图论选讲

4182_543_731

2024/07

温馨提示

题目首先按照大概的知识点分为若干部分。

为响应相关要求，每一部分内部大概按照难度排序。

所有数默认**非负**，边权默认 10^9 级别，时限默认合理范围（$1s \sim 2s$）。

Contents

[CF528C/527E] Data Center Drama

给一张**连通**无向图，点数为 n 边数为 m。

你需要加尽量少的边（允许重边自环），然后给每条边定向，使得任意点入度出度为偶数，构造方案。

$n, m \leq 10^5$

提示：合法的条件是什么？

[CF528C/527E] Data Center Drama

合法的条件是什么？显然有如下几点：

- 每个点定向后入度出度为偶数，因此每个点原度数为偶数。这够了吗？

[CF528C/527E] Data Center Drama

合法的条件是什么？显然有如下几点：

- 每个点定向后入度出度为偶数，因此每个点原度数为偶数。这够了吗？
- 每条边只贡献一个入度，因此总边数为偶数。

这够了吗？尝试能不能构造。

[CF528C/527E] Data Center Drama

生成树的一种用法：取原图的一棵生成树考虑。(在比较极端的情况下，可能会取 DFS 树以避免横叉边)

常见构造方式：从叶子开始往上考虑。删掉一个叶子后树还是树。

证明.

任取图中一生成树（图连通）。生成树外的边任意定向，然后考虑通过定向树边满足每个点的限制。

考虑任一叶子，我们可以通过调整它到父亲边的方向来改变其入度的奇偶性，这可以使其入度（及出度）均为偶数。

从下往上做，依次调整每个点和父亲连边的方向，这样可以满足除去根外所有点的限制。

[CF528C/527E] Data Center Drama

生成树的一种用法：取原图的一棵生成树考虑。(在比较极端的情况下，可能会取 DFS 树以避免横叉边)

常见构造方式：从叶子开始往上考虑。删掉一个叶子后树还是树。

证明.

任取图中一生成树（图连通)。生成树外的边任意定向，然后考虑通过定向树边满足每个点的限制。

考虑任一叶子，我们可以通过调整它到父亲边的方向来改变其入度的奇偶性，这可以使其入度（及出度）均为偶数。

从下往上做，依次调整每个点和父亲连边的方向，这样可以满足除去根外所有点的限制。

因为总边数为偶数，此时根的度数一定正确。 □

[CF528C/527E] Data Center Drama

- 每个点原度数为偶数。
- 总边数为偶数。

  只需让原图满足该限制。

[CF528C/527E] Data Center Drama

- 每个点原度数为偶数。
- 总边数为偶数。

只需让原图满足该限制。这是非常容易的：设本来有 k 个奇数点，则显然需要至少 $\frac{k}{2}$ 条边：配对连边即可。如果此时边数为奇数，那必然需要再加边，可以加一个自环。

然后用刚才的方式构造即可。复杂度 $O(n+m)$

[CF528C/527E] Data Center Drama

- 每个点原度数为偶数。
- 总边数为偶数。

只需让原图满足该限制。这是非常容易的：设本来有 k 个奇数点，则显然需要至少 $\frac{k}{2}$ 条边：配对连边即可。如果此时边数为奇数，那必然需要再加边，可以加一个自环。

然后用刚才的方式构造即可。复杂度 $O(n+m)$

另一种构造方式

考虑图的一个欧拉回路。由于每个点度数为偶数，此回路一定存在。然后考虑回路上交替定向（边数为偶数），这样每个点每次出现都贡献了 2 入度或 2 出度。

给一张 n 个点 m 条边的**简单连通**图，将边两两配对，使得每一对边有公共点。构造方案或输出无解。

$n, m \leq 10^5$

[ural2132] Graph Decomposition. Version 2

有解条件是什么？显然得 $2|m$。

有解条件是什么？显然得 $2|m$。这够吗？试试看。

取图的一棵生成树，我们需要考虑非树边和树边。

从下往上做，最好的想法是每次尝试删掉一个叶子，和所有与其相邻的边。考虑叶子 u，进行分类讨论：

[ural2132] Graph Decomposition. Version 2

有解条件是什么？显然得 $2|m$。这够吗？试试看。

取图的一棵生成树，我们需要考虑非树边和树边。

从下往上做，最好的想法是每次尝试删掉一个叶子，和所有与其相邻的边。考虑叶子 u，进行分类讨论：

- 如果 u 与至少两条非树边相邻，则可以将其配对。
- 如果 u 与一条非树边相邻，那么可以这条边与 u 到父亲的树边配对，这样就完全去掉了 u。

[ural2132] Graph Decomposition. Version 2

有解条件是什么？显然得 $2|m$。这够吗？试试看。

取图的一棵生成树，我们需要考虑非树边和树边。

从下往上做，最好的想法是每次尝试删掉一个叶子，和所有与其相邻的边。考虑叶子 u，进行分类讨论：

- 如果 u 与至少两条非树边相邻，则可以将其配对。
- 如果 u 与一条非树边相邻，那么可以这条边与 u 到父亲的树边配对，这样就完全去掉了 u。
- 如果 u 只连向其父亲 f_u，且 f_u 与非树边相邻，那么可以取出一条进行配对。
- 否则，如果 f_u 没有非树边，则必须与另一条树边配对。这可能会破坏树的结构，怎么办？

