最小边权和。
约束： n<=50,m 未给出

Soj 1426 Phone List

题意：有 n 个电话号码，每个号码长度不超过 10 位。如果存在一个号码是另一个号码的前缀，那么输出 NO，否则输出 YES。
约束：n<=10000

Soj 7766 Dark roads

题意：给定一个含 m 个点，n 条路的图。每条路有一个长度 z，建造道

路每米要花 1 元。政府决定为了省钱，不建完所有道路。问要使 m 个点全部连通，与建完所有道路相比最多能省多少钱？
约束：1<=m<=200000，m-1<=n<=200000，z 的总和小于 2^31，点的编号从 0 到 m-1

Soj 2002 Feeding Time

题意：给出一个 W*H 的地图，‘.’表示草，‘*’表示石头。一头牛想吃草，可以从任意一个地方开始吃，然后它只能横着走，竖着走和斜线走到相邻格子。牛不能走到有石头的地方。求它最多能吃多少格的草。
约束：1 <= W <= 750; 1 <= H <= 750

Soj 1170 countdown

题意：给出 n 个人以及每个人的子女建一个家庭树，找出有第 d 代子孙的名字(如 d=1，则找有最多孩子的，d=2 则找有最多孙子的，以此类推)将符合条件的人按照其所拥有的第 d 代子孙数目从大到小输出三个人名，如果有一样大小子孙数的，就按名字字母序从小到大输出直到和第三个人的该代子孙数不一样为止，如果小于三个人就全输出。
约束：1<=n<=1000

Soj 1490 Tree Grafting

题意：给出一颗树的遍历操作，d 表示向下一层，u 表示向上一层。求这棵树的高度以及将其转成二叉树后的高度。注意高度从 0 开始算。
约束：树的结点数至少为 2 且不超过一千。

Soj 14181 Flying Safely

题意：有一个 n 个城市，m 条航班的连通图。一个 spy 要坐航班，但是每个飞机他需要寻求帮助使安保相信他是好人。求最少的要让安保相信他是好人的飞机数，使 spy 可以在任意两个城市间来回。

2.实验目的

1.加深对图论各种算法的理解，学会如何应用它们。
2.提高运用课堂所学知识解决实际问题的能力。

3.程序设计

Soj 1140 国王的遗产

解题思路:首先，每个人都是贪婪的，一定会拿走尽量多的金块,也就是一

定会取中当前金块中刚好不超过一半的那部分。而如果金块刚好能分成数量相等的两堆，由于要“金块组编号”最小，即取编号最小那个金块所在那堆(因为每个金块的编号不同，所以两个金块组编号的比较其实就是各自最小编号的比较)。综上，该题要实现的过程实际上如下：

1. 对于每个人，在剩余金块组成的树中，寻找一条边切断，使得分得的两堆金块，数量较少那堆金块的数量尽量多，设这个最大值为 ans。如果存在两种切法，使得 ans 值一样，那么选取最小编号最小的那种切法。
2. 切完后，将数量较少那堆金块从图中删去。然后下一个人重复上述操作。

基于上述过程，我们可以给树中的每个结点两个属性 sizes 和 minID，其中 sizes 表示以当前结点为根的子树一共有多少个结点；minID 表示以当前结点为根的子树中最小的编号是多少。对于每一个人，我们首先将剩下的点递归建树，在建树过程中求出 sizes 和 minID。然后再深搜枚举切哪一条边假设当前搜到节点 x，要切它的某条边，这条边的另一端是 y，那么有两种情况：

1. x 这边这部分的结点数小于当前结点的一半，则判断其是否比已记录的最优值更优。所谓更优，就是结点数更多，或者结点数一样但最小编号更小
2. y 这边这部分的结点数小于当前结点的一半，同 1 的操作。

Ok，y 这边这部分的 sizes 和 minID 可以直接从 y 结点的这两个属性得出，那么 x 那边又该如何？x 这边的 sizes 可以用剩余的总结点数减去以 y 为根的子树的 sizes，但是 minID 似乎比较难办。有一个比较巧妙的方法就是，我们每次建树，都以剩余的金块中编号最小那块为根，这样子 x 这边的 minID 就是根的编号。至此，所有问题解决。

