有关图连通性的入门讲解

Aonynation

ASDFZ

2022 年 ζ 月 ϑ 日

前言

受本人智商的影响，本课件题目全部是入门难度而且准备的题目非常少，大家不要喷。
但是我自认为做的还算比较认真。
可能大家觉得小学知识很无聊，但还请大家耐着性子听一下。

Nowcoder 挑战 60 D. 三千道路

题目描述：
给你一个有向图，求是否存在一张竞赛图的连通性与给定的有向图连通性相同。

Nowcoder 挑战 60 D. 三千道路

题目描述：
给你一个有向图，求是否存在一张竞赛图的连通性与给定的有向图连通性相同。

什么是竞赛图：若简单有向图 G 满足任意不同两点间都有恰好一条单向边，则称 G 为竞赛图。

Nowcoder 挑战 60 D. 三千道路 Solution

其实是我用数量打败质量的行为。

Nowcoder 挑战 60 D. 三千道路 Solution

其实是我用数量打败质量的行为。
其实已经有神仙秒了，但是为了凑数，我也就讲一下。。。

Nowcoder 挑战 60 D. 三千道路 Solution

其实是我用数量打败质量的行为。
其实已经有神仙秒了，但是为了凑数，我也就讲一下。。。

首先，我们来了解一下竞赛图的一些性质。
(考虑到牛逼老哥要讲，就随便说说，反正这题 ztq 直接手推)

Nowcoder 挑战 60 D. 三千道路 Solution

其实是我用数量打败质量的行为。
其实已经有神仙秒了，但是为了凑数，我也就讲一下。。。

首先，我们来了解一下竞赛图的一些性质。
(考虑到牛逼老哥要讲，就随便说说，反正这题 ztq 直接手推)
竞赛图：有向完全图。
这题可能会用到的性质：缩点之后 DAG 是一条链的形状。

Nowcoder 挑战 60 D. 三千道路 Solution

至于上面的为什么，我来口胡一下。

Nowcoder 挑战 60 D. 三千道路 Solution

至于上面的为什么，我来口胡一下。
考虑逐个加入连通块。
接下来分几种不同的情况进行讨论：
1. 这个点连向所有点，直接放在最前面就可以了。
2. 所有点都连向这个点，直接放在最后面。
3. 其他的情况直接找到一个合适的位置，插入即可。

Nowcoder 挑战 60 D. 三千道路 Solution

知道这个之后，题目就异常简单。

Nowcoder 挑战 60 D. 三千道路 Solution

知道这个之后，题目就异常简单。
直接对原图缩点，然后上拓扑排序即可。
拓扑排序在任意一个时刻队列只能有一个元素。

Nowcoder 挑战 60 D. 三千道路 Solution

知道这个之后，题目就异常简单。
直接对原图缩点，然后上拓扑排序即可。
拓扑排序在任意一个时刻队列只能有一个元素。

注意：要特判一个强连通分量只有两个数的情况。

[CERC2016] 机棚障碍

题目描述：

给定一个 $n \times n$ 的网格图，其中部分格点有障碍物使得箱子不能置于其上。规定箱子是一个奇数边长的正方形，其坐标为其中心格点的坐标。箱子只能上下左右移动，每次询问从一个格点能移动到另一个格点的最大箱子。

[CERC2016] 机棚障碍 Sulotion

二分 + bfs + 缩点 + 最小生成树 + 树链剖分就没了。

FZOJ 2785

题目比较好理解，直接看吧。

FZOJ 2785 Solution

相信大家一眼秒。

FZOJ 2785 Solution

相信大家一眼秒。
直接跑一个 $Tarjan$ 然后拓扑排序一下，最后再 dp 就好了。

FZOJ 2785 Solution

相信大家一眼秒。
直接跑一个 $Tarjan$ 然后拓扑排序一下，最后再 dp 就好了。
注意 dp 时倒着做比较方便。

P5236 【模板】静态仙人掌

题目描述：

给你一个有 n 个点和 m 条边的仙人掌图，和 q 组询问。
每次询问两个点 u, v ，求两点之间的最短路。

P5236 【模板】静态仙人掌 Solution

先建出圆方树。

P5236 【模板】静态仙人掌 Solution

先建出圆方树。

圆方树

圆方树的建点、连边规则是这样的：
1、原图中的点都是圆点
2、对于每个点双，新建一个方点；这个方点和环上其它圆点连成菊花图
3、对于不在环上的两个圆点，保留原图中的边

P5236 【模板】静态仙人掌 Solution

考虑怎么把边权映射到圆方树上。

P5236 【模板】静态仙人掌 Solution

考虑怎么把边权映射到圆方树上。

设三个点 u,k,v，分别表示父亲，方点，儿子。

P5236 【模板】静态仙人掌 Solution

考虑怎么把边权映射到圆方树上。

设三个点 u,k,v，分别表示父亲，方点，儿子。
那么 $dis_{k,v}$ 表示 u,v 之间的最短距离。

P5236 【模板】静态仙人掌 Solution

考虑怎么把边权映射到圆方树上。

设三个点 u,k,v，分别表示父亲，方点，儿子。
那么 $dis_{k,v}$ 表示 u,v 之间的最短距离。

如果 v 不在环里，最短距离是原图中 u,v 的距离。
如果 v 在环里，直接对环顺时针/逆时针暴力求出最短路即可。

P5236 【模板】静态仙人掌 Solution

考虑怎么把边权映射到圆方树上。

设三个点 u,k,v，分别表示父亲，方点，儿子。
那么 $dis_{k,v}$ 表示 u,v 之间的最短距离。

如果 v 不在环里，最短距离是原图中 u,v 的距离。
如果 v 在环里，直接对环顺时针/逆时针暴力求出最短路即可。

查找答案直接求 LCA 即可。
(注意：如果两点 LCA 是方点，代价是 $dis_{x,y}$，x，y 是 LCA 的儿子，且分别是 u, v 的祖先)

Luogu P4606 [SDOI2018] 战略游戏

题目描述：

给出一个简单无向连通图。有 Q 次询问：
每次给出一个点集 S $(2 \leq |S| \leq n)$ 。
问有多少个点 u 满足 $u \notin S$ 且删掉 u 之后 S 不连通。

Luogu P4606 [SDOI2018] 战略游戏 Solution

感觉大家都是一眼秒掉啊。

Luogu P4606 [SDOI2018] 战略游戏 Solution

感觉大家都是一眼秒掉啊。
还是先建出圆方树考虑性质。

Luogu P4606 [SDOI2018] 战略游戏 Solution

感觉大家都是一眼秒掉啊。
还是先建出圆方树考虑性质。
先考虑两个点的情况，发现答案是两点间简单路径中圆点个数。

Luogu P4606 [SDOI2018] 战略游戏 Solution

感觉大家都是一眼秒掉啊。
还是先建出圆方树考虑性质。
先考虑两个点的情况，发现答案是两点间简单路径中圆点个数。
多模拟几组可以发现：

Luogu P4606 [SDOI2018] 战略游戏 Solution

感觉大家都是一眼秒掉啊。
还是先建出圆方树考虑性质。
先考虑两个点的情况，发现答案是两点间简单路径中圆点个数。
多模拟几组可以发现：
我们定义所有点的公共祖先是：LCA 。
那么答案是所有点到 LCA 的圆点的个数。注意去重。

