# 提高算法班

## 二分图、欧拉图、拓扑排序

Mas

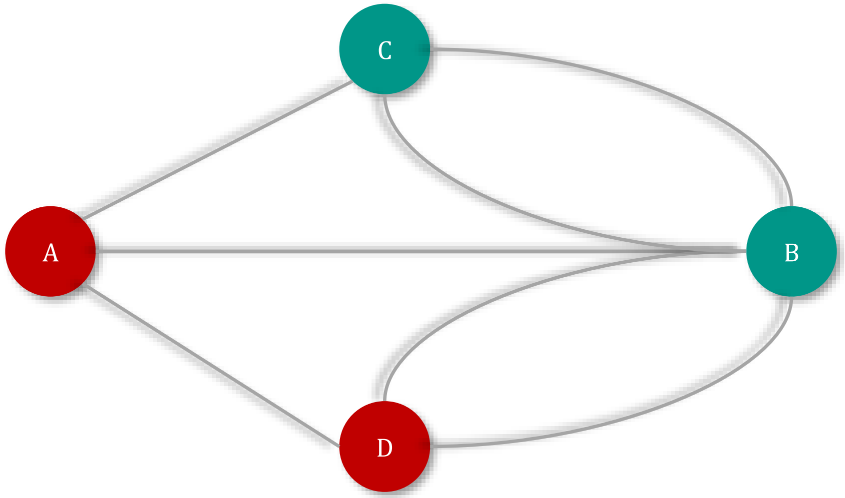
# 欧拉图

1736年 29 岁的数学家欧拉来到普鲁士的古城哥尼斯堡

普瑞格尔河从市中心流过,河中心有两座小岛,岛和两岸之间建筑有七座古桥

欧拉发现当地居民有一项消遣活动

市民们试图每座桥恰好走过一遍并回到原出发点,但从来没人成功过




若每座桥都恰好走过一次，对于每一个顶点，需要从某条边进入，从另一条边离开

进入/离开顶点的次数是相同的,每个顶点相连的边是成对出现的,即每个顶点的相连边的数量必须是偶数

上图中 $A$、$C$、$D$ 三个顶点的相连边都是 3 ,顶点 $B$ 的相连边为 5 为奇数

因此无法从一个顶点出发,遍历每条边各一次

# 欧拉图

### 欧拉通路

通过图中所有边恰好一次且行遍所有顶点的通路称为欧拉通路

### 欧拉回路

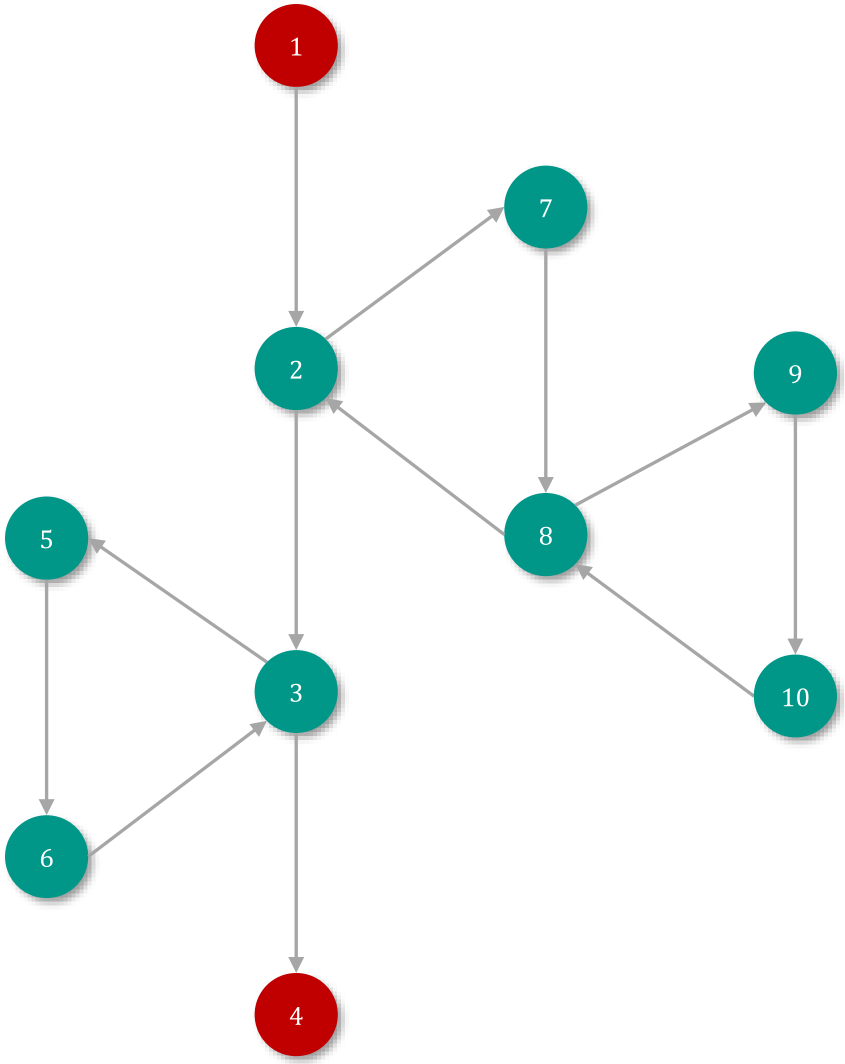
通过图中所有边恰好一次且行遍所有顶点的回路称为欧拉回路

### 欧拉图

具有欧拉回路的无向图或有向图称为欧拉图

### 半欧拉图

具有欧拉通路但不具有欧拉回路的无向图或有向图称为半欧拉图



# 欧拉图

对于无向图 $G$

是欧拉图当且仅当 $G$ 是连通的且没有奇度顶点

对于无向图 $G$

是半欧拉图当且仅当 $G$ 是连通的且恰有 2 个奇度顶点

对于有向图 $G$

是欧拉图当且仅当 $G$ 的所有顶点属于同一个强连通分量且每个顶点的入度和出度相同

对于有向图 $G$，是半欧拉图当且仅当

- 若将 $G$ 中的有向边退化为无向边，那么 $G$ 的所有顶点属于同一个连通分量
- 至多有一个顶点的出度与入度差为 1
- 至多有一个顶点的入度与出度差为 1
- 所有其他顶点的入度和出度相同

