图论问题的若干解题方法

中山纪念中学 刘海峰

May 15, 2025

讲题过程中可能讲题人无法严谨证明一些结论（或者一些题的做法不够优秀），如果大家更好的想法，欢迎和讲题人交流分享。

有一个 n 个点 m 条边的无向图，保证没有重边自环，你需要给每条边和每个点标上 $0/1$。

要求对于任意一个标了 0 的点，都有一条标了 0 的边和它相连。对于任意一个标了 1 的边，两个端点中至少有一个端点标了 1。

问所有 2^{n+m} 种方案中合法的方案数，对 2 取模。

$1 \leq n, m \leq 2 \times 10^3$。

AtCoder Xmas Contest 2024 A

可以发现答案永远是 1。
证明留作读者思考。

NWRRC 2015 Graph

给定一个 n 个点 m 条边的有向无环图，你需要新加至多 k 条有向边，使得图依然是有向无环图，并最大化字典序最小的拓扑序。

$1 \leq n, m, k \leq 10^5$。

考虑一个 DAG 如何求字典序最小的拓扑序。

显然是贪心，用堆维护当前入度为 0 的点，每次取最小的即可。

那一种很显然的想法就是当要取出一个点的时候，可以考虑加一条边，使得这个点的入度 > 0。

于是乎考虑开一个大根堆和一个小根堆，小根堆维护当前度数为 0 的点，大根堆维护当前度数为 0 但加了一条出边是该点入边还没确定的点。

直接维护即可，需要注意一下小根堆中只剩一个元素时应该如何选择的情况。

APIO2022 游戏

给定 n, k，有一个 n 个点的有向图（编号从 0 开始），初始没有边，现在依次加入 m 条边，你需要在加入每条边后判断是否存在 $0 < x < y < k$，满足 y 能到达 x。

$1 \leq k \leq n \leq 3 \times 10^5, 1 \leq m \leq 5 \times 10^5$，交互式强制在线。

首先考虑一种 $O(nk)$ 的做法。

对于每个 y，动态的维护一下当前能够到达的点，具体来说就是，每次加入一个新的边的时候看一下 y 是否能到达这条边的起点，如果能就去更新。

但这显然太浪费了。毕竟我们并没有要求起点唯一。

于是我们分治，看右半部分是否能到达左半部分。

但还有一个问题，就是一个遍不能在所有的层里面都加一遍，不然复杂度显然没变。

处理方法也很简单，看这条边属于右半部分还是左半部分。

具体来说，如果右半部分的点能到达这条边的起点，那么这条边属于右半部分。

如果左半部分的点能到达这条边的终点，那么这条边属于左半部分。

否则待定。

如果属于左半部分则往左半部分递归，如果属于右半部分就往右半部分递归。

复杂度 $O((n+m)\log k)$。

CF2080A

给定一个 n 个点 m 条边的有向图，保证不存在重边、自环，保证边 $x \to y, y \to x$ 不同时存在。

现在小 A 通过一些算法求出了这个有向图的所有强联通分量，小 B 想要将一些边的方向擦掉，使得小 A 可以通过之前求出的强联通分量唯一的确定这些边的方向。

问小 B 最多能擦多少条边，以及在擦掉的边数最多的情况下有多少种擦法，方案数对 $10^9 + 7$ 取模。

$n, m \leq 2 \times 10^3$。

首先简单分类讨论一下边是否在强连通分量内。

如果这条边不在强连通分量内，那么判断是否能删除是简单的，只要看删除这条边后起点是否能到达终点，直接 DFS 一下即可。注意对强连通分量缩点之后可能会出现重边。

于是问题变成了如何处理强连通内部的答案。

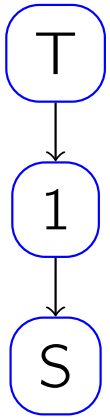
考虑一个强连通分量内的答案怎么算，现在我们只考虑整个图是个强连通分量的情况。

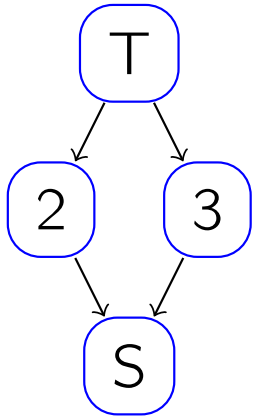
首先，如果删掉这条边之后整个依然强连通，那么这条边不能删。

否则的话，假设分成的强连通份量集合为 S。

把得到的 S 相同的边划分到一个等价类里面。

这里是删除一条 $S \to T$ 的边之后得到的图的两个例子。





对于一个边的等价类，设删掉其中的一条边得到强连通分量数量为 n_1，等价类内的边数为 n_2，可以发现 $n_1 \geq n_2$。

如果 $n_1 = n_2$，那么可以发现这个等价类里面的所有边需要留恰好一条，否则就可以全部删除。

这是为什么呢？如图可以发现，$T \to 2, 2 \to S$ 属于一个等价类，并且这个等价类里的边一定要保留恰好一条，$T \to 3, 3 \to S$ 同理。所以不难发现这种等价类里面的边可以全部删除，复杂度 $O(m(n+m))$。