Luogu P4606 [SDOI2018] 战略游戏 Solution

一种简单但是比较恶心的想法。

Luogu P4606 [SDOI2018] 战略游戏 Solution

一种简单但是比较恶心的想法。
设所有关键点的序列为 $city_i$。

Luogu P4606 [SDOI2018] 战略游戏 Solution

一种简单但是比较恶心的想法。
设所有关键点的序列为 $city_i$。
从左到右每次求 $city_i, city_{i-1}$ 的 lca，并把 lca 赋值给 $city_i$。
(就是求当前所有节点的 lca)。

Luogu P4606 [SDOI2018] 战略游戏 Solution

一种简单但是比较恶心的想法。
设所有关键点的序列为 $city_i$。
从左到右每次求 $city_i, city_{i-1}$ 的 lca，并把 lca 赋值给 $city_i$。
(就是求当前所有节点的 lca)。
统计答案的话直接树剖，注意每次求解后要把此路径清零。

Luogu P4606 [SDOI2018] 战略游戏 Solution

一种简单但是比较恶心的想法。
设所有关键点的序列为 $city_i$。
从左到右每次求 $city_i, city_{i-1}$ 的 lca，并把 lca 赋值给 $city_i$。
(就是求当前所有节点的 lca)。
统计答案的话直接树剖，注意每次求解后要把此路径清零。

额，反正我打了 7 kib 放弃了。
而且可能超时。

Luogu P4606 [SDOI2018] 战略游戏 Solution

来看另一种码量较小的做法。

Luogu P4606 [SDOI2018] 战略游戏 Solution

来看另一种码量较小的做法。
把上页的问题转化一下：

Luogu P4606 [SDOI2018] 战略游戏 Solution

来看另一种码量较小的做法。
把上页的问题转化一下：
所有关键点所在最小联通块中包含圆点的个数。

Luogu P4606 [SDOI2018] 战略游戏 Solution

来看另一种码量较小的做法。
把上页的问题转化一下：
所有关键点所在最小联通块中包含圆点的个数。
设 dis_i 表示根到 i 一路上圆点的个数。

Luogu P4606 [SDOI2018] 战略游戏 Solution

来看另一种码量较小的做法。
把上页的问题转化一下：
所有关键点所在最小联通块中包含圆点的个数。
设 dis_i 表示根到 i 一路上圆点的个数。
那么求任意两点的圆点个数可以用类似求树上两点间距离的做法解决。

Luogu P4606 [SDOI2018] 战略游戏 Solution

来看另一种码量较小的做法。
把上页的问题转化一下：
所有关键点所在最小联通块中包含圆点的个数。
设 dis_i 表示根到 i 一路上圆点的个数。
那么求任意两点的圆点个数可以用类似求树上两点间距离的做法解决。考虑把所有节点按照欧拉序排序。

Luogu P4606 [SDOI2018] 战略游戏 Solution

来看另一种码量较小的做法。
把上页的问题转化一下：
所有关键点所在最小联通块中包含圆点的个数。
设 dis_i 表示根到 i 一路上圆点的个数。
那么求任意两点的圆点个数可以用类似求树上两点间距离的做法解决。考虑把所有节点按照欧拉序排序。
计算相邻两点和 $1, n$ 两点间圆点的个数和。

Luogu P4606 [SDOI2018] 战略游戏 Solution

来看另一种码量较小的做法。
把上页的问题转化一下：
所有关键点所在最小联通块中包含圆点的个数。
设 dis_i 表示根到 i 一路上圆点的个数。
那么求任意两点的圆点个数可以用类似求树上两点间距离的做法解决。考虑把所有节点按照欧拉序排序。
计算相邻两点和 $1, n$ 两点间圆点的个数和。
画一个图可以发现，每一条边都计算了两次。

Luogu P4606 [SDOI2018] 战略游戏 Solution

来看另一种码量较小的做法。
把上页的问题转化一下：
所有关键点所在最小联通块中包含圆点的个数。
设 dis_i 表示根到 i 一路上圆点的个数。
那么求任意两点的圆点个数可以用类似求树上两点间距离的做法解决。考虑把所有节点按照欧拉序排序。
计算相邻两点和 $1, n$ 两点间圆点的个数和。
画一个图可以发现，每一条边都计算了两次。
对于结果除以二即可。
注意要考虑根节点是否有贡献，因为我们把其转化成了边上问题。

P3687 [ZJOI2017] 仙人掌

题目描述：

给定一个无向连通图，添加一些边，使其变为仙人掌，问方案数。

P3687 [ZJOI2017] 仙人掌

题目描述：

给定一个无向连通图，添加一些边，使其变为仙人掌，问方案数。

如果一个无自环无重边无向连通图的任意一条边最多属于一个简单环，我们就称之为仙人掌。所谓简单环即不经过重复的结点的环。

P3687 [ZJOI2017] 仙人掌 Solution

大家都一眼秒了吧。

P3687 [ZJOI2017] 仙人掌 Solution

大家都一眼秒了吧。
这道题好像重点不是仙人掌，不过还是分享一下吧。

P3687 [ZJOI2017] 仙人掌 Solution

大家都一眼秒了吧。
这道题好像重点不是仙人掌，不过还是分享一下吧。
首先考虑无解的情况。

P3687 [ZJOI2017] 仙人掌 Solution

大家都一眼秒了吧。
这道题好像重点不是仙人掌，不过还是分享一下吧。
首先考虑无解的情况。
根据定义，如果给出的图不是仙人掌，答案一定是 0 。

P3687 [ZJOI2017] 仙人掌 Solution

大家都一眼秒了吧。
这道题好像重点不是仙人掌，不过还是分享一下吧。
首先考虑无解的情况。
根据定义，如果给出的图不是仙人掌，答案一定是 0 。
所以先来考虑怎么判断一个图是仙人掌。

P3687 [ZJOI2017] 仙人掌 Solution

考虑直接从图上拉下来一颗 DFS 树。

P3687 [ZJOI2017] 仙人掌 Solution

考虑直接从图上拉下来一颗 DFS 树。
直接类似 $Tarjan$ 的方法。

P3687 [ZJOI2017] 仙人掌 Solution

考虑直接从图上拉下来一颗 DFS 树。
直接类似 $Tarjan$ 的方法。
如果对于某一个点出现了两次返祖边直接得出不是仙人掌。

P3687 [ZJOI2017] 仙人掌 Solution

考虑直接从图上拉下来一颗 DFS 树。
直接类似 $Tarjan$ 的方法。
如果对于某一个点出现了两次返祖边直接得出不是仙人掌。

有问题的话可以等一下看代码。

P3687 [ZJOI2017] 仙人掌 Solution

因为原来是环的地方我们不会再加边。

P3687 [ZJOI2017] 仙人掌 Solution

因为原来是环的地方我们不会再加边。
所以我们可以尝试把环拆散，变成若干个森林进行求解。

P3687 [ZJOI2017] 仙人掌 Solution

因为原来是环的地方我们不会再加边。
所以我们可以尝试把环拆散，变成若干个森林进行求解。
这一步可以直接在 $Tarjan$ 的过程中进行。

P3687 [ZJOI2017] 仙人掌 Solution

因为原来是环的地方我们不会再加边。
所以我们可以尝试把环拆散，变成若干个森林进行求解。
这一步可以直接在 $Tarjan$ 的过程中进行。
当然，最后的答案是森林中所有树方案的乘积。

P3687 [ZJOI2017] 仙人掌 Solution

因为原来是环的地方我们不会再加边。
所以我们可以尝试把环拆散，变成若干个森林进行求解。
这一步可以直接在 $Tarjan$ 的过程中进行。
当然，最后的答案是森林中所有树方案的乘积。

