Tutorials

EC-Final 2023 命题组

February 22, 2024

Statements

对于所有父亲编号小于自己的树，求有多少种可能的 DFS 序。

DFS 时，要求必须先走编号较小的儿子。

$1 \leq n \leq 800$

Solution

考虑如何判断 p 是否合法。

设 p_{i-1} 的父链从上往下依次设为 $anc_{1\ldots m}$。

如果 $p_{i-1} < p_i$ 那么可以直接将 p_i 接到 p_{i-1} 下方。

否则我们需要选择一个 k 满足 $anc_k < p_i$ 并将 p_i 接到 anc_{k-1} 下方。

因为 anc 递增，所以一定是选最大的 k。

Solution

考虑对这样贪心连出的树计数。等价于以下限制：

- 每个点的儿子按照编号从小到大依次排列。
- 对于某个点的两个相邻的儿子 u_1, u_2，设 u_1 的最后一个儿子为 v，则我们要求 $u_2 < v$。

一种思路是设 $dp_{i,j}$ 表示 i 个点，要求根节点的最后一个儿子为 j 的方案数。

复杂度 $O(n^4)$，难以通过。

Solution

若要求 $u_i > v_{i-1}$ 则可以删除 $u_i > u_{i-1}$ 的限制。

考虑一个点的所有儿子 $u_{1\dots m}$。设 u_i 的最后一个儿子为 v_i。

则我们要求 $u_{1\dots m}$ 递增且 $u_i < v_{i-1}$。

将问题看作在一个有向图上进行拓扑序计数。则我们将 $u_{1\dots m}$ 串成一条链。

而 $u_i < v_{i-1}$ 的限制会导致这个图不是一棵外向树。

我们将它容斥为 $[u_i < v_{i-1}] = 1 - [u_i > v_{i-1}]$，即要么删除要么改成一条反向边。

Solution

如果改成了反向边，显然我们可以删除 $u_i > u_{i-1}$ 这条限制。

这样操作之后当前这层的限制会形成一个链状结构，考虑其形态可以设计出如下状态。

设 $dp_{i,j}$ 表示大小为 i 的树，在根节点的最后一个儿子 u_m 后面额外增加一个子树大小为 j 的儿子 u_{m+1}，并且要求 $u_m < u_{m+1}$ 时前面 i 个点的拓扑序数量。

答案即为 $dp_{n,0}$。

时间复杂度为 $O(n^3)$。

Statements

给一个由 'V' 和 'I' 构成的长度为 n 的字符串，将它分成若干段，使得每一段是个合法的一位罗马数字，且最后形成的数字串最小。

$1 \leq n \leq 5 \times 10^5$

Solution

首先考虑数字的位数最小，从后往前 DP 即可。

然后构造最小的方案，你需要从前往后逐位确定，满足当前位尽量小，且后面的段能分成的数位比当前段小 1 即可。

Statements

有两个整数序列 a, b。a 的长度为 n，b 的长度为 m。a_i 在 $[la_i, ra_i]$ 中，b_i 在 $[lb_i, rb_i]$ 中。

你需要对于每组 $x \in [1, n], y \in [1, m]$ 求出 $\sum_{i=1}^{x} a_i = \sum_{i=1}^{y} b_i$ 的方案数。

$1 \leq n, m, la_i, ra_i, lb_i, rb_i \leq 500, la_i \leq ra_i, lb_i \leq rb_i$。

Solution

设 $dp_{i,j,k}$ 表示 $\sum_{l=1}^{i} a_i - \sum_{l=1}^{j} b_j = k$ 的方案数。朴素地做的状态数为 $O(n^3 V)$。

但我们实际上可以只求出所有 $k \in [-V, V]$ 的状态。

具体地，考虑 $dp_{i,j,k}$ 往后的转移。如果 $k > 0$，那么转移到 $dp_{i,j+1,k'}$，否则转移到 $dp_{i+1,j,k'}$。

这样做的状态数为 $O(n^2 V)$。转移时可以使用前缀和优化，总时间复杂度为 $O(n^2 V)$。

Statements

给一个随机的 $1 \sim n$ **的排列，求所有子区间的中位数之和。**

$1 \leq n \leq 3 \times 10^5$

Solution

本题因为数据随机，加上时限给的比较充裕，所以存在很多不同做法。这里给出一些参考的做法。

首先我们可以考虑这样一个事实，当区间足够大的时候，中位数会接近 $n/2$ 级别。也就是对于一个数，如果它距离 $n/2$ 比较远，那么它可能作为中位数的区间会比较小。

如果对于每个数 x，能够找到它可能作为中位数的区间 $[L_x, R_x]$，那么可以 $O(R_x - L_x + 1)$ 得求出它作为中位数的区间数。

具体就是把大于等于 x 作为 1，把小于 x 作为 -1，统计和为 $0, 1$ 的区间个数即可。

我们可以粗略估计，$\sum_x (R_x - L_x)$ 是 $O(n^{1.5})$ 级别的，估计见后。

Solution

关于如何找到 L_x, R_x，有很多种不同的方法，有些方法可能存在一定的常数/正确率的问题。

- 把大于等于 x 作为 1，小于 x 作为 -1。从小到大扫描 x，并且用线段树维护前缀和。找到前后缀的前缀和的区间求交，然后在线段树上二分即可。
- 设置一定的阈值，从 x 位置往前后扫描，超出这个阈值就 break 即可。
- ...