[ural2132] Graph Decomposition. Version 2

小技巧：考虑特殊的点。

考虑深度最大的 u，这样 f_u 的其它儿子也是叶子。因此我们考虑将 u 到父亲的树边和另一个到叶子的树边配对。

但其它叶子可能还有非树边，因此可以取深度最大，剩余度数（树边和非树边）最小的 u，这样处理到这种情况时别的叶子也没有非树边了。这时就可以拿两个叶子的树边配对。

小技巧：考虑特殊的点。

考虑深度最大的 u，这样 f_u 的其它儿子也是叶子。因此我们考虑将 u 到父亲的树边和另一个到叶子的树边配对。

但其它叶子可能还有非树边，因此可以取深度最大，剩余度数（树边和非树边）最小的 u，这样处理到这种情况时别的叶子也没有非树边了。这时就可以拿两个叶子的树边配对。

如果没有别的叶子怎么办？因为 f_u 也没有别的边了，可以将 $u - f_u - f_{f_u}$ 配对，然后删掉两个点。

这样一直做就搞完了。$2|m$ 可以确保最后不留下一条边。

[ural2132] Graph Decomposition. Version 2

小技巧：考虑特殊的点。

考虑深度最大的 u，这样 f_u 的其它儿子也是叶子。因此我们考虑将 u 到父亲的树边和另一个到叶子的树边配对。

但其它叶子可能还有非树边，因此可以取深度最大，剩余度数（树边和非树边）最小的 u，这样处理到这种情况时别的叶子也没有非树边了。这时就可以拿两个叶子的树边配对。

如果没有别的叶子怎么办？因为 f_u 也没有别的边了，可以将 $u - f_u - f_{f_u}$ 配对，然后删掉两个点。

这样一直做就搞完了。$2|m$ 可以确保最后不留下一条边。

因此对于一个连通块，合法条件就是边数为偶数。$O(m \log m)$ 或 $O(m)$ 实现上述过程即可。

[CF888G] Xor-MST

给 n 个点，点有非负点权 v_i。
任意两点 i,j 间存在边权为 $v_i \oplus v_j$ 的边，求最小生成树的边权和。
$n \le 2 \times 10^5, v < 2^{30}$

Boruvka's Algorithm

这类看起来有 $O(n^2)$ 条边的生成树题通常有两种做法：通过分析性质减少需要考虑的边数，或者直接~~力大飞砖~~使用 Boruvka 算法。

Kruskal 的 idea：按照边权从小到大尝试加进去。

Boruvka's Algorithm

根据上述 idea 有如下定理：对于任意点 u，与其相邻且边权最小的边必定在最小生成树中。因此我们对于每个 u 找到这样的边，然后将其加入最小生成树。这被称为一轮操作。每轮操作中，每个点必定和另一个合并，因此一轮后点数减半。从而只需要做 $O(\log n)$ 轮。

Boruvka's Algorithm

这类看起来有 $O(n^2)$ 条边的生成树题通常有两种做法：通过分析性质减少需要考虑的边数，或者直接~~力大飞砖~~使用 Boruvka 算法。

Kruskal 的 idea：按照边权从小到大尝试加进去。

Boruvka's Algorithm

根据上述 idea 有如下定理：对于任意点 u，与其相邻且边权最小的边必定在最小生成树中。因此我们对于每个 u 找到这样的边，然后将其加入最小生成树。这被称为一轮操作。每轮操作中，每个点必定和另一个合并，因此一轮后点数减半。从而只需要做 $O(\log n)$ 轮。

考虑使用并查集维护合并。在每一轮中，我们需要对于当前每个连通块，求出连通块内连向连通块外的最小权边。

Boruvka's Algorithm

常见的实现方式是：

1. 对于每个点，标记其颜色为其所在连通块。
2. 对于每个点，计算它与不同颜色点间的最小边。
3. 对于每个连通块，考虑其所有点并找到连通块内到连通块外的边。
4. 合并这些边，继续下一轮。

关键步骤：每个点有颜色，对于每个点，计算它与不同颜色点间的最小边。

[CF888G] Xor-MST

每个点有颜色，对于每个点，计算它与不同颜色点间的最小边。

现在边权是 $v_i \oplus v_j$。如果不存在颜色，那显然是 Trie 树上贪心：优先走这一位相同的，不行就走不同的。

怎么做不同颜色？对每个子树记录其中是否只有一种颜色的点，如果是那那个颜色是啥，然后就能询问颜色了。

复杂度 $O(n \log n \log v)$

[CF888G] Xor-MST

我们也可以冷静分析。

考虑按点权建 Trie 树，对于一个深度为 d 的子树，子树内的边权都小于 2^d，但子树外的边权显然至少是 2^d。或者说，两个点在 Trie 树上的 LCA 越深，边权就一定更小。

因此我们一定是从下往上加边：先把子树内合并，然后子树和它的兄弟合并，再继续向上合并。

那只需要考虑这一个问题：左子树里面选一个点，右子树里面选一个点，最小化异或和。

[CF888G] Xor-MST

我们也可以冷静分析。

考虑按点权建 Trie 树，对于一个深度为 d 的子树，子树内的边权都小于 2^d，但子树外的边权显然至少是 2^d。或者说，两个点在 Trie 树上的 LCA 越深，边权就一定更小。