考虑转化问题。

P3687 [ZJOI2017] 仙人掌 Solution

因为原来是环的地方我们不会再加边。
所以我们可以尝试把环拆散，变成若干个森林进行求解。
这一步可以直接在 $Tarjan$ 的过程中进行。
当然，最后的答案是森林中所有树方案的乘积。

考虑转化问题。
我们可以把一棵树统一变成仙人掌。
对于不在环中的边，我们可以连一条重边使其变成环。

P3687 [ZJOI2017] 仙人掌 Solution

因为原来是环的地方我们不会再加边。
所以我们可以尝试把环拆散，变成若干个森林进行求解。
这一步可以直接在 $Tarjan$ 的过程中进行。
当然，最后的答案是森林中所有树方案的乘积。

考虑转化问题。
我们可以把一棵树统一变成仙人掌。
对于不在环中的边，我们可以连一条重边使其变成环。
发现问题可以转化为用链覆盖树的方案数。

P3687 [ZJOI2017] 仙人掌 Solution

接下来考虑树形 DP。

P3687 [ZJOI2017] 仙人掌 Solution

接下来考虑树形 DP。
考虑到只有两种情况：没有链连到父亲，有链连到父亲。
同时如果有链，那么一定是一条。

P3687 [ZJOI2017] 仙人掌 Solution

接下来考虑树形 DP。
考虑到只有两种情况：没有链连到父亲，有链连到父亲。
同时如果有链，那么一定是一条。

我们设 f_i 和 g_i 分别表示以 i 为根的以上两种情况。
同时定义 $snum_i$ 为当前 i 节点的儿子个数。

P3687 [ZJOI2017] 仙人掌 Solution

接下来考虑树形 DP。
考虑到只有两种情况：没有链连到父亲，有链连到父亲。
同时如果有链，那么一定是一条。

我们设 f_i 和 g_i 分别表示以 i 为根的以上两种情况。
同时定义 $snum_i$ 为当前 i 节点的儿子个数。

然后来看怎么转移。

P3687 [ZJOI2017] 仙人掌 Solution

先来看第一种情况。

P3687 [ZJOI2017] 仙人掌 Solution

先来看第一种情况。

考虑到每个 i 的儿子都有两种可能：
1. 向外连到了 i 节点。 2. 没有向外连向 i 节点。

P3687 [ZJOI2017] 仙人掌 Solution

先来看第一种情况。

考虑到每个 i 的儿子都有两种可能：
1. 向外连到了 i 节点。 2. 没有向外连向 i 节点。

同时儿子们还可能内部消化（我在说什么）

P3687 [ZJOI2017] 仙人掌 Solution

先来看第一种情况。

考虑到每个 i 的儿子都有两种可能：
1. 向外连到了 i 节点。 2. 没有向外连向 i 节点。

同时儿子们还可能内部消化（我在说什么）
我们定义 num 个儿子互相连边的方案数是 p_{num}

P3687 [ZJOI2017] 仙人掌 Solution

先来看第一种情况。

考虑到每个 i 的儿子都有两种可能：
1. 向外连到了 i 节点。 2. 没有向外连向 i 节点。

同时儿子们还可能内部消化（我在说什么）
我们定义 num 个儿子互相连边的方案数是 p_{num}
考虑当前儿子连不连可以得出递推式：

$$p_i = p_{i-1} + (i-1) \times p_{i-2}$$

所以第一种情况的 DP 转移方程是：

$$f_i = p_{snum_i} \prod_{x \in son_i} g_x$$

P3687 [ZJOI2017] 仙人掌 Solution

再来考虑第二种情况。

P3687 [ZJOI2017] 仙人掌 Solution

再来考虑第二种情况。

首先他自己的 f_i 一定是一种可行的方案。

P3687 [ZJOI2017] 仙人掌 Solution

再来考虑第二种情况。

首先他自己的 f_i 一定是一种可行的方案。
然后我们在考虑枚举是它的哪一个孩子连边上来的。

P3687 [ZJOI2017] 仙人掌 Solution

再来考虑第二种情况。

首先他自己的 f_i 一定是一种可行的方案。
然后我们在考虑枚举是它的哪一个孩子连边上来的。
对于上述的每一个情况 i 的其它孩子只能选择不向上连边，所以和 f_i 转移一样。

P3687 [ZJOI2017] 仙人掌 Solution

再来考虑第二种情况。

首先他自己的 f_i 一定是一种可行的方案。
然后我们在考虑枚举是它的哪一个孩子连边上来的。
对于上述的每一个情况 i 的其它孩子只能选择不向上连边，所以和 f_i 转移一样。

$$g_i = f_i + \sum_{x \in son_i} g_x \times p_{snum_i - 1} \prod_{y \in son_i, y \neq x} g_y$$

P3687 [ZJOI2017] 仙人掌 Solution

但是这个式子好像不太友好，我们化简一下：

P3687 [ZJOI2017] 仙人掌 Solution

但是这个式子好像不太友好，我们化简一下：

设 $\prod_{x \in son_i} g_x$ 为 G_i 。

$$g_i = f_i + \sum_{x \in son_i} g_x \times p_{snum_i - 1} \prod_{y \in son_i, y \neq x} g_y$$

$$g_i = f_i + \sum_{x \in son_i} g_x \times p_{snum_i - 1} \frac{G_i}{g_x}$$

P3687 [ZJOI2017] 仙人掌 Solution

但是这个式子好像不太友好，我们化简一下：

设 $\prod_{x \in son_i} g_x$ 为 G_i 。

$$g_i = f_i + \sum_{x \in son_i} g_x \times p_{snum_i - 1} \prod_{y \in son_i, y \neq x} g_y$$

$$g_i = f_i + \sum_{x \in son_i} g_x \times p_{snum_i - 1} \frac{G_i}{g_x}$$

$$g_i = f_i + p_{snum_i - 1} \times G_i$$

P3687 [ZJOI2017] 仙人掌 Solution

但是这个式子好像不太友好，我们化简一下：

设 $\prod_{x \in son_i} g_x$ 为 G_i 。

$$g_i = f_i + \sum_{x \in son_i} g_x \times p_{snum_i - 1} \prod_{y \in son_i, y \neq x} g_y$$

$$g_i = f_i + \sum_{x \in son_i} g_x \times p_{snum_i - 1} \frac{G_i}{g_x}$$

$$g_i = f_i + p_{snum_i - 1} \times G_i$$

然后直接树形 DP 就可以啦。

CF487E Tourists

题目描述：

给定一张简单无向连通图，要求支持两种操作：
1. 修改一个点的点权。
2. 询问两点之间所有简单路径上点权的最小值。

$1 \leq n, m, q \leq 10^5$

CF487E Tourists Solution

大家都一眼秒了吧。/kk

CF487E Tourists Solution

大家都一眼秒了吧。/kk
看到这道题首先想到的此题的树上版本。(不就是树链剖分的板子题么?)

CF487E Tourists Solution

大家都一眼秒了吧。/kk
看到这道题首先想到的此题的树上版本。(不就是树链剖分的板子题么?)
但是此题是图上的两点间的走法，自然要想到是圆方树。
我们先无脑构建出圆方树。

CF487E Tourists Solution

大家都一眼秒了吧。/kk
看到这道题首先想到的此题的树上版本。(不就是树链剖分的板子题么?)
但是此题是图上的两点间的走法，自然要想到是圆方树。
我们先无脑构建出圆方树。
我们先猜测：设后加入的节点权值为 inf ，直接再圆方树上做述链剖分?

