2023.8.3 图论题单-上

1. Fox And Names（easy）

【问题描述】

Fox Ciel打算在FOCS（Foxes Operated Computer Systems，发音为“Fox”）上发表一篇论文。她听说一个谣言：论文上的作者列表总是按照字典序排序的。

经过检查一些例子，她发现有时并不总是真的。在一些论文上，作者的名字在常规意义上并不是按照字典序排序的。但是在一些字母顺序的修改后，作者的顺序总是按照字典序排列的！

她想知道是否存在拉丁字母表的字母顺序，使得提交的论文中的名字按照字典序排序。如果存在这样的顺序，您应该找出任何一种满足条件的顺序。

字典序排序定义如下：当我们比较字符串 s 和 t 时，首先找到最左边的不同字符位置：s_i ≠ t_i。如果没有这样的位置（即 s 是 t 的前缀或反之亦然），较短的字符串较小。否则，根据字母表中的顺序比较字符 s_i 和 t_i。

【链接】

https://www.luogu.com.cn/problem/CF510C

【输入格式】

第一行包含一个整数 n (1 <= n <= 100)：表示名字的数量。

接下来的 n 行，每行包含一个字符串 name_i (1 <= |name_i| <= 100)：表示第 i 个名字。每个名字只包含小写拉丁字母。所有名字都是不同的。

【输出格式】

如果存在这样的字母顺序，使得给定的名字按照字典序排序，则输出该顺序作为字符'a'-'z'的排列（即先输出被修改字母表的第一个字母，然后是第二个字母，依此类推）。

否则，输出一个单词 "Impossible"（不带引号）。

【输入样例】

3

rivest

shamir

adleman

【输出样例】

bcdefghijklmnopqrsatuvwxyz

【数据范围】

字符输入的字符串最长为100个字符。

题解

拓扑排序即可

代码

```
#include<bits/stdc++.h>#define int long longusing namespace std;
char ans[105];
string s[105];
queue<char> q;
int n,cnt,in[205];
vector<char> nbr[205];
signed main(){
        cin >> n;
        for(int i=1; i<=n; i++)
        {
                cin >> s[i];
                if (i==1)
                        continue;
                int j;
                for(j=0; j<min(s[i-1].size(),s[i].size()); j++)
                {
                        char c1=s[i-1][j],c2=s[i][j];
                        if (c1!=c2)
                        {
                                nbr[c1].push_back(c2);
                                in[c2]++;
```

```
                                break;
                        }
                }
                if (j==min(s[i-1].size(),s[i].size())&&s[i-1].size()>s[i].size()
                {
                        cout << "Impossible";
                        return 0;
                }
        }
        for(char i='a'; i<='z'; i++)
                if (in[i]==0)
                        q.push(i);
        while(q.empty()==false)
        {
                char cur=q.front();
                q.pop();
                ans[++cnt]=cur;
                for(int i=0; i<nbr[cur].size(); i++)
                {
                        char nxt=nbr[cur][i];
                        in[nxt]--;
                        if (in[nxt]==0)
                                q.push(nxt);
                }
        }
        if (cnt==26)
                for(int i=1; i<=cnt; i++)
                        cout << ans[i];
        elsecout << "Impossible";
        return 0;
}
```

2. Graph and String

【问题描述】

某天，学生Vasya在课堂上发现了一个字符串，由字母"a"、"b"和"c"组成，并写在他的课桌上。由于课程很无聊，Vasya决定通过构造一个满足以下条件的图G来完成这个字符串：

- 图G有恰好n个顶点，编号从1到n。

- 对于所有不同的顶点i和j（其中i≠j），只有当字符si和sj相等或在字母表中相邻时才存在连接它们的边。即，字母对"a"-"b"和"b"-"c"是相邻的，而字母"a"-"c"是不相邻的。

Vasya在字符串旁边画了结果图G，然后抹掉了字符串。第二天，Vasya的朋友Petya来上课，在他的课桌上发现了一个图G。他听说了Vasya的冒险，并想知道这个图G是否可能是Vasya绘制的原始图G。为了验证这一点，Petya需要知道是否存在一个字符串s，如果Vasya使用这个s，他就会产生给定的图G。

