一些图论知识

清华大学
neither_nor

写在前面的话

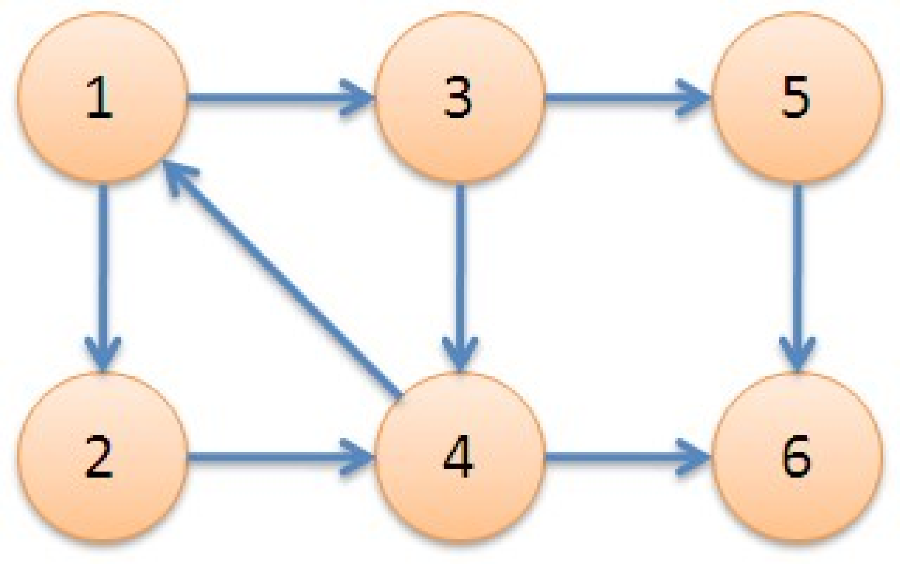
- 图论曾经是 OI 中必不可少的内容
- 然而随着最近几年大量数据结构题、字符串题与数学题慢慢风靡起来，以及一些偏题怪题的出现，你会发现在重大考试当中
- 图论仍然是 OI 中必不可少的内容
- 而且一般来讲这些题目都呈现一种又难又简单的状态
- 比起拼码力的数据结构，拼智商的数学题，拼脑洞的奇葩题
- 一般图论题经常能造成一种，考试时就是想不出来，结束之后一拍别人肩膀，一拍自己脑袋，最后一拍大腿，然后由衷发出一句“诶呦我 x ，这题这么简单”的情况

首先是目录

- 强连通分量
- 双连通分量
- 二分图匹配
- 2-SAT

什么是强连通分量？

- 根据百度百科的定义
  - 有向图 G 中，若两个顶点 u,v 之间能够**相互**到达，则称 u,v 是强联通的
  - 若有向图 G 的每两个顶点都强连通，则 G 是一个强连通图
  - 有向图的**极大**强连通子图，称为强连通分量



