以下题解仅供学习参考使用

抄袭、复制题解，以达到刷 AC 率/AC 数量或其他目的的行为，在洛谷是严格禁止的。

洛谷非常重视学术诚信。此类行为将会导致您成为作弊者。 具体细则请查看洛谷社区规则。

提交题解前请务必阅读题解审核要求及反馈要求。

3 篇题解　　默认排序　按时间排序

Sata_moto　更新时间：2019-10-23 21:49:45

在 Ta 的博客查看

前言：

1　　1 条评论

顺着LRJ蓝皮书刷到这题的...

本来不想写题解...

但是看了看，本题就一篇题解...而且图片爆炸了...

于是我就跑来写篇题解了...0.0...

题目大意：

原文戳此0.0

给你一个n个点m条边的无向图，不一定联通。现在你需要把原有的无向边变为有向边，并加入一些新的有向边。问最少加入多少条有向边使得图只有一个强连通分量。

题目分析：

拿到这题...像我一样的蒟蒻大概会首先想到分情况讨论...

我们先举个最特殊的情况：

假设n个点连成了环...

很明显，我们不需要在环内加任何有向边...

下一步，我们让这个情况稍微一般点...

假设我们得到的无向图中有一个环...

我们可以把它缩成一个点...

因为环内部不需要加边，所以缩点对答案不会有影响......

我们再一般一点，把视野扩展到连通无向图（题目没说连通）

让我们想一想一个连通无向图缩点之后变成了什么...

我们得到的一定是个仅有n-1条边(如果有更多边，就一定会形成环）的连通图，即是树...

那么，如何处理这颗树?

我们可以用一种贪心的思想：

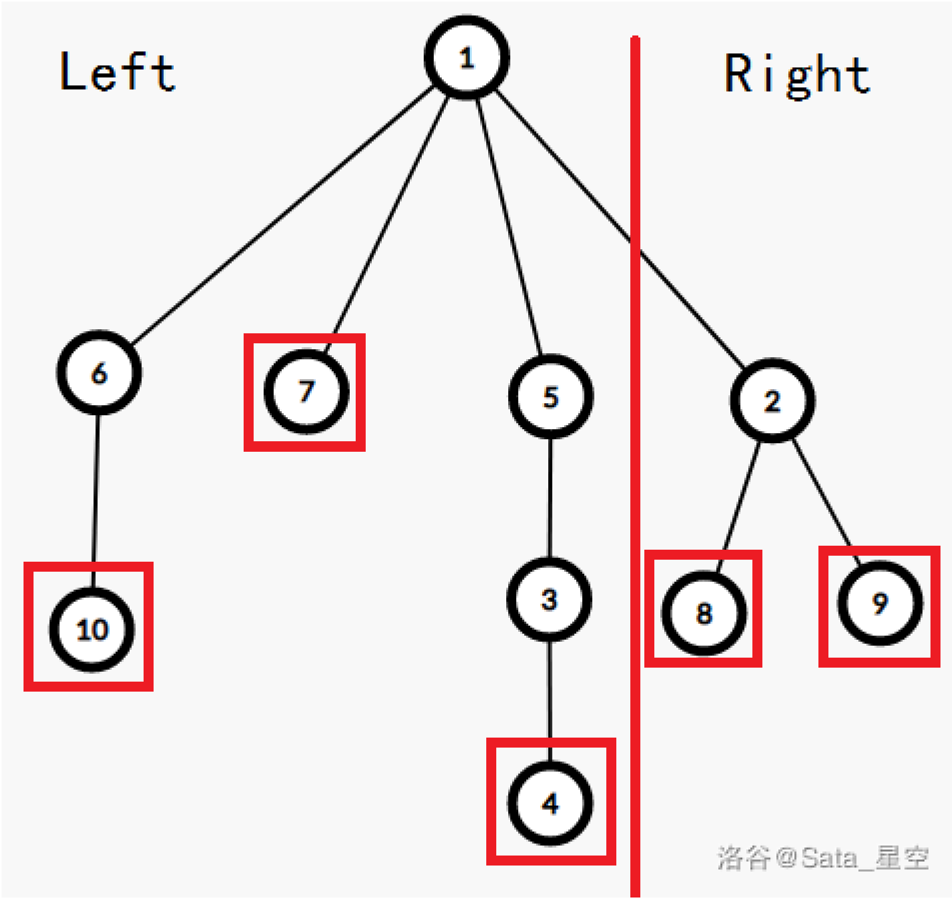
树无非是由很多条链组成的...

加一条边最多可以合并一条链的两端..