CF487E Tourists Solution

大家都一眼秒了吧。/kk
看到这道题首先想到的此题的树上版本。(不就是树链剖分的板子题么?)
但是此题是图上的两点间的走法，自然要想到是圆方树。
我们先无脑构建出圆方树。
我们先猜测：设后加入的节点权值为 inf ，直接再圆方树上做述链剖分?
看上去很简单，但是完全不对，考虑同一个点双的情况。
(提示：可以绕道走啊!)

CF487E Tourists Solution

大家都一眼秒了吧。/kk
看到这道题首先想到的此题的树上版本。(不就是树链剖分的板子题么?)
但是此题是图上的两点间的走法，自然要想到是圆方树。
我们先无脑构建出圆方树。
我们先猜测：设后加入的节点权值为 inf ，直接再圆方树上做述链剖分?
看上去很简单，但是完全不对，考虑同一个点双的情况。
(提示：可以绕道走啊!)

他们在圆方树中是兄弟的关系，根本剖不到上面去，但是同样对答案有贡献。

CF487E Tourists Solution

大家都一眼秒了吧。/kk
看到这道题首先想到的此题的树上版本。(不就是树链剖分的板子题么?)
但是此题是图上的两点间的走法，自然要想到是圆方树。
我们先无脑构建出圆方树。
我们先猜测：设后加入的节点权值为 inf ，直接再圆方树上做述链剖分?
看上去很简单，但是完全不对，考虑同一个点双的情况。
(提示：可以绕道走啊!)

他们在圆方树中是兄弟的关系，根本剖不到上面去，但是同样对答案有贡献。
所以不能树链剖分去做？

CF487E Tourists Solution

我们模拟一下数据。（这个图在之后会很常见）

CF487E Tourists Solution

我们模拟一下数据。(这个图在之后会很常见)
有一个大胆的设想:

CF487E Tourists Solution

我们模拟一下数据。(这个图在之后会很常见)
有一个大胆的设想:
因为每一个新建的方形节点都对应一个点双。
可以把新建的节点权值更新成和它相连的所有节点权值的最小值?

CF487E Tourists Solution

我们模拟一下数据。(这个图在之后会很常见)
有一个大胆的设想:
因为每一个新建的方形节点都对应一个点双。
可以把新建的节点权值更新成和它相连的所有节点权值的最小值?
答案是可以的。

CF487E Tourists Solution

我们模拟一下数据。(这个图在之后会很常见)
有一个大胆的设想:
因为每一个新建的方形节点都对应一个点双。
可以把新建的节点权值更新成和它相连的所有节点权值的最小值?
答案是可以的。
于是我们得到以下思路:

CF487E Tourists Solution

我们模拟一下数据。(这个图在之后会很常见)
有一个大胆的设想:
因为每一个新建的方形节点都对应一个点双。
可以把新建的节点权值更新成和它相连的所有节点权值的最小值?
答案是可以的。
于是我们得到以下思路:
1. 先构建出圆方树。
2. 每个新节点为与它相连的节点的权值最小值。
3. 树链剖分，每次修改时更新对应的新节点。

CF487E Tourists Solution

看上去非常完美，结果 TLE 在了第 18 个点。

CF487E Tourists Solution

看上去非常完美，结果 TLE 在了第 18 个点。
来看一个图。（它是一个菊花图）

CF487E Tourists Solution

看上去非常完美，结果 TLE 在了第 18 个点。
来看一个图。(它是一个菊花图)
发现复杂度变成了 $O(n^2)$

CF487E Tourists Solution

看上去非常完美，结果 TLE 在了第 18 个点。
来看一个图。(它是一个菊花图)
发现复杂度变成了 $O(n^2)$
那么我们之前的要推翻重来吗？

CF487E Tourists Solution

看上去非常完美，结果 TLE 在了第 18 个点。
来看一个图。(它是一个菊花图)
发现复杂度变成了 $O(n^2)$
那么我们之前的要推翻重来吗？
不是我们来考虑圆方树的性质。

CF487E Tourists Solution

看上去非常完美，结果 TLE 在了第 18 个点。
来看一个图。(它是一个菊花图)
发现复杂度变成了 $O(n^2)$
那么我们之前的要推翻重来吗？
不是我们来考虑圆方树的性质。
因为圆方树就是树，我们直接从树的性质入手。

CF487E Tourists Solution

看上去非常完美，结果 TLE 在了第 18 个点。
来看一个图。(它是一个菊花图)
发现复杂度变成了 $O(n^2)$
那么我们之前的要推翻重来吗？
不是我们来考虑圆方树的性质。
因为圆方树就是树，我们直接从树的性质入手。
考虑到每个树上节点只会有一个父亲，那么我们可以把一个方节点记录其子节点的最小值。

CF487E Tourists Solution

看上去非常完美，结果 TLE 在了第 18 个点。
来看一个图。(它是一个菊花图)
发现复杂度变成了 $O(n^2)$
那么我们之前的要推翻重来吗？
不是我们来考虑圆方树的性质。
因为圆方树就是树，我们直接从树的性质入手。
考虑到每个树上节点只会有一个父亲，那么我们可以把一个方节点记录其子节点的最小值。
所以有什么区别么？有！这样的话我们每个更新就从他的所连接点变成了父亲节点！

CF487E Tourists Solution

看上去非常完美，结果 TLE 在了第 18 个点。
来看一个图。(它是一个菊花图)
发现复杂度变成了 $O(n^2)$
那么我们之前的要推翻重来吗？
不是我们来考虑圆方树的性质。
因为圆方树就是树，我们直接从树的性质入手。
考虑到每个树上节点只会有一个父亲，那么我们可以把一个方节点记录其子节点的最小值。
所以有什么区别么？有！这样的话我们每个更新就从他的所连接点变成了父亲节点！
我们对每一个方形节点建一棵平衡树，维护一下最小值。
(我用的是 $multiset$，有人用 $Treap$ 我也不反对 /tuu)

CF487E Tourists Solution

看上去非常完美，结果 TLE 在了第 18 个点。
来看一个图。(它是一个菊花图)
发现复杂度变成了 $O(n^2)$
那么我们之前的要推翻重来吗？
不是我们来考虑圆方树的性质。
因为圆方树就是树，我们直接从树的性质入手。
考虑到每个树上节点只会有一个父亲，那么我们可以把一个方节点记录其子节点的最小值。
所以有什么区别么？有！这样的话我们每个更新就从他的所连接点变成了父亲节点！
我们对每一个方形节点建一棵平衡树，维护一下最小值。
(我用的是 $multiset$，有人用 $Treap$ 我也不反对 /tuu)

注意：当两个节点的 lca 是方形节点时，要考虑方形节点的的父亲节点。

P6966 Cactus Construction

题目描述：

有 n 个集合，初始第 i 个集合只包含第 i 个点。
每个点的初始颜色都为 1。要求使用下列 3 种操作构建出给定的仙人掌。
输出方案，操作数不得多于 10^6。

1.$j\ a\ b$，将 a 所在的集合和 b 所在的集合合并成一个集合。
2.$r\ a\ c_1\ c_2$，将 a 所在的集合中，所有颜色为 c_1 的点的颜色改为 c_2。
3.$c\ a\ c_1\ c_2$，将 a 所在的集合中，每一个颜色为 c_1 的点连向每一个颜色为 c_2 的点，**不允许出现重边**。

