最小生成树+Tarjan算法+拓扑排序

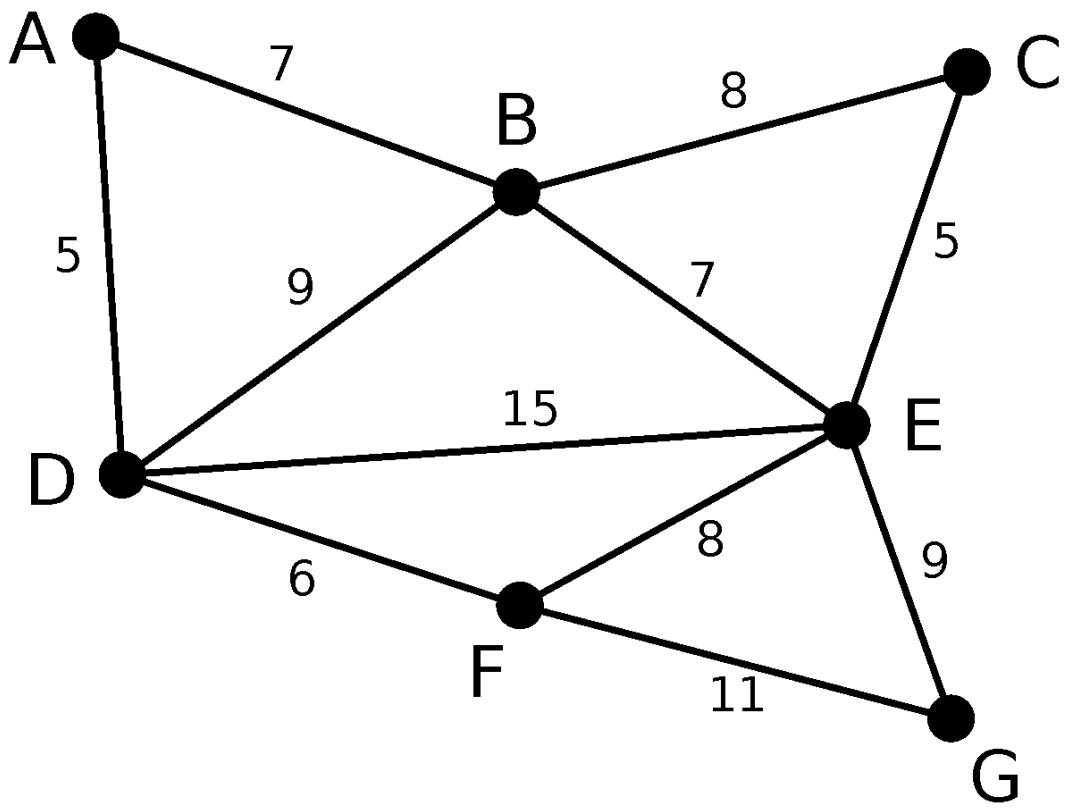
王琦

最小生成树(MST)