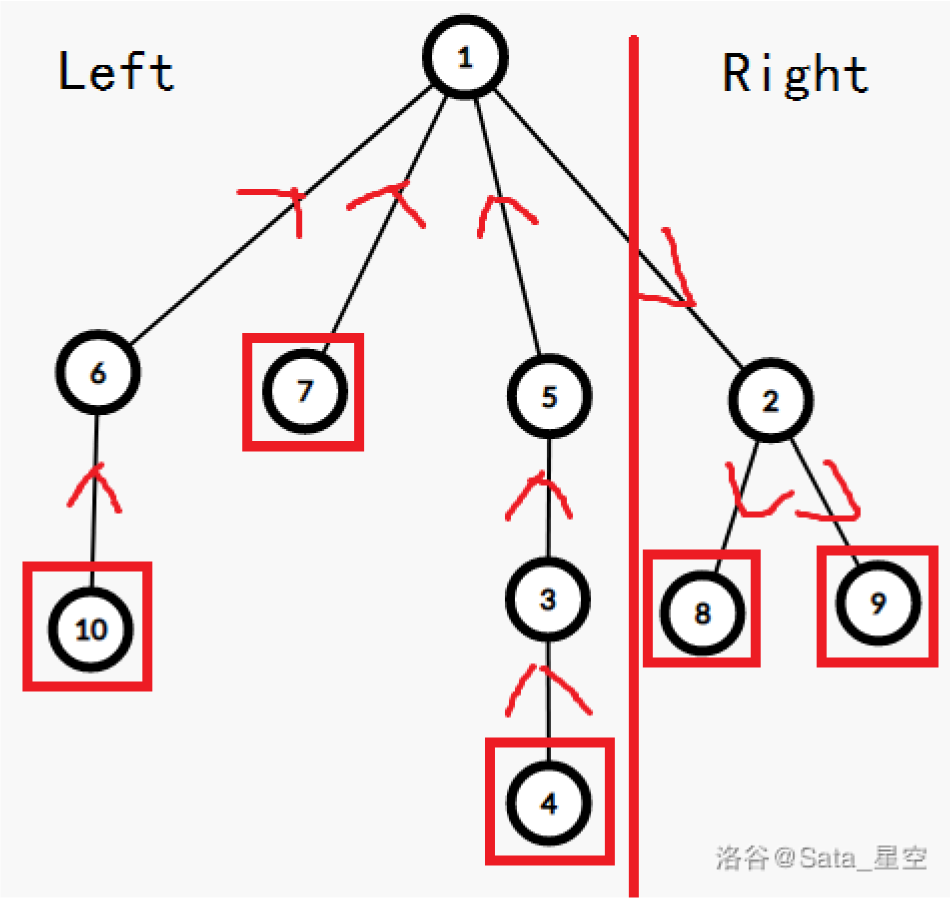
也就是说合并两个叶节点...

所以我们的贪心策略是：我们取一个非叶节点作为根，左边那一半叶节点把已有的边连向父亲，右边那一半则相反...然后两两相连...

大概这样说没啥用...我们看图：



我们先把叶节点（度数为1的点）平均分到两边



然后左边的边上连，右边的边下连...

这个时候我们发现，向上的边和向下的边一定会在它们对应的节点的LCA处相遇...

这意味任意的两个叶节点，在'上面'是连通的...