P6966 Cactus Construction Solution

大家都切了吧。

P6966 Cactus Construction Solution

大家都切了吧。

先不要考虑仙人掌的情况，看看树上情况如何构造。

P6966 Cactus Construction Solution

大家都切了吧。
先不要考虑仙人掌的情况，看看树上情况如何构造。
经过思考，发现树上情况时最多只需要三种颜色。

P6966 Cactus Construction Solution

大家都切了吧。
先不要考虑仙人掌的情况，看看树上情况如何构造。
经过思考，发现树上情况时最多只需要三种颜色。

col_1 表示这个点所有的边都相连了。
col_2 表示子树全部完成，但是还没有和父亲连接。
col_3 表示子树都没有建好。

P6966 Cactus Construction Solution

大家都切了吧。
先不要考虑仙人掌的情况，看看树上情况如何构造。
经过思考，发现树上情况时最多只需要三种颜色。

col_1 表示这个点所有的边都相连了。
col_2 表示子树全部完成，但是还没有和父亲连接。
col_3 表示子树都没有建好。

三种颜色的表示不断递进的三种状态。

P6966 Cactus Construction Solution

一开始所有的点颜色一定是 col_3。

P6966 Cactus Construction Solution

一开始所有的点颜色一定是 col_3。
那么很显然我们从叶子节点向上不断进行构造连边。

P6966 Cactus Construction Solution

一开始所有的点颜色一定是 col_3。
那么很显然我们从叶子节点向上不断进行构造连边。
不难发现此时 now 的儿子们颜色都是 col_2。

P6966 Cactus Construction Solution

一开始所有的点颜色一定是 col_3。
那么很显然我们从叶子节点向上不断进行构造连边。
不难发现此时 now 的儿子们颜色都是 col_2。

在这里给出算法流程：

P6966 Cactus Construction Solution

一开始所有的点颜色一定是 col_3。
那么很显然我们从叶子节点向上不断进行构造连边。
不难发现此时 now 的儿子们颜色都是 col_2。

在这里给出算法流程：
1. 将 now 和其儿子 son 连边。
2. 所有颜色时 col_3 的变成 col_2。
3. 将颜色 col_2 变成 col_1

P6966 Cactus Construction Solution

一开始所有的点颜色一定是 col_3。
那么很显然我们从叶子节点向上不断进行构造连边。
不难发现此时 now 的儿子们颜色都是 col_2。

在这里给出算法流程：
1. 将 now 和其儿子 son 连边。
2. 所有颜色时 col_3 的变成 col_2。
3. 将颜色 col_2 变成 col_1

也是很好理解的捏。

P6966 Cactus Construction Solution

考虑仙人掌的构造方法。

P6966 Cactus Construction Solution

考虑仙人掌的构造方法。
要建圆方树吗？

P6966 Cactus Construction Solution

考虑仙人掌的构造方法。
要建圆方树吗？
其实是不需要的。
因为已知是一颗仙人掌，所以可以直接拉出 DFS 树。
然后判断返祖边即可。

P6966 Cactus Construction Solution

考虑仙人掌的构造方法。
要建圆方树吗？
其实是不需要的。
因为已知是一颗仙人掌，所以可以直接拉出 DFS 树。
然后判断返祖边即可。

这时需要引入 col_4 表示当前点没有连返祖边。

P6966 Cactus Construction Solution

考虑仙人掌的构造方法。
要建圆方树吗？
其实是不需要的。
因为已知是一颗仙人掌，所以可以直接拉出 DFS 树。
然后判断返祖边即可。

这时需要引入 col_4 表示当前点没有连返祖边。
考虑当前节点 now 和其儿子 son 的关系：

P6966 Cactus Construction Solution

考虑仙人掌的构造方法。
要建圆方树吗？
其实是不需要的。
因为已知是一颗仙人掌，所以可以直接拉出 DFS 树。
然后判断返祖边即可。

这时需要引入 col_4 表示当前点没有连返祖边。
考虑当前节点 now 和其儿子 son 的关系：
1. son 不在环里，直接按照树的方法进行处理。
2. son 所在的环正好经过 now，那么直接连接两点，$col_4 \to col_1$
3. son 所在的环深度小于 $deep_{now}$，把 son 节点直接赋值 col_4。

P6966 Cactus Construction Solution

考虑仙人掌的构造方法。
要建圆方树吗？
其实是不需要的。
因为已知是一颗仙人掌，所以可以直接拉出 DFS 树。
然后判断返祖边即可。

这时需要引入 col_4 表示当前点没有连返祖边。
考虑当前节点 now 和其儿子 son 的关系：
1. son 不在环里，直接按照树的方法进行处理。
2. son 所在的环正好经过 now，那么直接连接两点，$col_4 \to col_1$
3. son 所在的环深度小于 $deep_{now}$，把 son 节点直接赋值 col_4。

为了防止冲突，我们把 low 按从大到小排序，再分类讨论。

P3731 新型城市化

题目描述：

感觉不太好讲，大家直接看题目吧。

P3731 新型城市化 Solution

首先根据题意我们可以得知所有城市分成互不干扰的两部分。

P3731 新型城市化 Solution

首先根据题意我们可以得知所有城市分成互不干扰的两部分。
所以对于本题中所给出的补图实际上是一个 **二分图**。

P3731 新型城市化 Solution

首先根据题意我们可以得知所有城市分成互不干扰的两部分。所以对于本题中所给出的补图实际上是一个 **二分图**。

再继续观察我们可以发现要求的原图最大团就是补图 **最大独立集**。

P3731 新型城市化 Solution

首先根据题意我们可以得知所有城市分成互不干扰的两部分。
所以对于本题中所给出的补图实际上是一个 **二分图**。

再继续观察我们可以发现要求的原图最大团就是补图 **最大独立集**。
因为最大团中的点两两相连，所以补图上表示为没有任何联系的点的集合，就是 **最大独立集**。

我们考虑怎么求最大独立集：

P3731 新型城市化 Solution

首先根据题意我们可以得知所有城市分成互不干扰的两部分。
所以对于本题中所给出的补图实际上是一个 **二分图**。

再继续观察我们可以发现要求的原图最大团就是补图 **最大独立集**。
因为最大团中的点两两相连，所以补图上表示为没有任何联系的点的集合，就是 **最大独立集**。

我们考虑怎么求最大独立集：
这里给出结论：最大独立集 = 点数 - 最大匹配

P3731 新型城市化 Solution

首先根据题意我们可以得知所有城市分成互不干扰的两部分。
所以对于本题中所给出的补图实际上是一个 **二分图**。

再继续观察我们可以发现要求的原图最大团就是补图 **最大独立集**。
因为最大团中的点两两相连，所以补图上表示为没有任何联系的点的集合，就是 **最大独立集**。

我们考虑怎么求最大独立集：
这里给出结论：最大独立集 = 点数 - 最大匹配
要证明的话大家自己证一下吧。

P3731 新型城市化 Solution

首先根据题意我们可以得知所有城市分成互不干扰的两部分。
所以对于本题中所给出的补图实际上是一个 **二分图**。

再继续观察我们可以发现要求的原图最大团就是补图 **最大独立集**。
因为最大团中的点两两相连，所以补图上表示为没有任何联系的点的集合，就是 **最大独立集**。

我们考虑怎么求最大独立集：
这里给出结论：最大独立集 = 点数 - 最大匹配
要证明的话大家自己证一下吧。

所以我们现在就是求那些边删掉之后**最大独立集**变大，也就是要把**最大匹配数**减小。

P3731 新型城市化 Solution

首先根据题意我们可以得知所有城市分成互不干扰的两部分。
所以对于本题中所给出的补图实际上是一个 **二分图**。

再继续观察我们可以发现要求的原图最大团就是补图 **最大独立集**。
因为最大团中的点两两相连，所以补图上表示为没有任何联系的点的集合，就是 **最大独立集**。