- 由定义我们可以知道：

    1. 强连通分量是针对有向图的

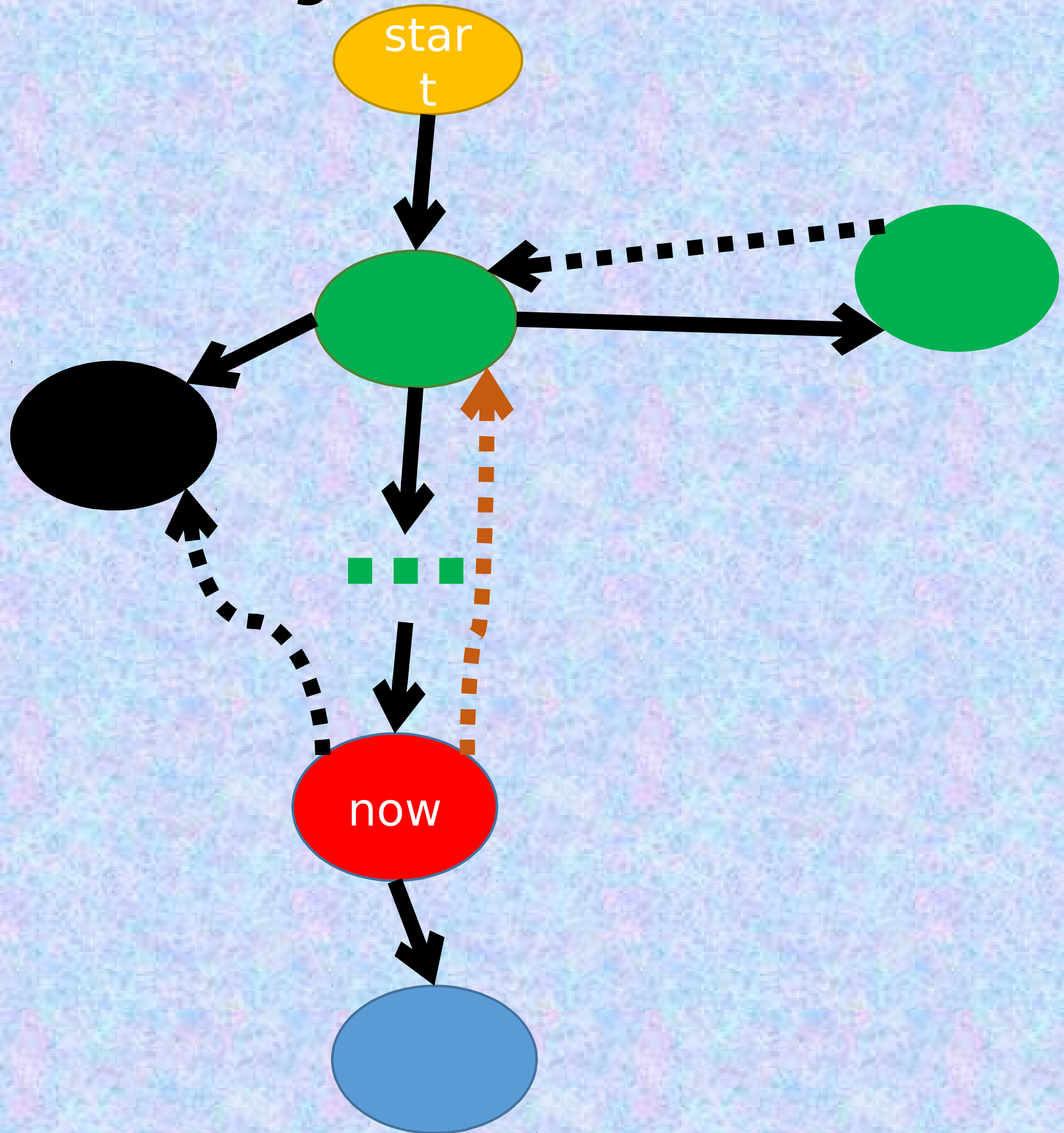
    2. 有向图 G 中的每个点都属于且只属于一个强连通分量

    3. 若 A,B 属于同一个强连通分量， B,C 属于同一个强连通分量，则 A,C 也属于同一个强连通分量

如何求强连通分量?

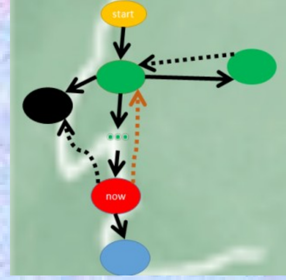
- **Tarjan** 算法
- **Kosaraju** 算法

Tarjan 求强连通分量：感性思考



从某个点开始 DFS
now 连出的边无非分三种情况：
绿色：之前访问过的能到达 now 的点
黑色：之前访问过的到不了 now 的点
蓝色：还没访问的点

Tarjan 求强连通分量：实现




```
int dfn[N],low[N],ans,cn;
bool ins[N];
int s[N],t;
void tarjan(int x)
{
    cn++;
    dfn[x]=low[x]=cn;
    ins[x]=true;t++;s[t]=x;
    int i,j;
    for(i=pre[x];i;i=nxt[i])
    {
        j=to[i];
        if(!dfn[j])
        {
            tarjan(j);
            low[x]=min(low[x],low[j]);
        }
        else if(ins[j]) low[x]=min(low[x],dfn[j]);
    }
    if(low[x]==dfn[x])
    {
        ans++;
        while(s[t]!=x)
        {
            ins[s[t]]=false;
            t--;
        }
        ins[x]=false;
        t--;
    }
}
```

主体过程（看图说话）：

从某个点开始， DFS 整个图，每访问到一个新的节点就将其压入栈中
为每个点记录两个量 dfn 和 low
其中 dfn[x] 是时间戳，代表 x 是第几个被访问到的点
low[x] 代表 x **最多只经过一条非树边**（虚线）能到达的点中， dfn 的最小值

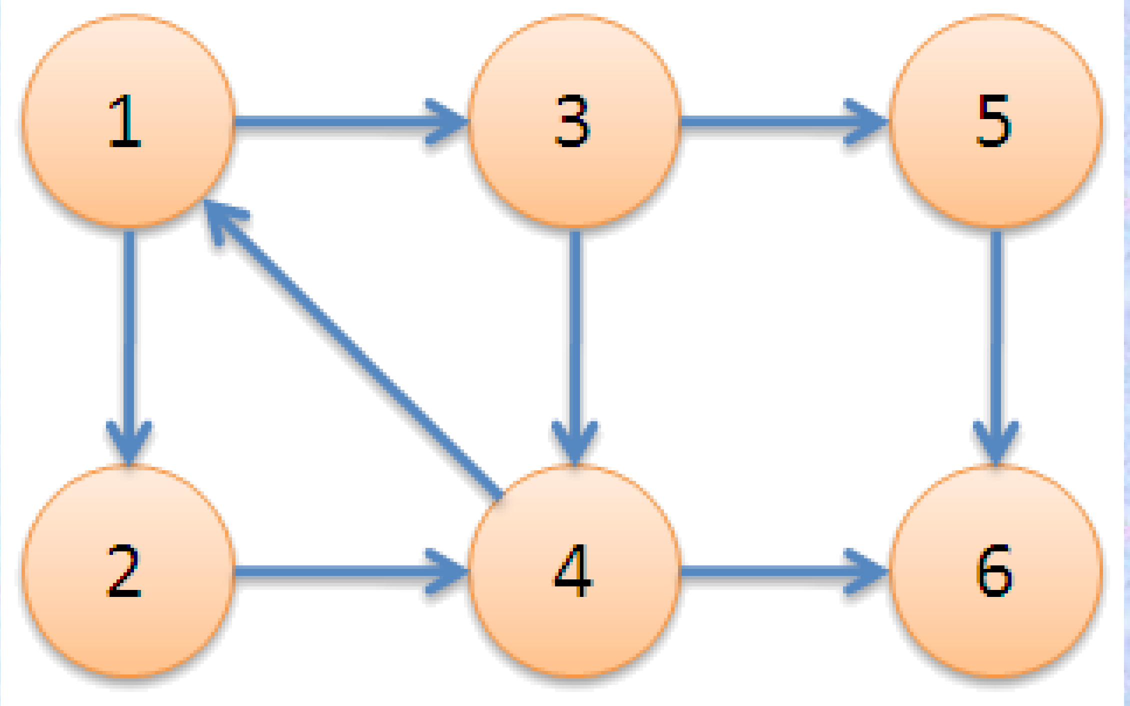
维护 low 值的方式：对于 x 的每个出点 j

若 j 还没被搜到（蓝色），则先搜索 j ，然后用 low[j] 来更新 low[x]
否则若 j 还在栈中（绿色），则直接用 dfn[j] 来更新 low[x]
为什么 dfn[x]==low[x] 时就是同一个强连通分量了？