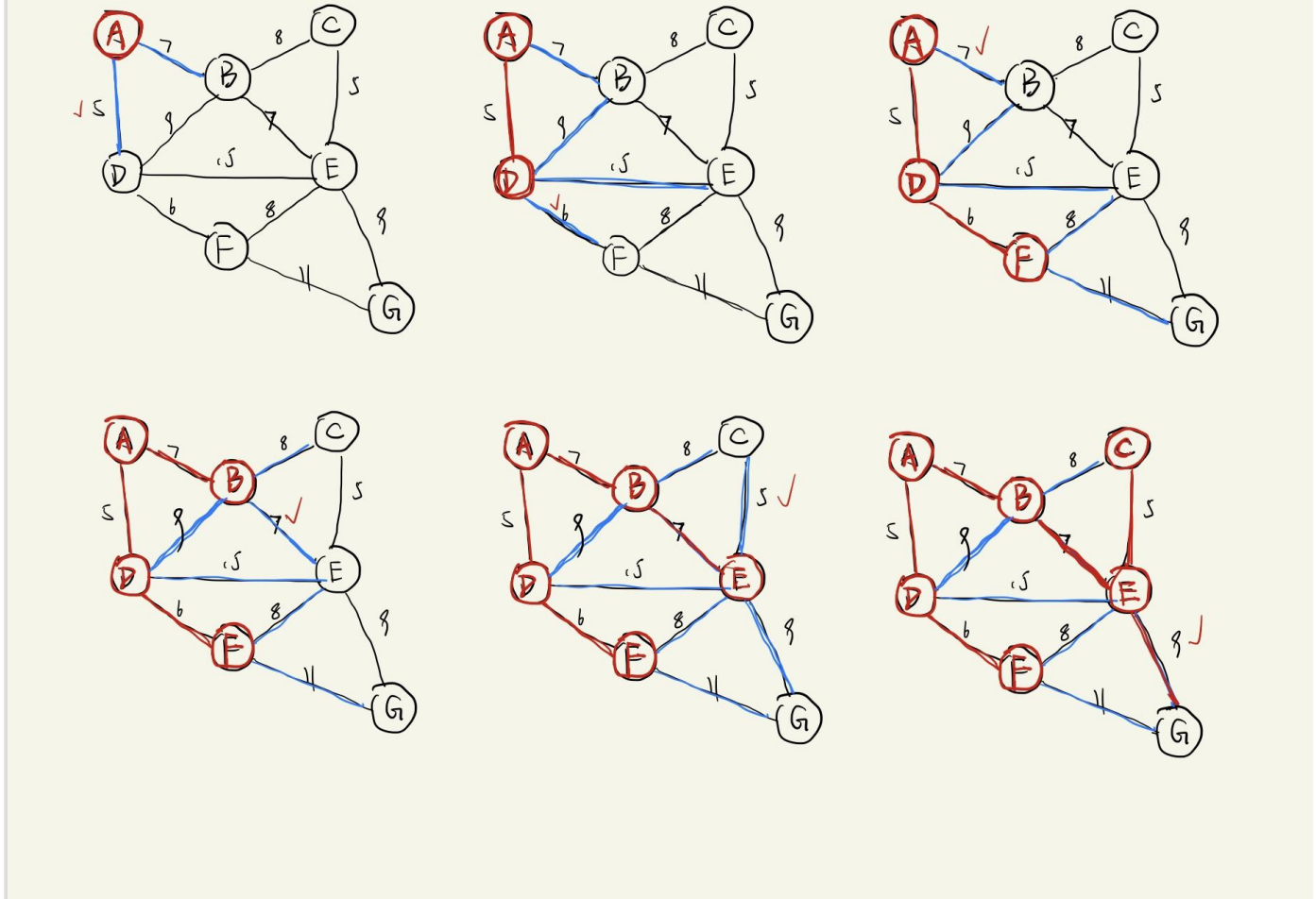
- 给定一个无向图，如果它的某个子图中任意两个顶点都互相连通并且是一棵树，那么这棵树就叫最小生成树(Spanning Tree)。如果边上有权值，那么使得边权和最小的生成树叫做最小生成树(Minimum Spanning Tree)。