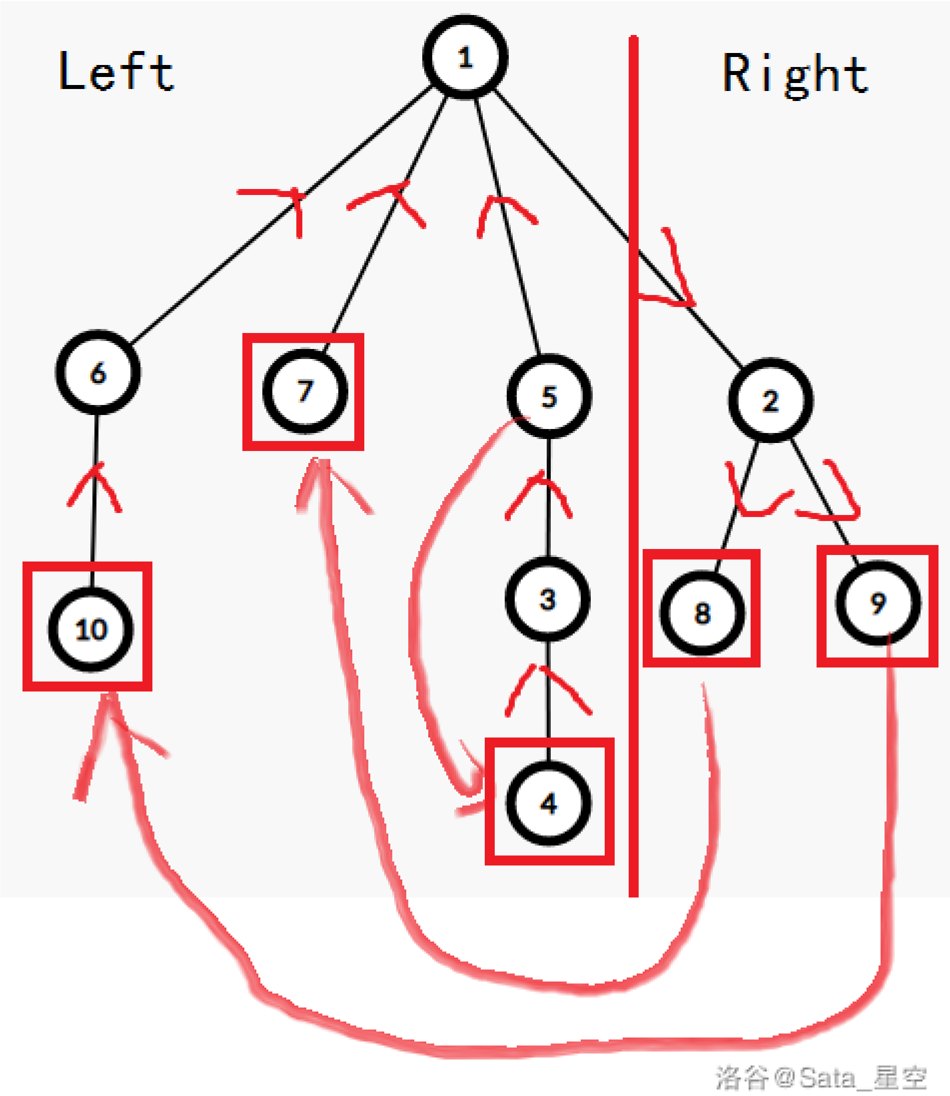
现在我们要让它们在下面也连通...

而我们的任意假如一条边，可以连通两个点（明白为什么把叶节点均分了吗？）

很明显，我们至少要ceil（叶节点数目/2）条边------这一定是在左右均分的情况下实现的...

多出来的那一个怎么办?

随便连一连就好...



搞定.0.0.

不过没有完，这里有一个需要注意的地方：

如果我们的根节点（R)度数为1，那么他也是一个广义上的'叶子节点'

你可以理解为，我们选取一个树上的"饱满节点"A，以A为根重构树形，那么R会变成一个叶子节点...

还有一种特殊情况，一个度数为0的点至少要向外界连两条边（均摊1条边(因为边的另一端也连了点啊)）...所以它对答案的贡献为叶子节点的两倍（叶子节点均摊需要½条边）...

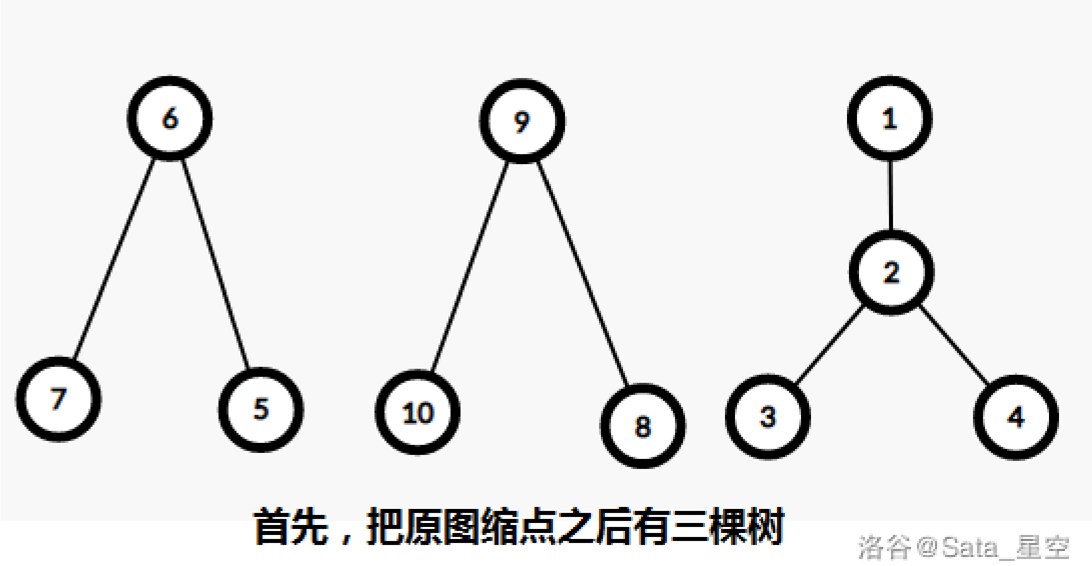
至此，我们可以推出对连通图的公式：ceil（（度数为1的节点的个数（广义叶子节点）+度数为0的节点的个数*2）/2）