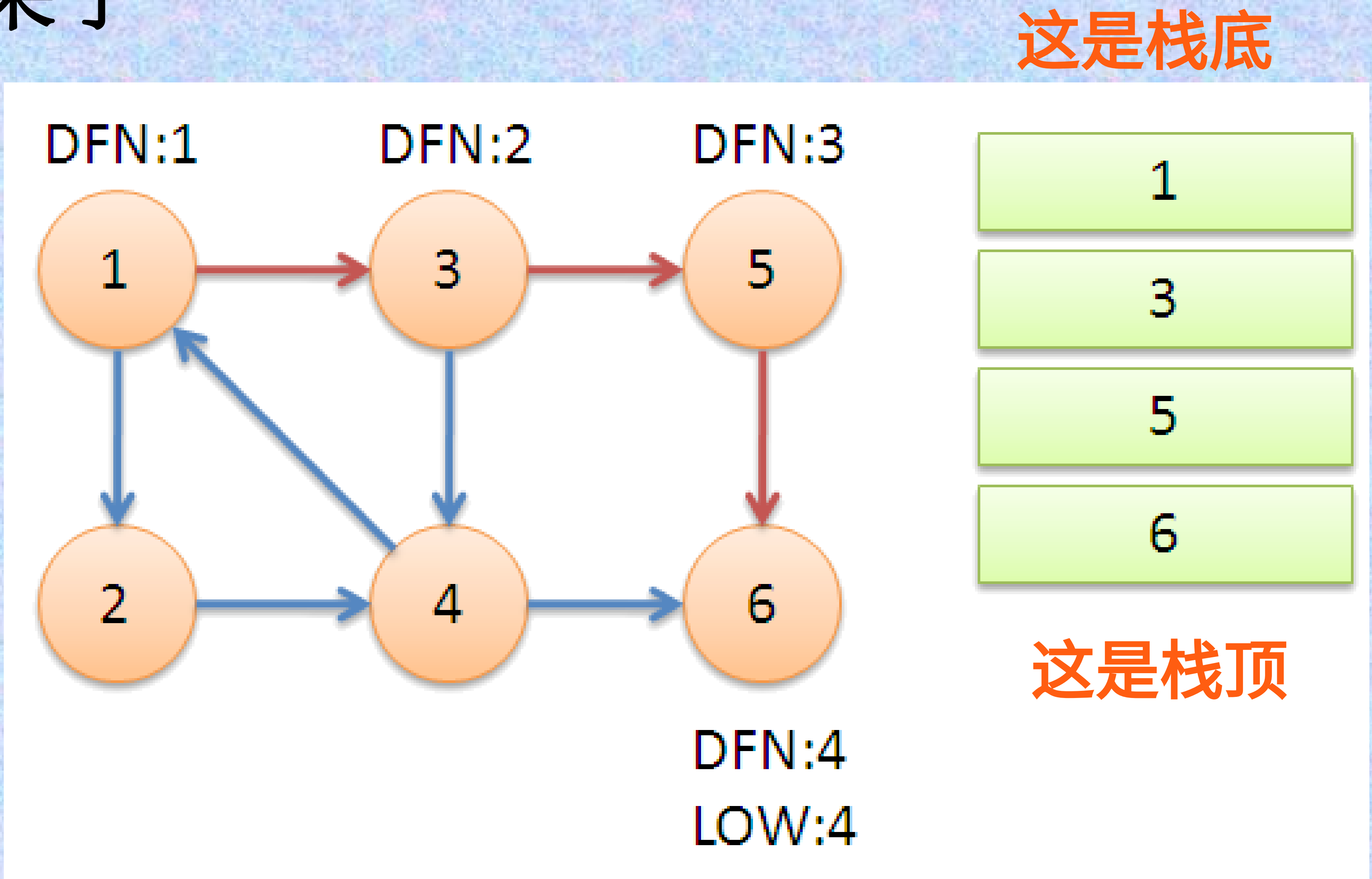
j 还在栈中时如果用 low 来更新会怎么样?