时间复杂度 O(nk)。

代码:

```
#include<iostream>
#include<vector>
#include<cstdio>
using namespace std;
#define N 30010
#define rep(i,u,v) for (int i=(u);i<=(v);i++)
struct node{
    vector<int>child;
    int minID,sizes;
}t[N];
int n,k,tot,ans,root,st,cnt=0,start_from_root,minID;
int bo[N];
vector<int>g[N];
void build_tree(int x){
    bo[x]=-1;
    t[x].child.clear();
    t[x].sizes=1;t[x].minID=x;
    int len=g[x].size();
    rep(i,0,len-1){
        int y=g[x][i];
        if (!bo[y]){
            build_tree(y);
            t[x].child.push_back(y);
            t[x].sizes+=t[y].sizes;
            t[x].minID=min(t[x].minID,t[y].minID);
        }
    }
    bo[x]=0;
}
```

```
void dfs(int x){
    //深搜寻找取哪一部分，start_from_root=1表示取切掉的边的靠近
    //根节点那部分start_from_root=0表示取切掉的边远离根结点那边
    //st记录的是被切那条边远离根结点的那个结点
    int len=t[x].child.size();
    rep(i,0,len-1){
        int y=t[x].child[i];
        int res=tot-t[y].sizes;
        if (t[y].sizes>=res){//取靠近根结点那部分
            if (res<ans)continue;
            else if (res>ans||res==ans&&t[y].minID>t[st].minID){
                ans=res;
                start_from_root=1;
                minID=root;
                st=y;
            }
        }
        else {//取远离根结点那部分
            if (t[y].sizes<ans)continue;
            else if (t[y].sizes>ans||(!start_from_root&&t[y].minID<minID)){
                ans=t[y].sizes;
                start_from_root=0;
                minID=t[y].minID;
                st=y;
            }
        }
        dfs(y);
    }
}
void del(int x, int y){
  int len=t[x].child.size();
  rep(i,0,len-1){
    int y=t[x].child[i];
    del(y,y);
  }
  bo[x]=y;
}
int main(){
    int x,y;
    scanf("%d%d",&n,&k);
    rep(i,1,n-1){
        scanf("%d%d",&x,&y);
        g[x].push_back(y);
        g[y].push_back(x);
    }
    rep(i,1,k-1){
        root=1;
        //用还没被拿走的编号最小的金块为根节点建树
        while (bo[root] && root<n)root++;
        build_tree(root);
        ////////////////////////////////

        //////////////////深搜寻找切哪条边
        tot=t[root].sizes;
        ans=0;
        dfs(root);
        ////////////////////

        //**删去被拿走的金块
        if (start_from_root)del(root,1),del(st,0);//删除靠近根那部分
        else del(st,1);//删除远离根那部分
        //***************
        printf("%d ", ans);
        cnt+=ans;
    }
    printf("%d\n",n-cnt);
    return 0;
}
```

Soj 1034 Forest

解题思路:要求深度和宽度，显然是要遍历一遍。但在遍历之前，我们首先要保证森林是“合法”的，即要保证没有回路，且一个点没有两条及以上的入边。这个判断可以放在遍历过程中进行，但我选择分离。我的判断方法如下：

1. 将所有点插入到一个名为 root 的 set 中。
2. 读入所有边，由于是有向边，我将此边的终点从 set 中删除。
3. 若删除过程中发现 set 中并无此点，则说明此前已经删除过一次，也就是说出现了一个点有两条入边的情况。
4. 读边的时候还要判断一下是否出现了自环，即该边是自己到自己。这一步容易遗漏，我因为这个 WA 了几次。
5. 所有边读入完毕后，判断一下 root 这个集合是否为空，若为空，说明每个点都有一条入边，组成了一个大环。

判断完成后，root 中剩下的点就是每棵树的根结点。从 root 中的每个点分别开始进行深度搜索，记录下每棵树的最大深度，取最大值为答案。同时用一个数组 widths 记录森林每一层的宽度，遍历到某个点时，假如它是第 dep 层，那么 widths[dep]++。最后取 widths 中的最大值作为答案。
时间复杂度为 O(nlogn+m)