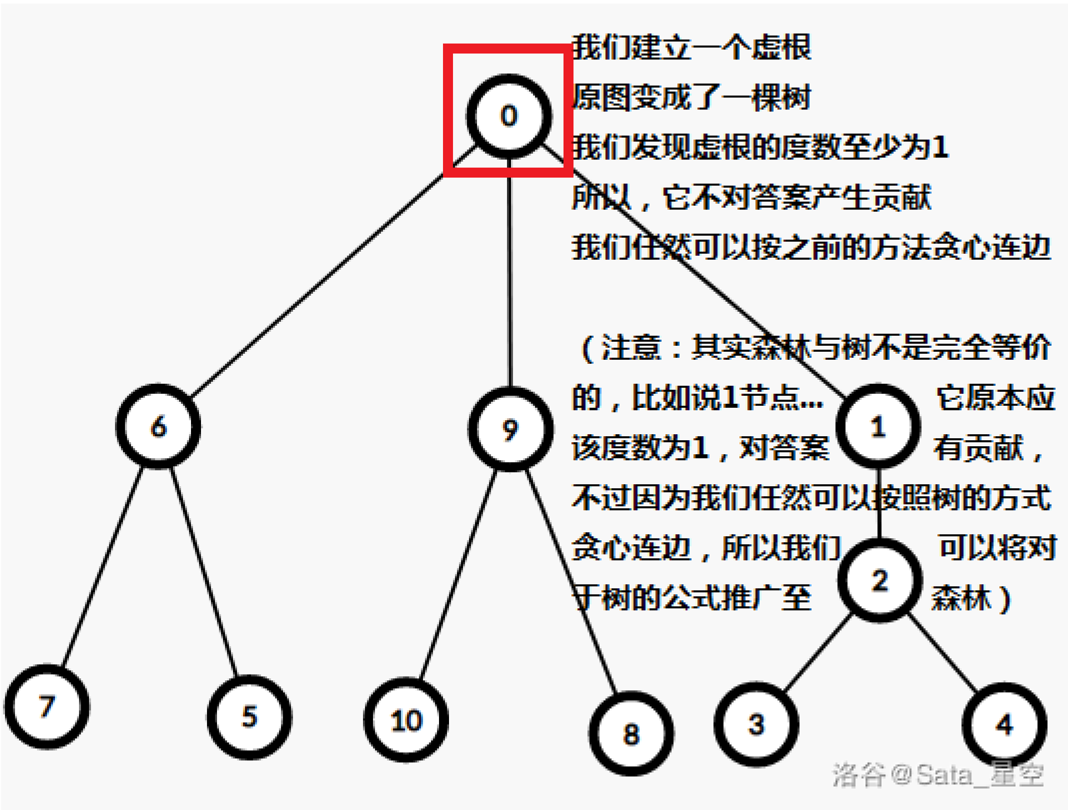
至于更一般的情况：图可能不连通...

很简单...我们假设有一个虚根，把它与森林的根连起来就好...

易证（因为虚根度数至少为2）...需要的边数任然是ceil（（度数为1的节点的个数（广义叶子节点） + 度数为0的节点的个数*2）/2）

不明所以就看图：





至此，我们得到了一般性的公式：

对于任意连通无向图，将它进行缩点以后，所需要新增的边数一定是ceil（(度数为0的节点个数*2+度数为1的节点个数)/2）...

剩下细节的自己处理吧0.0...