[PA 2022] Mędrcy

有 n 个人和 m 条咒语，每条咒语都有恰好 $n-2$ 个人知道，并且每条知道该咒语的人同时也知道有哪些人也知道这条咒语。

每个人不知道总共有多少条咒语，但每个人都知道任意咒语有 $n-2$ 个人知道。

所有人中午会在一起聚会，聚会过程中不能交流，然后晚上回到自己的小屋，如果这个人通过白天得到的信息发现有自己不知道的咒语，那么他会在当晚默默离开。

有一天，有个外地人来到了这里并在中午聚会的时候告诉大家："这里至少有一个人不知道至少一条咒语。"这个外地人还会在这里停留 k 天，他好奇第一次有人离开的时间以及会有哪些人在这天离开。如果 k 天之内不会有人离开则输出 -1。

$3 \le n \le 10^3, 1 \le m \le 3 \times 10^3, 1 \le k \le 30$。

首先考虑如何分析这个问题，因为每个咒语都有恰好 $n-2$ 个人知道，很显然应该在不知道的那两个人之间连一条边。

然后考虑如何分析，首先所有人都会假设自己这里没有连边，所有人都可以看到其他人的连边。

如果一个人 x 没有看到任何连边，那么 x 第一天就走了。

如果删掉 x 后 y 本应当第一天走，但 y 第一天没走，那么 x 第二天就走了。

如果 x 第三天走掉，那么相当于删掉 x 之后第二天本应当有人走掉，但最后没人走掉。

那相当于是如果存在点的序列 $a_{1\cdots m}$ 满足删掉 $a_{1\cdots m}$ 后不存在边，那么 $a_{1\cdots m}$ 都应该在第 m 天走掉。

因为我们要找第一次有人离开的时间，所以相当于是求该图的最小点覆盖。

该问题显然不能在多项式时间复杂度内解决。

注意到 k 很小，考虑一种搜索方法：每次找到当前度数最大的点并考虑该点的状态删去然后。

如果该点选，那么相当于要求剩下的点中最多只能选 $k-1$ 个的答案。

如果该点不选，那么和该点有连边的点都要选，那么把这些点一并删掉剩下的点中最多只能选出 $k-deg$ 个点。

注意到最大点度数 ≤ 2 时该问题是可以快速做的。

所以复杂度为 $T(k)=T(k-1)+T(k-3)$，计算得 $T(30)$ 大概是 5e4 级别，不过这个显然是卡不满的。

复杂度 $O(T(k)(n+m))$。

结束了？

并没有，其实复杂度还可以分析得更紧。

设 D 为度数最大的点，如果边数 $> k \times D$，那么显然没救了。

于是边数 $\leq k \times D$，也就是说要考虑的点、边的数量也不是很多，只对有边连的点进行考虑，复杂度就是 $O(\max(k \times (n+m), T(k) \times k))$。

关于最大独立集

实际上，最大独立集问题也可以用上述方法解决，复杂度为 $T(n) = O(n^2) + T(n-1) + T(n-1-D)$，其中 D 为当前度数最大的点，当 $D < 3$ 时可以直接暴力做，可以发现这个做法复杂度显然无法达到上界，即使达到上界效率也远高于 $O(n \times 2^{n/2})$ 的做法。

关于最大独立集

上述的搜索算法本质上相当于搜了所有可能的独立集，但如果只追求最大独立集的话，还可以进一步的发掘性质并优化。

比如说，如果存在一度点，那么这个点一定选，如果存在二度点，那么要么这个点两边的点都选，要么只选这个点，于是可以把两边的点合起来并删掉中间的点。

目前学术界最优的据说是特殊处理最大点度数 ≤ 9 的情况，然后在度数 > 9 再递归，复杂度是 $T(n) = T(n-1) + T(n-11) + O(\text{poly}(n))$，不过我目前只找到了度数 > 8 的时候递归的论文：

https://arxiv.org/pdf/1312.6260

AGC072E

有 n 个机场和 m 个航班，第 i 个航班会从 a_i 机场起飞，b_i 机场降落，飞行距离为 c_i，费用为 $c_i \times F$。

若乘坐一个飞行距离为 s 的航班，则会获得 s 的积分，初始时积分为 0。积分可以用来兑换钱，具体来说，k 积分在 j 机场可以兑换 $k \times R_j$ 单位的钱，保证 $0 \le R_j \le F-1$，注意这里 k 可以不是整数，但要求 $k \ge 0$。