代码:

```
#include<iostream>
#include<vector>
#include<cstring>
#include<set>
using namespace std;
#define rep(i,u,v) for (int i=(u);i<=(v);i++)
int n,m;
int widths[110];
vector<int> g[110];
set<int> root;
void Initialize(){//初始化
    root.clear();
    memset(widths,0,sizeof(widths));
    rep(i,1,n){
        g[i].clear();
        root.insert(i);
    }
}
void dfs(int x,int dep,int &ansdep,int &answidths){
    widths[dep]++;
    ansdep=dep>ansdep?dep:ansdep;
    answidths=answidths>widths[dep]?answidths:widths[dep];
    int len=g[x].size();
    rep(i,0,len-1)
            dfs(g[x][i],dep+1,ansdep,answidths);
}
```

```
int work(){
    int x,y;
    int flag=1;
    rep(i,1,m){
            cin>>x>>y;
            g[x].push_back(y);
            if (root.find(y)==root.end()||x==y)flag=0;//有两条入边或自环
            else root.erase(y);
        }
    if (!flag)return 0;
    set<int>::iterator it;
    if (root.empty())return 0;//整体为一个大环。
    int ansdep=0,answidths=0;
    for (it=root.begin();it!=root.end();it++)//遍历每棵树求解
        dfs(*it,0,ansdep,answidths);
    cout<<ansdep<<" "<<answidths<<endl;
    return 1;
}
int main(){
    while (cin>>n>>m,n>0){
        Initialize();
        if (!work())cout<<"INVALID"<<endl;
    }
    return 0;
}
```

Soj 1219 新红黑树

分析和解法:这题咋看之下与 1140 相像，但是不一样。按照题意，这是一题博弈题。但凡博弈题，都可以写成一棵博弈树的形式。根节点为初始状态，执行一步操作之后得到一系列子状态，从这些子状态寻找一个最优值去更新根状态的值，而每个子状态的值也用上述方法得出。对于这题同样如此。这里的最优值毫无疑问是 D 的最大值。那么如何表示一个局面的状态呢。我们发现 N 很小，只有 20，一般来说这么小的数据就是告诉我们要用二进制状态去表示了。于是我们可以用 N 位二进制状态去表示 N 条边的状况。我们以 0 为根节点建树，那么边的编号我们规定为后继点的编号。比如样例中第一条边的后继点是 1，那么他的编号就是 1。N 位二进制状态 S 表示树的情况为 1，如果 S 的第 i 位为 1，那么证明第 i+1 条边没被砍了(二进制数的位编号从 0 开始，而我们的边编号从 1 开始)，为 0 则被砍。设 f[s][turn]表示当前状态为 s，轮到 turn 去砍(turn=1 为 A 去砍，turn=0 为 B 去砍)。采用记忆化搜索，在搜索的每一层，枚举砍哪一条边，维护当前状态的最优值即可(A 维护 f[s][turn]的最大值，B 维护最小值。)为了方便起见，我们可以预处理出砍了某条边后，一共有哪几条边消失，方便状态转移。

代码:

```
#include<iostream>
#include<cstdio>
#include<algorithm>
#include<cstring>
using namespace std;
#define INF 0x3f3f3f3f
#define rep(i,u,v) for (int i=(u);i<=(v);i++)
int n;
int f[1<<20][2];
int c[22][22],w[22][22],pre[22],bo[22],sta[22];
```

```
void dfs(int x){//预处理砍掉某条边后会消失哪条边
    bo[x]=1;
    if (x>0)sta[x]|=(1<<(x-1));
    rep(i,1,n){
        if(!bo[i]&&w[x][i]!=0){
            dfs(i);
            sta[x]|=sta[i];
            pre[i]=x;
        }
    }
}
int dp(int s,int turn){
    if (!s)return 0;
    int k=(turn+1)/2;
    if (f[s][k]!=INF)return f[s][k];//搜过的状态不再搜
    int maxx=-INF,minn=INF;
    rep(i,1,n){
        if (s&(1<<(i-1))){
            if (turn!=c[pre[i]][i])
                continue;
            int temp=dp(s&~sta[i],-turn);
            maxx=max(maxx,w[pre[i]][i]+temp);//A砍
            minn=min(minn,w[pre[i]][i]+temp);//B砍
        }
    }
    if(turn==1&&maxx==-INF)maxx=dp(s,-turn);//A没得砍
    if(turn==-1&&minn==INF)minn=dp(s,-turn);//B没得砍
    if (turn==1)f[s][k]=maxx;
    else f[s][k]=minn;
    return f[s][k];
}
```

```
int main(){
    int x,y,color,weight;
    cin>>n;
    rep(i,1,n){
        cin>>x>>y>>color>>weight;
        c[x][y]=c[y][x]=color;
        w[x][y]=w[y][x]=weight*color;
    }
    pre[0]=-1;
    dfs(0);
    memset(f,INF,sizeof(f));
    cout<<dp((1<<n)-1,1)<<endl;
    return 0;
}
```