代码:

```
#include <stack>
#include <cmath>
#include <vector>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <algorithm>

const int N = 1010;

int n, m, color, tot;
int col_p[N], dfn[N], low[N];
bool in[N], visit[N];
std::vector <int > link[N];
std::vector <int > new_link[N];
std::stack <int > s;

void tarjan(int wz, int fa)//无向图求环...类似有向图
{
    dfn[wz] = low[wz] = ++tot;
    in[wz] = true, s.push(wz);

    for(int k = 0; k < (int )link[wz].size(); k++)
    {
        int to = link[wz][k];

        if(!dfn[to])
        {
            tarjan(to, wz);
            low[wz] = std::min(low[wz], low[to]);
        }
        else if(in[to] && to != fa)
            low[wz] = std::min(low[wz], dfn[to]);
    }

    if(low[wz] == dfn[wz])
    {
        color++;

        while(s.top() != wz)
        {
            col_p[s.top() ] = color;

            in[s.top() ] = false, s.pop() ;
        }
```

```
        col_p[wz] = color;

        in[s.top() ] = false, s.pop() ;
    }
}

void build()
{
    for(int k = 1; k <= n; k++)
        for(int i = 0; i < (int )link[k].size() ; i++)
        {
            int to = link[k][i];

            if(col_p[to] != col_p[k])
                new_link[col_p[to]].push_back(col_p[k]);
        }
}

int answ;

int dfs(int wz, int fa)
{
    visit[wz] = true;

    int size1 = (int )new_link[wz].size();

    int size0 = size1 == 0 ? 2 : size1 == 1 ? 1 : 0;

    for(int k = 0; k < (int )new_link[wz].size(); k++)
    {
        int to = new_link[wz][k];

        if(to != fa)
            size0 += dfs(to, wz);
    }

    return size0;
}

void work()
{
    memset(link, 0, sizeof(link));
    memset(new_link, 0, sizeof(new_link));
    memset(dfn, 0, sizeof(dfn));
    memset(low, 0, sizeof(low));
    memset(visit, 0, sizeof(visit));
    tot = color = answ = 0;
```

```
    for(int k = 1; k <= m; k++)
    {
        int l, r;
        scanf("%d %d", &l, &r);

        link[l].push_back(r);
        link[r].push_back(l);
    }

    //缩点
    for(int k = 1; k <= n; k++)
        if(!dfn[k])
            tarjan(k, 0);

    //重建图
    build();

    //搜索度数为0或1的点
    for(int k = 1; k <= color; k++)
        if(!visit[k])
            answ += dfs(k, 0);

    if(color!=1)
        printf("%d\n", (int )ceil(answ / 2.0));
    else
        printf("0\n");
}

int main()
{
    while(scanf("%d %d", &n, &m) != EOF)
    {
        work();
    }

    return 0;
}
```

结语：

如果本题解有BUG...

那么...那么...那么...

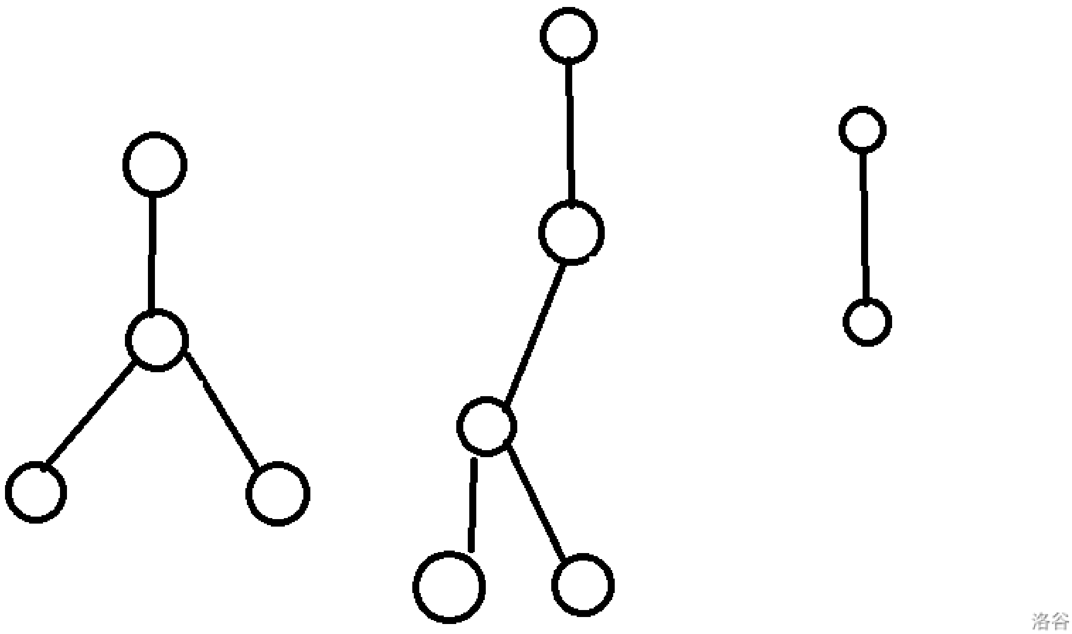
（随意了）还请私信作者....