x 的出边全部遍历之后，若 dfn[x]==low[x]，则从栈顶到 x 的这部分属于同一个强连通分量，把这段弹栈

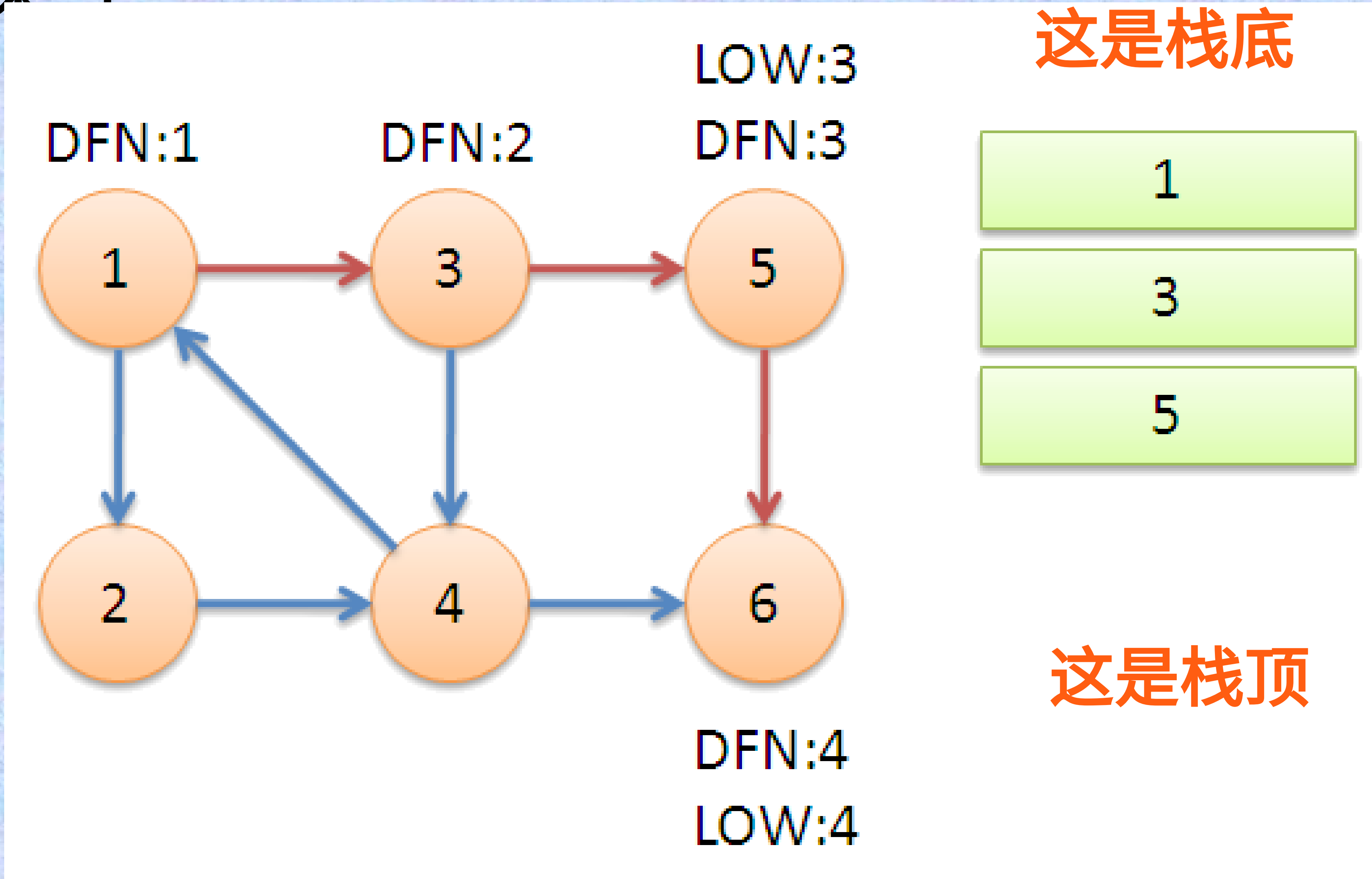
举个栗子



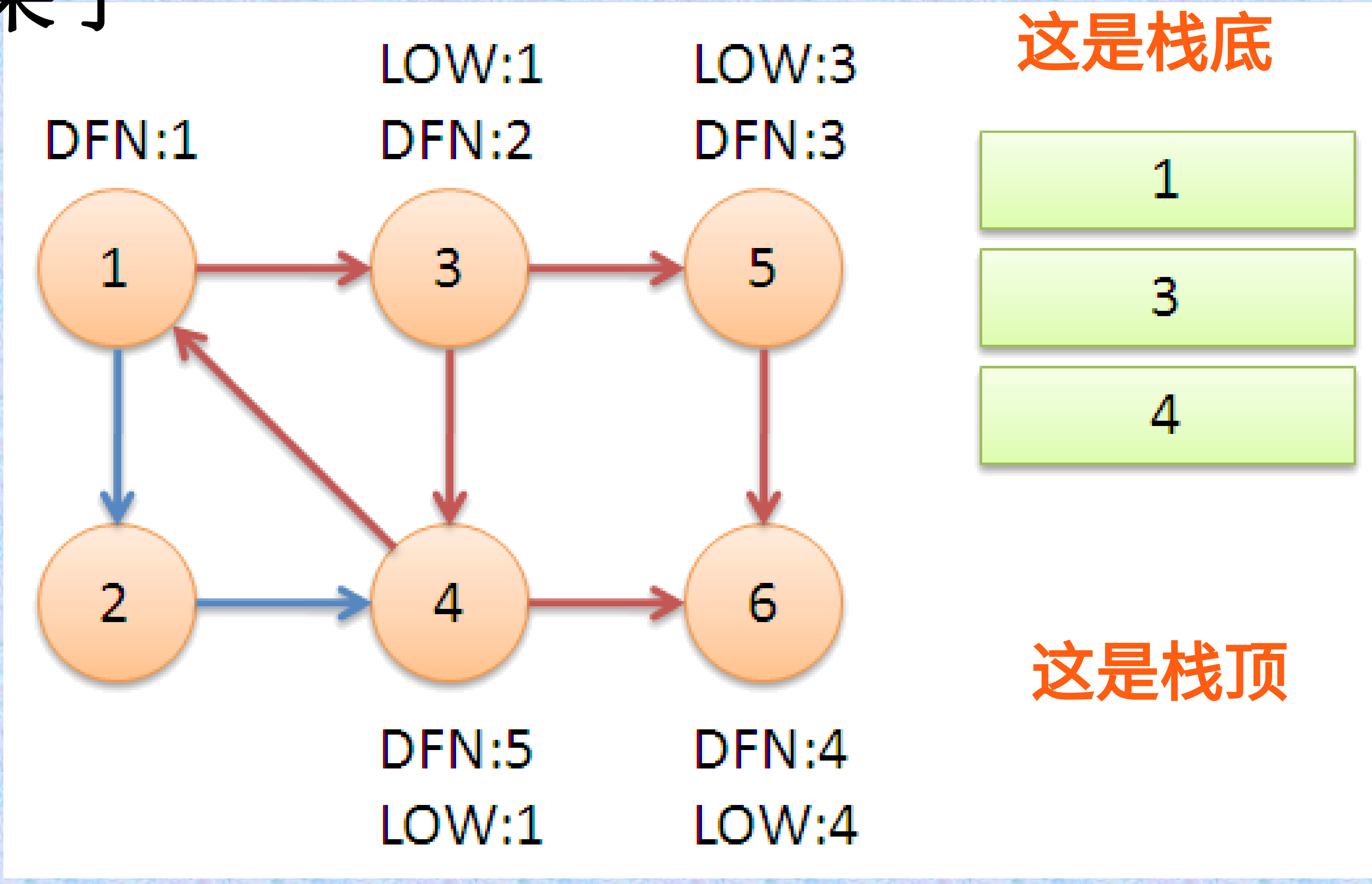
举个栗子



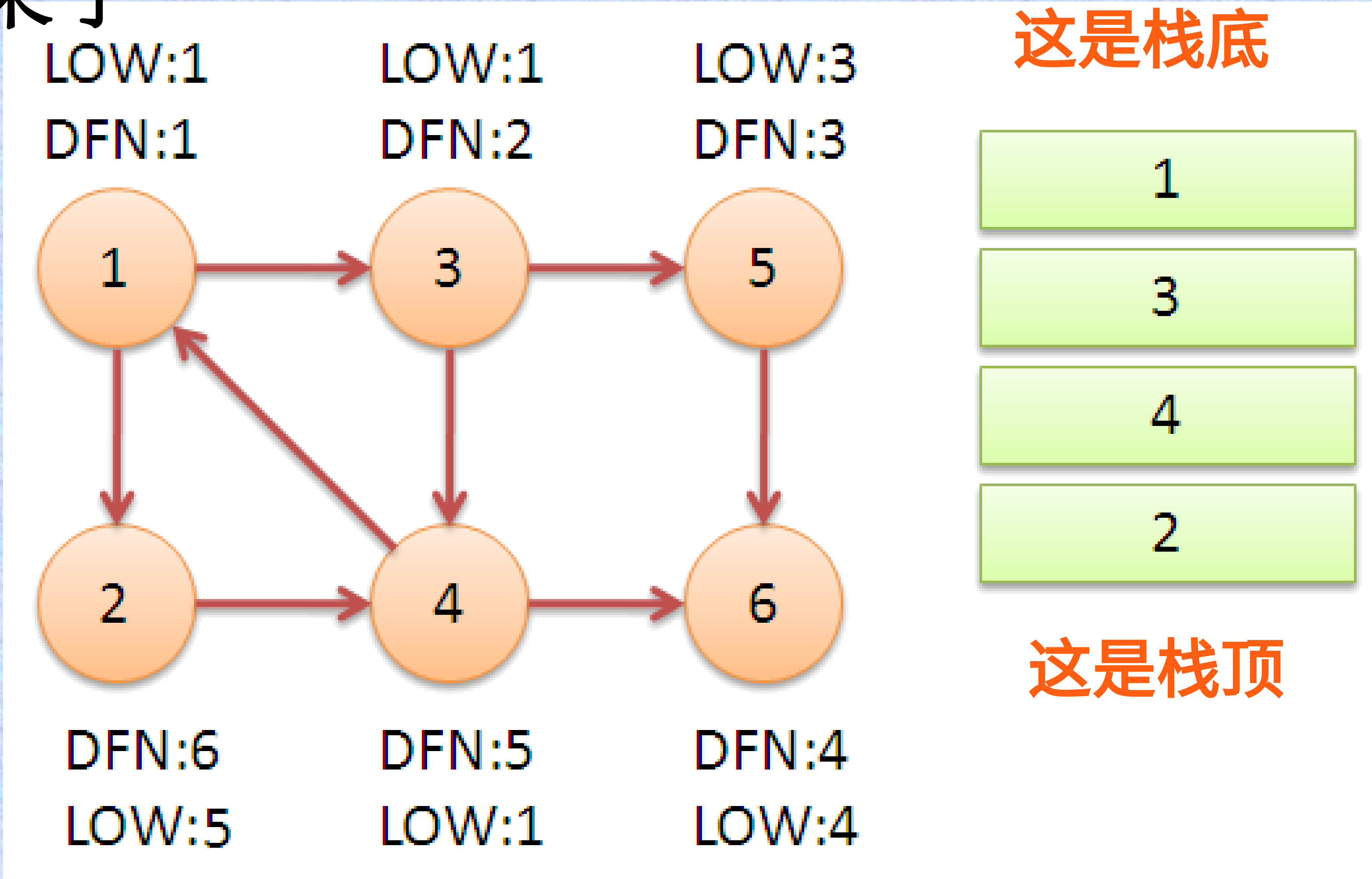
举个栗子



举个栗子


LOW:1
LOW:3
DFN:1
DFN:2
DFN:3
1
3
5
2
4
6
DFN:5
DFN:4
LOW:1
LOW:4
这是栈底
1
3
4
这是栈顶

举个栗子



LOW:1
DFN:1
LOW:1
DFN:2
LOW:3
DFN:3
这是栈底
1
3
5
1
3
4
2
4
6
DFN:6
LOW:5
DFN:5
LOW:1
DFN:4
LOW:4
这是栈顶

缩点

- 求出强连通分量之后我们能干什么呢?
- 很套路的一步是缩点，把同一个强连通分量里的点缩成一个点

- 这样做的好处?
- 原图中的环消失了，变成了一个 DAG
- ~~卡常暴力~~

[Usaco2006 Jan]The Cow Prom

- 这题废话贼多
- 给出一个有向图 G，问有多少个大小 >1 的强连通分量

- 直接上 Tarjan 就好了，顺便维护每个强连通分量的大小

[HAOI2006] 受欢迎的牛

- 给出一个有向图，问有多少个点能被所有点到达

- 点数 <=10000, 边数 <=50000

- 考虑没有环时的做法
- 看出度为 0 的点是否只有一个，如果是则这个点能被所有点到达
- 否则没有能被所有点到达的点

- 有环的做法?
- 先缩点，然后看出度为 0 的强连通分量是否只有一个
- 如果有，则答案为这个强连通分量的大小
- 否则没有能被所有点到达的点

BZOJ2140： 稳定婚姻

- 这题废话也多…
- 给定 n 对夫妻关系，再给定 m 对情人关系。
- 如果在第 i 对夫妻离婚的前提下，这些人仍然能组成 n 对夫妻 (情人可以结为夫妻)，就称 i 这个婚姻是不稳定的。
- 要求判断每一对婚姻稳不稳定

