Terv

Egy nagyszabású építkezés terve n különböző munka elvégzését írja elő. Minden munkát egy nap alatt lehet elvégezni. Egy napon több munkát is el lehet végezni párhuzamosan, feltéve, hogy a megelőzési feltételt betartjuk. Ez ezt jelenti, hogy vannak olyan előírások, hogy egy adott u munka elvégzése meg kell, hogy előzze más adott v munka elégzését. Tehát a v munkát csak akkor lehet elkezdeni, ha már az u munkát befejeztük.

Feladat

Ijunk olyan programot, amely kiszámítja, hogy a teljes építkezést legkevesebb hány nap alatt lehet befejezni és meg is adja, hogy ehhez mely napokon mely munkákat kell elvégezni!

Bemenet

A standard bemenet első sora két egész számot tartalmaz, az elvégzendő munkák n számát ($1 \leq n \leq 200000$), és a megelőzési előírások m számát ($0 \leq m \leq 1000000$).

Kimenet

A standard kimenet első sora az összes munka elvégzéséhez szükséges napok k számát tartalmazza. Ha a megelőzési előírások miatt nem lehet elvégezni az összes munkát, akkor az első ás egyetlen sorba a 0 számot kell írni. A további k sor mindegyike egy napon elvégzendő munkák sorszámait tartalmazza egy-egy szóközzel elválasztva. Több megoldás esetén bármelyik megadható.
Ha nincs megoldás, akkor az első és egyetlen sor a 0 számot tartalmazza.

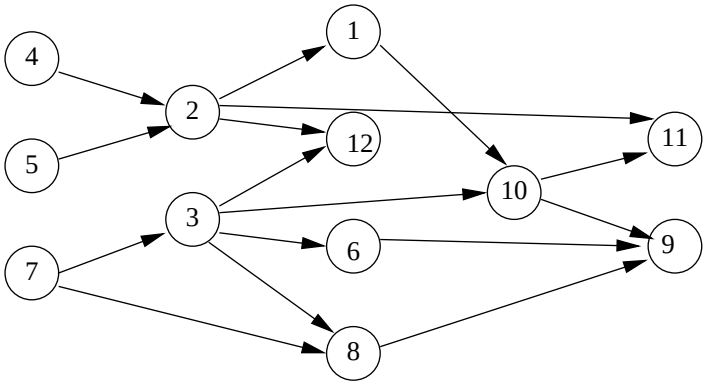
Példa

Bemenet

```
12 16
4 2
5 2
7 3
7 8
2 1
2 11
2 12
3 12
3 10
3 6
3 8
1 10
6 9
8 9
10 11
10 9
```

Kimenet

```
5
4 5 7
2 3
1 12 6 8
10
11 9
```



Korlátok

Időlimit: 0.1 mp.
Memórilimit: 32 MiB
Pontozás: a tesztesetek 40%-ában $n < 1000$

Megoldás

Tekintsük azt az irányított $G = (V, E)$ gráfot, amelynek pontjai az elvégzendő munkák, azaz $V = \{1, \ldots, n\}$, és (u, v) akkor és csak akkor él a gráfban, ha az u munkát előbb kell elvégezni, mint a v munkát.
A feladatnak akkor és csak akkor van megoldása, ha G körmentes. Jelölje $BeFok[p]$ azon q pontok számát, amelyre (q, p) él a gráfban.
Nyilvánvaló, hogy mindazon p pontok, amelyekre $BeFok[p] = 0$ elvégezhetők az első napon. Vegyük azt a $\overline{G}$ gráfot, amelyet úgy kapunk G-ből, hogy töröljük G 0-befokú pontjait (és az ilyen pontokhoz kapcsolódó éleket). Ha $\overline{G}$ opztimális megoldása k nap, akkor G megoldható $k + 1$ nap alatt. G nem oldható meg k+1-nél kevesebb nap, mert akkor lenne $\overline{G}$-nak k-nal kisebb megoldása.
Az alábbi algoritmus egy lehetséges megoldása a problémának.

```
hany:=1; van:=n
ciklus i=1-töl n-ig
    ha BeFok[i]=0 akkor
        tegyük i-t a Nap[1]halmazba
    elágazás vége
ciklus vége

ciklus amíg Nap[hany] <> Üres
    Nap[hany+1]:=Üres
    ciklus Nap[hany] minden p elemére
        van:=van-1
        ciklus minden olyan q elemre, amelyre (p,q) él a gráfban
            BeFok[q]:=BeFok[q]-1
            ha BeFok[q]=0 akkor
                tegyük q-t az Nap[hany+1] halmazba
            elágazás vége
        ciklus vége
    ciklus vége
    hany:=hany+1
ciklus vége
ha van<>0 akkor
    KiÍr(0)
egyébként
    KiÍr(hany-1)
    ciklus i=1-töl hany-1-ig
        ciklus Nap[i] minden p elemére
            KiÍr(p)
        ciklus vége
        KiÍr(újsor)
    ciklus vége
elágazás vége
```

Megvalósítás C++ nyelven

```
1  #include <iostream>
2  #include <vector>
3  #define maxN 100001
4  using namespace std;
5  vector<int> G[maxN];
6  int BeFok[maxN];
7  int n;
8
9  void Beolvas(){
10    int m,p,q;
11    cin>>n>>m;
12    for(int i=1;i<=n;i++) BeFok[i]=0;
13    for (int i=0;i<m;i++){
14       cin>>p>>q;
15       G[p].push_back(q);
16       BeFok[q]++;
17    }
18 }
19
20 int main(){
21    Beolvas();
22    vector<int> Nap[n];
23    int hany=0, van=n;
24    for (int p=1;p<=n;p++)
25       if(BeFok[p]==0)
26          Nap[0].push_back(p);
27
28    while(Nap[hany].size()>0){
29       for(int p:Nap[hany]){
30          van--;
31          for (int q:G[p]){
32             BeFok[q]--;
33             if(BeFok[q]==0)
34                Nap[hany+1].push_back(q);
35          }
36       }//for p
37       hany++;
38    }
39    if(van>0)
40       cout<<hany<<endl;
41    else{
42       cout<<hany<<endl;
43       for(int i=0;i<hany; i++){
44          for (int p:Nap[i])
45             cout<<p<<"␣";
46          cout<<endl;
47       }
48    }
49  return 0;
50  }
```