因此我们一定是从下往上加边：先把子树内合并，然后子树和它的兄弟合并，再继续向上合并。

那只需要考虑这一个问题：左子树里面选一个点，右子树里面选一个点，最小化异或和。

Trie 树上按位贪心/搜索 $dfs(u, v)$，如果 u, v 同时存在左子树或同时存在右子树，那么这一位可以是 0，分别搜两侧子树。否则，这一位一定是 1，然后搜另一方向 $(ls_u - rs_v, rs_u - ls_v)$。从而每个子树只会访问一次，复杂度为子树大小。

复杂度 $O(n \log v)$

[CF888G] Xor-MST

如果存在冷静分析的做法，它通常比 Boruvka 少 $\log n$，从而更快。

但 Boruvka 简单，不需要复杂分析。且存在必须 Boruvka 的情况。具体怎么采用需要具体情况具体分析。

[Atcoder keyence2019_e] Connecting Cities

n 个点最小生成树，点 i,j 间边权为 $D|i-j|+A_i+A_j$。

$n \le 2\times 10^5$

Hint: 如果用 Boruvka，一轮可以线性。

一道题

有 n 个点 m 条边的无向图，每条边的边权是 $c_i + x$ 或 $c_i - x$。

q 次询问，每次给一个 x，求最小生成树边权和。

$n, q \le 10^5, m \le 4 \times 10^5$

一道题

Kruskal 的 idea：按照边权从小到大尝试加进去。

随着 x 增大，$c_i - x$ 的边的相对顺序只会向前，$c_i + x$ 只会向后。

考虑固定一个 x，求出生成树。

- 如果一条 $c_i - x$ 的边在生成树中，那 x 进一步增加时，这条边顺序只会向前，从而其必定仍在生成树中。
- 而如果它不在生成树中，那 x 减小时，这条边顺序只会向后，从而其必定不在生成树中。

$+x$ 的边同理。

一道题

Kruskal 的 idea：按照边权从小到大尝试加进去。

随着 x 增大，$c_i - x$ 的边的相对顺序只会向前，$c_i + x$ 只会向后。

考虑固定一个 x，求出生成树。

- 如果一条 $c_i - x$ 的边在生成树中，那 x 进一步增加时，这条边顺序只会向前，从而其必定仍在生成树中。
- 而如果它不在生成树中，那 x 减小时，这条边顺序只会向后，从而其必定不在生成树中。

$+x$ 的边同理。那么求出 x 的答案后，我们可以分治做 $< x$ 的部分和 $> x$ 的部分。对于每一部分，我们可以固定一些边（缩一些点），再删去一些边。这样一条边只会进入一侧分治。

由于边权形式简单，排序可以预处理解决。复杂度 $O(m\alpha(n)\log v)$

[JOISC 2022] Reconstruction Project

有 n 个点 m 条边的无向图，每条边有一个权值（不是边权）c_i。
q 次询问，每次给一个 x，然后每条边的边权为 $|c_i - x|$，求最小生成树边权和。
$n \le \cancel{500}\ 10^5, m \le 10^5, q \le 10^6$

[JOISC 2022] Reconstruction Project

和刚才类似，$x = c_i$ 时这条边一定会在最小生成树中；随着 x 增大，它的顺序逐渐向后，直到某个 x 后被移出生成树；x 减小时同理。

考虑类似刚才的分治，但只在 c_i 已经不在当前分治区间内时用之前的优化剪枝。这样分治复杂度已经是 $O(m\alpha(n)\log v)$。

这里 q 很大，但我们可以分治直接找到每条边使用的区间，然后就容易 $O(\log m)$ 回答询问。

Contents

[CF786B/787D] Legacy

有 n 个点和 m 条边，但每条边为如下形式之一：

- 可以用 c_i 的代价，从 u_i 走到 v_i，即正常的边。
- 可以用 c_i 的代价，从 u_i 走到 $[l_i, r_i]$ 中的任意一个点。
- 可以用 c_i 的代价，从 $[l_i, r_i]$ 中的任意一个点走到 v_i。

求 s 出发的单源最短路。

$n, m \leq 10^5$

优化建图

对于一些特定形式的“边”，我们可以通过加入辅助点的方式，减少加边需要的边数。

最简单的例子：向一个前缀连边。

优化建图

对于一些特定形式的“边”，我们可以通过加入辅助点的方式，减少加边需要的边数。

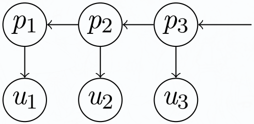
最简单的例子：向一个前缀连边。

$$
\begin{tikzcd}
p_1 \arrow[d] & p_2 \arrow[l] \arrow[d] & p_3 \arrow[l] \arrow[d] & {} \arrow[l] \\
u_1 & u_2 & u_3 &
\end{tikzcd}
$$

那如何向一个区间连边呢？

优化建图

对于一些特定形式的“边”，我们可以通过加入辅助点的方式，减少加边需要的边数。最简单的例子：向一个前缀连边。



那如何向一个区间连边呢？最经典的方式是建一棵线段树，上面连向下面。这样线段树上一个点就连向区间内所有点。那么连向一个区间只需要区间在线段树上分解为 $O(\log n)$ 个区间，然后分别连过去。

[CF786B/787D] Legacy

- 第一类边直接连。
- 对于第二类边，建一棵上面连向下面的线段树，每个连向区间的操作在线段树上变为 $O(\log n)$ 条边。
- 对于第三类边，将上面的东西反过来做一次。