# #1245、欧拉路径

## 题目描述

求有向图字典序最小的欧拉路径

> *如果方案* $A$ *比方案* $B$ *大，是指存在* $k \geq 1$ *使得* $A_i = B_i$ *对所有* $i < k$ *成立,并且* $A_k > B_k$

## 输入格式

第一行两个整数 $n, m$ 表示有向图的点数和边数

接下来 $m$ 行每行两个整数 $u, v$ 表示存在一条 $u \to v$ 的有向边

## 输出格式

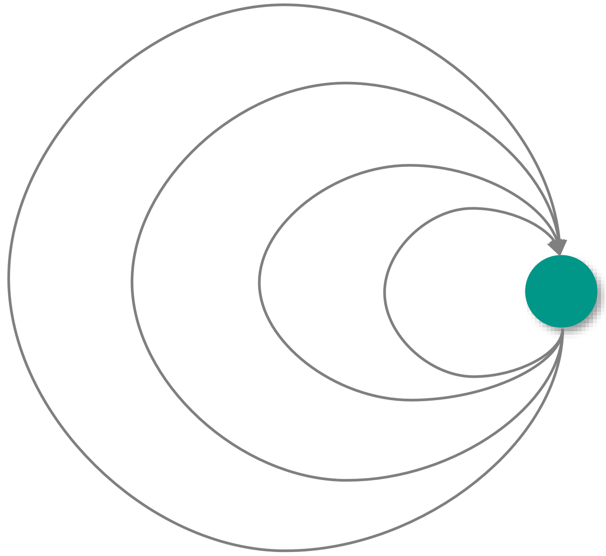
如果不存在欧拉路径,输出一行 `No`

否则输出一行 $m+1$ 个数字,表示字典序最小的欧拉路径

## 数据规模

对于 $100\%$ 的数据, $1 \leq u, v \leq n \leq 10^5, 1 \leq m \leq 2 \times 10^5$

保证将有向边视为无向边后图连通



不可在遍历过程中对点进行标记

# #1245、欧拉路径

从起点开始 DFS

每次从当前节点的边中任选一条将其**删除**

然后进入这条边的另一端继续进行 DFS

最终所有边一定会被经过

且在回溯时记录经过的边的编号,其逆序即为欧拉路/欧拉回路

时间复杂度 $O(n+m)$

要求点编号的字典序最小,对每个点的出边排序即可

若不先删边直接 $DFS$ , 时间复杂度将为 $O(nm)$

```
void dfs(int u)
{
    for (int i = idx[u]; i < g[u].size(); i = idx[u])
    {
        idx[u] = i + 1; //下次遍历u的出边时跳过(等价于删边)
        dfs(g[u][i]);
    }
    ans[++pos] = u;
}
```