【链接】

https://www.luogu.com.cn/problem/CF623A

【输入格式】

第一行包含两个整数n和m，表示图G中的顶点数和边数。

接下来m行，每行包含两个整数ui和vi (1<=ui, vi<=n, ui≠vi)，表示图G中的边。

【输出格式】

如果存在字符串s满足条件，第一行输出"Yes"，第二行输出满足条件的字符串s。

如果不存在这样的字符串s，只在第一行输出"No"。

【输入样例】

2 1

1 2

【输出样例】

Yes

aa

题解

由题意可知，*b*与所有的点都有连边，我们很容易就可以找出*b*，

反过来想，只有*a*,*c*之间有没有连的边，那么我们把图反过来，准确的说是建原图的补图，

如果有符合条件的字符串，那这个补图一定是个二分图，那么对补图进行二分图染色就完成了。

注意一下，在补图中一条边都没有的即为点*b*，我们并不需要对它进行处理，只在最后输出就可以啦。

最后需要检查一下原图有没有问题。

代码

```
#include<iostream>#include<cstdio>#include<ctype.h>#include<cstring>#include<cst
inline int read(){
    int x=0,f=0;char ch=getchar();
    while(!isdigit(ch))f|=ch=='-',ch=getchar();
    while(isdigit(ch))x=x*10+(ch^48),ch=getchar();
    return f?-x:x;
}
int head[507],cnt;
struct Edge{int next,to;
}edge[500007];
bool ma[507][507];//用邻接矩阵存图，方便建补图int col[507];
inline void Byebye(){printf("No\n");exit( ~~(0^_^0) );}//有点闲..inline void add_
```

```
    edge[++cnt].next=head[from];
    edge[cnt].to=to;head[from]=cnt;
}
bool dfs(int x){//二分图染色for(int i=head[x];i;i=edge[i].next){
        int to=edge[i].to;
        if(col[to]==col[x])return 0;
        if(~col[to])continue;col[to]=col[x]^1;
        if(!dfs(to))return 0;
    }
    return 1;
}
int main(){
    int n=read(),m=read();
    for(int i=1;i<=m;++i){
        int u=read(),v=read();
        ma[u][v]=ma[v][u]=1;
    }
    for(int i=1;i<=n;++i)for(int j=1;j<i;++j)//建补图if(!ma[i][j])add_edge(i,j),a
    memset(col,-1,sizeof col);//染色数组置为-1，最后若依旧为-1,即为bfor(int i=1;i<=n
        if(~col[i] || !head[i])continue;
        col[i]=0;
        if(!dfs(i))Byebye();//染色基本套路，若不符合条件输出No，退出程序
    }
    for(int i=1;i<=n;++i)for(int j=1;j<i;++j)
        if(ma[i][j] && col[i]+col[j]==1)Byebye();//检查a,c之间是否有连边printf("Ye
    for(int i=1;i<=n;++i)putchar(~col[i]?!col[i]?'a':'c':'b');//输出一种即可retur
}
```

3. Just one（easy）

【题目描述】

给定一个 $n\left(n \leq 2 * 10^5\right)$ 个点, $m\left(m \leq 2 * 10^5\right)$ 条边的无向图 (无自环和重边) 。你需要为每条边赋一个方向。使得对于每一个点来说, 有且仅有一条出边。