你任意时刻积分和钱都不能为负数，问你至少需要准备多少钱，才能从 1 机场到达 n 机场。数据保证一定有解。

$n \le 400, m \le n \times (n-1), 1 \le F \le 100, 1 \le c_i \le 100$。

首先做一点简单的观察，假如我们已经确定了整个路程，并且确定了初始有多少钱，考虑如何安排积分的兑换。

考虑到当前机场后，记到下一个 R 值大于等于该机场的 R 值的机场的的开销为 C，那么肯定是尽量少兑换积分使得资金能达到 C，如果无法达到则全部兑换。

这个贪心过程限制有点多，总结起来就几句话：很多情况下要么没钱，要么没积分。

考虑二分答案，然后 dp。

设 f_i, g_i 分别表示当前在点 i，没积分/钱的情况下钱/积分最多能有多少。

先不考虑转移顺序，经过分析发现转移有四种：

1. 当前只有钱，走到一个点，然后把积分全部兑换成钱。
2. 当前只有积分，把一些积分兑换成钱，使得刚好走到另一个点时没钱。
3. 当前只有钱，走到一个点 x 时把一部分积分兑换成钱，再走到另一个 y 时刚好花完所有的钱。
4. 当前只有积分，在这个点上全部兑换成钱。

现在问题变成了转移顺序，显然可以像 Bellman-ford 一样迭代 $O(n)$ 次，复杂度 $O(n^4)$，显然无法接受。

可以发现 f 的转移中有一条复杂度是 $O(n^2)$ 的，但 g 的转移复杂度为 $O(n)$。

如果只有 f 的转移或只有 g 的转移我们可以 dijkstra，具体的，每次找到最大值的位置并进行转移，正确性比较显然。

但现在 f, g 一起转移的时候“最大的”就不好刻画了。

但其实有更简单粗暴的方法，就是发现把所有的 g 做一遍 dijkstra 复杂度也是 $O(n^2)$ 的，和 f 单次转移复杂度相同，于是我们可以搞一个 dijkstra 套 dijkstra，也就是每次对 f 更新之后都把所有的 g 拉出来做一次 dijkstra。

于是复杂度就是 $O(n^3)$ 了，但发现外面还有个二分，所以是 $O(n^3 \log n)$，显然无法通过。

其实处理方法也很简单，把算法转置一下（或者说，从终点开始倒着做）

具体来说就是现在 f_i, g_i 表示从 i 出发要到终点且没积分/钱的情况下至少需要多少钱/积分。转移类似，不过现在变成了外层对 g 做 dijkstra，内层对 f 做 dijkstra。

复杂度 $O(n^3)$。

2024 集训队互测 R1T3

对于一个 $n \times m$ 的 01 矩阵 A（下标从 1 开始，下同），定义一个 $n \times m$ 的 01 矩阵 B 的是好的当且仅当：

1. 对于第 $i\ (1 \le i \le n)$ 行，不存在 $1 \le j, k \le m, j \ne k$ 同时满足 $A_{i,j} = B_{i,k} = 1, A_{i,k} = B_{i,j} = 0$。
2. 对于第 $i\ (1 \le i \le m)$ 列，不存在 $1 \le j, k \le n, j \ne k$ 同时满足 $A_{j,i} = B_{j,i} = 1, A_{k,i} = B_{k,i} = 0$。

对于一个 $n \times m$ 的 01 矩阵 A，有一个 $n \times m$ 的 01 矩阵 B，初始 B 的所有位置都是 0，每次可以选择 B 中若干个等于 0 的位置，将它们同时变成 1，要求做完操作之后 B 矩阵仍然是好的。设最多能操作 k 次，则矩阵 A 的权值为 k。

小 L 现在有一个 $N \times N$ 的 01 矩阵 M，由于 N 很大，所以 M 由特殊的方式生成。

初始 M 中所有位置均为 0，然后有 Q 次操作，每次操作给定 x_1, x_2, y_1, y_2，对于所有位置 (i, j) 满足 $x_1 \leq i \leq x_2, y_1 \leq j \leq y_2$，将 $M_{i,j}$ 变成 $1 - M_{i,j}$。