求解最小生成树的主要算法有Prim算法和Kruskal算法。

Prim算法



在不构成环的情况下，每次选择距离最小的一个结点，用新的边更新其他结点的距离

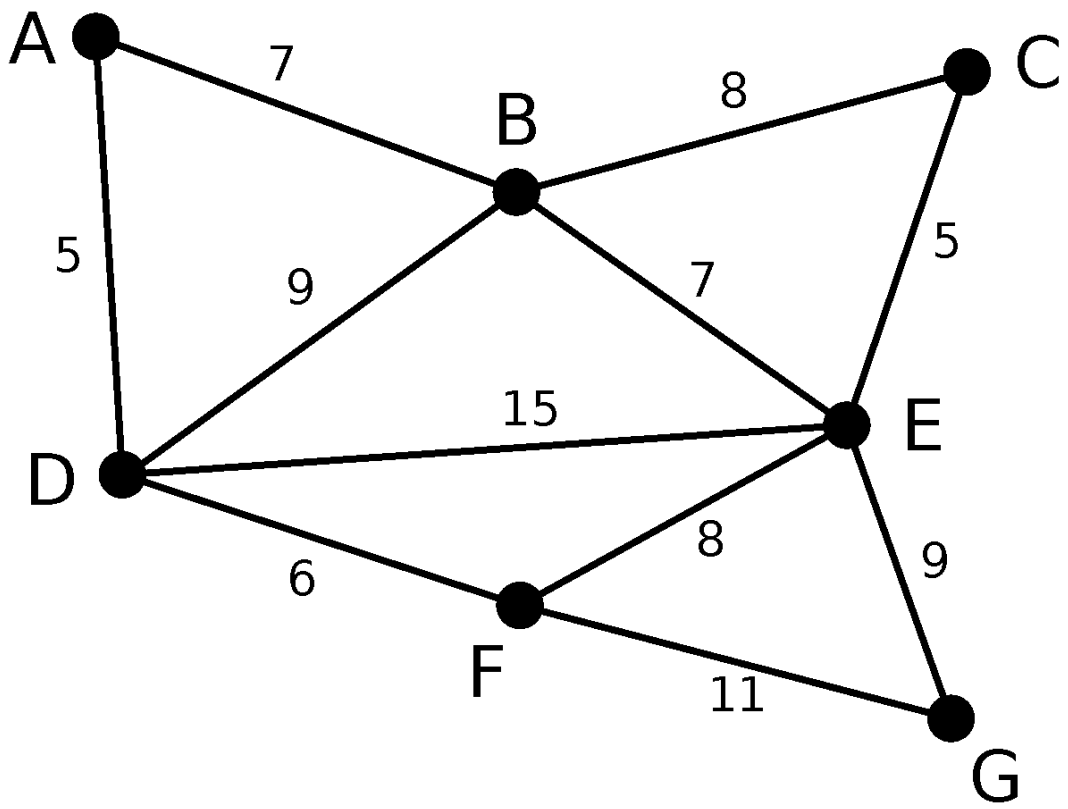

A
B
C
D
E
F
G

```cpp
#include <iostream>//Prim
#include <string.h>
#define maxn 5001
using namespace std;
long long inf = 0x3f3f3f3f;
int edge[maxn][maxn],dis[maxn];//dis被选中点连接的最短边
int n,m;//n个点，m条边  u->v  权为w
bool node[maxn];//点是否被访问
```

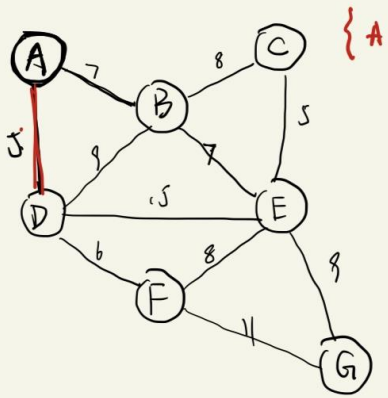
```cpp
int main() {
    cin>>n>>m;
    memset( b: edge,  c: inf,  len: sizeof(edge));
    while(m--)
    {
        int u,v,w;
        cin>>u>>v>>w;
        edge[u][v] = min(w, edge[u][v]);
        edge[v][u] = edge[u][v];
    }
    int ans = prim();
    if(ans==inf)
        cout<<"orz"<<endl;
    else
        cout<<ans<<endl;

    return 0;
}
```

```cpp
int prim()
{
    memset( b: dis,  c: inf,  len: sizeof(dis));
    int ans = 0;
    for(int i=1;i<=n;i++) {
        dis[i] = edge[1][i];//从第一个点开始
    }
    dis[1]=0;
    node[1]=true;

    for(int i=1;i<=n-1;i++)//结点1已访问
    {
        int min = inf,index=0;
        for (int j = 1; j <= n; j++)
        {
            if (!node[j] && dis[j] < min) {
                min = dis[j];
                index = j;
            }
        }
        if(min==inf)
            return inf;
        else
        {
            node[index] = true;
            ans += dis[index];
        }


        for (int j = 1; j <= n; j++)
        {
            if (!node[j] && edge[index][j] < dis[j])//如果未访问，并且这个点到到任意一点的距离比现在到树的距离近
                dis[j] = edge[index][j];
        }
    }
    return ans;
}
```