时间复杂度 $O((n+m)\log n)$ 或者 $O(m\log^2 n)$，空间复杂度 $O(m\log n)$。

这里假设 Dijkstra 是 $O(m+n\log n)$ 的，但一般写 $O((n+m)\log n)$ 也问题不大

优化建图

还可以做更复杂的情况。

练习 1：二维情况，向一维在一个区间，另一维在一个前缀的区域连边。

优化建图

还可以做更复杂的情况。

练习 1：二维情况，向一维在一个区间，另一维在一个前缀的区域连边。

类似线段树套 vector，每个线段树节点按照第二维顺序维护区间内所有点，定位到该节点时是一个前缀连边。
点数 $O(n\log n)$，边数 $O(m\log n)$

练习 2：二维情况，向二维区间连边。

优化建图

还可以做更复杂的情况。

练习 1：二维情况，向一维在一个区间，另一维在一个前缀的区域连边。

类似线段树套 vector，每个线段树节点按照第二维顺序维护区间内所有点，定位到该节点时是一个前缀连边。

点数 $O(n\log n)$，边数 $O(m\log n)$

练习 2：二维情况，向二维区间连边。

把 vector 换成另一层线段树，或者叫做树套树。更好的写法有可持久化线段树合并等。

点数 $O(n\log n)$，边数 $O(m\log^2 n)$

[NOI 2019] 弹跳

二维平面上有 n 个点，给定 m 条如下形式的边：

- 从点 u_i 出发，花费 c_i 的代价，到达矩形 $[lx_i, rx_i] \times [ly_i, ry_i]$ 中的任意一个点。

求 s 出发的单源最短路。

$n \le 7 \times 10^4, m \le 1.5 \times 10^5$

空间限制 **128MB**

[NOI 2019] 弹跳

二维平面上有 n 个点，给定 m 条如下形式的边：

- 从点 u_i 出发，花费 c_i 的代价，到达矩形 $[lx_i, rx_i] \times [ly_i, ry_i]$ 中的任意一个点。

求 s 出发的单源最短路。

$n \le 7 \times 10^4, m \le 1.5 \times 10^5$

空间限制 **128MB**

直接做？直接 MLE！

[NOI 2019] 弹跳

能不能不把边建出来?

回顾 Dijkstra 的过程：我们每步拿出距离最小的点，用它更新其它点的距离。

放在这里的话，我们需要每一步拿走当前 dis 最小的点 u，然后对于它的每个出边 i，用 $dis_u + c_i$ 更新一个矩形内的距离，或者说矩形取 $\min$。

我们只需要维护这样一个数据结构，支持矩形 $\min$，拿出一个权值最小的点。而不需要真的把边建出来。

这种思路通常被称为隐式建边，或者模拟 Dijkstra。

[NOI 2019] 弹跳

直接做二维 $\min$ 还是比较困难。但查询只需要最小的权值，同时这里没有负权，一个点不可能更新出更小的距离。

考虑记录每个矩形 $\min$ 操作。需要拿出最小点时，找到最小的一个矩形 $\min$ 操作，那这个操作覆盖的那些点距离一定是这个 $\min$ 操作的值，把它们全部拿出来更新即可。

操作只需要支持矩形删点，可以直接线段树 $+$set。

时间复杂度 $O(m\log^2 n)$，空间复杂度 $O(n\log n)$

差分约束

差分约束问题

给许多形如 $x_j - x_i \leq v$ 的限制，问是否有解，并构造一组解。

在最短路中，我们有三角形不等式的限制：记 d_i 表示某个点到 i 的最短路，则对于边 (i, j, v) 有 $d_j \leq d_i + v$，这就是我们想要的。

将每个限制变为有向边 (i, j, v)。如果存在负环，则将限制加起来可以得到 $0 < -c$ 的矛盾，从而无解。否则，最短路 d_i 可以给出一组解。

一般来说造出来的图会强连通，所以直接做就行。但有些时候需要仔细加点边避免神必情况。

显然一个正常的差分约束会有负权边，因此不能用 Dijkstra。

「联合省选 2021 A」矩阵游戏

给一个 $(n-1)\times(n-1)$ 的整数矩阵 b，你需要找到一个 $n\times n$ 的整数矩阵 a 满足：

- $a_{i,j}\in[0,10^6]$
- $b_{i,j}=a_{i,j}+a_{i,j+1}+a_{i+1,j}+a_{i+1,j+1}$

构造方案或输出无解。

$n,m\le 300$

「联合省选 2021 A」矩阵游戏

如果没有值域限制会发生什么？把第一行第一列设成 0，然后剩下的每个位置可以逐一确定。

那么我们只需要考虑填第一行第一列造成的影响，然后尝试把所有数扔到 $[0,10^6]$ 里面。考虑如果我们改了一个数，按照刚才的方式逐一确定会有什么影响：

	$\cdots$	$+$	
	$\cdots$	$-$	
	$\cdots$	$+$	
	$\cdots$	$\cdots$	

对于除去左上角之外的格子，改第一行相当于这一列上交替加减，改第一列相当于这一行上交替加减。但对于左上角的格子——

「联合省选 2021 A」矩阵游戏

$+$			
	$-$	$+$	$-$
	$+$	$-$	$+$
	$-$	$+$	$-$

这显然是不好的。不过这可以拆成如下形式：先给第一行做 $+-+-\cdots$，然后第二行做 $+-+-\cdots$，第三行做 $-+-+\cdots$，以此类推。这样就变回了之前的操作形式。因此可以看成如下问题：