设 $S_{i,j}$ 为 M 前 i 行前 j 列构成的子矩阵的权值。

给定询问参数 $op \in \{0, 1\}$，若 $op = 0$ 则输出 $S_{N,N}$，若 $op = 1$ 则输出 $S_{N,1}, S_{N,2}, \cdots, S_{N,N}$。

$1 \leq N, Q \leq 2 \times 10^5$。

考虑把这个奇怪判定条件对应到图上面。

具体的，考虑一个左侧 n 个点，右侧 m 个点的有向二分图。

如果 $A_{i,j}=1$ 则左侧 i 连向右侧 j，如果 $A_{i,j}=0$ 则右侧 j 连向左侧 i。

于是考虑把 $B_{i,j}$ 对应到左侧 i 到右侧 j 的连边的边权。

观察一下判定条件，发现合法当且仅当对于两条边 i,j 满足 i 的终点和 j 起点相同，那么 i 的权值 $\leq j$ 的权值。

考虑一个图该如何计算答案。

发现我们只关注强连通分量个数以及在强连通分量内部的边数。

直接建图复杂度无法接受，考虑图的特性。考虑对这个图跑缩点，缩完点之后发现大概是一条链。

为什么说是大概呢，因为两个属于同一部分的点可能互相都不能到达，两个点互相不能到达当且仅当它们连向的点的集合相同。

如果把这类点缩起来，那么整个图就是一条链了。具体的，每个强连通分量要么只包含一个点，要么包含了两种颜色的点，对于后者，相邻两个强连通分量之间一定存在连边，对于前者，假设有两个同色的单点 a, b，那么肯定存在一个和它们不同色的点 c 是的 $a \to c, c \to b$（否则两个点连出的边完全相等，就可以缩起来了），所以缩完点后就是一条链了。

于是考虑如何快速处理这些东西，或者说如何快速处理出图的每一个强连通分量信息。

事实上我们可以只关注每个点的出度。

首先把左右侧的所有点分别按照出度从小到大排序。

我们发现链的一个前缀一定是左侧点的一个前缀和右侧点的一个前缀构成的点集。

并且合法当且仅当度数之和 $=$ 左侧点数 $\times$ 右侧点数。

暴力枚举 x, y，单次复杂度 $O(n^2)$。不过实际上单次复杂度可以达到 $O(n)$。

具体来讲，设 d_i 表示左侧第 i 个点的度数。

性质：如果 $d_i > d_j$，则 i 能够到达 j。

我们将左右侧的点排序之后出度大的往出度小的连（在本题中，相等的话可以随意指定一个顺序），然后就变成了一般竞赛图问题了。

因为 d_i 变化非常复杂，可以这一会儿 $d_i > d_j$，过一会儿 $d_j > d_i$，所以考虑对于每一对 (i, j) 求出第一次 $d_i \neq d_j$ 的时刻以及此时 d_i, d_j 的大小关系。

首先找出对应两行的最长公共前缀，在这一段里面两个点都互相不能到达，然后比较下一位，大的那一个此时可以到达小的那一个。感受一下，这是一个比较字典序的过程。相当于是把矩阵每一行按照字典序排序即可。

用主席树维护字符串哈希即可 $O(n \log^2 n)$ 完成排序，事实上，用类似 SA 的思路或者分治均可以做到 $O(n \log n)$，不过后者常数过大再加上不是本题重点所有出题人选择了 $O(n \log^2 n)$ 的做法。