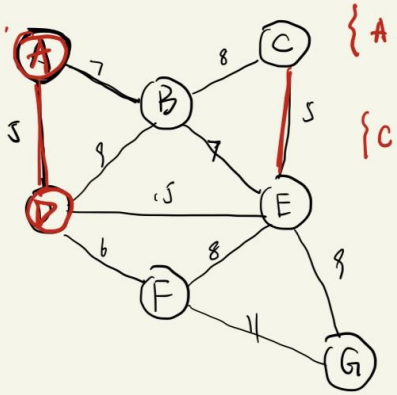
Kruskal算法



从权值最小的边开始，在不构成环的情况下，按照从小到大的顺序选择，当加入了n-1条边，形成了一颗树

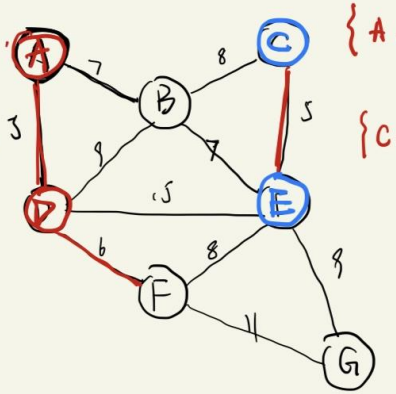


{A.D}



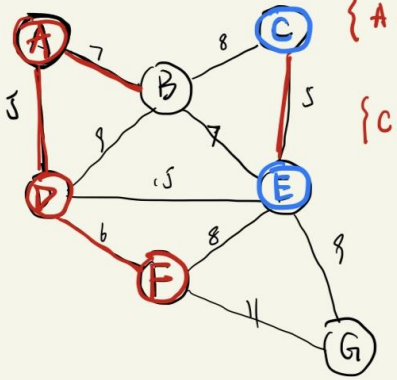
{A.D}

{C.E}



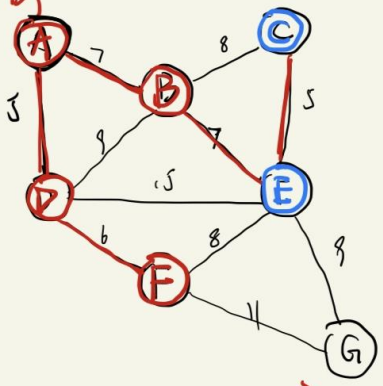
{A.D.F}

{C.E}

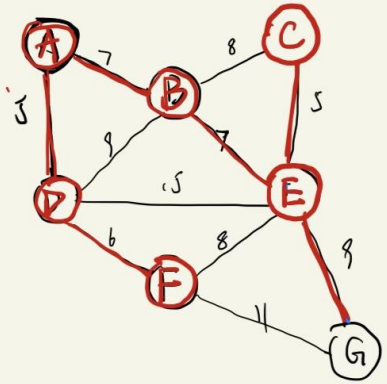


{A.D.F.B}

{C.E}



{A.D.F.B.C.E}



{A.D.F.B.C.E.G}

```cpp
#include <iostream>//Kruskal
#include <algorithm>
using namespace std;
#define maxn 5001
#define maxm 200001
typedef long long ll;
ll inf=0x3f3f3f, n, m;
ll f[maxn];  //并查集
struct e{
    ll u,v,w;
}edge[maxm]; //存储边
bool cmp(e n1, e n2)//排序函数
{
    return n1.w<n2.w;
}
ll find(ll k)
{
    if(k!=f[k])
        f[k]=find( k: f[k]);
    else
        return k;
}
```

```cpp
ll kruskal()
{
    ll ans=0, cnt=0;
    for(ll i=1;i<=n;i++)
        f[i]=i;
    for(ll i=1;i<=m;i++)
    {
        ll u=edge[i].u, v=edge[i].v, w=edge[i].w;
        u = find( k: u);
        v = find( k: v);
        if(u==v)
            continue;
        else if(u!=v)
        {
            f[v] = u;
            ans += w;
            cnt++;
        }
    }
    return cnt==n-1? ans:inf;
}
```

```cpp
int main()
{
    cin>>n>>m;
    for(ll i=1;i<=m;i++)
    {
        ll u,v,w;cin>>u>>v>>w;
        edge[i].u=u, edge[i].v=v, edge[i].w=w;
    }
    sort( first: edge+1, last: edge+m+1, comp: cmp);

    ll ans = kruskal();
    if(ans==inf)
        cout<<"orz"<<endl;
    else
        cout<<ans<<endl;

    return 0;
}
```

拓扑排序

- 对于一个有向无环图(DAG)G，将G中的所有顶点排序为一个线性序列，使得图中任意一对顶点u和v，如果u、v之间存在一条从u指向v的边，那么u的线性序列一定在v前。

Kahn算法

- 1)找到入度为0 的顶点找到并记录到队列或者栈中；
- 2)移除找到的入度为0的顶点和对应的以该顶点为起点的边，并将被移除的顶点加入到list集合中，同时移除的顶点作为起点的边的终点的如度减去1；继续循环1的步骤，直至队列或者栈为空。
- 3)此时list集合中的顶点的顺序输出就是拓扑排序的结果；如果list集合的元素数量少于顶点数量则说明该有向图存在环。

- **题目描述**
- 一个不同的值的升序排序数列指的是一个从左到右元素依次增大的序列，例如，一个有序的数列 A,B,C,D表示A<B,B<C,C<D。在这道题中，我们将给你一系列形如 A<BA<B 的关系，并要求你判断是否能够根据这些关系确定这个数列的顺序。
- 输入格式
- 第一行有两个正整数 n,mn,m，nn 表示需要排序的元素数量，2≤n≤262≤n≤26，第 11 到 nn 个元素将用大写的 A,B,C,D…A,B,C,D… 表示。mm 表示将给出的形如 A<BA<B 的关系的数量。
- 接下来有 mm 行，每行有 33 个字符，分别为一个大写字母，一个 < 符号，一个大写字母，表示两个元素之间的关系。