Soj 1310 Right-Heavy Tree

解题思路:按照题目要求建树，然后遍历求中序，前序，后序数列即可。要注意本题不是 n=0 时结束。

建树过程：

1. 先读入第一个数字作为数的根结点。
2. 此后读入每个数字，从根结点开始比较，设树中与当前数字 x 作比较的是结点是 t[now]。则若 t[now]结点储存的数字 t[now].x>=x，说明

x 应该在 t[now].x 的左子树中，令 now=t[now].lc(即去 t[now]的左孩子节点继续比较),若发现 t[now]不存在左孩子,则将 x 变成 t[now]的左孩子。若 t[now]结点储存的数字 t[now].x<x，说明 x 应该在 t[now].x 的右子树中，令 now=t[now].rc(即去 t[now]的右孩子节点继续比较)，若发现 t[now]不存在右孩子，则将 x 变成 t[now]的右孩子。

3. 为了方便判断 t[now]是否存在左右孩子，将 t[now].lc 和 t[now].rc 设为-1 来表示其不存在左右孩子。

建好树之后，分别进行中序，前序，后序遍历输出即可。过程分别如下：

1. 前根序遍历：先遍历根结点，输出当前点的值,然后遍历左子树，最后遍历右子树。
2. 中根序遍历：先遍历左子树，然后遍历根结点，输出当前点的值，最后遍历右子树。
3. 后根序遍历：先遍历左子树，然后遍历右子树，最后遍历根节点，输出当前点的值。

时间复杂度 O(nlogn)

代码:

```
#include<iostream>
#include<cstdio>
using namespace std;
#define rep(i,u,v) for (int i=(u);i<=(v);i++)
#define N 200010
struct node{
    int x,lc,rc;
}t[N];
int n,m;
void preorder(int now){
    if (now==-1)return;
    printf(" %d",t[now].x);
    preorder(t[now].lc);
    preorder(t[now].rc);
}
void inorder(int now){
    if (now==-1)return;
    inorder(t[now].lc);
    printf(" %d",t[now].x);
    inorder(t[now].rc);
}
void postorder(int now){
    if (now==-1)return;
    postorder(t[now].lc);
    postorder(t[now].rc);
    printf(" %d",t[now].x);
}
```

```
int main(){
    int x,n,T=0;
    while (scanf("%d",&n)!=EOF){
        if (++T>1)printf("\n");
        if (!n){
            printf("Inorder:\n");
            printf("Preorder:\n");
            printf("Postorder:\n");
            continue;
        }
        scanf("%d",&t[0].x);
        t[0].lc=t[0].rc=-1;
        rep(i,2,n){
            scanf("%d",&t[i].x);
            t[i].lc=t[i].rc=-1;
            int now=0;
            while (1){
                if (t[now].x>=t[i].x){
                    if(t[now].lc==-1){
                        t[now].lc=i;
                        break;
                    }
                    now=t[now].lc;
                }
                else {
                    if(t[now].rc==-1){
                        t[now].rc=i;
                        break;
                    }
                    now=t[now].rc;
                }
            }
        }
        printf("Inorder:");inorder(0);printf("\n");
        printf("Preorder:");preorder(0);printf("\n");
        printf("Postorder:");postorder(0);printf("\n");
    }
    return 0;
}
```

Soj 1083 Networking

分析和解法:用最少的边将 n 个点连通，且边权和最小，这就是很直白地让我们求最小生成树。这里我选择使用 Kruskal 算法。先将所有边按边权从小到大排序。然后从头访问每一条边，用并查集判断这条边的两个端点是否在同一集合中。若是，代表这两个点已经连通，那么这条边放弃，否则，将两个点所在集合合并，答案中加入这条边的边权。这题要注意的是，题目没有给出边数，我们可以使用 stl 中的 vector 动态存储边。
时间复杂度 O(mlogm)