分析

- 设大写字母为男性，小写字母是女性
- 则一对婚姻 Aa 是不稳定的当且仅当 A 与 a 离婚， a 与 B 情人，b 与 C 情人， c 与 D 情人 .....x 与 A 情人
- 也就是说我们能找到 a->B->b->C->c->D->d->…>X->x->A 这样一条路径
- 我们发现形如 B->b （男 -> 女）的都是夫妻关系， b->B （女 -> 男）的都是情人关系
- 所以我们按照这种方式来建图，若 Aa 在同一个强连通分量中，则这对婚姻是不稳定的

- 强连通分量
- 双连通分量
- 二分图匹配
- 2-SAT

割点和桥

- 无向图 G 中，若去掉点 x 及其所有相邻的边之后，G 的连通块个数增加了，则称 x 是 G 的一个割点

- 无向图 G 中，若去掉边 e 之后，G 的连通块个数增加了，则称 e 是 G 的一个桥

割点的求法

- 类似之前的 tarjan 算法
- 如果点 x 有一个出点的 low 值 >=dfn[x]，则 x 是一个割点

- 特殊情况：第一个搜索的点
- 终于能解释为什么用 dfn 值更新了 QAQ

双连通分量

- 对于一个无向图 G ，若其不存在桥，则称其是一个边双连通图
- 一个无向图 G 的极大边双连通子图成为 G 的边双连通分量

- 对于一个无向图 G ，若其不存在割点，则称其是一个点双连通图
- 一个无向图 G 的极大点双连通子图成为 G 的点双连通分量

边双连通分量

用通俗一点的话说， A,B 在同一个边双中当且仅当 A,B 之间有两条边不相交路径

根据定义我们可以知道，每个点属于且只属于一个边双连通分量

注意：边双连通分量缩点后会变成一棵树

边双连通分量的求法

- 还是用 Tarjan 算法，几乎和求强连通分量的方式一样

- 为什么这样做是对的?
- 从 DFS 树的角度考虑，我们把树边当成从上到下的有向边，非树边当成从下到上的有向边，这个图中的强连通分量就是正常无向图中的边双连通分量

[Usaco2006 Jan] Redundant Paths

- 为了从 F(1≤F≤5000) 个草场中的一个走到另一个，贝茜和她的同伴们有时不得不路过一些她们讨厌的可怕的树．奶牛们已经厌倦了被迫走某一条路，所以她们想建一些新路，使每一对草场之间都会至少有两条相互分离的路径，这样她们就有多一些选择．
- 　每对草场之间已经有至少一条路径．给出所有 R(F-1≤R≤10000) 条双向路的描述，每条路连接了两个不同的草场，请计算最少的新建道路的数量，路径由若干道路首尾相连而成．两条路径相互分离，是指两条路径没有一条重合的道路．但是，两条分离的路径上可以有一些相同的草场． 对于同一对草场之间，可能已经有两条不同的道路，你也可以在它们之间再建一条道路，作为另一条不同的道路．

- 先把边双连通分量缩起来，这样就变成了一棵树的问题
- 那么我们就只需要考虑树的最优策略

- 显然我们可以贪心地把所有度数为 1 的点一对一对的连起来
- 所以答案 =⌈ 缩点之后度数为 1 的点的个数 /2⌉

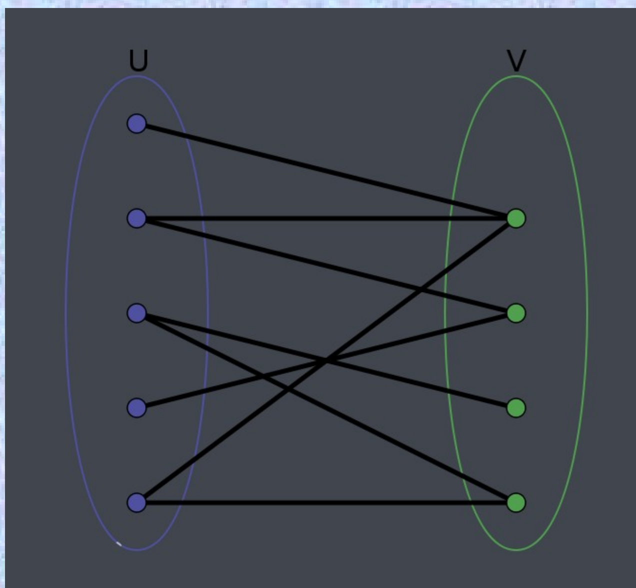
点双连通分量的求法

- 由于一个割点可能在多个点双连通分量里，所以我们需要在普通的 tarjan 算法基础上做一些改动
- 我们尝试把边压入栈中而不是点，这样我们开始弹栈的时候最后剩下的这条边就会把割点留下了