END

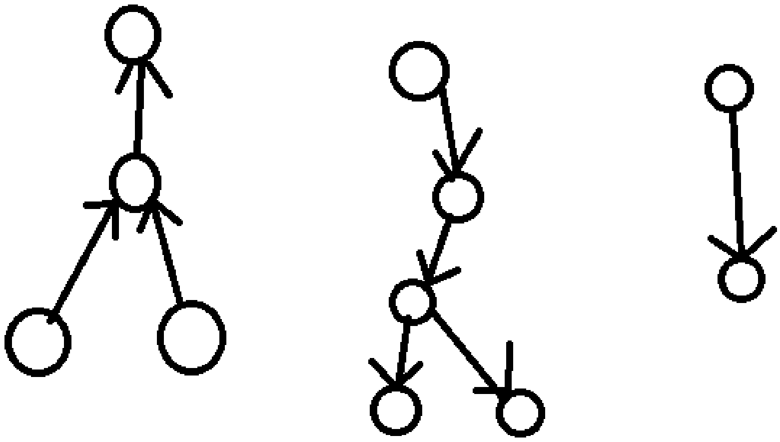
地表最强男人　更新时间：2019-11-01 22:14:14

在 Ta 的博客查看

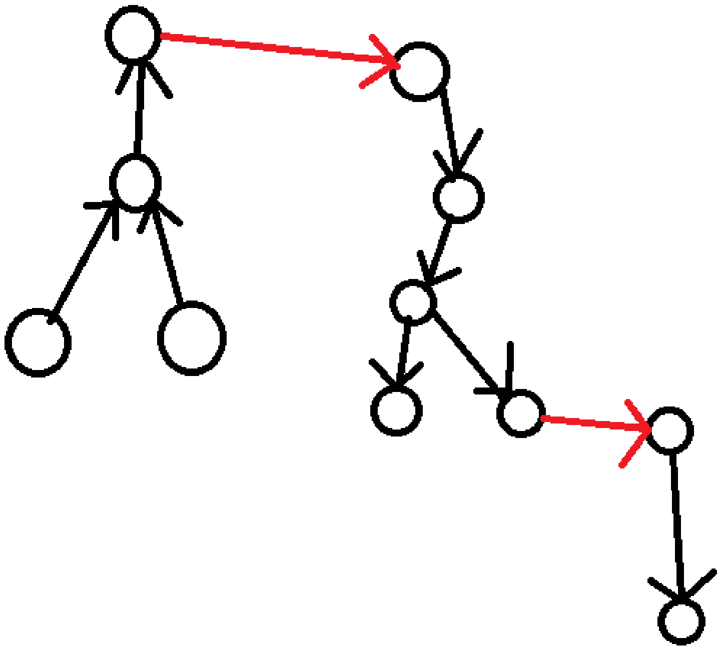
- 其实这一题这个博客已经大部分讲的很清楚了，但是还是有一些地方没有说明清楚，所以这里做一些补充。
- 主要还是先将所有的环缩点，注意这里能够缩点的是环，而不是点双连通分量，也不是边双连通分量。因为只有环改成有向图后才能形成强连通分量。将所有的环缩成点以后，那么剩下的就是一颗树或者是森林，那么怎么改边和加边才是最优的呢？
- 可以发现，如果一个有向图要是一个强连通分量，那么一个点入度和出度都至少要为1。因为一个点只有能够走到其他点，并且至少能够被一个点走到，才有可能是强连通分量。考虑一下最优的情况，每一次的加边的边的两端都是缩点以后度为0或者度为1的点，这样每一次加：(入度为0的点的数量*2+入度为1的点的数量)/2向上取整。但是能不能构造出这样的情况呢？
- 如果针对一颗树来说的话，处理方法就是那一个博客的处理方法，那么如果是森林的话呢？
- 可以想办法将森林合并成一棵树，我们还是分成左边和右边，即按照叶子节点的个数平均分配，然后把左边的树连接在左边的树上，把右边的树连接在右边的树上。但是连接需要加边，所以为了符合将入度为1的点连接起来，所以可以先将每一棵树度为一的节点提出一个来。