在刚才找到的解上，我们可以选一行或一列，再选择一个 k，然后给这一行/列依次 $+k,-k,+k,\cdots$。求能不能把所有数放进 $[0,10^6]$。

「联合省选 2021 A」矩阵游戏

记第 i 行加了 r_i，第 j 列加了 c_i，那么一个元素的限制就是

$$v_{i,j} + (-1)^{j+1} r_i + (-1)^{i+1} c_j \in [0, 10^6]$$

但这里两个数可以同符号，所以不是差分约束的形式。

「联合省选 2021 A」矩阵游戏

记第 i 行加了 r_i，第 j 列加了 c_i，那么一个元素的限制就是

$$v_{i,j} + (-1)^{j+1} r_i + (-1)^{i+1} c_j \in [0, 10^6]$$

但这里两个数可以同符号，所以不是差分约束的形式。但如果我们给每一行的 $+-+-$ 做个交替：

$+$	$-$	$+$	$-$
$-$	$+$	$-$	$+$
$+$	$-$	$+$	$-$
$-$	$+$	$-$	$+$

或者说，我们把 r_i 变成 $(-1)^i r'_i$，c_j 变成 $(-1)^{j+1} c'_j$，这样上面的不等式中 r', c' 的系数正好相反。然后跑差分约束，复杂度 $O(nm(n+m))$

Contents

2-SAT

2-SAT

有 n 个变量，每个变量可以取 True 或 False。有 m 个限制，每个限制形如

- 如果变量 x_i 取某个值，那变量 x_j 必须取某个值。

构造方案或输出无解。

做法：每个变量间两个点 $x_i, \neg x_i$ 表示这个变量为真或假。对于一个限制 $x_i \implies x_j$，连边 $x_i \to x_j$，**同时加入逆否命题** $\neg x_j \to \neg x_i$。

对该有向图求强连通分量。如果有 i 使得 $x_i, \neg x_i$ 在同一强连通分量内，则任一取值均可推出矛盾，无解。否则，任取一拓扑序，对每一变量取其出现较晚的点对应的值。可以证明这是一组合法解。(证明用到了加入逆否命题的步骤)

复杂度 $O(n+m)$

2-SAT

$\Longrightarrow$ 可以表示任意一个四种情况中三种合法的二元限制。

如果给的是一般的二元限制，那也能做：可能的情况有：

- 固定某些变量的取值。
- 要求两变量值相同。
- 要求两变量值相反。

随便处理后两者即可。

2-SAT

$\Longrightarrow$ 可以表示任意一个四种情况中三种合法的二元限制。

如果给的是一般的二元限制，那也能做：可能的情况有：

- 固定某些变量的取值。
- 要求两变量值相同。
- 要求两变量值相反。

随便处理后两者即可。

需要注意的是，如果给的是三元限制（3SAT），那问题是 NP-Complete 的，或者说目前一定不可能比指数更快。因此如果出现了这种情况，应当反思哪一步推导忽略了特殊性质。

[NOI 2017] 游戏

有 n 个变量，每个变量有三种取值 a, b, c。除去 d 个变量外，每个变量被禁止了一种取值。

有 m 个限制：如果第 u_i 个变量取 p_i，那么第 v_i 个变量取 q_i。

构造方案或输出无解。

$n \le 5 \times 10^4, m \le 10^5, d \le 8$

[NOI 2017] 游戏

如果没有那 d 个变量，那显然是标准的 2SAT 问题。

怎么处理那 d 个变量？最直接的想法是枚举这 d 个变量的取值，然后做 2SAT。但这样复杂度是 $O(3^d(n+m))$，不能通过。

如果没有那 d 个变量，那显然是标准的 2SAT 问题。

怎么处理那 d 个变量？最直接的想法是枚举这 d 个变量的取值，然后做 2SAT。但这样复杂度是 $O(3^d(n+m))$，不能通过。

但我们会变量有两个取值的问题。因此对于能取三种值 a, b, c 的变量，我们可以先枚举能取 a, b，然后枚举能取 a, c 两种情况。每种情况都是一个 2SAT，一共有 2^d 种情况。

复杂度 $O(2^d(n+m))$

「2021 集训队互测」序列

构造一个长度为 n 的正整数序列 v，满足如下 m 个限制：

- $v_{a_i}, v_{b_i}, v_{c_i}$ 的中位数为 w_i。

$n, m \leq 10^5$

对于取值的问题，一个常见的操作是，对每个值 v 建 10^9 个变量 $x_{v,1},\cdots,x_{v,10^9}$，分别表示 v 是否大于等于对应值。只需要要求 $x_{v,i} \implies x_{v,i-1}$（一条链）就可以保证选的是一段前缀。

这些变量可以表示所有 $v_i \le c, v_i \ge c$ 的限制。

显然我们不会真的建 10^9 个点，而是找出与 v 相关限制的那些值，然后链上只保留这些值。

回到原题，根据刚才的经验，我们不可能处理三元的限制。那能不能把中位数的条件变成二元限制?

