拓扑排序

拓扑排序：是指将有向无环图（DAG）G 中的所有顶点排成一个线性序列，使得图中任意一对顶点 u 和 v，如果有∈E， 则 u 在线性序列中出现在 v 之前，这样的序列称为拓扑序列。

注意：

1、若将图中顶点按拓扑序排成一样，图中所有有向边是从 左指向右

2、若有向图有环，则没有拓扑序列

3、一个 DAG 的拓扑序列通常表示某种方案可行

4、一个 DAG 如果有多个拓扑序列称为没有严格拓扑序，如果 只有一个拓扑序列称为存在严格拓扑序。

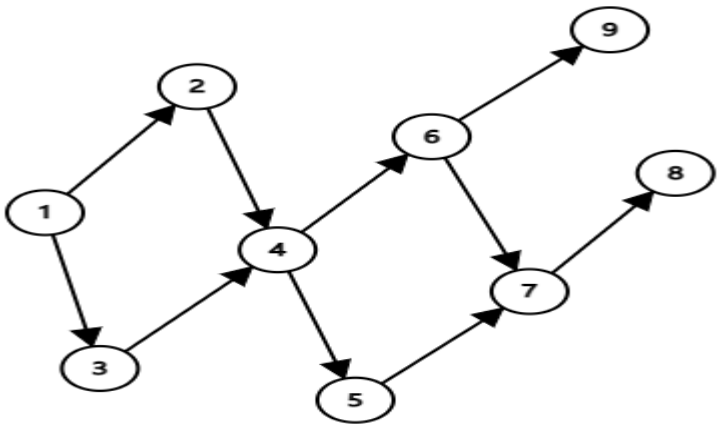
拓扑排序算法思想 ：

(1)在有向图中选取一个入度为 0 的结点输出；

(2)在图中删除该顶点和所有与它连接的边；

重复上面两个步骤，直到所有顶点输出或没有前驱顶点为止。

如下图：



请写出一种拓扑序列：_____

观察上图发现，该图存在多种拓扑序列，我们称这样的 DAG 图具有非严格拓扑序列，如果图只有唯一的拓扑序，我们称该图有严格拓扑序列。

例一：P3074．士兵排队问题

题目描述：有 N 个士兵（1<=N<=100),编号依次为 1,2,...,N.队列训练时，指挥官要把士兵从高

到矮排成一行，但指挥官只知道“1 比 2 高，7 比 5 高”这样的比较结果。如果不行输出"No answer"

分析:士兵的身高关系对应一张有向图,图中的顶点对应一个士兵,有向边<vi,vj>表示士兵 i 高于士兵 j。我们按照从高到矮将士兵排出一个线性的顺序关系,即为对有向图的顶点进行拓扑排序。

参考代码：

```
#include<iostream>
using namespace std;
int a[1001],into[1001],map[1001][1001],n,m;
bool topsort(){
```

```
    for(int i=1;i<=n;i++)  {
        int j=1;
        while((into[j]!=0)&&(j<=n))
            j++;
        if(j>n)
            return false;
        else{
            a[i]=j;
            into[j]=0xffffffff;
            for(int k=1;k<=n;k++)
                  if(map[j][k]!=0)into[k]--;
        }
   }
    return true;
}
int main(){
    cin>>n>>m;
    while(m--){
        int i,j;
        cin>>i>>j;
        map[i][j]=1;into[j]++;
    }
    if(topsort())
        for(int i=1;i<=n;i++)cout<<a[i]<<" ";
    else
        cout<<"No answer";
  return 0;
}
```

例二：P3075 奖金

【题目描述】YY 总经理 XXX 心情好，决定给每位员工发奖金。公司决定以每个人本年在公司的贡献为标准来计算他们得到奖金的多少。

于是 XXX 下令召开 m 方会谈。每位参加会谈的代表提出了自己的意见：“我认为员工 a 的奖金应该比 b 高!” XXX 决定要找出一种奖金方案，满足各位代表的意见，且同时使得总奖金数最少。每位员工奖金最少为 100 元。

【解析】首先构图，若存在条件“a 的钱比 b 多”则从 b 引一条有向指向 a；然后拓扑排序，若无法完成排序则表示问题无解（存在圈）；若可以得到完整的拓扑序列，则按序列顺序进行递推：

设 f[i]表示第 i 个人能拿的最少奖金数；

首先所有 f[i]=100（题目中给定的最小值）；

然后按照拓扑顺序考察每个点 i，若存在有向边(j,i)，则表示 f[i]必须比 f[j]大，因此我们令 f[i] = Max { f[i] , f[j]+1 }即可；

递推完成之后所有 f[i]的值也就确定了，而答案就等于 f[1]+…+f[n]。

【参考代码】

```
#include<bits/stdc++.h>
using namespace std;
struct Edge{
    int v,nxt;
}e[20010];
int cnt=0,first[10010]={},f[10010]={},ru[10010]={};
void add(int u,int v){
    e[++cnt].v=v;
    e[cnt].nxt=first[u];
    first[u]=cnt;
}
int n,m;
int topsort(){
    queue<int>q;int tot=0;
    for(int i=1;i<=n;i++)
        if(!ru[i]){
            q.push(i);    f[i]=100;tot++;
        }
    while(!q.empty()){
        int u=q.front();q.pop();
        for(int i=first[u];i;i=e[i].nxt){
            int v=e[i].v;
            ru[v]--;
            f[v]=max(f[v],f[u]+1);
            if(ru[v]==0){
                q.push(v);tot++;
            }
        }
    }
    if(tot<n)return 0;
    else return 1;
}
int main(){
    cin>>n>>m;
    for(int i=1;i<=m;i++){
        int x,y;
        cin>>x>>y;
        add(y,x);ru[x]++;
    }
    if(topsort()){
        int ret=0;
        for(int i=1;i<=n;i++)ret+=f[i];
        cout<<ret;
    }
```

```
    else cout<<"Poor Xed";
}
```