- 强连通分量
- 双连通分量
- 二分图匹配
- 2-SAT

二分图

- 二分图又称作二部图，是图论中的一种特殊模型
- 设 G=(V,E) 为一个无向图，若 V 存在两个子集 A,B ，使得 A∪B= V ， A∩B=∅ ，且保证对于∀ (i,j)∈E ，有 i∈A&&j∈B 或 i∈B&&j∈A ，则称 G 是一个二分图 (二部图)
- 简而言之就是如果能把 G 划分成两部分，使得每一部分内都不自己和自己连边， G 就是一个二分图



二分图的特征及判定

- 可以用两种颜色染色 (不存在长度为奇数的环)
- 所以我们可以从每个连通块开始 DFS ，尝试染色，每次从 x 向外扩展时分两种情况讨论：
- 1. 没染过色，那就染成相反的颜色然后从他开始 DFS
- 2. 染过色了，检验其是否合法，不合法退出，合法跳过
- 时间复杂度 O(N)

二分图

- 对于二分图 G 的一个子图 M=(V,E) 若 E 的任意两条边都不依附于 V 中的同一个顶点，则称 M 是一个匹配

- 若 M 是 G 的所有匹配当中边数最多的一个匹配，则称 M 是一个最大匹配
- 二分图的最大匹配可能有多个

- 如果一个匹配中图中的每个顶点都和匹配中的某条边关联，则称这个匹配是一个完美匹配

Hall 定理

- 有二分图 G=(V,E)，其中 V 划分为 A,B， C 是 A 的子集，且 |C|=k，
- 存在一个匹配 M 使得 C 中的每一个点都存在于匹配中，当且仅当对于∀ 1<=i<=k， C 中任意 i 个点都至少和 B 中 i 个点相连
- 必要性显然
- 充分性：假设对于 i<k 全部成立且 k 不成立。先找到一个 C 的最大匹配，此时有一个点不在匹配当中，由于 i=1 成立，所以至少有一个点在 B 中并且和他关联，且这个点必须已经在匹配中，那么我们再找到和这个点匹配的点，由于 i=2 成立，所以至少有两个点在 B 中并且和它们相连，然后那个点也必须已经在匹配中⋯⋯最后得出 B 中至少有 k 个点在匹配中，这和 C 中只有 k-1 个点在匹配中矛盾

匈牙利算法

- 匈牙利算法的目的是求出二分图的最大匹配
- 算法的主要思想是不断地寻找增广路来使得总匹配数 +1
- (先找出大小为 1 的匹配，由此产生大小为 2 的匹配… .)
- 其主体过程和 Hall 定理的证明十分类似

增广路

- 一条增广路指的是我们从某个点出发，沿着 (不在匹配中)->(在匹配中)->(不在匹配中)->(在匹配中)…. 的方式走，最后停在一个不依附于当前任何匹配的点
- 如果我们找不到这样一条增广路，当前匹配就是最大匹配 (Hall 定理)
- 否则我们把这条增广路取反 (在匹配中 <-> 不在匹配中)，这样匹配数 +1

代码实现

```
bool find(int x)
{
    int i;
    for(i=1;i<=n;i++)
    {
        if(f[x][i]&&!used[i])
        {
            used[i]=true;
            if(!c[i]||find(c[i]))
            {
                c[i]=x;
                return true;
            }
        }
    }
    return false;
}
```

```
int ans=0;
for(i=1;i<=m;i++)
{
    memset(used,0,sizeof(used));
    if(find(i)) ans++;
}
cout<<ans;
```

时间复杂度：最差O(N^3)

- 二分图最小点覆盖 = 二分图最大匹配
- 二分图最大点独立集 = 总点数 - 二分图最大匹配

- 第一个证明：首先最小点覆盖大于等于最大匹配
- 接下来用构造法，从右侧每个非匹配点出发，寻找一条匹配边与非匹配边相间的路径，且以匹配边结束
- 选择所有左侧的在上面的路径内的点和右侧的不在上面的路径内的点，即是一个点覆盖

- 第二个证明：点覆盖和独立集是互补的，点覆盖的补必定是独立集，独立集的补必定是点覆盖

飞行员配对方案问题

- 第二次世界大战时期，英国皇家空军从沦陷国征募了大量外籍飞行员。由皇家空军派出的每一架飞机都需要配备在航行技能和语言上能互相配合的 2 名飞行员，其中 1 名是英国飞行员，另 1 名是外籍飞行员。
- 在众多的飞行员中，每一名外籍飞行员都可以与其他若干名英国飞行员很好地配合。如何选择配对飞行的飞行员才能使一次派出最多的飞机。对于给定的外籍飞行员与英国飞行员的配合情况，试设计一个算法找出最佳飞行员配对方案，使皇家空军一次能派出最多的飞机。

[Usaco2005 nov]Asteroids

- 贝茜想驾驶她的飞船穿过危险的小行星群．小行星群是一个 NxN 的网格 (1≤N≤500)，在网格内有 K 个小行星 (1≤K≤10000)． 幸运地是贝茜有一个很强大的武器，一次可以消除所有在一行或一列中的小行星，这种武器很贵，所以她希望尽量地少用．给出所有的小行星的位置，算出贝茜最少需要多少次射击就能消除所有的小行星．

- 我们把对行和列建一个二分图，若存在小行星 (i,j)，就在行 i 和列 j 之间连一条边
- 现在问题就转化为了求二分图最小点覆盖
- 直接跑匈牙利算法即可

泥泞的牧场

- 大雨侵袭了牧场．牧场是一个 R * C 的矩形，其中 1≤R， C≤50.
- 大雨将没有长草的土 地弄得泥泞不堪，可是小心的奶牛们不想在吃草的时候弄脏她们的蹄子． 为了防止她们的 蹄子被弄脏，农场主决定在泥泞的牧场里放置一些木板．每一块木板的宽度为 1 个单位，长度任意．每一个板必须放置在平行于牧场的泥地里。农场主想使用最少的木板覆盖所有的 泥地。
- 一个木板可以重叠在另一个木板上，但是不能放在草地上