「2021 集训队互测」序列

- $v_{a_i}, v_{b_i}, v_{c_i}$ 的中位数为 w_i。

- (如果 $v_{a_i} > w_i$，则 $v_{b_i} \le w_i$)×6
- (如果 $v_{a_i} < w_i$，则 $v_{b_i} \ge w_i$)×6

  这样就对了。复杂度 $O(n+m)$

[gym 102059B] Dev, Please Add This!

有一个 $n \times m$ 的网格，其中一些位置是障碍。

有一个球初始在某个位置，每次操作可以选择上下左右的一个方向。球向这个方向移动，遇到障碍或边界停止。

给定一些特殊位置，求是否可以让球经过所有特殊位置。

$n, m \leq 50$

[gym 102059B] Dev, Please Add This!

一个时刻的状态是球的位置和它移动的方向。如果前面不是障碍就必须继续向前，否则可以重新选择方向。这些状态可以看成新的点，然后球的移动过程就是在这个状态图上游走。

考虑限制，每个位置有四种经过它的状态，那么一个特殊位置相当于这四个状态必须至少被经过一个。但这是一个 4SAT，显然不能做。

[gym 102059B] Dev, Please Add This!

一个时刻的状态是球的位置和它移动的方向。如果前面不是障碍就必须继续向前，否则可以重新选择方向。这些状态可以看成新的点，然后球的移动过程就是在这个状态图上游走。

考虑限制，每个位置有四种经过它的状态，那么一个特殊位置相当于这四个状态必须至少被经过一个。但这是一个 4SAT，显然不能做。

玩过这游戏可以发现，如果沿着一个方向移过去，那接下来可以向回走然后再走过去，也就是在这个方向上来回走。

那么一个状态和其反方向的状态显然是互相可达的，其可以合并为一个状态。事实上状态可以进一步设计为图中每一个“横条”和“竖条”。

这样就变成两个状态至少需要被经过一个，这看起来很像 2SAT。

[gym 102059B] Dev, Please Add This!

通过上述分析，问题变为有一个 $O(nm)$ 大小的有向图。有若干组限制，每组限制形如两个点中至少需要访问一个。求能否从 s 出发游走，满足所有限制。

那如何描述从 s 出发，一条路径经过的点?

[gym 102059B] Dev, Please Add This!

通过上述分析，问题变为有一个 $O(nm)$ 大小的有向图。有若干组限制，每组限制形如两个点中至少需要访问一个。求能否从 s 出发游走，满足所有限制。

那如何描述从 s 出发，一条路径经过的点?

拿简单情况（DAG）考虑，可以发现：

- 不能选两个相互不能到达的点。
- 不能选 s 到不了的点。

可以发现对一般情况也对：它选出了一条 s 出发的路径……的一个子集。但这就够了。

通过上述分析，问题变为有一个 $O(nm)$ 大小的有向图。有若干组限制，每组限制形如两个点中至少需要访问一个。求能否从 s 出发游走，满足所有限制。

那如何描述从 s 出发，一条路径经过的点?

拿简单情况（DAG）考虑，可以发现：

- 不能选两个相互不能到达的点。
- 不能选 s 到不了的点。

可以发现对一般情况也对：它选出了一条 s 出发的路径……的一个子集。但这就够了。

总的限制就是：选一些点，某些 pair 至少需要选一个，不能互相到达的点对不能同时选。某些点不能选。

点对有 $O(n^2m^2)$ 个，所以直接 2SAT 做的复杂度是 $O(n^2m^2)$

Contents

点双/边双

点双定义：删掉一个点之后，剩余点连通。
边双定义：删掉一条边之后，所有点连通。
有什么性质?

点双/边双

点双定义：删掉一个点之后，剩余点连通。

边双定义：删掉一条边之后，所有点连通。

有什么性质?

点双/边双的经典求法：Tarjan。

边双的另类求法：先取一棵生成树，然后对于每条非树边，把它在树上的路径全部合并（树上并查集)。

点双的另类求法：取 DFS 树，把上面做法中合并点改成合并边。没有横叉边从而只用考虑直上直下的合并。

点双/边双

考虑连通图。

如果把每个边双缩成一个点，则剩余部分必定是一棵树：树上每条边对应原图的割边。

两个点双可能有一个公共点。对此我们可以使用圆方树的方式：对应每个点双新建一个方点，其向点双内所有点连边。如果只考虑这些边，则得到的图为原点和方点间的树，称为圆方树。不同点双之间的点为割点。