问最后可以产生多少种不同的有向图。

【链接】

https://www.luogu.com.cn/problem/AT_abc226_e

【输入】

第一行n，m

接下来m行，每行表示一条边

【输出】

一个数字，表示方案数对998244353取模后的结果

【输入样例】

3 3
1 2
1 3
2 3

【输出样例】

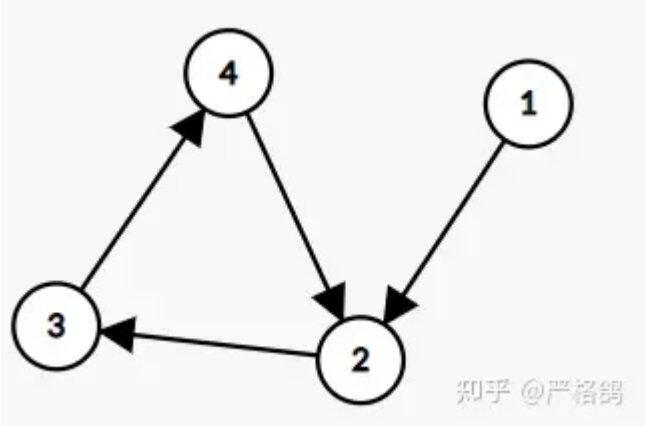
2

题解

每个点只有一条出边，所以对于一个联通块来说，边的数量必须等于点的数量。

> 产生一个出度的同时一定会产生一个入度，一个点的个数为n的联通块，其入度出度之和为2n，也就是有n条边

而n个点n条边，一定会产生一个环，所以一个联通块只能构造出两种不同的图。环上逆时针/顺时针各一种。



这是顺时针

所以我们直接并查集就可以了。

代码

```
#include<bits/stdc++.h>using namespace std;
const int maxn=1000000,mod=998244353;
vector<int>e[maxn];int vis[maxn],cnt;
void dfs(int u,int f){
        vis[u]=1;
        for(int v:e[u]){
                if(v!=f){
                        if(!vis[v]){
                                dfs(v,u);
```

```
                        }
                        else{
                                cnt++;
                        }
                }
        }
}
int main(){
        int n,m;
        cin>>n>>m;
        for(int i=1;i<=m;i++){
                int u,v;
                cin>>u>>v;
                e[u].push_back(v);
                e[v].push_back(u);
        }
        int ans=1;
        for(int i=1;i<=n;i++){
                if(!vis[i]){
                        cnt=0;
                        dfs(i,0);
                        // cout<<cnt<<"\n";
                        ans=(ans*(cnt==2?2:0))%mod;
                }
        }
        cout<<ans;
        return 0;
}
```

4. 灾难（medium）

【问题描述】

阿米巴和小强在草原上捉蚂蚱。小强突然想，如果蚂蚱被他们捉灭绝了，那么吃蚂蚱的小鸟就会饿死，而捕食小鸟的猛禽也会跟着灭绝，从而引发一系列的生态灾难。学过生物的阿米巴告诉小强，草原是一个极其稳定的生态系统。如果蚂蚱灭绝了，小鸟照样可以吃别的虫子，所以一个物种的灭绝并不一定会引发重大的灾难。

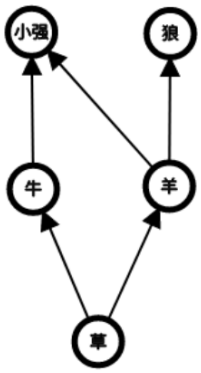
我们现在从专业一点的角度来看这个问题。我们用一种叫做食物网的有向图来描述生物之间的关系：

- 一个食物网有 n 个点，代表 n 种生物，生物从 1 到 n 编号。
- 如果生物 x 可以吃生物 y，那么从 y 向 x 连一个有向边。
- 这个图没有环。

- 图中有一些点没有连出边，这些点代表的生物都是生产者，可以通过光合作用来生存。
- 而有连出边的点代表的都是消费者，它们必须通过吃其他生物来生存。
- 如果某个消费者的所有食物都灭绝了，它会跟着灭绝。

我们定义一个生物在食物网中的“灾难值”为，如果它突然灭绝，那么会跟着一起灭绝的生物的种数。

举个例子：在一个草场上，生物之间的关系如下



如果小强和阿米巴把草原上所有的羊都给吓死了，那么狼会因为没有食物而灭绝，而小强和阿米巴可以通过吃牛、牛可以通过吃草来生存下去。所以，羊的灾难值是 1。但是，如果草突然灭绝，那么整个草原上的 5 种生物都无法幸免，所以，草的灾难值是 4。

给定一个食物网，你要求出每个生物的灾难值。

$1 \leqslant n \leqslant 65534$

【链接】

https://www.luogu.com.cn/problem/P2597

【输入格式】

第一行有一个整数，表示食物网的结点个数 n。

第 2 到第 $(n+1)$ 行，每行若干个互不相同的整数，第 $(i+1)$ 行的整数 $a_{i,j}$ 表示编号为 i 的生物可以吃编号为 $a_{i,j}$ 的生物。每行结尾有一个整数 0 表示本行结束。