# #2494、无序字母对

根据字母对建立无向图

求字典序最小的欧拉路/欧拉回路

## 题目描述

给定 $n$ 个各不相同的无序字母对(区分大小写，无序即字母对中的两个字母可以位置颠倒)

请构造一个有 $(n+1)$ 个字母的字符串使得每个字母对都在这个字符串中出现

## 输入格式

第一行输入一个正整数 $n$

第二行到第 $(n+1)$ 行每行两个字母，表示这两个字母需要相邻

## 输出格式

输出满足要求的字符串

如果没有满足要求的字符串，请输出 `No Solution`

如果有多种方案，请输出**字典序最小的方案**(即满足前面的字母的 $ASCII$ 编码尽可能小)

## 输入样例

```
4
aZ
tZ
Xt
aX
```

## 输出样例

```
XaZtX
```

## 说明/提示

不同的无序字母对个数有限，$n$ 的规模可以通过计算得到

# 二分图

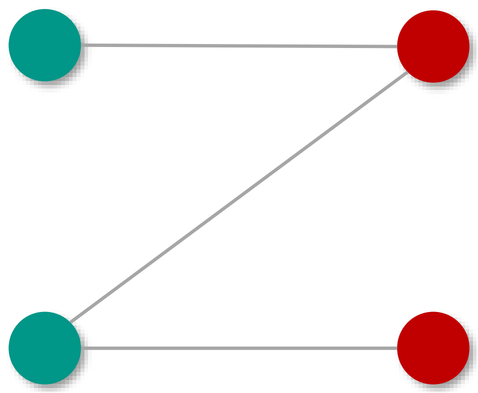
二分图又叫二部图（Bipartite graph）,是图论中的一种特殊模型

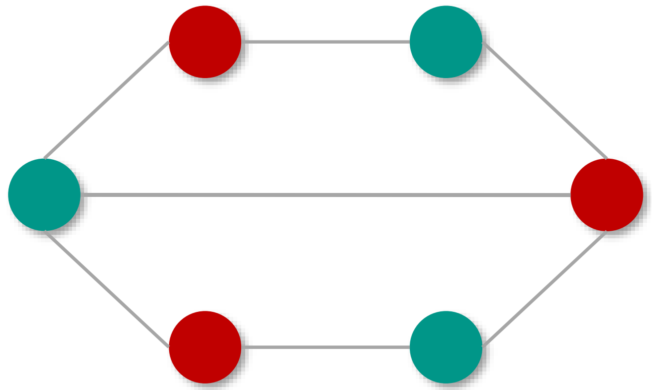
$G = (V, E)$ 是一个无向图

若点集 $V$ 可分为两个互不相交的子集 $A, B$

图中每条边 $(u, v)$ 的两个点 $u$ 和 $v$ 分属于两个不同的顶点集 $(u \in A, v \in B)$

满足上述定义,称图 $G$ 为一个二分图





# 二分图

二分图不存在长度为奇数的环

每一条边都从一个集合走到另一个集合,只有走偶数次才可能回到同一个集合

考虑性质,可以使用 DFS 或者 BFS 来遍历这张图(或种类并查集)