基础应用

询问删去点 w / 边 e 后，两点 u, v 是否连通。

这相当于问边双缩点后，e 是否是 u, v 路径上的割边，或者圆方树上 w 是否是 u, v 路径上的割点。这都容易解决。

~~圆方树还可以拿来做仙人掌题目~~

[NOIP 2022] 建造军营

给一张 n 个点 m 条边的无向图，选择一些点和一些边，使得删掉任意一条没有选择的边不会使得两个选择的点不连通。求方案数。

$n \le 5 \times 10^5, m \le 10^6$

[NOIP 2022] 建造军营

删边不连通，那显然是考虑边双。一个边双里面删边不影响连通性。那我们可以把一个边双缩起来：里面选了点则选的啥不重要，删里面的边也不影响。

缩完边双图就变成了树，那么问题变为：给一棵树，选择一个点有 $2^k - 1$ 种方案，剩余限制和之前相同。这就相当于，选中点之间的边必须选，剩下的任意。

然后考虑树形 DP。

考虑一个子树内的情况。如果子树外还选了点，那子树内所有选的点到子树根的边都需要选。否则子树外就任意了。

设 f_u 表示 u 子树内选了一些点，且子树内选的边满足选的点都可以通过这些边到达根的方案数。

转移是非常容易的。加一个子树时，如果它不选点就乘上边任意的方案数；如果它选那么这条边必选所以直接乘，也可能本来没有选、这个子树选了。

但怎么统计答案？

[NOIP 2022] 建造军营

考虑一个子树内的情况。如果子树外还选了点，那子树内所有选的点到子树根的边都需要选。否则子树外就任意了。

设 f_u 表示 u 子树内选了一些点，且子树内选的边满足选的点都可以通过这些边到达根的方案数。

转移是非常容易的。加一个子树时，如果它不选点就乘上边任意的方案数；如果它选那么这条边必选所以直接乘，也可能本来没有选、这个子树选了。

但怎么统计答案？一种方式是，考虑连接这些点的边连通块向上延申到了哪里，那个点满足 f_u 的形式，且 u 到父亲的边不选、外面任意。对这个求和即可。

复杂度 $O(n+m)$

[CF555E] Case of Computer Networks

给一张 n 个点 m 条边的无向连通图，和 q 对 (s_i, t_i)。问是否可以给每条边定向，使得定向后对于每对 (s_i, t_i)，s_i 可以到达 t_i。

$n, m, q \le 2 \times 10^5$

[CF555E] Case of Computer Networks

什么东西一定满足条件？换言之，什么样的图必定存在定向，使得图强连通（任意两点均可达)?

[CF555E] Case of Computer Networks

什么东西一定满足条件？换言之，什么样的图必定存在定向，使得图强连通（任意两点均可达）？

答案是边双。

证明.

首先，如果图存在割边，则无论这条边怎么定向，总有一边到不了另一边，此时必定不可。如果图是边双，那考虑 dfs 树，然后树边向下，非树边向上。因为非树边都是返祖边，且边双的性质保证每条树边必定被一条非树边覆盖，从任意点开始总可以不断用非树边向上到根，然后即可到达任一点。 □

[CF555E] Case of Computer Networks

根据上述结论，一个边双内部可以通过定向直接强连通，那么我们又一次可以把它合并为一个点。

缩点后，问题变为树上的问题。此时树的性质保证路径唯一，那么每个限制就相当于钦定了树上一条路径的方向。我们只需要再判断是否存在矛盾。

直接的方式是对每组限制拆 LCA，然后限制是直上直下的一段必须向上一段必须向下，这可以用树上差分标记（在下面的点 $+1$，上面的点向上 -1，然后向上求和）

复杂度 $O(m+q\log n)$

[CF639F] Bear and Chemistry

给一张 n 个点 m 条边的无向**连通**图。多组询问，每次给一个点集和额外的边集，问在原图上加入本次询问给出的边集后，是否对于给定点集中任意两个点 x, y，都存在一条从 x 出发、经过 y 并返回 x、不经过重复边的环路。强制在线。

$n, m, q, \sum |S|, \sum |E| \leq 3 \times 10^5$

[CF639F] Bear and Chemistry

这次的限制是，任意两点间存在两条边不相交的路径。

[CF639F] Bear and Chemistry

这次的限制是，任意两点间存在两条边不相交的路径。可以发现答案还是边双。

证明.