代码:

```cpp
using namespace std;
#define rep(i,u,v) for (int i=(u);i<=(v);i++)
struct edge{
    int x,y,z;
    edge(int xx,int yy,int zz){
        x=xx;y=yy;z=zz;
    }
};
int f[55];
int findfa(int x){
    if (f[x]!=x)f[x]=findfa(f[x]);
    return f[x];
}
bool cmp(const edge &n1,const edge &n2){
    return n1.z<n2.z;
}
int main(){
    int n,m,x,y,z,ans;
    while (cin>>n&&n){
        cin>>m;
        ans=0;
        vector<edge>e;
        rep(i,1,m){
            cin>>x>>y>>z;
            e.push_back(edge(x,y,z));
        }
        rep(i,1,n)f[i]=i;
        sort(e.begin(),e.end(),cmp);
        rep(i,0,m-1){
            int x=findfa(e[i].x),y=findfa(e[i].y);
            if (x==y)continue;
            f[x]=y;
            ans+=e[i].z;
        }
        cout<<ans<<endl;
    }
    return 0;
```

Soj 1426 Phone List

分析和解法:求一个字符串是否是另一个字符串的前缀，是 Trie 字母树的经典应用。Trie 树有如下性质：

（1）根节点不包含字符，除根节点意外每个节点只包含一个字符。

（2）从根节点到某一个节点，路径上经过的字符连接起来，为该节点对应的字符串。

（3）每个节点的所有子节点包含的字符串不相同。

Trie 树的基本实现：字母树的插入（Insert）、删除（ Delete）和查找（Find）都非常简单，用一个一重循环即可，即第 i 次循环找到前 i 个字母所对应的子树，然后进行相应的操作。实现这棵字母树，我们用最常见的数组保

存（静态开辟内存）即可，当然也可以开动态的指针类型（动态开辟内存）。
至于结点对儿子的指向，一般有三种方法：

1、对每个结点开一个字母集大小的数组，对应的下标是儿子所表示的字母，内容则是这个儿子对应在大数组上的位置，即标号；
2、对每个结点挂一个链表，按一定顺序记录每个儿子是谁；
3、使用左儿子右兄弟表示法记录这棵树。

三种方法，各有特点。第一种易实现，但实际的空间要求较大；第二种，较易实现，空间要求相对较小，但比较费时；第三种，空间要求最小，但相对费时且不易写。在本题中，我选择使用第一种实现方法。
知道 Trie 树如何构建之后，这道题就迎刃而解了。我们将每个字符串插入到 Trie 树中，每个结点使用一个变量 flag 记录这个结点的两种状态：

1. flag=1，某个字符串经过了这个点。
2. flag=2，这个点代表的字符是某个字符串的最后一个字符。

当我们插入一个新字符串时，遇到下面两种情况就说明出现了某个字符串是另一个字符串的前缀：

1. 遇到一个 flag=2 的结点，说明前面插入的某一个字符串是正在进行插入的这个字符串的前缀。
2. 遇到一个 flag=1 的结点且当前插入的字符串插入到了最后一个字符，说明当前字符串是某个字符串的前缀。

如果插入完所有字符串都没有出现上述情况就说明这些电话是不相容的，否则就是相容的。
时间复杂度 O(n*len)，len 为字符串长度。
代码：

```
#include<iostream>
#include<cstring>
using namespace std;
#define rep(i,u,v) for (int i=(u);i<=(v);i++)
#define N
struct node{
    int flag;
    int ch[10];
    void init(){
        memset(ch,0,sizeof(ch));
        flag=0;
    }
}t[101000];
int flag,m;
void insert(char *s){
    int p=0;
    int len=strlen(s);
    for (int i=0;i<len;i++){
        int x=s[i]-'0';
        if (!t[p].ch[x]){
            t[p].ch[x]=++m;
            t[t[p].ch[x]].init();
        }
        p=t[p].ch[x];
        if (t[p].flag==2){//当前点是某个字符串的终点
            flag=0;
            return;
        }
        else if(t[p].flag==1&&i==len-1){
        //当前点是当前字符串的终点，且同时被另一个字符串经过
            flag=0;
            return;
        }
        t[p].flag=1;
    }
    t[p].flag=2;
}
```

```
int main(){
    int T,n;
    char s[15];
    cin>>T;
    rep(tt,1,T){
        cin>>n;
        flag=1;m=0;
        t[0].init();
        rep(i,1,n){
            cin>>s;
            insert(s);
        }
        if (flag)cout<<"YES"<<endl;
        else cout<<"NO"<<endl;
    }
    return 0;
}
```