然后每次加入右侧的点的时候用线段树求一下连出去的点中排名最大的和连进来的点中排名最小的，做一个类似区间覆盖的数据结构（出题人用的是并查集）维护即可。

总复杂度 $O(n \log^2 n + Q \log n)$。

CTSC2011 无穷图的桥

本题的目标是求一个点数无穷的无向图的所有桥的边权和。

这个无向图具有如下性质：

1. 这个图是一个连通图。

2. 这个图有无穷层（层的标号从 1 开始），每层都有 n 个节点。为了描述方便，以下用 (i, x) 表示第 i 层的 x 号节点。

3. 同一层内的节点可以相互连边，相邻两层的节点之间可以相互连边，除此之外，其他节点之间不能相互连边。

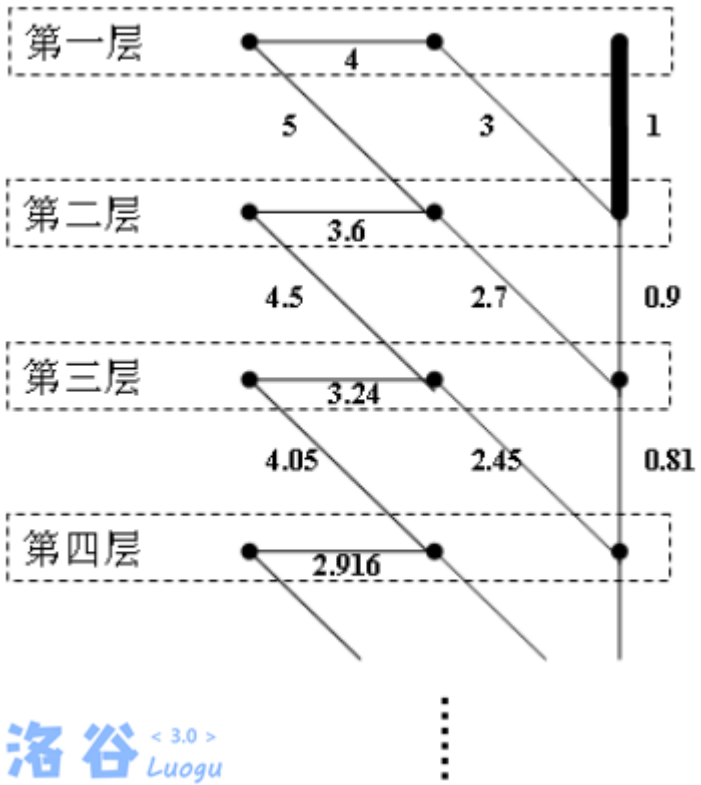
4. 对于任意 $i, j \geq 1$，$1 \leq x, y \leq n$，$p \in \{0, 1\}$，若 (i, x) 与 $(i + p, y)$ 之间有一条权值为 d 的边，那么 (j, x) 与 $(j + p, y)$ 之间也有一条边，它的权值为 $0.9^{j-i} \times d$。

给定这个无穷图第一层内部以及第一层和第二层之间的连边信息，求这个无穷图的所有桥的权值和（$m1$ 为第一层与第一层之间的连边数量，$m2$ 为第一层与第二层之间的连边数量）。

$1 \leq n \leq 3 \times 10^5$，$1 \leq m1, m2 \leq 5 \times 10^5$。

例子

如下所示的无向图就符合上面的所有性质。加粗的边为这个图的桥。



首先注意到一个容易被忽视的条件：图连通。这相当于对于任意 i，第 i 层的任意 x, y 都只通过层数 $\geq i$ 的点互相到达。

我们不妨定义横边为第一层内部的边，竖边为两层之间的边。

我们先只考虑第一层的横边和第一层第二层之间的竖边有哪些在整个图中是桥。

可以发现如果第一层的横边形成的一个连通块中如果有超过一条竖边，那么这时所有的竖边都一定不是桥。

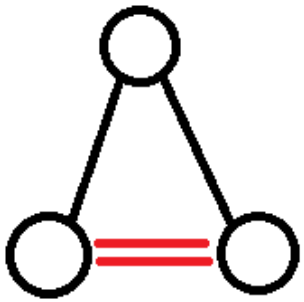
考虑把一个连通块内的边双树给建出来，其中边双树的根为有竖边的那个边双。

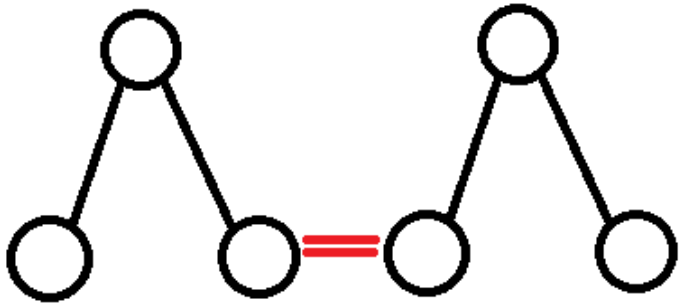
考虑如何去确定接下来的层的边双树。

也就是考虑当前层的竖边对接下来的层的影响。对于一个边双里面的竖边，对下一层的影响。

相当于是合并这些竖边的另一个端点所在的边双，我们考虑如下两种合并操作：

1. 合并两个处于同一个连通块的边双。
2. 合并两个不处于同一个连通块的边双。





对于第一种情况，直接合并这两个边双对应到树上的这条路径上的所有边双。

对于第二种情况，县合并这两个边双到根的所有边双，然后再合并两个根。具体来说，我们可以在建树的时候维护一下深度，这样可以方便后续合并。

我们可以在合并的过程中顺便计算一下哪些横边和竖边现在不是桥了，并加入答案。

总复杂度 $O((n+m1+m2)\log n)$。