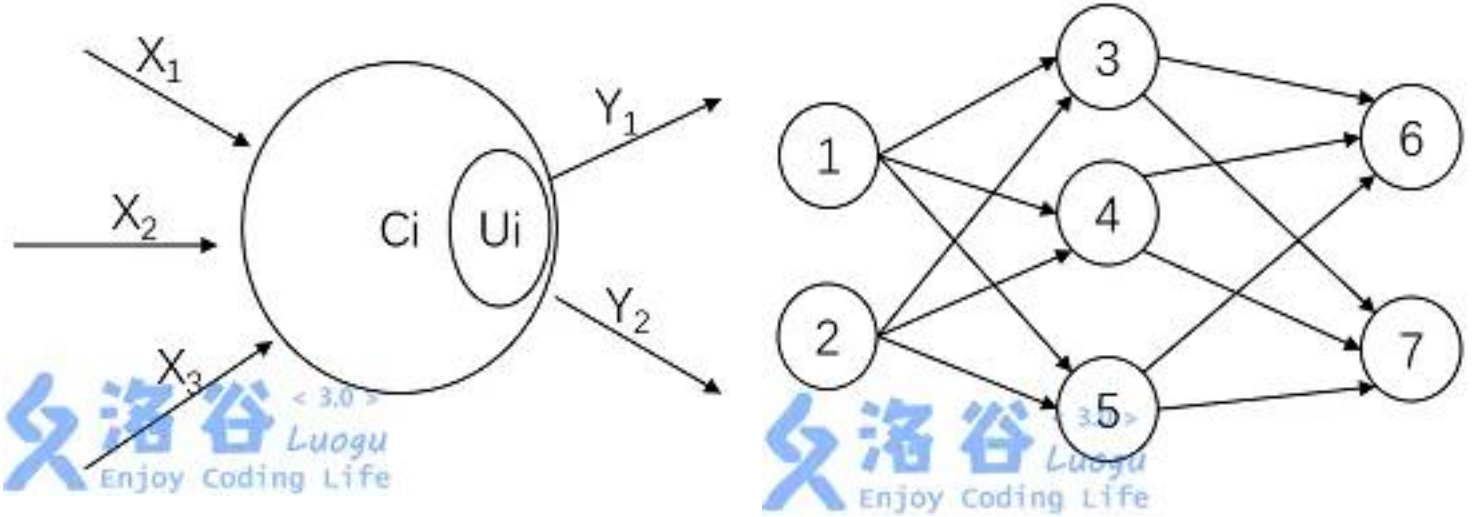
https://www.luogu.com.cn/problem/P1038

题目描述

在兰兰的模型中，神经网络就是一张有向图，图中的节点称为神经元，而且两个神经元之间至多有一条边相连，下图是一个神经元的例子：

图中，X1~X3是信息输入渠道，Y1~Y2是信息输出渠道，Ci表示神经元目前的状态，Ui是阈值，可视为神经元的一个内在参数。

神经元按一定的顺序排列，构成整个神经网络。在兰兰的模型之中，神经网络中的神经元分为几层；称为输入层、输出层，和若干个中间层。每层神经元只向下一层的神经元输出信息，只从上一层神经元接受信息。下图是一个简单的三层神经网络的例子。



兰兰规定，Ci 服从公式：

（其中 n 是网络中所有神经元的数目）

$$C_i = \left(\sum_{(j,i) \in E} W_{ji} C_j \right) - U_i$$

公式中的 Wji（可能为负值）表示连接 j 号神经元和 i 号神经元的边的权值。当 Ci 大于 00 时，该神经元处于兴奋状态，否则就处于平静状态。当神经元处于兴奋状态时，下一秒它会向其他神经元传送信号，信号的强度为 Ci。

如此．在输入层神经元被激发之后，整个网络系统就在信息传输的推动下进行运作。现在，给定一个神经网络，及当前输入层神经元的状态（Ci），要求你的程序运算出最后网络输出层的状态。