Soj 7766 Dark roads

分析和解法:这题要求最多能省多少钱，即求不用修的路的建造费之和。反过来说，只要我们知道建路的最小花费，用建完所有道路的花费减去这个最小花费不就得到了最多能省的钱数了么。要求建路的最小花费，按照题意，无非就是建的路越少越好，建的每条路的花费越少越好，同时建完这些路之后能使 n 个点成为一个连通图。将 n 个点连成一个连通图最少需要 n-1 条边，同时要使这 n-1 条边的边权和最小，这就是要求这幅图的最小生成树。求出最小生成树之后，用建完所有道路的花费减去最小生成树的边权和就得到了答案。
时间复杂度 O(nlogn)
代码:

```
#include<iostream>
#include<algorithm>
#include<cstdio>
#include<vector>
using namespace std;
#define rep(i,u,v) for (int i=(u);i<=(v);i++)
struct edge{
    int x,y,z;
    edge(int xx,int yy,int zz){
        x=xx;y=yy;z=zz;
    }
};
int f[200005];
vector<edge>e;
int findfa(int x){
    if (f[x]!=x)f[x]=findfa(f[x]);
    return f[x];
}
bool cmp(const edge &n1,const edge &n2){
    return n1.z<n2.z;
}
```

```cpp
int main(){
    int n,m,x,y,z,ans;
    while (scanf("%d%d",&m,&n)!=EOF){
        if (!m&&!n)break;
        ans=0;e.clear();
        rep(i,1,n){
            scanf("%d%d%d",&x,&y,&z);
            e.push_back(edge(x,y,z));
            ans+=z;
        }
        rep(i,0,m)f[i]=i;
        sort(e.begin(),e.end(),cmp);
        rep(i,0,n-1){
            int x=findfa(e[i].x),y=findfa(e[i].y);
            if (x==y)continue;
            f[x]=y;
            ans-=e[i].z;
        }
        cout<<ans<<endl;
    }
    return 0;
}
```

Soj 2002 Feeding Time

分析和解法:由题目知牛不能走到有石头的地方，所以相当于用石头把整个农场分割成了若干个都是草区域，小牛要选择一个面积最大的区域。所以问题就转变为求每一个区域的面积，取最大值。我们使用一个二维数组 bo 记录 bo[i][j]这个地方的状态。假设(i，j)这个地方已经统计过或者这个地方是石头，则 bo[i][j]=1，否则 bo[i][j]=0；所以我们从头遍历区域中的每一个格子，若遇到 bo[i][j]=0，则使用洪水算法 floodfill 统计(i，j)这个点所在区域的面积，然后将这片区域的 bo 值设为 1。

上面提到的洪水算法其实就是从一点开始向四周寻找未遍历过且能够遍历的点去遍历，可以使用深度搜索或者广度搜索实现。在本题中，由于地图面积较大，使用深度搜索为造成栈空间溢出，所以只能使用广度搜索实现。

时间复杂度 O(W*H)

代码:

```cpp
#include<iostream>
#include<cstdio>
#include<cstring>
using namespace std;
#define rep(i,u,v) for (int i=(u);i<=(v);i++)
struct node{
    int x,y;
    void add(int xx,int yy){
        x=xx;y=yy;
    }
}list[752*752];
const int dx[]={0,-1,-1,-1,0,1,1,1};
const int dy[]={-1,-1,0,1,1,1,0,-1};
int W,H,ans=0,sum;
char s[753][753];
int bo[753][753];
```

```
void floodfill(int stx,int sty){
    int l=1,r=1;
    list[1].add(stx,sty);
    bo[stx][sty]=1;
    while (l<=r){
        int x=list[l].x,y=list[l].y;
        l++;sum++;
        rep(i,0,7){
            int xx=x+dx[i];
            int yy=y+dy[i];
            if (xx<=0||xx>H)continue;
            if (yy<=0||yy>W)continue;
            if (bo[xx][yy]||s[xx][yy]=='*')continue;
            list[++r].add(xx,yy);
            bo[xx][yy]=1;
        }
    }
}
int main(){
    cin>>W>>H;
    rep(i,1,H)scanf("%s",s[i]+1);
    memset(bo,0,sizeof(bo));
    rep(i,1,H)
        rep(j,1,W)
            if (s[i][j]=='.'&& !bo[i][j]){
                sum=0;
                floodfill(i,j);
                if (sum>ans)ans=sum;
            }
    cout<<ans<<endl;
    return 0;
}
```