我们考虑怎么求最大独立集：
这里给出结论：最大独立集 $=$ 点数 - 最大匹配
要证明的话大家自己证一下吧。

所以我们现在就是求那些边删掉之后**最大独立集**变大，也就是要把**最大匹配数**减小。

直接来个匈牙利跑个暴力就能得到 $60\ pts$ 的好成绩。

P3731 新型城市化 Solution

考虑 $100\ pts$ 的满分做法。

P3731 新型城市化 Solution

考虑 $100\ pts$ 的满分做法。

把题目转换为：求出二分图中关键边的个数。
一条边是关建边当且仅当满足之前我们要求的性质。

P3731 新型城市化 Solution

考虑 $100\ pts$ 的满分做法。

把题目转换为：求出二分图中关键边的个数。
一条边是关建边当且仅当满足之前我们要求的性质。

应用我们先跑个网络流，如果当前的边以跑满则说明这条边是二分图最大比配中的边，**但不一定是关建边**。

P3731 新型城市化 Solution

考虑 $100\ pts$ 的满分做法。

把题目转换为：求出二分图中关键边的个数。
一条边是关建边当且仅当满足之前我们要求的性质。

应用我们先跑个网络流，如果当前的边以跑满则说明这条边是二分图最大比配中的边，**但不一定是关建边**。
然后我们判断这条边的两个端点是不是在一个**强联通分量**中。

P3731 新型城市化 Solution

考虑 $100\ pts$ 的满分做法。

把题目转换为：求出二分图中关键边的个数。
一条边是关建边当且仅当满足之前我们要求的性质。

应用我们先跑个网络流，如果当前的边以跑满则说明这条边是二分图最大比配中的边，**但不一定是关建边**。
然后我们判断这条边的两个端点是不是在一个**强联通分量**中。
如果不是就是关建边。

P3731 新型城市化 Solution

来看看第二个结论的证明：

P3731 新型城市化 Solution

来看看第二个结论的证明：

显然，如果在一个强联通分量中，删掉这条边，会有另一边顶替。

[ZJOI2007] 最大半连通子图

题目描述：
题目还是比较不好描述，大家直接看吧。

[ZJOI2007] 最大半连通子图

题目描述：
题目还是比较不好描述，大家直接看吧。

这里就给出半连通子图的定义：
对任意 $u, v \in V$，满足 $u \to v$ 或 $v \to u$，即对于图中任意两点 u, v，存在一条 u 到 v 的有向路径或者从 v 到 u 的有向路径。

[ZJOI2007] 最大半连通子图 Solution

稍稍看一下，可以看出一个强联通分量一定是半连通子图。
（注意：可以是一条或是两条都可以）

[ZJOI2007] 最大半连通子图 Solution

稍稍看一下，可以看出一个强联通分量一定是半连通子图。
(注意：可以是一条或是两条都可以)
同时呢，可以发现对于一个有向无环图一条路径其实都是半连通子图

[ZJOI2007] 最大半连通子图 Solution

稍稍看一下，可以看出一个强联通分量一定是半连通子图。
(注意：可以是一条或是两条都可以)
同时呢，可以发现对于一个有向无环图一条路径其实都是半连通子图

那直接先跑一个 $Tarjan$ 求出强连通分量，然后缩点。

[ZJOI2007] 最大半连通子图 Solution

稍稍看一下，可以看出一个强联通分量一定是半连通子图。
(注意：可以是一条或是两条都可以)
同时呢，可以发现对于一个有向无环图一条路径其实都是半连通子图

那直接先跑一个 $Tarjan$ 求出强连通分量，然后缩点。
同时我们把边权赋值为强联通分量的点的个数。

[ZJOI2007] 最大半连通子图 Solution

然后直接把问题转换成了：

[ZJOI2007] 最大半连通子图 Solution

然后直接把问题转换成了：
给你一棵树，求最长路径的长度以及出现的次数。

[ZJOI2007] 最大半连通子图 Solution

然后直接把问题转换成了：
给你一棵树，求最长路径的长度以及出现的次数。

至于为什么是树，直接跑一个拓扑排序就好了。

[ZJOI2007] 最大半连通子图 Solution

然后直接把问题转换成了：
给你一棵树，求最长路径的长度以及出现的次数。

至于为什么是树，直接跑一个拓扑排序就好了。
然后直接 dp 就好了。

[ZJOI2007] 最大半连通子图 Solution

然后直接把问题转换成了：
给你一棵树，求最长路径的长度以及出现的次数。

至于为什么是树，直接跑一个拓扑排序就好了。
然后直接 dp 就好了。

```
for (int t : dis[now]) {
  if (len[t] < len[now] + siz[t]) {
    len[t] = len[now] + siz[t];
    ans[t] = ans[now];
  } else if (len[t] == len[now] + siz[t]){
    ans[t] = (ans[t] % mod + ans[now]) % mod;
  }
}
```

FZOJ 3608

题目描述：
给你一个 $0/1$ 矩阵，求出最小的 k, d 满足 $A^k = A^{k+d}$。
注：计算幂时要用布尔运算，即乘法为与，加法为或。

FZOJ 3608

题目描述：
给你一个 $0/1$ 矩阵，求出最小的 k, d 满足 $A^k = A^{k+d}$。
注：计算幂时要用布尔运算，即乘法为与，加法为或。

注意，只有在 n 较小时才需要求 k，之前没看到想了一个下午。

FZOJ 3608 Sulotion

发现大家都会做……

FZOJ 3608 Sulotion

发现大家都会做……

但还是讲一下吧。

FZOJ 3608 Sulotion

发现大家都会做……

但还是讲一下吧。
考虑 $0/1$ 矩阵的多次幂的性质：
对于 n 次幂也就是 A^n 。
$A^n_{i,j} = 1$ 表示走 n 步能从 i 到 j。
$A^n_{i,j} = 0$ 表示走 n 步不能从 i 到 j。

FZOJ 3608 Sulotion

发现大家都会做……

但还是讲一下吧。
考虑 $0/1$ 矩阵的多次幂的性质：
对于 n 次幂也就是 A^n 。
$A^n_{i,j} = 1$ 表示走 n 步能从 i 到 j。
$A^n_{i,j} = 0$ 表示走 n 步不能从 i 到 j。

可以发现，这个 d 显然和强联通分量有关的。

FZOJ 3608 Sulotion

发现大家都会做……

但还是讲一下吧。
考虑 $0/1$ 矩阵的多次幂的性质：
对于 n 次幂也就是 A^n 。
$A^n_{i,j}=1$ 表示走 n 步能从 i 到 j。
$A^n_{i,j}=0$ 表示走 n 步不能从 i 到 j。

可以发现，这个 d 显然和强联通分量有关的。
我们设共有 n 个强连通分量，对于第 i 个强联通分量，我们设它的循环的长度是 len_i，那么很容易得到：

$$d=\mathrm{lcm}(len_1,len_2,\cdots,len_{n-1})$$

FZOJ 3608 Sulotion

现在问题是求出每一个强联通分量的循环长度 len_i 。

FZOJ 3608 Sulotion

现在问题是求出每一个强联通分量的循环长度 len_i 。
先给出结论： len_i 等于强联通分量中所有环长度的 $\gcd$ 。

FZOJ 3608 Sulotion

现在问题是求出每一个强联通分量的循环长度 len_i 。
先给出结论： len_i 等于强联通分量中所有环长度的 $\gcd$ 。
代码就像这样：

```
1  inline void dfs(int now) {
2    for (int t : dis[now]) {
3      if (col[now] != col[t]) continue;
4      if (deep[t] == 0) {
5        deep[t] = deep[now] + 1;
6        dfs(t);
7      } else if (deep[t])
8        circle = std::__gcd(circle, std::abs(deep[t] - deep[now])
             + 1);
9    }
10 }
```