若发现了奇环,那么就不是二分图,否则是

从任意起点 DFS 或者 BFS ,对节点进行染色标记

若相邻节点 $v$ 未被标记颜色,将 $v$ 标记成不相同颜色,将 $v$ 作为新的起点

若相邻节点 $v$ 已被染色,若颜色相同说明存在奇环

# #2415、二分图判定

## 题目描述

给出一张 $n$ 个点 $m$ 条边的无向图,请你判定其是否是一张二分图

也即,是否存在一种对图中每个节点进行黑白染色的方案,使得任意两个相邻的节点拥有不同的颜色

## 输入格式

本题包含多组测试数据,输入的第一行包含一个整数 $T$ 表示数据组数

对于每组测试数据,输入的第一行为两个整数 $n, m$ ,表示图的点数和边数

接下来 $m$ 行,每行两个整数 $x, y$ ,表示图中的一条边 $(x, y)$

## 输出格式

输出共 $T$ 行,每行一个字符串 `Yes` 或 `No` 表示一组数据中的图是不是二分图

## 数据范围

对于 $30\%$ 的数据, $n \le 20, m \le 30$

对于 $50\%$ 的数据, $n \le 2000, m \le 3000$

对于 $100\%$ 的数据, $T \le 15, n \le 2 \times 10^5, m \le 3 \times 10^5$

## 样例输入

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

## 样例输出

```
Yes
No
No
```

# 二分图匹配

## 匹配

对于图 $G = (V, E)$ ,若 $E' \in E$ 且 $E'$ 中任意两条不同的边都没有公共的端点,且 $E'$ 中任意一条边都不是自环

则 $E'$ 是图 $G$ 的一个**匹配** (matching),也可以叫作 **边独立集** (independent edge set)

若一个点是匹配中某条边的一个端点,则称这个点是 **被匹配的** (matched)/饱和的 (saturated)

否则称这个点是 **不被匹配的** (unmatched)

## 最大匹配

边数最多的匹配被称作一张图的 **最大匹配** (maximum − cardinality matching)

图 $G$ 的最大匹配的大小记作 $\nu(G)$

## 完美匹配

所有的点都在匹配边上的匹配

# 二分图匹配

## 交替路

对于一个匹配 M，若一条路径以非匹配点为起点，每相邻两条边的其中一条在匹配中而另一条不在匹配中

则这条路径被称作一条**交替路径** (alternating path)

## 增广路

一条在非匹配点终止的交替路径,被称作一条**增广路径** (agumenting path)