【输出格式】

输出 n 行，每行一个整数，第 i 行输出编号为 i 的生物的灾难值。

【样例输入】

```
5
0
1 0
1 0
2 3 0
2 0
```

【样例输出】

4
1
0
0
0

题解

首先，我们可以发现什么时候一种动物不会灭绝，就是有一条路从生产者到达这个点，这样说明这种动物没有灭绝。因为只能假设灭绝一种动物，所以一旦动物A跟着动物B灭绝，那么B一定是A所有通往生产者路径上的交集，也就是必经点。

由于有多个生产者，为了方便我们引入超级生产者，所有的生产者都以他为食，所以必经点就是到超级点路径上的必经点。

必经点具有传递性，A的必经点是B，B的必经点是C，那么C一定是A的必经点。这样，我们只要考虑离某个点最近的必经点即可，这种传递关系形成了一棵树，即支配树，或必经点树。

具有传递性的性质通常可以用树结构来表示。

我们考虑如何计算一个点的必经点，这个必经点一定是所有它能到达的点的必经点。也就是所有食物的在支配树上的lca。这就需要维护动态加点的lca，显然倍增都是支持单端加入和删除的，这样就可以找到每个点的lca，前提是所有食物的必经点都计算好了。

一个点要被计算，连向它的点必须先被计算，这是典型的拓扑排序。排过序后顺序加入即可。

代码

```
#include<iostream>
#include<cstdio>
#include<algorithm>
#include<cstring>
#include<cmath>
#include<set>
#include<vector>
#include<ctime>
#define ll long long
#define pr(x) cerr<<#x<<"="<<x<<endl
using namespace std;
#define N 700000
struct node
{
 int to,from,next;
}e[N],e2[N];
int size,g[N],g2[N],size2,st[N],t,top[N],n,cnt,tot,du[N],ans[N],i,o,f,l;
```

```
int dep[N],fa[N][20];
void add(int o,int p)
{
        e[++size].to=p;
        e[size].from=o;
        e[size].next=g[o];
        g[o]=size;
}
void add2(int o,int p)
{
        e2[++size2].to=p;
        e2[size2].from=o;
        e2[size2].next=g2[o];
        g2[o]=size2;
}
void tuopu()
{
        st[++t]=n+1;
 while (t)
        {
  int x=st[t--];
                top[++cnt]=x;
  for (int k=g[x];k;k=e[k].next)
                {
   int y=e[k].to;
                        du[y]--;
   if (du[y]==0) st[++t]=y;
                }
        }
}
int lca(int x,int y)
{
 if (dep[x]<dep[y]) swap(x,y);
 int d=dep[x]-dep[y];
 for (int i=0;i<=18;i++)
 if ((1ll<<i)&d) x=fa[x][i];
 if (x==y) return x;
 for (int i=18;i>=0;i--)
        {
  if (fa[x][i]!=fa[y][i])
                {
                        x=fa[x][i];
                        y=fa[y][i];
                }
        }
 return fa[x][0];
}
```

```
void dfs(int x)
{
        tot++;
        ans[x]-=tot;
 for (int k=g2[x];k;k=e2[k].next)
        {
  dfs(e2[k].to);
        }
        ans[x]+=tot;
}
int main()
{
 scanf("%d",&n);
 for (i=1;i<=n;i++)
        {
  while(1)
                {
   scanf("%d",&o);
   if (o==0) break;
   add(o,i);
                        du[i]++;
                }
        }
 for (i=1;i<=n;i++)
 if (du[i]==0) add(n+1,i),du[i]++;
 tuopu();
 //for (i=1;i<=n+1;i++) printf("top[%d]=%d\n",i,top[i]);
 memset(g,0,sizeof(g));
 for (i=1;i<=size;i++)
        {
  swap(e[i].to,e[i].from);
                e[i].next=g[e[i].from];
                g[e[i].from]=i;
        }
 for (i=2;i<=n+1;i++)
        {
  int x=top[i];f=1;
  for (int k=g[x];k;k=e[k].next)
                {
   if (f==1) f=0,l=e[k].to;
   else
                        {
                                l=lca(l,e[k].to);
                        }
                }
                fa[x][0]=l;
                dep[x]=dep[l]+1;
```

```
112    //printf("fa-->son %d-->%d\n",l,x);
113    for (int k=1;k<=18;k++)
114                  {
115                          fa[x][k]=fa[fa[x][k-1]][k-1];
116                  }
117    add2(l,x);
118          }
119   dfs(n+1);
120   for (i=1;i<=n;i++)
121          {
122    printf("%d\n",ans[i]);
123          }
124   return 0;
125  }
```