FZOJ 3608 Sulotion

大家先来想一下这个结论的证明。

FZOJ 3608 Sulotion

大家先来想一下这个结论的证明。
大家应该都证出来了，在这里我来口胡一下。

FZOJ 3608 Sulotion

大家先来想一下这个结论的证明。
大家应该都证出来了，在这里我来口胡一下。

看到有 $\gcd$ ，想到互质也就是 $\gcd$ 为 1 的情况，其它的情况可以直接按照比例扩大即可。

FZOJ 3608 Sulotion

大家先来想一下这个结论的证明。
大家应该都证出来了，在这里我来口胡一下。

看到有 $\gcd$ ，想到互质也就是 $\gcd$ 为 1 的情况，其它的情况可以直接按照比例扩大即可。
由结论得知，当互质时强联通分量的循环节长度为 1 。

FZOJ 3608 Sulotion

大家先来想一下这个结论的证明。
大家应该都证出来了，在这里我来口胡一下。

看到有 $\gcd$ ，想到互质也就是 $\gcd$ 为 1 的情况，其它的情况可以直接按照比例扩大即可。
由结论得知，当互质时强联通分量的循环节长度为 1 。
不难发现，如果从所有 st 点出发，走 i 步和走 $i+1$ 步能走到的点的集合完全一样，那么只有一种可能：**就是邻接矩阵所有的元素都是** 1 。

FZOJ 3608 Sulotion

大家先来想一下这个结论的证明。
大家应该都证出来了，在这里我来口胡一下。

看到有 $\gcd$ ，想到互质也就是 $\gcd$ 为 1 的情况，其它的情况可以直接按照比例扩大即可。
由结论得知，当互质时强联通分量的循环节长度为 1 。
不难发现，如果从所有 st 点出发，走 i 步和走 $i+1$ 步能走到的点的集合完全一样，那么只有一种可能：**就是邻接矩阵所有的元素都是** 1 。
现在问题转化成几个互质的环套在一起能在某一个时刻所有的点都能互相走到。

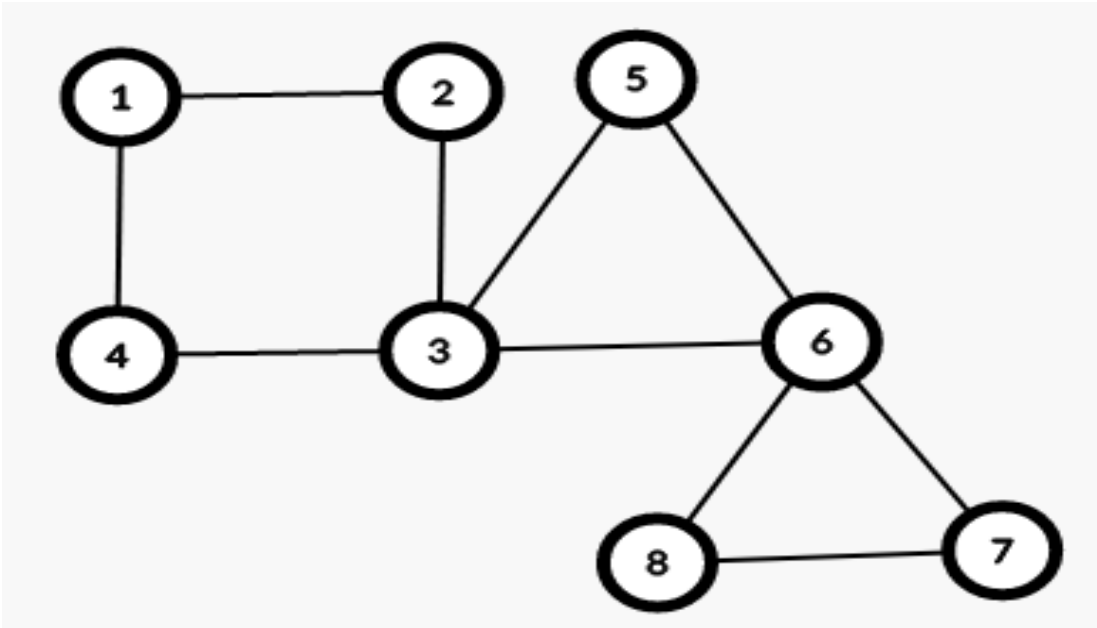
FZOJ 3608 Sulotion

因为所有的点都在强联通分量里，所以成立。(bushi

FZOJ 3608 Sulotion

因为所有的点都在强联通分量里，所以成立。(bushi

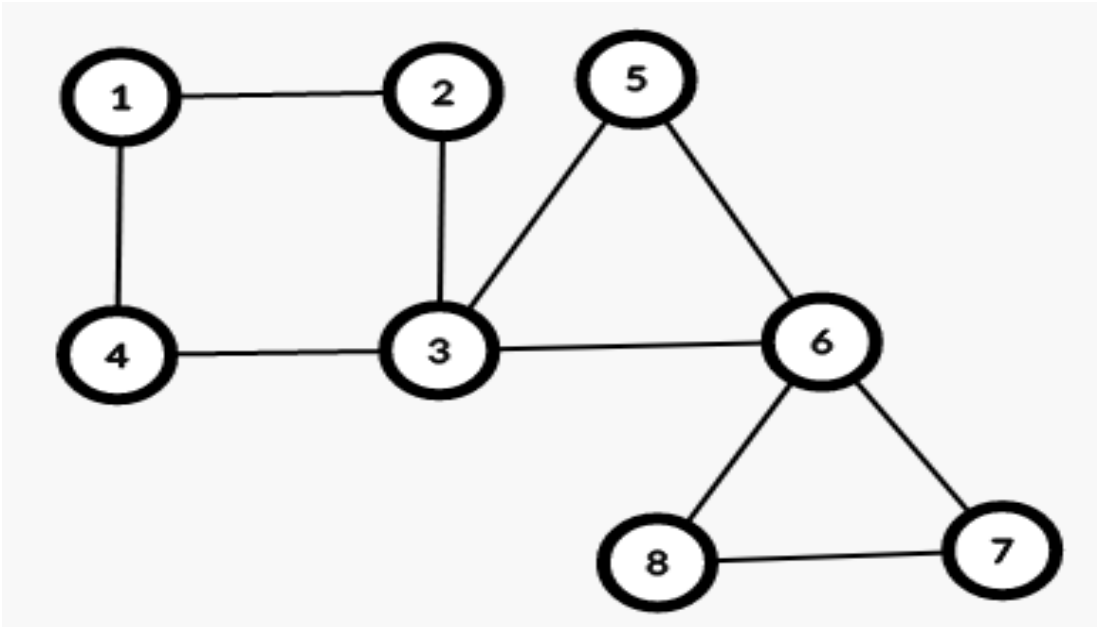
考虑一种最简单的情况，如下图所示。



FZOJ 3608 Sulotion

因为所有的点都在强联通分量里，所以成立。(bushi

考虑一种最简单的情况，如下图所示。



其它的情况可以通过拆点得到。

FZOJ 3608 Sulotion

我们令起点和终点都在最左边的那个环里。

FZOJ 3608 Sulotion

我们令起点和终点都在最左边的那个环里。
通过观察，这样的话除了最短路径意外，每一个其它的环一定会走整数个。

FZOJ 3608 Sulotion

我们令起点和终点都在最左边的那个环里。
通过观察，这样的话除了最短路径意外，每一个其它的环一定会走整数个。
我们定义最短路径为 dis ，每个环的长度为 l_i ，走过的次数为 k_i 。

