**拓扑排序**

下面是一个使用 Kahn 算法进行拓扑排序的示例代码：

from collections import deque, defaultdict def topological\_sort(graph):

indegree = defaultdict(int) result = []

queue = deque()

# 计算每个顶点的入度

for u in graph:

for v in graph[u]: indegree[v] += 1

# 将入度为 0 的顶点加入队列

for u in graph:

if indegree[u] == 0: queue.append(u)

# 执行拓扑排序

while queue:

u = queue.popleft() result.append(u)

for v in graph[u]: indegree[v] -= 1

if indegree[v] == 0:

queue.append(v)

# 检查是否存在环

if len(result) == len(graph): return result

else:

return None#所以拓扑排序可以检查有向图是否成环 graph = {

'A': ['B', 'C'],

'B': ['C', 'D'],

'C': ['E'],

'D': ['F'],

'E': ['F'], 'F': []

}

sorted\_vertices = topological\_sort(graph) if sorted\_vertices:

print("Topological sort order:", sorted\_vertices) else:

print("The graph contains a cycle.")

# Topological sort order: ['A', 'B', 'C', 'D', 'E', 'F']

# 008: 最小奖金方案（kahn）

现在有n个队伍参加了比赛，他们进行了m次PK。现在赛事方需要给他们颁奖（奖金为整数），已知参加 比赛就可获得100元，由于比赛双方会比较自己的奖金，所以获胜方的奖金一定要比败方奖金高。请问赛 事方要准备的最小奖金为多少？奖金数额一定是整数。

输入

一组数据，第一行是两个整数n(1≤n≤1000)和m(0≤m≤2000)，分别代表n个队伍和m次pk，队伍编号从0

到n-1。接下来m行是pk信息，具体信息a，b，代表编号为a的队伍打败了编号为b的队伍。输入保证队伍 之间的pk战胜关系不会形成有向环。输出 给出最小奖金w

5 6

1 0

2 0

3 0

4 1

4 2

4 3

505

import collections

n,m = map(int,input().split()) G = [[] for i in range(n)] award = [0 for i in range(n)]

inDegree = [0 for i in range(n)] for i in range(m):

a,b = map(int,input().split())

G[b].append(a)

inDegree[a] += 1#因为所以获胜方的奖金一定要比败方奖金高，所以要增加获胜方的入度来递增奖金

数目

q = collections.deque() for i in range(n):

if inDegree[i] == 0: q.append(i) award[i] = 100

while len(q) > 0:

u = q.popleft() for v in G[u]:

inDegree[v] -= 1

award[v] = max(award[v],award[u] + 1) if inDegree[v] == 0:

q.append(v)

total = sum(award) print(total)

**04084: 拓扑排序**

给出一个图的结构，输出其拓扑排序序列，要求在同等条件下，编号小的顶点在前。

## 输入

若干行整数，第一行有2个数，分别为顶点数v和弧数a，接下来有a行，每一行有2个数，分别是该条弧所 关联的两个顶点编号。 v<=100, a<=500

**输出** 若干个空格隔开的顶点构成的序列(用小写字母)。 样例输入

6 8

1 2

1 3

1 4

3 2

3 5

4 5

6 4

6 5

样例输出

v1 v3 v2 v6 v4 v5

from collections import deque def topo\_sort(n, edges):

neigh = {i: [] for i in range(1, n + 1)}

in\_degree = {i: 0 for i in range(1, n + 1)}

# 处理边，填充邻接表和入度字典

for u, v in edges: neigh[u].append(v) in\_degree[v] += 1

# 找到所有入度为 0 的顶点，作为起始点

queue = deque([v for v in in\_degree if in\_degree[v] == 0]) topo\_order = []

while queue:

u = sorted(queue)[0]#每次都要排序，因为有新增进来的 topo\_order.append(u)

queue.remove(u)

for v in neigh[u]: in\_degree[v] -= 1

if in\_degree[v] == 0: queue.append(v)

# 检查是否所有顶点都被排序

if len(topo\_order) == n: return topo\_order

else:

return []

n, m = map(int, input().split())

edges = [tuple(map(int, input().split())) for \_ in range(m)] result = topo\_sort(n, edges)

for v in result:

print('v' + str(v), end=' ')