## 增广路性质

- 路径长度必定为奇数,第一条边和最后一条边都不属于 M
- 编号为奇数的一定是非匹配边
- 增广路上非匹配边比匹配边数量多一,

  将边取反(匹配变为未匹配,未匹配变为匹配),匹配大小会增加一且依然是交错路
- M 为 $G$ 的最大匹配当且仅当不存在相对于 M 的增广路径

# Hungarian Algorithm

利用增广路找最大匹配的算法,就叫做匈牙利算法

算法流程

- 置 M 为空
- 找出一条增广路径 P ,通过取反操作获得更大的匹配 M′ 代替 M
- 重复 2 操作直到找不出增广路径为止

增广路长度为奇数,路径起始点非左即右, 先考虑从左边的未匹配点找增广路

增广路上的第奇数条边都是非匹配边,第偶数条边都是匹配边,于是左到右都是非匹配边,右到左都是匹配边

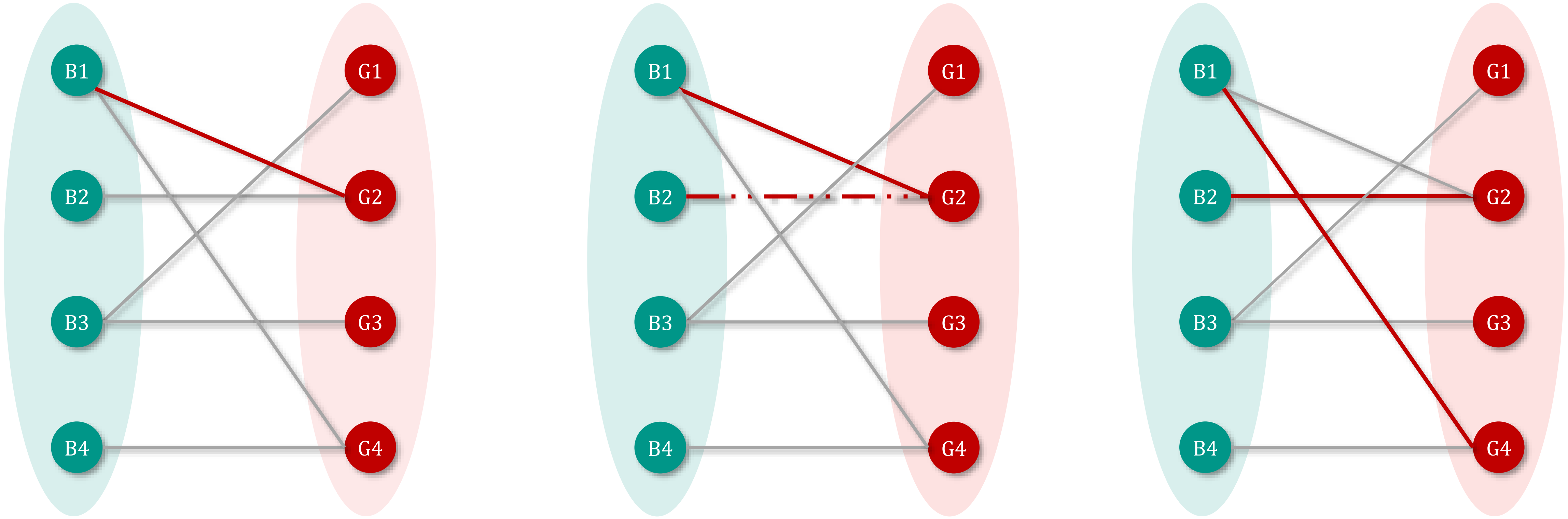
给二分图定向问题转换成

有向图中从给定起点找一条简单路径走到某个未匹配点,此问题等价给定起始点能否走到终点

只需从起始点开始 DFS 遍历直到找到某个未匹配点

# Hungarian Algorithm



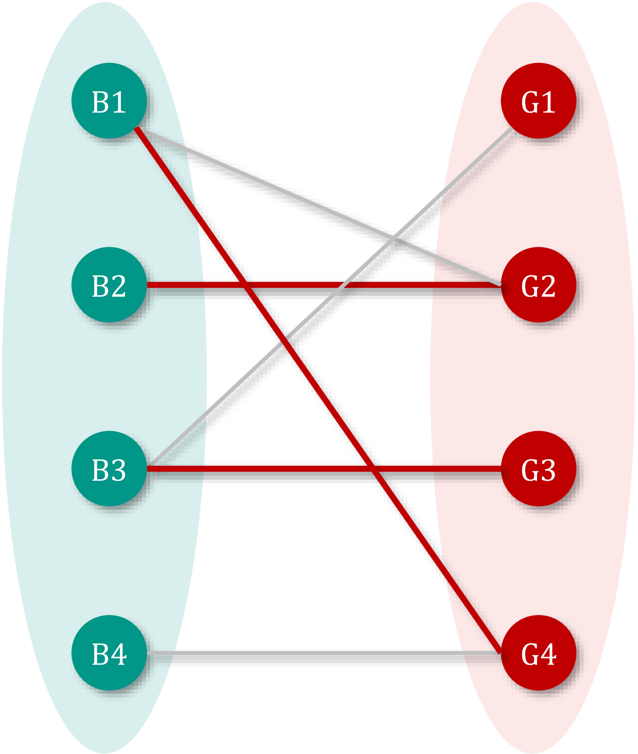



先将 $B1$ 和 $G2$ 匹配

$B2$ 尝试与 $G2$ 匹配,但是 $G2$ 已和 $B1$ 匹配, 倒回去看看 $B1$ 是否有其它选择

发现可以给 $B1$ 安排 $G4$ ,$B2$ 安排 $G2$

# Hungarian Algorithm




```
bool match(int u)
{
  for (int i = head[u]; i; i = e[i].nxt)
    if (!vis[e[i].v])
    {
      vis[e[i].v] = true;
      if (!sel[e[i].v] || match(sel[e[i].v]))
      {
        sel[e[i].v] = u;
        return true;
      }
    }
  return false;
}
```

给 $B_3$ 安排上 $G_1$

$B_4$ 只能选 $G_4$，但给 $B_1$ 重新分配一个已经不可能了

时间复杂度 $O(nm)$

# #2664、二分图的最大匹配

## 题目描述

给定一个二分图,其左部点的个数为 $n_1$ ,右部点的个数为 $n_2$ ,边数为 $e$ ,求其最大匹配的边数

左部点从 $1 \sim n_1$ 编号,右部点从 $1 \sim m_2$ 编号

## 输入格式

输入的第一行是三个整数,分别代表 $n_1, n_2, m$

接下来 $m$ 行,每行两个整数 $u, v$ ,表示存在一条连接左部点 $u$ 和右部点 $v$ 的边

## 输出格式

输出一行一个整数,代表二分图最大匹配的边数

## 数据规模与约定

对于全部的测试点,保证: $1 \leq n_1, n_2 \leq 500, 1 \leq m \leq 5 \times 10^4, 1 \leq u \leq n_1, 1 \leq v \leq n_2$