- 然后把边改成有向边，分成左边和右边（按照题解的法则，即将叶子节点平均分，分成左边和右边。

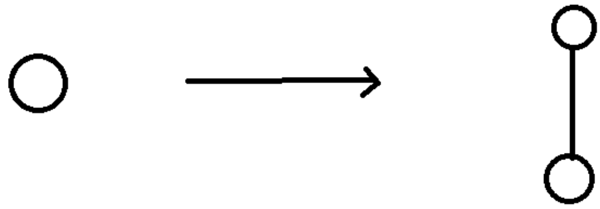


- 然后将右边的树接在右边的树的下面，左边的接在左边的下面，最后把根接起来。（红色的是我们多添加的边）

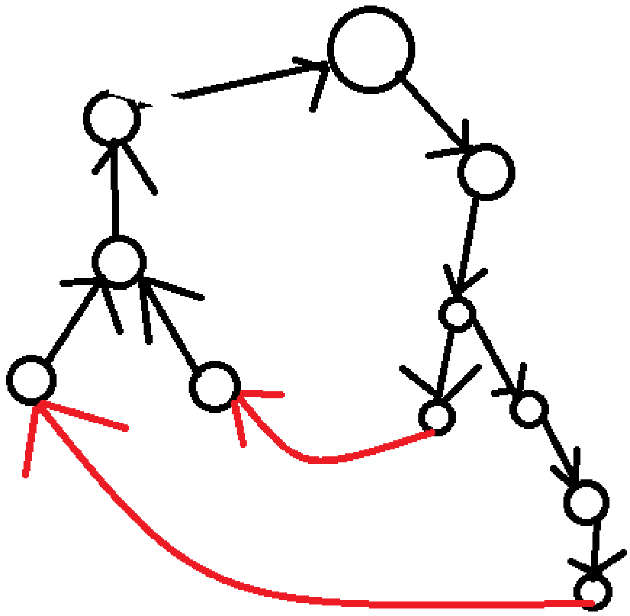


)

- 这样连边一定是最优的，因为每一次连的边都是将度为1的点连起来，无论是对于第一步将度数为1的节点提出来，还有最后一步的将根连起来，都保证了加边是符合原则的。而如果有度数为0的节点，那么可以将它拆开，拆成两个度数为1的节点。



- 然后依旧按照我们加边的法则来处理。把这个森林变成树了以后，就可以按照树的方法处理了，也就是将每一个右边的叶子向左边的叶子连边，如果多余的话就自己向根节点连边。



- 红色的就是连的边，那么这样的树（或者是森林）就构造完了，证明完成。

```
#include<iostream>
#include<cstring>
#include<algorithm>
#include<cstdio>
#include<cmath>
using namespace std;
const int N=1010,M=1001000;
int n,m;
int head[N],ver[M<<1],Next[M<<1],tot,stack[N<<1],top,num,cnt,edge[N];//edge用来记录度
int head1[N],ver1[M<<1],Next1[M<<1],tot1;
int c[N],anss,dfn[N],low[N];
bool ins[N];
void pre()
{
    memset(head,0,sizeof(head));
    memset(head1,0,sizeof(head1));
    memset(dfn,0,sizeof(dfn));
    memset(ins,0,sizeof(ins));
    memset(c,0,sizeof(c));
    memset(edge,0,sizeof(edge));
    memset(stack,0,sizeof(stack));
    tot=tot1=top=num=cnt=anss=0;
}
```

```
void add(int x,int y)
{
        ver[++tot]=y;
        Next[tot]=head[x],head[x]=tot;
}
void Add(int x,int y)
{
        ver1[++tot1]=y;
        Next1[tot1]=head1[x],head1[x]=tot1;
}
void tarjan(int x,int root)
{
        dfn[x]=low[x]=++num;
        stack[++top]=x;
        ins[x]=1;
        if(x==root&&head[x]==0)
        {
            c[x]=++cnt;
            ins[x]=0;
            return ;
        }
        for(int i=head[x];i;i=Next[i])
        {
            int y=ver[i];
            if(!dfn[y])
            {
                tarjan(y,x);
                low[x]=min(low[x],low[y]);
            }
            else if(ins[y]&&y!=root)
                low[x]=min(low[x],dfn[y]);
        }
        if(dfn[x]==low[x])
        {
            int y;
            cnt++;
            do
            {
                y=stack[top--];
                ins[y]=0;
                c[y]=cnt;
            }while(y!=x);
        }
}
int main()
{
        while(scanf("%d %d",&n,&m)!=EOF)
        {
            pre();
```

```
        for(int i=1;i<=m;i++)
        {
            int x,y;
            cin>>x>>y;
            add(x,y);
            add(y,x);
        }
        for(int i=1;i<=n;i++)
            if(!dfn[i])
                tarjan(i,i);
        for(int x=1;x<=n;x++)
            for(int i=head[x];i;i=Next[i])
            {
                int y=ver[i];
                if(c[x]!=c[y])
                {
                    Add(c[x],c[y]);
                    edge[c[x]]++;
                    edge[c[y]]++;
                }
            }
        for(int i=1;i<=cnt;i++)
        {
            if(edge[i]==0)
                anss+=2;
            if(edge[i]==2)
                anss+=1;
        }
        if(cnt!=1)
            cout<<((int)ceil(anss/2.0))<<endl;
        else
            cout<<0<<endl;
    }
    return 0;
}
```