# 09202: 舰队、海域出击！（判断成环）

作为一名海军提督，Pachi将指挥一支舰队向既定海域出击！ Pachi已经得到了海域的地图，地图上标识 了一些既定目标和它们之间的一些单向航线。如果我们把既定目标看作点、航线看作边，那么海域就是 一张有向图。不幸的是，Pachi是一个会迷路的提督QAQ，所以他在包含环(圈)的海域中必须小心谨慎， 而在无环的海域中则可以大展身手。 受限于战时的消息传递方式，海域的地图只能以若干整数构成的数 据的形式给出。作为舰队的通讯员，在出击之前，请你告诉提督海域中是否包含环。

## 输入

每个测试点包含多组数据，每组数据代表一片海域，各组数据之间无关。 第一行是数据组数T。 每组数 据的第一行两个整数N，M，表示海域中既定目标数、航线数。 接下来M行每行2个不相等的整数x,y，表 示从既定目标x到y有一条单向航线（所有既定目标使用1~N的整数表示）。 描述中的图片仅供参考，其 顶点标记方式与本题数据无关。

1<=N<=100000，1<=M<=500000，1<=T<=5 注意：输入的有向图不一定是连通的。 输入中的两张图就是描述中给出的示例图片。

拓扑排序检查有向图是否存在环

from collections import deque,defaultdict#用defaultdict避免不联通的情况 def topo\_sort(graph):

in\_degree={u:0 for u in range(1,n+1)} for u in graph:

for v in graph[u]: in\_degree[v]+=1

q=deque([u for u in in\_degree if in\_degree[u]==0])

topo\_order=[] while q:

u=q.popleft()

topo\_order.append(u) for v in graph[u]:

in\_degree[v]-=1

if in\_degree[v]==0: q.append(v)

if len(topo\_order)!=len(graph): return 'Yes'

return 'No'

for \_ in range(int(input())): n,m=map(int,input().split()) graph=defaultdict(list)

for \_ in range(m): u,v=map(int,input().split()) graph[u].append(v)

print(topo\_sort(graph))

# 01094: Sorting It All Out

不同值的升序排序序列是使用某种形式的小于运算符对元素从小到大进行排序的序列。例如，排序序列 A、B、C、D 意味着 A < B、B < C 和 C < D.在本题中，我们将为您提供一组形式为 A < B 的关系，并要 求您确定是否指定了排序顺序。

## 输入

输入由多个问题实例组成。每个实例都以包含两个正整数 n 和 m 的行开头，第一个值表示要排序的对象 数，其中 2 <= n <= 26。要排序的对象将是大写字母表的前 n 个字符。第二个值 m 表示在本问题实例中 将给出的 A < B 形式的关系数。接下来是 m 行，每行包含一个由三个字符组成的此类关系：一个大写字 母、字符“<”和第二个大写字母。任何字母都不会超出字母表的前 n 个字母的范围。n = m = 0 的值表示 输入结束。

4 6

A<B A<C B<C C<D B<D A<B 3 2 A<B B<A

26 1

A<Z 0 0

Sorted sequence determined after 4 relations: ABCD. Inconsistency found after 2 relations.

Sorted sequence cannot be determined.

from collections import deque def topo\_sort(graph):

in\_degree = {u:0 for u in graph} for u in graph:

for v in graph[u]:

in\_degree[v] += 1

q = deque([u for u in in\_degree if in\_degree[u] == 0]) topo\_order = [];flag = True

while q:

if len(q) > 1:

flag = False#topo\_sort不唯一确定 u = q.popleft()

topo\_order.append(u) for v in graph[u]:

in\_degree[v] -= 1

if in\_degree[v] == 0: q.append(v)

if len(topo\_order) != len(graph): return 0 return topo\_order if flag else None

while True:

n,m = map(int,input().split())

if n == graph edges

for i

=

=

0: break

{chr(x+65):[] for x in range(n)} [tuple(input().split('<')) for \_ in range(m)]

in range(m):

a,b = edges[i] graph[a].append(b)

t = topo\_sort(graph) if t:

s = ''.join(t)

print("Sorted sequence determined after {} relations:

{}.".format(i+1,s))

break elif t == 0:

print("Inconsistency found after {} relations.".format(i+1))

break

else:

print("Sorted sequence cannot be determined.")