拓扑排序题目选讲

NKOJ1506 比武定工资

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个人能拿的最少工资数；
首先所有f[i]=100（题目中给定的最小值）；
然后考察拓扑序列中的每个点i，若在它之前的点j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]。

注意：可能出现m条意见并没有把所有保镖讨论完的情况，也就是某些保镖没有出现在m个人的意见中，那么这些人的工资直接就是100元

车站分级(NKOJ2502)

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）

例如，下表是5趟车次的运行情况。其中，前4趟车次均满足要求，而第5趟车次由于停靠了3号火车站（2级）却未停靠途经的6号火车站（亦为2级）而不满足要求。

车站编号	1		2		3		4		5		6		7		8		9
车站级别 / 车次	3		1		2		1		3		2		1		1		3
1	始	→	→	→	停	→	→	→	停	→	终						
2					始	→	→	→	停	→	终						
3	始	→	→	→	→	→	→	→	停	→	→	→	→	→	→	→	终
4							始	→	停	→	停	→	停	→	停	→	终
5					始	→	→	→	停	→	→	→	→	→	→	→	终

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

输入格式：

第一行包含2个正整数n, m，用一个空格隔开。

第i+1行（1≤i≤m）中，首先是一个正整数s_i（2≤s_i≤n），表示第i趟车次有s_i个停靠站；接下来有s_i个正整数，表示所有停靠站的编号，从小到大排列。每两个数之间用一个空格隔开。输入保证所有的车次都满足要求。

输出格式：

输出只有一行，包含一个正整数，即n个火车站最少划分的级别数。

样例输入1：

```
9 2
4 1 3 5 6
3 3 5 6
```

样例输出1：

```
2
```

样例输入2：

```
9 3
4 1 3 5 6
3 3 5 6
3 1 5 9
```

样例输出2：

```
3
```

分析

如何分级?

如果两个点(A和B)被同一辆车经过,并且车停在了A,没有停在B,那么A的等级一定比B高。

根据这个,把等级高的向等级低的连一条边,表示它们不能在同一个等级.这个做法还是很好实现的,而且代码也很短.只需在读入时记录停留的点和没停留的点,把停留的点向所有没停留的点连一条边就好了。

只要算出不同等级的数量,就得到了答案。

不同等级的数量怎么算?

回想拓扑排序入栈的过程：

初始，我们将入度为0的点入栈，显然它们是同一个等级……

当第一个等级的节点都依次弹出删除，产生的新的一批入度为0的节点是第二个等级……

重复以上的过程，我们就能记录出最少需要的等级数。

```cpp
int n, m, s;
int g[maxn][maxn], vis[maxn], lst[maxn], ind[maxn];
int ans;
stack<int> s1;
void toposort(){
    for(int i = 1; i <= n; i++) { //入度为0的节点入栈
        if(ind[i] == 0) s1.push(i);
    }
    while(!s1.empty()){
        ans++; //记录操作数
        stack<int> s2;
        while(!s1.empty()){
            int u = s1.top();
            s1.pop();
            for(int v = 1; v <= n; v++) if(g[u][v]){
                ind[v]--;
                if(ind[v] == 0) s2.push(v); //用s2来临时保存新产生的入度为0的点
            }
        }
        s1 = s2;//更新s1，进行下一轮操作
    }
}
int main() {
    cin >> n >> m;
    for(int i = 1; i <= m; i++){
        cin >> s;
        memset(vis,0,sizeof(vis));
        for(int j = 1; j <= s; j++) {
            cin >> lst[j];
            vis[lst[j]] = 1;
        }
        for(int j = lst[1]; j <= lst[s]; j++) {//等级低的向等级高的连边
            if(!vis[j]) {
                for(int k = 1; k <= s; k++) {
                    if(!g[lst[k]][j]) {
                        g[lst[k]][j] = 1;
                        ind[j]++;
                    }
                }
            }
        }
    }
    toposort();
    cout << ans << endl;
```