Soj 1170 countdown

分析和解法:首先对每个人给予一个数字编号,这里我使用 stl 的 map 实现。而后根据给出的关系建立家族树。但实际上我只是一边读入，一边将父亲与孩子建一条有向边。由于数据量较小，我们直接枚举每一个成员，以这个成员为根节点用深度搜索搜索到第 d 层，统计该层的结点数量，就是枚举的这个成员的第 d 代子孙数。最后将所有含有第 d 代子孙的成员按照题目要求排序输出即可。

时间复杂度 O(n^2)

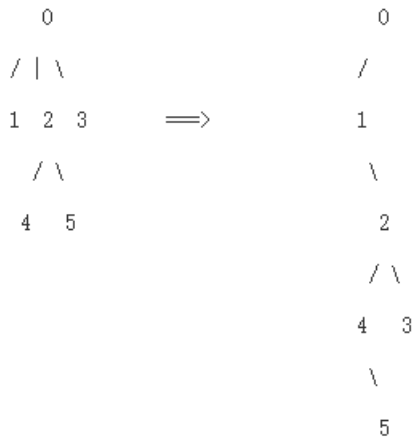
代码:

```
#include<iostream>
#include<cstring>
#include<algorithm>
#include<map>
#include<vector>
using namespace std;
#define rep(i,u,v) for (int i=(u);i<=(v);i++)
int n,m,d;
int root[1002],sizes[1002];
map<string,int>num;
map<int,string>name;
vector<int>g[1002];
vector<pair<string,int> >ans;
bool cmp(pair<string,int>n1,pair<string,int>n2){
    return n1.second>n2.second ||
           n1.second==n2.second && n1.first<n2.first;
}
```

```
void dfs(int now,int x,int dai){
    sizes[now]=0;
    for (int i=0;i<g[now].size();i++){
        if (dai+1==d)sizes[x]++;
        else {
            int y=g[now][i];
            dfs(y,x,dai+1);
        }
    }
}
int main(){
    int T,k;
    cin>>T;
    string s1,s2;
    rep(tt,1,T){
        cin>>n>>d;
        num.clear();name.clear();
        ans.clear();m=0;
        memset(root,0,sizeof(root));
        rep(i,1,n){
            cin>>s1>>k;
            int &x=num[s1];
            if (!x){
                x=++m;
                name[x]=s1;
                g[x].clear();
            }
            rep(j,1,k){
                cin>>s2;
                int &y=num[s2];
                if (!y){
                    y=++m;root[y]=1;
                    name[y]=s2;
                    g[y].clear();
                }
                g[x].push_back(y);
            }
        }
        rep(i,1,m){
            dfs(i,i,0);
            if (sizes[i]>0)
                ans.push_back(make_pair(name[i],sizes[i]));
        }
        if (tt>1)cout<<endl;
        cout<<"Tree "<<tt<<":"<<endl;
        sort(ans.begin(),ans.end(),cmp);
        for (int i=0;i<ans.size();i++){
            if (i>2&&ans[i].second!=ans[i-1].second)break;
            cout<<ans[i].first<<" "<<ans[i].second<<endl;
        }
    }
    return 0;
}
```

Soj 1490 Tree Grafting

分析和解法:题目意思很明显，就是叫我们先建一颗多叉树，求出该树的高度，再将该树转化成二叉树，再求二叉树的高度。然而这样太麻烦了，实际上我们直接建二叉树而可以了。下面讨论如何直接建二叉树求解两个高度。首先以第一个样例来分析，如下图：



从上图可以看出，多叉树转二叉树，就是将某一结点的第一个孩子结点作为左孩子结点，将第二个孩子结点作为左孩子结点的右孩子结点，将第三个孩子结点作为左孩子结点的右孩子结点的右孩子结点……明了这个过程，就好办了。按照递归过程建树，设递归函数有三个参数分别为当前结点 now，当前结点在多叉树中的高度 h1，当前结点在二叉树中的高度 h2.
若遇到’u’则返回上一层递归，若遇到‘d’，则有两种情况:

1. 当前结点无左孩子，记为 now->lc=NULL，那么将新的结点作为当前结点的左孩子，并递归下去这个点，同时 h1+1，h2+1，因为由图可以看出此时无论在哪种树中都是向下一层。
2. 当前结点有左孩子，右图中可以看出，我们应把新的点插入到当前点的左子树的最右边。用一个循环寻找到这个第二点，记录下包括当前点的左孩子在内途中经过了多少个点，记为 cnt。此时递归下去，h1+1，h2+cnt+1。因为在多叉树中是向下一层，而在二叉树中是在 h2+cnt 这层的下一层。

由于不确定每一层的点数，这题最好使用指针动态建树，按照上述做法即可求解。
时间复杂度 O(n)
代码:

```
#include<iostream>
using namespace std;
struct node{
    struct node* lc;
    struct node* rc;
};
int n,h1,h2,i,len;
string s;
void dfs(node* &now,int H1,int H2){
    while (1){
        if (++i==len)return;
        if (H1>h1)h1=H1;
        if (H2>h2)h2=H2;
        if (s[i]=='d'){
            node* n1=new node;
            n1->lc=NULL;n1->rc=NULL;
            if (now->lc==NULL){
                now->lc=n1;
                dfs(now->lc,H1+1,H2+1);
            }
```

```
            else {
                node *p=now->lc;
                int cnt=1;
                while (p->rc!=NULL)p=p->rc,cnt++;
                p->rc=n1;
                dfs(p->rc,H1+1,H2+cnt+1);
            }
        }
        else return;
    }
}
void del(node* now){
    if (now==NULL)return;
    del(now->lc);
    del(now->rc);
    delete now;
}
int main(){
    int T=0;
    while (cin>>s){
        if (s[0]=='#')break;
        len=s.length();
        node* root=new node;
        root->lc=NULL;root->rc=NULL;
        h1=h2=0;i=-1;
        dfs(root,0,0);
        printf("Tree %d: %d => %d\n",++T,h1,h2);
        del(root);
    }
    return 0;
}
```

Soj 14181 Flying Safely

分析和解法:题目这么绕，要让 spy 可以在任意两个城市往返，其实就是要让这些城市组成一个两两可直接或间接到达的连通图。而要求的最小飞机数就是要用最少的边将所有城市连接起来。那么显然，将 n 个点连接在一起的最小边数就是 n-1.
时间复杂度 O(1)
代码:

```
#include<iostream>
using namespace std;
#define rep(i,u,v) for (int i=(u);i<=(v);i++)
int main(){
    int T,n,m,x,y;
    cin>>T;
    rep(t,1,T){
        cin>>n>>m;
        rep(i,1,m)cin>>x>>y;
        cout<<n-1<<endl;
    }
    return 0;
}
```

4.程序运行与测试

11 道题均通过成功运行并通过 sicily 的测试。

smie15352408	1219	C++	Accepted	0.09sec	8504 KB	1268 Bytes
smie15352408	1140	C++	Accepted	0.28sec	4508 KB	1946 Bytes
smie15352408	1170	C++	Accepted	0.03sec	328 KB	1744 Bytes
smie15352408	1490	C++	Accepted	0.5sec	440 KB	1161 Bytes
smie15352408	1310	C++	Accepted	0.58sec	3152 KB	1645 Bytes
smie15352408	2002	C++	Accepted	0.07sec	7496 KB	913 Bytes
smie15352408	1426	C++	Accepted	0.17sec	4648 KB	992 Bytes
smie15352408	7766	C++	Accepted	0.39sec	5704 KB	919 Bytes
smie15352408	1083	C++	Accepted	0.02sec	308 KB	839 Bytes
smie15352408	14181	C++	Accepted	0.1sec	308 KB	239 Bytes

smie15352408	1034	C++	Accepted	0sec	312 KB	1191 Bytes

5.实验总结与心得

这次图论专题的练习算是对以前所学知识的一次巩固。对多叉树、二叉树、字母数、深度搜索、广度搜索、树型 dp，最短路和最小生成树的知识得到巩固和提高。图论图论，其难点就在于构图，如何能将问题抽象成一个数学模型且与上面所说的几个知识的模型相对应是解决这类问题的关键。一般来说，一旦构好了图，直接套模型的模板或者作一些修改就可以解决。而要能快速地想到应该怎么构图只能够多做题，培养题感。

附录、提交文件清单

实验报告.pdf
1140.cpp
1034.cpp
1219.cpp
1310.cpp
1083.cpp
1426.cpp
7766.cpp
2002.cpp
1170.cpp
1490.cpp

14181.cpp