**不保证给出的图没有重边**

## 输入样例1

```
1 1 1
1 1
```

## 输出样例1

```
1
```

# 二分图匹配

## 点覆盖

对于图 $G = (V, E)$ ,若 $V' \in V$ 且 $\forall e \in E$ 满足 $e$ 至少一个端点在 $V'$ 中,则称 $V'$ 是图 $G$ 的一个 **点覆盖** (vertex cover)

二分图的最小点覆盖 = 二分图的最大匹配

## 边覆盖

对于图 $G = (V, E)$ ,若 $E' \in E$ 且 $\forall v \in V$ 满足 $v$ 与 $E'$ 中至少一条边相连,则称 $E'$ 是图 $G$ 的一个 **边覆盖** (edge cover)

最小边覆盖的大小记作 $\rho(G)$

贪心选一组最大匹配的边放进集合

对于剩下未匹配的点，任选一条关联的边放进集合，得到的集合即最小边覆盖

$$\rho(G) = \nu(G) + |V| - 2\nu(G) = |V| - \nu(G)$$

即

二分图的最少边覆盖 = 点数 - 二分图的最大匹配

# 二分图匹配

## 独立集

对于图 $G = (V,E)$，若 $V' \in V$ 且 $V'$ 中任意两点都不相连，则称 $V'$ 是图 $G$ 的一个**独立集**( independent set )

图 $G$ 最大的独立集的大小记作 $\alpha(G)$

在二分图中, 选最多的点使得任意两个点之间没有直接边连接

二分图的最大独立集 = 点数 - 二分图的最大匹配

把所有的点放进集合，然后删去最少的点和与之相关联的边

使得全部边都被删完(最小点覆盖)

# #2666、舞会

## 题目描述

某学校要召开一个舞会

已知学校所有 $n$ 名学生中,有些学生曾经互相跳过舞

当然跳过舞的学生一定是一个男生和一个女生

在这个舞会上,要求被邀请的学生中的任何一对男生和女生互相都不能跳过舞

求这个舞会最多能邀请多少个学生参加

## 输入格式

输入的第一行是 $n$ 和 $m$

其中 $n$ 是可选的学生的总数, $m$ 是已知跳过舞的学生的对数( $n \leq 1000, m \leq 2000$ )

然后有 $m$ 行,每行包括两个非负整数,表示这两个编号的学生曾经跳过舞

学生的编号从 $0$ 号到 $n-1$ 号

## 输出格式

输出一行一个数字,即能够邀请的最多的学生数

**二分图的最大独立集**

将图染色，对同一种颜色点开始求最大匹配

# 拓扑排序

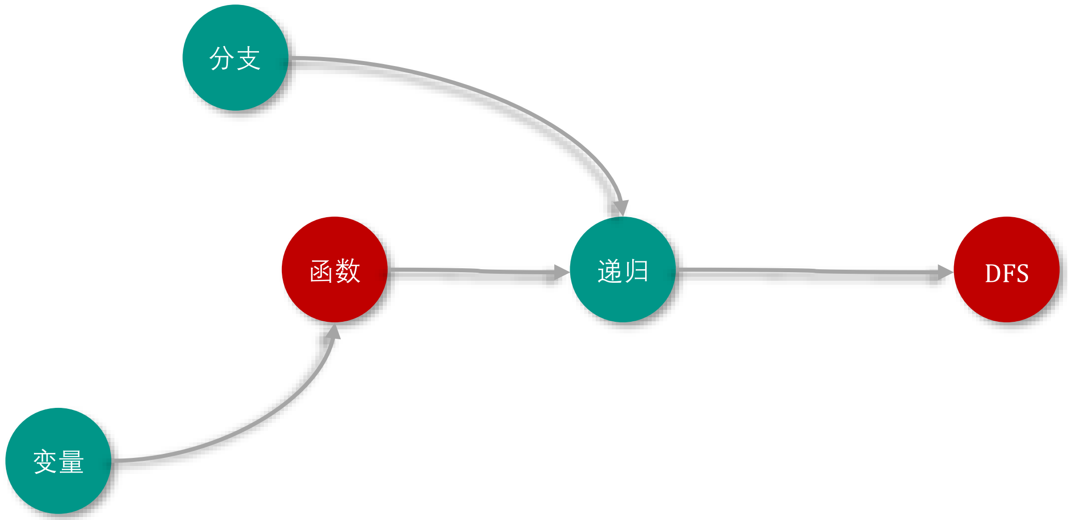
拓扑排序(Topological sorting)是对 DAG (有向无环图) 上的节点进行排序