7. King Bombee

【问题描述】

给定一个包含N个顶点、M条边的简单无向图。顶点从1到N编号，边从1到M编号，第i条边连接顶点Ui和顶点Vi。给定整数N、M、K、S、T、X。计算满足以下条件的数列A=(A0, A1, ..., AK)有多少种：

- Ai是介于1和N（包含1和N）之间的整数。

- A0 = S，AK = T。

- 数列A中相邻元素Ai和Ai+1之间有一条直接连接的边。

- 整数X（X ≠ S，X ≠ T）在数列A中出现偶数次（可以为零）。

由于答案可能很大，输出答案对998244353取模的结果。

【链接】

https://www.luogu.com.cn/problem/AT_abc244_e

【输入格式】

输入以以下形式从标准输入给出：

N M K S T X

U1 V1

U2 V2

...

UM VM

【输出格式】

输出一个整数，表示满足条件的数列A的种数对998244353取模后的结果。

【输入样例】

4 4 4 1 3 2
1 2
2 3
3 4
1 4

【输出样例】

4

【数据范围】

约束条件：

- 2 ⩽ N ⩽ 2000

- 1 ⩽ M ⩽ 2000

- 1 ⩽ K ⩽ 2000

- 1 ⩽ S, T, X ⩽ N

- S ≠ X ≠ T

- 1 ⩽ Ui < Vi ⩽ N

- 对于所有i ≠ j，(Ui, Vi) ≠ (Uj, Vj)

题解

f[i][j][k] 走了i步停在j点经过X点的奇偶性为k（0或者1)的路径数。
初始化
f[0][s][0] = 1 ;
转移
对于k步枚举所有边，如果是X点就改变k的奇偶性，不是就直接加。

代码

```cpp
#include <bits/stdc++.h>
//#include <unordered_map>
//priority_queue
#define PII pair<int,int>
#define ll long long

using namespace std;

const  int  INF =  0x3f3f3f3f ;
const  int  N =  200100 ;
```

```cpp
const int mod = 998244353 ;
vector <PII> sk ;
long long f[2010][2010][3] ;
void solve()
{
    int n , m , k , s , t , x ;
    cin >> n >> m >> k >> s >> t >> x ;
    for (int i = 1 ; i <= m ; i++ )
    {
        int t1 , t2 ;
        cin >> t1 >> t2 ;
        sk.push_back({t1,t2}) ;
    }
    f[0][s][0] = 1 ;
    for (int i = 1 ; i <= k ; i++ )
    {
        for ( auto j : sk )
        {
            for (int u = 0 ; u <= 1 ; u++ )
            {
                if ( j.first == x )
                {
                    f[i][j.first][u] = (f[i][j.first][u] + f[i-1][j.second][(u+1
                    f[i][j.second][u] = (f[i][j.second][u] + f[i-1][j.first][u]
                }else if ( j.second == x )
                {
                    f[i][j.second][u] = (f[i][j.second][u] + f[i-1][j.first][(u+
                    f[i][j.first][u] = (f[i][j.first][u] + f[i-1][j.second][u] )
                }
                else
                {
                    f[i][j.first][u] = (f[i][j.first][u] + f[i-1][j.second][u] )
                    f[i][j.second][u] = (f[i][j.second][u] + f[i-1][j.first][u]
                }
            }
        }
    }
    cout << f[k][t][0] << "\n" ;
}
int main()
{
    ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);
        solve() ;
    return 0 ;
}
```