GoldenPotato137 更新时间：2019-04-09 11:48:50

在 Ta 的博客查看

戳我获得更好的阅读体验qwq

Solution

这题就比较牛皮。

我们先来考虑一下图联通的话怎么做。显然，我们可以先把图按边双缩点，边双内部是肯定不用加任何一条有向边就能改成强连通分量的（易证）。
缩完点之后，图一定会变成一颗树。接下来我们依旧可以像这道题那样贪心。我们数一下广义叶子数有多少，要加的边的个数一定为$sum/2$（向上取整）。
连边方式如图所示：


接下来再来考虑不连通的情况。显然，我们可以发现，对于多颗树来说，我们依旧可以照样刚刚那样贪心。我们左右两棵树在叶子那里连边即可。

因此，我们的总答案依旧是$sum/2$（向上取整）

时间复杂度$O(n)$
就酱，这题就被我们切掉啦(*≧▿≦)

Code

```
//UVA10972 RevolC FaeLoN
//Apr,9th,2019
//边双
#include<iostream>
#include<cstdio>
#include<vector>
#include<cstring>
using namespace std;
long long read()
{
    long long x=0,f=1; char c=getchar();
    while(!isdigit(c)){if(c=='-') f=-1;c=getchar();}
    while(isdigit(c)){x=x*10+c-'0';c=getchar();}
    return x*f;
}
const int N=1000+10;
vector <int> e[N],e2[N];
int dfn[N],dfn_to,low[N],mstack[N],top,belong[N],cnt;
bool vis[N],InStack[N];
void Tarjan(int now,int father)
{
    vis[now]=InStack[now]=true;
    mstack[++top]=now;
    dfn[now]=low[now]=++dfn_to;
    for(int i=0;i<int(e[now].size());i++)
        if(vis[e[now][i]]==false)
        {
```

```
            Tarjan(e[now][i],now);
            low[now]=min(low[now],low[e[now][i]]);
        }
        else if(e[now][i]!=father and InStack[e[now][i]]==true)
            low[now]=min(low[now],dfn[e[now][i]]);
    if(low[now]==dfn[now])
    {
        cnt++;
        while(mstack[top+1]!=now)
            InStack[mstack[top]]=false,
            belong[mstack[top--]]=cnt;
    }
}
int n,m;
int dfs(int now,int father)
{
    vis[now]=true;
    int t_ans=0;
    for(int i=0;i<int(e2[now].size());i++)
        if(e2[now][i]!=father)
            t_ans+=dfs(e2[now][i],now);
    if(e2[now].size()==1)
        t_ans++;
    if(e2[now].size()==0)
        t_ans+=2;
    return t_ans;
}
int main()
{
    for(int o=1;;o++)
    {
        if(scanf("%d%d",&n,&m)==EOF) break;

        for(int i=0;i<=n;i++)
            e[i].clear(),e2[i].clear();
        for(int i=1;i<=n;i++)
            e[i].reserve(4),e2[i].reserve(4);
        for(int i=1;i<=m;i++)
        {
            int s=read(),t=read();
            e[s].push_back(t);
            e[t].push_back(s);
        }

        memset(vis,0,sizeof vis);
        memset(mstack,0,sizeof mstack);
        dfn_to=cnt=0;
        for(int i=1;i<=n;i++)
            if(vis[i]==false)
```

```
                Tarjan(i,i);

            for(int i=1;i<=n;i++)
                for(int j=0;j<int(e[i].size());j++)
                    if(belong[i]!=belong[e[i][j]])
                        e2[belong[i]].push_back(belong[e[i][j]]);
            memset(vis,0,sizeof vis);
            if(cnt==1)
                printf("0\n");
            else
            {
                int ans=0;
                for(int i=1;i<=n;i++)
                    if(vis[belong[i]]==false)
                        ans+=dfs(belong[i],belong[i]);
                printf("%d\n",ans/2+ans%2);
            }
        }
        return 0;
    }
```

0 0 条评论 收起