使得对于每一条有向边 $u \to v$, $u$都在 $v$ 之前出现

拓扑排序的目标是将所有节点排序,使得排在前面的节点不能依赖于排在后面的节点

简单地说,是在不破坏节点先后顺序的前提下,把 DAG 拉成一条链

*拓扑序列往往不唯一*



# Kahn's Algorithm

初始状态下,集合 $S$ 装着所有入度为 0 的点, $L$ 是一个空列表

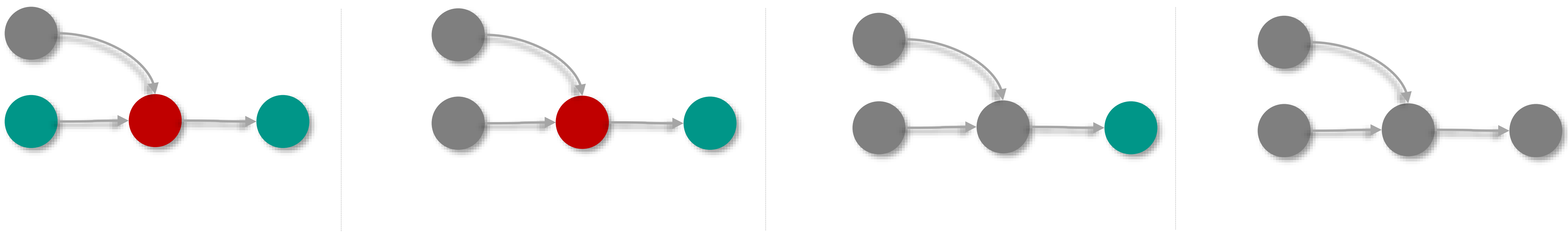
每次从 $S$ 中取出一个点 $u$ 放入 $L$, 然后将 $u$ 的所有边 $(u,v_1),(u,v_2),(u,v_3)\cdots$ 删除

对于边 $(u,v)$,若将该边删除后点 $v$ 的入度变为 0 , 则将 $v$ 放入 $S$ 中

不断重复以上过程,直到集合 $S$ 为空

若图中依然有边,那么这个图一定有环路

$L$ 中顶点的顺序就是拓扑排序的结果



# #2665、DAG路径计数

## 题目描述

给定一个 $n$ 个节点, $m$ 条边的 $DAG$

以及图中两个点 $st$ 和 $et$ ,求 $st$ 到 $et$ 之间简单路径的数量

答案可能很大,你只需要输出答案对 $1000000007$ 取模的结果

## 输入格式

第一行两个正整数 $n, m$

接下来 $m$ 行,每行两个正整数 $u, v$ ,表示有 $u$ 到 $v$ 节点间连有一条有向边

## 输出格式

输出一个整数,表示路径数对 $1000000007$ 取模后的结果

## 数据范围与提示

对于 $100\%$ 的数据: $1 \leq n \leq 10^5, 1 \leq m \leq 2 \times 10^5$

拓扑序列从前到后无依赖(无后效性)