- 类似之前的那道题，区别在于这道题不能盖住好的草坪

- 首先能想到贪心，即每块板子延伸至能盖的最长情况
- 所以对于一个泥坑，他就一定要被横着或竖着的两块板子的至少一块来覆盖，这就很像之前的最小点覆盖了
- 所以我们建二分图， A 部是所有横着的板子， B 部是所有竖着的板子，然后对于所有泥坑，把他相关的两个板子连起来，这样选择一个点就代表选择了一块板子，最小点覆盖就相当于用最小的板子数来覆盖所有泥坑了

- 强连通分量
- 双连通分量
- 二分图匹配
- 2-SAT

2-SAT 模型

- SAT 是 Satisfiability(适应性) 的缩写， SAT 问题 (适应性问题) 指的是给出一些条件 (元素经过逻辑运算的结果，如 x1 and x2 or x3 xor x4 = a) ，问是否存在一种合法的方案使得其满足所有的条件。SAT 问题已经被证明是一个 NP 完全问题

- 2-SAT 问题是一种特殊的 SAT 问题，它限制了每个表达式当中最多有两个未知量。 2-SAT 问题已经可以在多项式复杂度内求出 100% 正确解

二元逻辑运算的转化

- 为了解决 2-SAT 问题，我们可以把给出的二元关系式进行一些转换，变成形如“若 A 取值为 x 则 B 必须取值为 y” 这样的条件。
- 为了叙述方便，我们拿取值为 0 和 1 举例子
- A&B=0 -> 若 A 取 1 则 B 必须取 0，若 B 取 1 则 A 必须取 0
- A|B=1 -> 若 A 取 0 则 B 必须取 1，若 B 取 0 则 A 必须取 1
- A^B=1 -> …
- 对称性
- 对于特殊的限制条件，例如 A=1(A&B=1)，我们可以构建“若 A 取 0 则 A 必须取 1” 这样的限制条件

建图

- 对于所有元素，我们拆成两个点 a_0, a_1 分别代表他的两种取值
- 对于所有的“若A取x则B必须取y”条件，从 A_x 向 B_y 连一条边
- 这样对于这张图中两个点的P和Q，若P能到达Q，则代表“若P成立，则Q一定成立”
- 那么我们显然知道，如果出现 a_0 能到达 a_1 且 a_1 能到达 a_0，则矛盾
- 这一步的检验我们可以用Tarjan求强连通分量，O(N)解决
- 如果没有这种情况出现，我们断言必定存在一种合法方案
- 证明？如下：

证明

- 我们构造一种这样的方案：缩点之后按照”逆拓扑序”挑选一个未被染色的强连通分量将其染成黑色，然后把所有与这个强连通分量里的点矛盾的点所在的强连通分量全部染成白色。最后所有染成黑色的强连通分量里的点就是选择的方案
- 由于建出的图有对称性，即对于不同的点a,b，若存在边(a_0,b_1)，则一定存在边(b_0,a_1)，则有图中的到达关系也存在对称性(形式与上面类似)
- 当我们把a_0染成黑色时，对应的a_1应被染成白色，此时意味着a_1的前代节点永远不能染成黑色了，而a_1的前代节点的对应相反结点正是a_0的后代节点，它们已经全都是黑色了，所以不存在矛盾
- 这也是构造可行解的一种方式，时间复杂度O(N+M)

判定某个元素的取值是否确定

- 有的时候题目并不只是让你判定是否存在合法解，而是要求你判断某个元素的取值是否为定值
- 若a_0是可选的，则在图中加入(a_1,a_0)这条边后，应该仍然合法
- 若a_1是可选的，则在图中加入(a_0,a_1)这条边后，应该仍然合法
- 若二者皆合法，则这个元素的取值是不确定的

基于上一个方法的另一种求可行解的方法

- 我们可以从 1 到 n 枚举元素，再枚举取值，如果这种取值是合法的，就继续下去，否则就取另外一种取值，然后继续下去
- 由于是否合法可以及时判断，所以没有回溯的过程，时间复杂度是 O(NM) 的 ······
- 但是他可以用来求字典序最小的可行解

两道裸题

- **[JSOI2010] 满汉全席**
- **[Usaco2011 Jan]** 奶牛议会

noi2017 游戏

- 小 L 计划进行 n 场游戏，每场游戏使用一张地图，小 L 会选择一辆车在该地图上完成游戏。
- 小 L 的赛车有三辆，分别用大写字母 A、 B、 C 表示。地图一共有四种，分别用小写字母 x、 a、 b、 c 表示。其中，赛车 A 不适合在地图 a 上使用，赛车 B 不适合在地图 b 上使用，赛车 C 不适合在地图 c 上使用，而地图 x 则适合所有赛车参加。适合所有赛车参加的地图并不多见，最多只会有 d 张。
- 小 L 对游戏有一些特殊的要求，这些要求可以用四元组 (i,hi,j,hj) 来描述，表示若在第 ii 场使用型号为 hi 的车子，则第 jj 场游戏要使用型号为 hj 的车子。
- 你能帮小 L 选择每场游戏使用的赛车吗？如果有多种方案，输出任意一种方案。如果无解，输出 “-1” （不含双引号）。
- n<=50000,d<=8,m<=100000

游戏

- 考虑如果没有 x 的情况，那么由于每个赛道只有两种可用的车，所以就是一个经典的 2sat 问题
- 考虑 x ，我们可以 3^d 枚举每个 x 是不能用哪辆车，然后再 2-sat ，但是这样会超时
- 观察发现其实不需要 3^d ，因为枚举一个不能用之后剩下的都能用，只需要 2^d 枚举即可涵盖所有可能的情况，

谢谢大家