例三： J2013D 车站分级

一条单向的铁路线上，依次有编号为 1，2，…，n 的 n 个火车站。每个火车站都有一个级别，最低为 1 级。现有若干趟车次在这条线路上行驶，每一趟都满足如下要求：如果这趟车次停靠了火车站 x，则始发站、终点站之间所有级别大于等于火车站 x 的都必须停靠。（注意：起始站和终点站自然也算作事先已知需要停靠的站点）现有 m 趟车次的运行情况（全部满足要求），试推算这 n 个火车站至少分为几个不同的级别

【解析】车停靠站的级别总是高于未停靠站的，类似于拓扑排序里的先后顺序。

• 假设一列火车的起点站为 s，终点站为 e，则途中的所有停靠站向未停靠站连一条边，表示停靠站的级别高于未停靠站，这样处理之后，就得到一个 AOV 图。

• 然后做拓扑，这里拓扑变下形，需要每一次处理所有入度为 0 的点，处理次数就是答案。其实相当于求一个有向图最长链（最短路变形也可以）

```
#include<bits/stdc++.h>
using namespace std;
#define N 5010
#define M 2000001
int n,m, vis[N],h[N],ru[N],chu[N];
int ans=0,s,cnt;
#define in read()
inline int read(){
    int x=0,f=1;    char ch;
    for(ch=getchar();(ch<'0'||ch>'9') && ch!='-';ch=getchar());
    if(ch=='-') {    f=-1;      ch=getchar();   }
    while(ch>='0' && ch<='9') {
        x=(x<<3)+(x<<1)+ch-'0';ch=getchar();
    }//i=i*10+c-'0'
    return x*f;
}
int first[N],tot=0,dis[N];;
struct edge{
    int v,w,nxt;
}e[M];
inline void add(int u,int v,int w){
    e[++tot].v=v;e[tot].w=w;e[tot].nxt=first[u];first[u]=tot;
}
void readin(){//读入建图，停靠站向不停靠站连边
    n=in;m=in;
    cnt=n;
    for(int i=1;i<=m;i++){
        s=in;
        memset(vis,0,sizeof(vis));
        for(int j=1;j<=s;j++){
```

```
            h[j]=in;
            vis[h[j]]=1;
        }
        cnt++;
        for(int jj=h[1];jj<=h[s];jj++)
            if(vis[jj]){
                add(jj,cnt,1);ru[cnt]++;
            }else{
                add(cnt,jj,0);ru[jj]++;
            }
    }
}
int top( ){
    memset(dis,128,sizeof(dis));
    memset(vis,0,sizeof(vis));
    queue<int>q;
    for(int i=1;i<=n;i++)
        if(ru[i]==0)q.push(i),dis[i]=1;
    while(!q.empty()) {
        int u=q.front();q.pop();   //cout<<u<<endl;
        for(int i=first[u];i;i=e[i].nxt){
            int v=e[i].v;
            dis[v]=max(dis[v],dis[u]+e[i].w);
            ru[v]--;
            if(ru[v]==0)q.push(v);
        }
    }
    int ret=0;
    for(int i=1;i<=n;i++)ret=max(ret,dis[i]);
    return ret;
}
int main(){
    readin();
    ans=top();
    cout<<ans<<"\n";
    return 0;
}
```

例四： P3076 球的序列

【题目描述】

有 n 个球，编号 1-n，现在知道一些重量关系，请你给每个球赋一个重量，使得所有的球的重量满足约束，任意两个球的重量不能相同，且重量也是 1-n。

【解析】

拓扑排序，注意根据题的要求，要先保证 1 号球最轻，如果我们由轻的向重的连边，然后我们依次有小到大每次把重量分给一个入度为 0 的点，那么在拓扑时我们面对多个入度为 0

的点，我们不知道该把最轻的分给谁才能以最快的速度找到 1 号（使 1 号入度为 0），并把当前最轻的分给 1 号。所以我们要由重的向轻的连边，然后从大到小每次把一个重量分给一个入度为 0 的点。这样我们就不用急于探求最小号。我们只需要一直给最大号附最大值，尽量不给小号赋值，这样自然而然就会把轻的重量留给小号。

【参考代码】

```
#include<bits/stdc++.h>
using namespace std;
struct Edge{
    int v,nxt;
}e[20010];
int cnt=0,first[10010]={},f[10010]={},ru[10010]={};
void add(int u,int v){
    e[++cnt].v=v;
    e[cnt].nxt=first[u];
    first[u]=cnt;
}
int n,m;
int topsort(){
    priority_queue<int>q;
    int tot=0;
    int k=n;
    for(int i=1;i<=n;i++)
        if(!ru[i]){
            //cerr<<i<<endl;
            q.push(i);tot++;
        }
    while(!q.empty()){
        int u=q.top();q.pop();//cerr<<u<<endl;
        f[u]=k--;
        for(int i=first[u];i;i=e[i].nxt){
            int v=e[i].v;
            ru[v]--;
            if(ru[v]==0){
                q.push(v);tot++;
            }
        }
    }
    if(tot<n)return 0;
    else return 1;
}
int main(){
    cin>>n>>m;
    for(int i=1;i<=m;i++){
        int x,y;
```

```
        cin>>x>>y;
        add(y,x);ru[x]++;
    }
    if(topsort()){
        int ret=0;
        cout<<"YES\n";
        for(int i=1;i<=n;i++)
            cout<<f[i]<<" ";
    }
    else cout<<"NO";
}
```