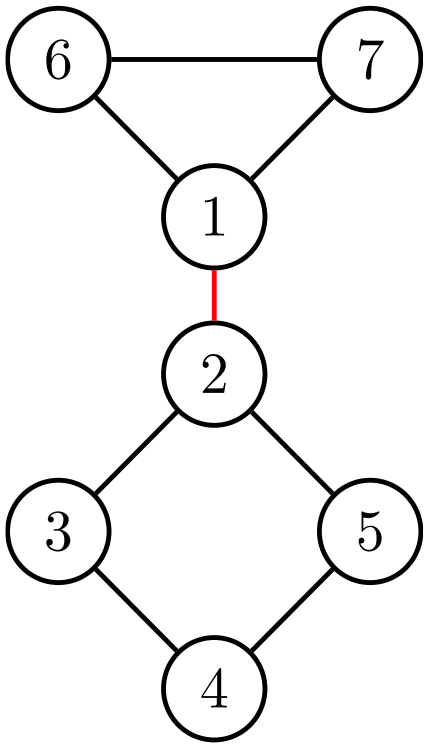
Tarjan算法

- dfn[x]：点x到dfs树被访问的时间
- low[x]：点x在dfs树中，由该点及其后代连接的边中可以返回的最小的时间

case(1) 割点 & 桥

- **割点**：无向连通图中，去掉某点和连接点的边后，连通分量数量增加
- **桥**：无向连通图中，去掉某条边后，连通分量增加

无向图中，dfs树中 子->父 的边不回溯



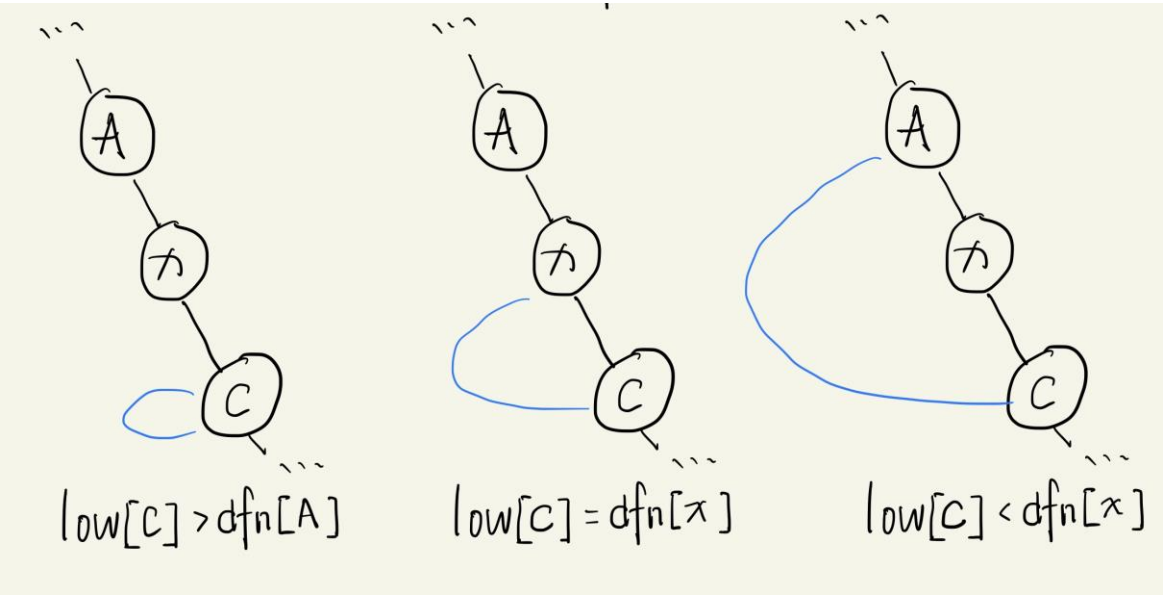
X是割点：

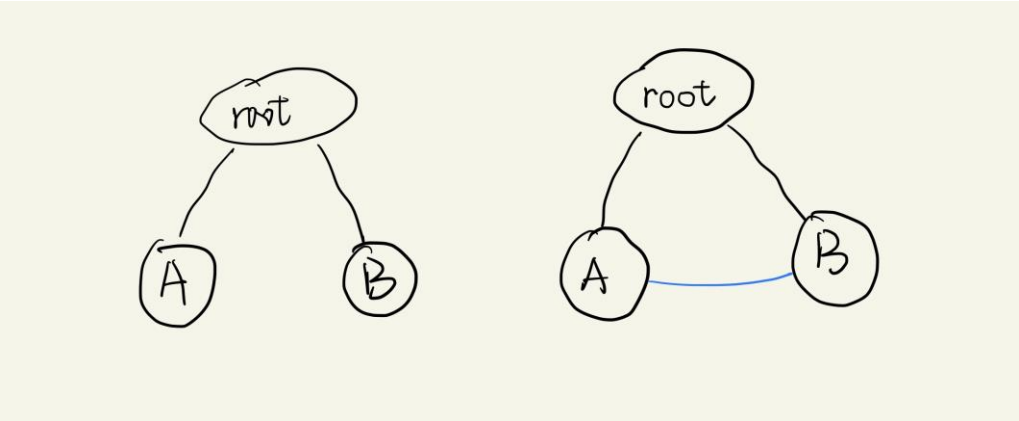
Case1: 非root&有儿子

low[x的儿子]>=dfn[x]