令 $cnt_i$ 表示 $s \to i$ 的路径数量

初始时 $cnt_s = 1$

在拓扑排序的过程中

$cnt_v \leftarrow cnt_v + cnt_u$ , 其中 $u \to v$

答案为 $cnt_e$

# #356、可达统计

## 题目描述

给定一张 $N$ 个点 $M$ 条边的有向无环图

分别统计从每个点出发能够到达的点的数量

## 输入格式

第一行两个整数 $N, M$ ,接下来 $M$ 行每行两个整数 $x, y$ ,表示从 $x$ 到 $y$ 的一条有向边

## 输出格式

输出共 $N$ 行,表示每个点能够到达的点的数量

## 数据范围

对于全部的数据 $1 \leq N, M \leq 30000$

## 输入样例

```
10 10
3 8
2 3
2 5
5 9
5 9
2 3
3 9
4 8
2 10
4 9
```

## 输出样例

```
1
6
3
3
2
1
1
1
1
1
```

# #356、可达统计

$S_u$ 表示 $u$ 能到达点的集合

$$S_u = S_{v_1} \cup S_{v_2} \cdots \cup S_{v_x}$$

其中 $v_x$ 表示 $u$ 能直接到达的点

拓扑排序求出拓扑排序,逆序递推求出 $S_u$

对于每一个点用长度为 $n$ 的 bool 数组表示 $S$

时间复杂度 $O(nm)$

bitset优化

$S$ 使用一个 bitset 元素代替,合并集合可以直接使用 按位或

时间复杂度$O\left(\frac{nm}{W}\right)$ , 其中 $W$ 为计算机一个整型变量的大小

```
while (q.size())
{
    int u = q.front();
    q.pop();
    seq.push_back(u);
    for (int i = head[u]; i; i = e[i].next)
        if (!--in[e[i].v])
            q.push(e[i].v);
}
reverse(seq.begin(), seq.end());
for (auto &&u : seq)
{
    ans[u][u] = 1;
    for (int i = head[u]; i; i = e[i].next)
        ans[u] |= ans[e[i].v];
}
for (int i = 1; i <= n; i++)
    printf("%d\n", ans[i].count());
```

# #842、车站分级

## 题目描述

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

## 说明

对于 $20\%$ 的数据， $1 \le n,m \le 10$
对于 $50\%$ 的数据， $1 \le n,m \le 100$
对于 $100\%$ 的数据， $1 \le n,m \le 1000$

## 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 $(1 \le i \le m)$ 中,首先是一个正整数 $s_i$ ,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

一个正整数,即 $n$ 个火车站最少划分的级别数

# #842、车站分级

两个停靠之间所有站点优先级一定低于所有的停靠站点

将 $S_1 \sim S_k$ 之间的所有未停站点连有向边

得到一个 DAG

答案为 DAG 的最大层数(Dilworth′s theorem)

最坏情况下有 $1000 \times 500 \times 500$ 条有向边, MLE

其中有大量重边,考虑邻接矩阵存图

# #842、车站分级

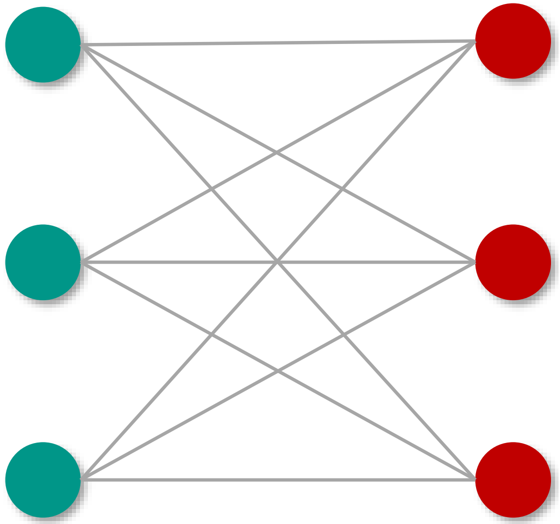
每条行车计划新增一个虚拟点

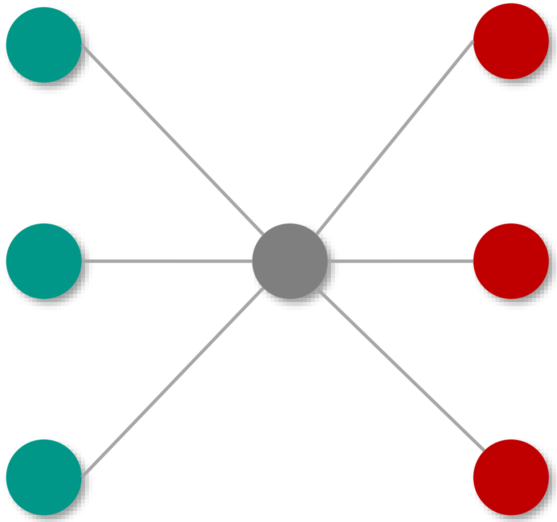
所有未停靠的站点连一条有向边到虚拟点,将虚拟点分别连一条有向边到 $S_1 \sim S_k$

在拓扑排序过程中

若遇到虚拟点,层数不加一,答案依然是层数最大值

能否更优?





实验舱
青少年编程
走近科学 走进名校

# 谢谢观看