首先如果有割边，那显然最多一条路径能用这条割边。

另一方面，如果只存在一条边不相交的路径，那根据最大流最小割定理，两点间最小割大小为 1，也就是存在割边。

也可以直接取生成树然后用路径覆盖的方式来直接构造，但较为繁琐。~~可以网络流增广构造~~ □

[CF639F] Bear and Chemistry

这里会反复加边，因此 Tarjan 算法是不大合适的。

考虑另一个描述方式：先取一棵生成树，然后对于每条非树边，把两个端点在树上路径的所有点合并。

先对原图做一遍，然后询问就相当于：给若干条树上路径，问把它们分别合并后，给定的若干点是否在一个边双内。

[CF639F] Bear and Chemistry

这里会反复加边，因此 Tarjan 算法是不大合适的。

考虑另一个描述方式：先取一棵生成树，然后对于每条非树边，把两个端点在树上路径的所有点合并。

先对原图做一遍，然后询问就相当于：给若干条树上路径，问把它们分别合并后，给定的若干点是否在一个边双内。

或者说，也可以只标记合并，那相当于给若干条树上路径，问它们的并能否让给定点连通。

[CF639F] Bear and Chemistry

给若干条树上路径，问它们的并能否让给定点连通。

最直接的做法是直接给所有点建虚树，然后虚树上直接 dfs。复杂度 $O(n + (\sum |S| + |E|) \log n)$

$n \times m$ 的网格上有一些障碍，一个箱子和一个人。现在玩推箱子游戏（如果人和箱子相邻且这一方向上箱子后面是空位，则人可以把箱子推过去）

求箱子可以推到哪些位置。

$n, m \leq 1500$

直接的状态设计：记录人和箱子的位置。这显然不可接受。

但人必须去推箱子，所以可以发现只有人在箱子旁边的状态是有用的，这样就只有 $O(nm)$ 个状态。

记状态为 $((x, y), d)$ 表示箱子在 (x, y)，人在箱子的某一方向。我们需要判断从初始状态出发（人先走到箱子旁边），能走到哪些状态。

「USACO 2017.12 Platinum」Push a Box

直接的状态设计：记录人和箱子的位置。这显然不可接受。

但人必须去推箱子，所以可以发现只有人在箱子旁边的状态是有用的，这样就只有 $O(nm)$ 个状态。

记状态为 $((x, y), d)$ 表示箱子在 (x, y)，人在箱子的某一方向。我们需要判断从初始状态出发（人先走到箱子旁边），能走到哪些状态。

状态转移显然只有两种：人推箱子，或者箱子不动人走到另一个方向。第一种转移是显然的。对于第二种转移，我们需要求出在箱子不动的情况下，人能不能在四个相邻格子间走过去。

这是个删点连通性，那显然是点双的问题。我们需要判断删一个点后，另外两个点是否连通。那当然可以直接建圆方树，然后大力判断是否割点在路径上……

「USACO 2017.12 Platinum」Push a Box

但这必要吗？

我们判定的是，删掉一个格子后，其相邻的两个格子是否可达。

显然这两个格子是可以通过删掉的格子直接到达的，所以它们之间的割点只可能是我们判定的这个点。

小结论：如果删掉任意一点都不影响两点连通，则这两点在同一点双中。(否则，这两个点双可以通过这两个点合并)

所以问题等价于相邻三个格子（两条边）是否在同一点双中，然后更简单了。

复杂度 $O(nm)$

「2020-2021 集训队作业」Tom & Jerry

给一张 n 个点 m 条边的无向连通简单图，Alice 和 Bob 在图上进行如下博弈：

两人轮流行动，Alice 先手。Alice 每次可以经过任意条边（可以不走）但不能经过 Bob 当前位置。Bob 每次最多走一条边，Bob 抓到 Alice 则 Bob 获胜。如果经过 n^2 轮游戏还未结束则 Alice 获胜。

q 次询问，每次给定双方初始位置，求谁获胜。

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

什么是简单情况？每一轮 Alice 可以走除了 Bob 当前所在点外的所有点，因此能到的点是删去 Bob 所在点后当前点所在的连通块。删一个点的连通性，这对应着点双。

考虑最简单的情况：图是单个点双。此时 Alice 每步可以走到任意点，但不能被 Bob 一步抓到。可以发现：

「2020-2021 集训队作业」Tom & Jerry

什么是简单情况？每一轮 Alice 可以走除了 Bob 当前所在点外的所有点，因此能到的点是删去 Bob 所在点后当前点所在的连通块。删一个点的连通性，这对应着点双。

考虑最简单的情况：图是单个点双。此时 Alice 每步可以走到任意点，但不能被 Bob 一步抓到。可以发现：

在一个点双内，如果存在点连向所有点则 Bob 胜，否则 Alice 胜。

证明：Bob 走到这个点显然直接获胜，否则 Alice 每步只需要走到 Bob 不能一步到的点。

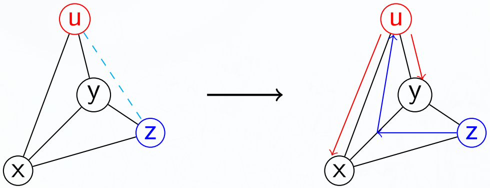
称这样的点双是好的。

回到圆方树上考虑。根据割点的简单性质，Alice 每次能到的点就是圆方树上不经过 Bob 能到的点。如果 Alice 能直接走到一个这样的点双，那走过去就直接赢了：如果有外面的点连向点双内所有点，那它也应该加入点双。

否则，Alice 就不能停在一个点双里面，因此我们需要回到圆方树上考虑问题。

如果这是一棵普通的树，那 Bob 只需要不断向 Alice 所在的子树走就能抓住。因此这里 Alice 想要不被抓住必须逃出子树：在 Bob 从上面进入一个点双时离开当前子树。这相当于：

如果 Bob 进入点双时所在的点不连向点双内所有点，则 Alice 就可以等在这然后出去：



另一方面，如果这个点（圆方树上的父亲）连向点双内所有点，那对于 Alice 在某个子树内的情况，Bob 一定可以一步走到连接更深子树的圆点，然后就可以继续向下。从而如果 Alice 第一步不能找到一个好的点双或者上述结构（以 Bob 当前点为根考虑），则 Alice 必输。

考虑 Alice 从这里出去后的情况。和刚才类似如果外面有个好的点双那直接赢了，否则又变成了刚才的情况，Alice 需要在换了根的情况下再找一个上述结构，形成如下形式。

但如果有这种情况，Alice 就能在这两个结构上来回绕，从而获胜。这样就完成了讨论。

总结起来，Alice 有三种情况获胜：

1. 圆方树上初始能到的子树中包含一个好的点双。
2. 子树内在 Bob 过来的方向上存在一个能出去的结构，且外面存在一个好的点双。
3. 子树内存在这样一个结构，且这个结构可以和另一个类似结构如上一页所示配对。

多次 dfs/树形 DP 求出：每个子树内是否存在好的点双，每个子树内（这个方向上）是否存在好的结构，每个结构是否可以配对，子树内是否存在可以配对的结构。

细节留作练习。

因为询问要定位子树，比较容易的写法是 $O((n+q)\log n)$

Thanks!