Case2: root

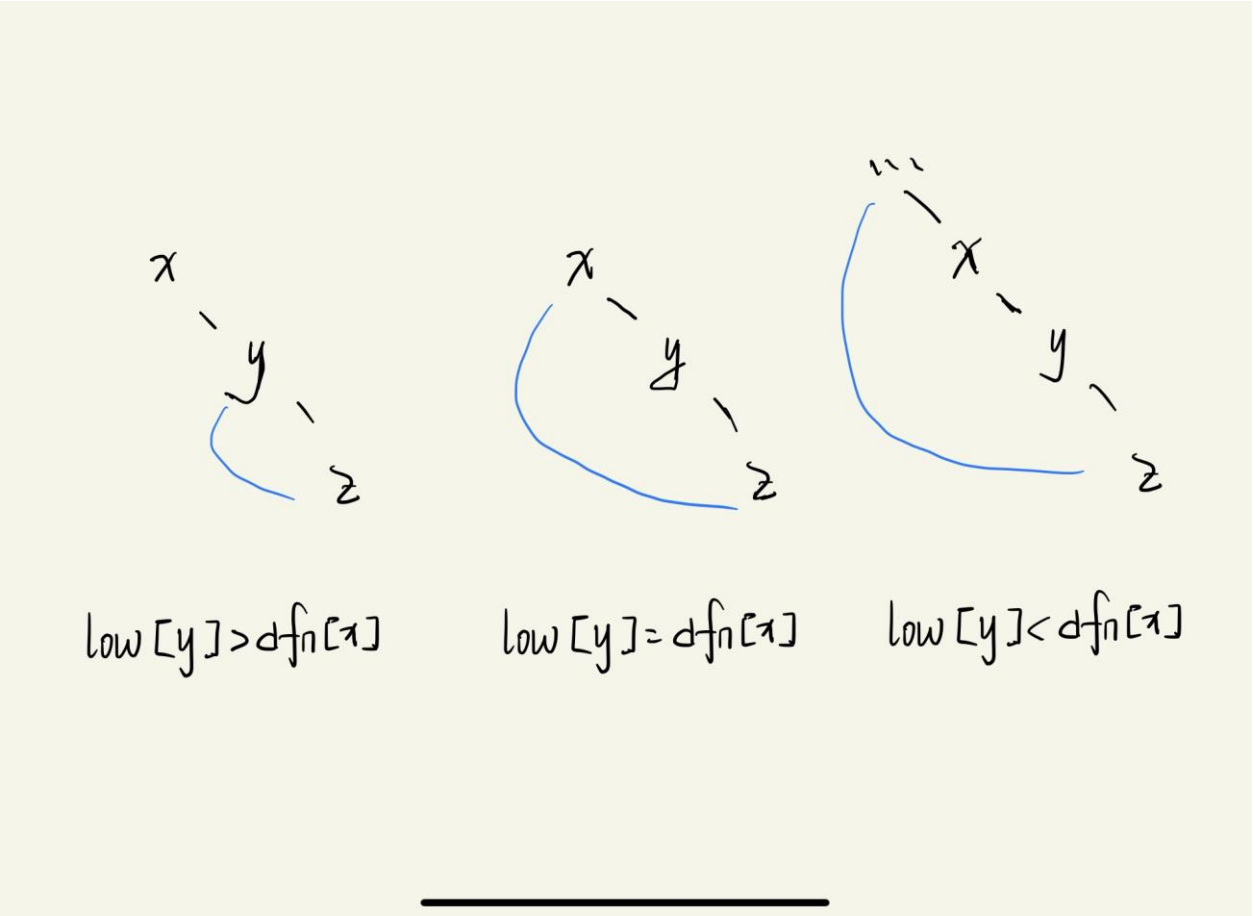
有>=2个儿子





x->y是桥：

low[y] > dfn[x]



- https://www.luogu.com.cn/problem/P1656

A 国派出将军uim，对 B 国进行战略性措施，以解救涂炭的生灵。

B 国有 n 个城市，这些城市以铁路相连。任意两个城市都可以通过铁路直接或者间接到达。

uim 发现有些铁路被毁坏之后，某两个城市无法互相通过铁路到达。这样的铁路就被称为 key road。

uim 为了尽快使该国的物流系统瘫痪，希望炸毁铁路，以达到存在某两个城市无法互相通过铁路到达的效果。

然而，只有一发炮弹（A 国国会不给钱了）。所以，他能轰炸哪一条铁路呢？

输入格式:

第一行 n,m(1≤n≤150，1≤m≤5000)，分别表示有 n 个城市，总共 m 条铁路。

以下 m 行,每行两个整数 a,b，表示城市 a 和城市 b 之间有铁路直接连接。

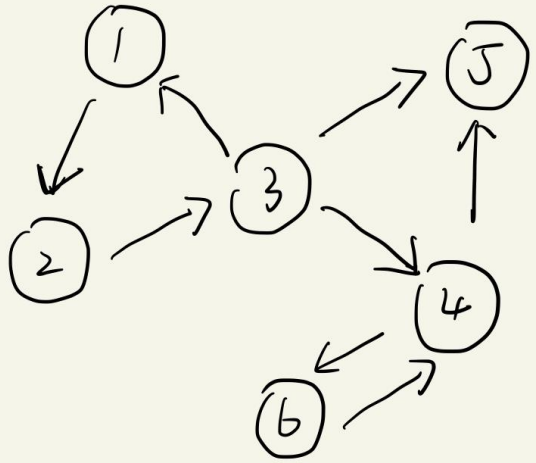
输出格式:

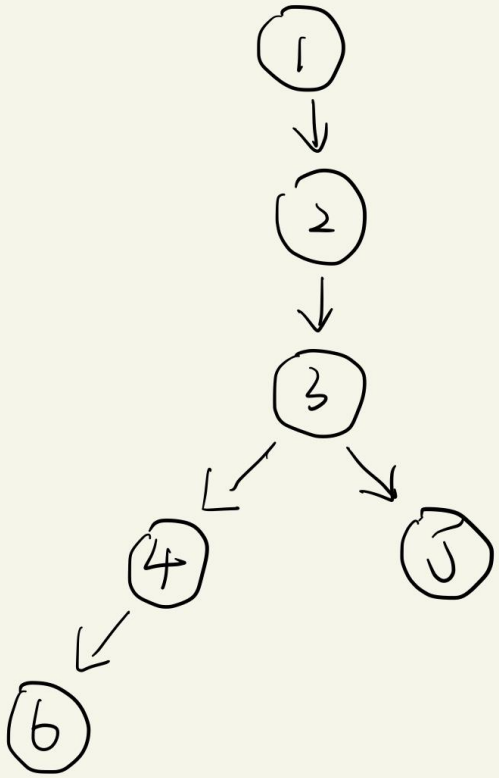
输出有若干行。

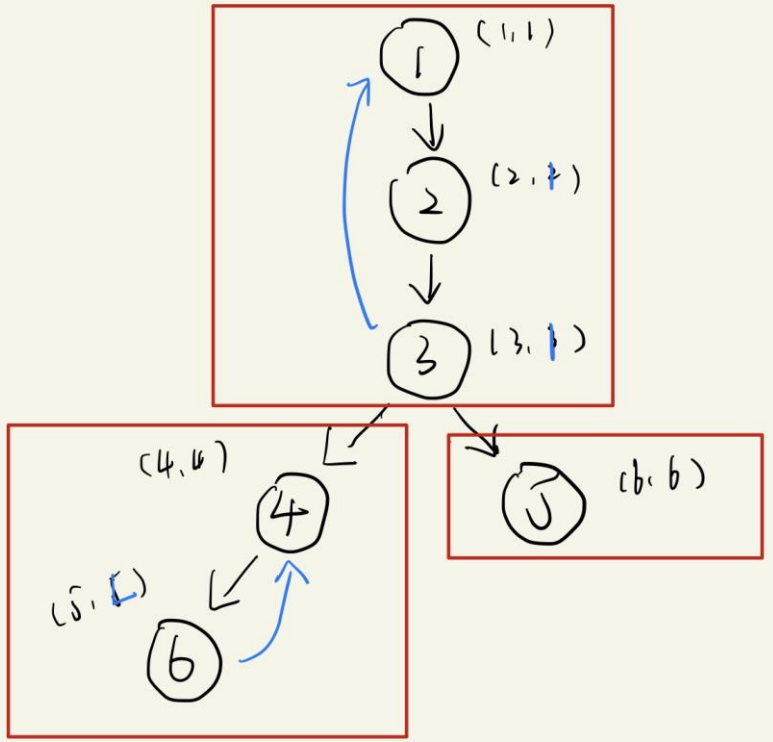
每行包含两个数字 a，b，其中 a<b，表示 <a,b> 是 key road。

请注意：输出时，所有的数对 <a,b>必须按照 a 从小到大排序输出；如果a 相同，则根据 b 从小到大排序。

case(2) 强连通分量







- https://www.luogu.com.cn/problem/P2746

一些学校连入一个电脑网络。那些学校已订立了协议：每个学校都会给其它的一些学校分发软件（称作“接受学校”）。注意即使 B在 A学校的分发列表中，A也不一定在 B学校的列表中。

你要写一个程序计算，根据协议，为了让网络中所有的学校都用上新软件，必须接受新软件副本的最少学校数目（子任务 A）。更进一步，我们想要确定通过给任意一个学校发送新软件，这个软件就会分发到网络中的所有学校。为了完成这个任务，我们可能必须扩展接收学校列表，使其加入新成员。计算最少需要增加几个扩展，使得不论我们给哪个学校发送新软件，它都会到达其余所有的学校（子任务 B）。一个扩展就是在一个学校的接收学校列表中引入一个新成员。

输入格式

输入文件的第一行包括一个正整数 N，表示网络中的学校数目。学校用前 N 个正整数标识。

接下来 N 行中每行都表示一个接收学校列表（分发列表），第 i+1 行包括学校 i 的接收学校的标识符。每个列表用 0 结束，空列表只用一个 0 表示。

输出格式

你的程序应该在输出文件中输出两行。

第一行应该包括一个正整数，表示子任务 A 的解。

第二行应该包括一个非负整数，表示子任务 B 的解。

thanks