FZOJ 3608 Sulotion

我们令起点和终点都在最左边的那个环里。
通过观察，这样的话除了最短路径意外，每一个其它的环一定会走整数个。
我们定义最短路径为 dis ，每个环的长度为 l_i ，走过的次数为 k_i 。
所以走过的路径长度：

$$D = \sum_{i=1}^{num} k_i \times l_i + dis$$

FZOJ 3608 Sulotion

我们令起点和终点都在最左边的那个环里。
通过观察，这样的话除了最短路径意外，每一个其它的环一定会走整数个。
我们定义最短路径为 dis ，每个环的长度为 l_i ，走过的次数为 k_i 。
所以走过的路径长度：

$$D = \sum_{i=1}^{num} k_i \times l_i + dis$$

推导一下：

$$D - dis = \sum_{i=1}^{num} k_i \times l_i$$

FZOJ 3608 Sulotion

由裴蜀定理:
$ax + by = c$ **有解的充要条件是** $\gcd(x, y)|c$ 。

FZOJ 3608 Sulotion

由裴蜀定理：
$ax + by = c$ **有解的充要条件是** $\gcd(x, y)|c$ **。**
因为都是互质的，所以

$$D - dis = \sum_{i=1}^{num} k_i \times l_i$$

一定有解。

FZOJ 3608 Sulotion

由裴蜀定理：
$ax + by = c$ 有解的充要条件是 $\gcd(x, y)|c$ 。
因为都是互质的，所以

$$D - dis = \sum_{i=1}^{num} k_i \times l_i$$

一定有解。
所以本问题得证。

FZOJ 3608 Sulotion

由裴蜀定理：
$ax + by = c$ 有解的充要条件是 $\gcd(x, y)|c$ 。
因为都是互质的，所以

$$D - dis = \sum_{i=1}^{num} k_i \times l_i$$

一定有解。
所以本问题得证。
找 k 直接二分就好了，很简单。

P6658 最大边三连通分量

不要问我题目，问我就是标题就是。

P6658 最大边三连通分量 Solution

边三联通分量定义就是任意删除两条边之后图是联通的，跟边双还是比较类似的捏。

P6658 最大边三连通分量 Solution

边三联通分量定义就是任意删除两条边之后图是联通的，跟边双还是比较类似的捏。

首先还是先做一个边双连通分量。
一个非常显然的东西：不在一个边双一定不是一个边三。
所以直接看每个边双内的就好了。

P6658 最大边三连通分量 Solution

边三联通分量定义就是任意删除两条边之后图是联通的，跟边双还是比较类似的捏。

首先还是先做一个边双连通分量。
一个非常显然的东西：不在一个边双一定不是一个边三。
所以直接看每个边双内的就好了。

对于每一个边双直接建出 DFS 树。
1. 因为是无向图所以 DFS 树只存在树边和非树边。
2. 因为是边双所以每条树边都最少被非树边覆盖一次。

接下来从性质出发，进行进一步思考。

P6658 最大边三连通分量 Solution

如果我们随机删去两条边就有点不连通，那么不是边三。

现在我们考虑出现上述情况的两种可能。

P6658 最大边三连通分量 Solution

如果我们随机删去两条边就有点不连通，那么不是边三。

现在我们考虑出现上述情况的两种可能。
1. 删除一条树边和一条非树边，这种情况在某条边只被一个非树边覆盖时会满足这个情况。
2. 删除两条树边，当两个树边的覆盖情况完全相同时这种情况是成立的，画个图也就发现很好理解。

P6658 最大边三连通分量 Solution

在考虑如何进行删边之前，先统计如何解决非树边的覆盖情况。

P6658 最大边三连通分量 Solution

在考虑如何进行删边之前，先统计如何解决非树边的覆盖情况。
自认为然算是比较巧妙的。（当然，不看题解想不出来）

1. 对每一个非树边随机一个权值 p 满足 $p \in [0, 2^{64} - 1]$ 。
2. 定义每个点 i 的权值为 P 满足：

$$P = \texttt{xor}_{i \to j} P_j$$

这样的话，可以发现对于一个节点 i 他的子树中所有节点的异或和其实就是所有从 i 的父亲节点连过来的所有非树边的异或和。

P6658 最大边三连通分量 Solution

在考虑如何进行删边之前，先统计如何解决非树边的覆盖情况。
自认为然算是比较巧妙的。（当然，不看题解想不出来）

1. 对每一个非树边随机一个权值 p 满足 $p \in [0, 2^{64} - 1]$ 。
2. 定义每个点 i 的权值为 P 满足：

$$P = \texttt{xor}_{i \to j} P_j$$

这样的话，可以发现对于一个节点 i 他的子树中所有节点的异或和其实就是所有从 i 的父亲节点连过来的所有非树边的异或和。

换一句话说，其实就是覆盖 i 父边的非树边的异或和。

P6658 最大边三连通分量 Solution

接着就是来考虑如何进行割断的操作。

P6658 最大边三连通分量 Solution

接着就是来考虑如何进行割断的操作。

相信大家其实第一种可能已经完全会了。

P6658 最大边三连通分量 Solution

接着就是来考虑如何进行割断的操作。

相信大家其实第一种可能已经完全会了。
上面说到，要求被一条边覆盖，我们直接调用上一页的做法。
考虑到把所有的边的权值全部压入一个 Hash 里面。
如果正好能配对就说明可以直接从第一种情况排除。

P6658 最大边三连通分量 Solution

接着就是来考虑如何进行割断的操作。

相信大家其实第一种可能已经完全会了。
上面说到，要求被一条边覆盖，我们直接调用上一页的做法。
考虑到把所有的边的权值全部压入一个 Hash 里面。
如果正好能配对就说明可以直接从第一种情况排除。

不难发现此时我们的原图又被分成了若干个连通块，接下来只要对于每一个连通块搞一下第二种可能即可。

P6658 最大边三连通分量 Solution

题目经过简化已经越来越像人做的题目了。

P6658 最大边三连通分量 Solution

题目经过简化已经越来越像人做的题目了。

发现多条边是配对的，选择最短距离的两条边。
原因很简单，要求最大的边三连通分量。

P6658 最大边三连通分量 Solution

题目经过简化已经越来越像人做的题目了。

发现多条边是配对的，选择最短距离的两条边。
原因很简单，要求最大的边三连通分量。

所以直接在 `dfs` 中直接维护一个 `Hash` 即可。
直接对应这条边下方和他最近的边，然后删除就可以啦。

P6658 最大边三连通分量 Solution

题目经过简化已经越来越像人做的题目了。

发现多条边是配对的，选择最短距离的两条边。
原因很简单，要求最大的边三连通分量。

所以直接在 `dfs` 中直接维护一个 `Hash` 即可。
直接对应这条边下方和他最近的边，然后删除就可以啦。

所以题目就做完了捏，$O(n+m)$ 。

Thanks for watching.

所以这次交流就愉快的结束了。
不像之前的同学的内容让人容易掉线。
我的交流内容并不深奥甚至可以说是普及组内容。
所以请同学们看得上的话就补一补觉得比较好的题目吧。

Thanks for watching