这样就可以在 $O(n^{1.5})$ 内通过本题。

Solution

不妨设 $x = n/2 - k$，由于序列的随机性，如果一个区间长度为 l，这 l 个标记每个有 $1/2 - k/n$ 概率为 $+1$，$1/2 + k/n$ 概率为 -1，它们和的期望是 $-2kl/n$。

若近似他们为独立分布，由 Chernoff bound，x 之后的连续 len 个元素的标记的和大于等于 0 的最大的 len 的期望是 $O(n^2/k^2)$。

设这 len 个元素的最大前缀标记和是 m，这些元素与 x 之前连续且标记和为 $-m$ 的元素组成组成中位数为 x 的区间。可以估计 m 的期望是 $O(n/k)$ 级别。

Solution

x 之前连续且标记和为 $-m$ 的元素至多有期望 $mn/k + n^2/k^2 = n^2/k^2$ 级别。其中第一项表示期望达到 $-m$，第二项表示保证不再大于等于 $-m$。

即如果统计以 x 为中位数，且 x 之后大于 x 的数字少于小于 x 的数字的区间，那只需考虑 x 前后 $O(n^2/k^2)$ 个元素。

对称地，如果统计以 x 为中位数，x 之前大于 x 的数字少于小于 x 的数字的区间，也只需考虑 $O(n^2/k^2)$ 长度。这个长度与 n 取 min 之后求和，是 $O(n^{1.5})$ 级别的。

这是对暴力运行时间的估计，我们在随机数据上测试符合预期。

Solution

在上面做法中，我们发现只需要统计 x 这个数前后，前缀和等于某个值的出现次数即可，如果我们可以快速维护出每个值的出现次数，那么我们不需要线性扫描整个序列。

假设对于 x，需要关注的前缀和的值域为 $[l_x, r_x]$，通过实践和粗略的估计，$\sum_x (r_x - l_x)$ 是 $O(n \log n)$ 级别的。

令 $d_x = r_x - l_x + 1$，那么我们可以直接用线段树维护前缀和，以及这个区间内前缀和距离最小值或距离最大值不超过 $O(d_x)$ 的数字出现次数。

所以我们可以做到理论上 $O((\sum_x d_x) \log n)$ 也就是 $O(n \log^2 n)$ 的时间复杂度。

基于这个想法以及一些实现技巧，可以在实践中比上面的 $O(n^{1.5})$ 更快。

Statements

给你若干个点，每个点有一种颜色，一共 n 种颜色。

你需要给这些点之间连边，使得同色的点距离不小于 d，或者不连通。

$1 \leq n \leq 5 \times 10^5$

Solution

当 $d = 1$ 时，所有点之间两两连边即可。

当 $d = 2$ 时，所有不同色的点之间两两连边即可。

当 $d \geq 3$ 时，容易猜想，我们可以连若干个所有点颜色不同的团。

Solution

证明：当 $d \geq 3$ 时，考虑两种颜色的点 A, B，出现次数分别是 c_A, c_B，不妨设 $c_A \leq c_B$。

对于 A 中的每个点，如果有两个 B 里的邻居，那么这两个邻居的距离等于 2，矛盾。

所有 A 中的每个点，至多一个 B 里的邻居，两种颜色之间的边不超过 $c_A = \min(c_A, c_B)$。

上述的若干个团的构造，满足取等号的条件。

需要对每个颜色的大小进行排序，时间复杂度 $O(n \log n)$。

Statements

给一个 1 到 n 的排列 $p_1, p_2, \ldots, p_n$，你可以交换相邻的元素，问最少多少次，能把整个排列变好。

一个排列是好的，当且仅当 $i \cdot p_i$ 单调不降。

$1 \leq n \leq 5 \times 10^5$

Solution

可以归纳证明，一个好的排列是好的，当且仅当它可以通过 $1, 2, \ldots, n$ 然后交换若干组不同的相邻对得到。

可以先求出将序列变成 $1, 2, \ldots, n$ 的代价，这是经典的逆序对问题。

然后考虑相邻两个元素，如果 i 的位置在 $i+1$ 后面，那么把它们排成 $i+1, i$ 可以减少 1 的代价。

问题就变成考虑每组 $i, i+1$ 是否交换，可以减少多少代价，并且每个数至多被交换一次。

可以使用贪心或者 DP 解决，时间复杂度为 $O(n \log n)$。

Solution

证明如下：

- 首先考虑，1 什么位置，如果在 i 的位置，那么 $i-1$ 位置的数有不超过 $i/(i-1)$，所以 1 只能在 $1,2$ 位置，并且如果 $p_2 = 1$，那么 $p_1 = 2$。
- 然后考虑，假设 $1, 2, \ldots, x$ 在前 x 位，可以用类似的方法得到 $x+1$ 只能在 $x+1$ 和 $x+2$ 位置，然后归纳证明即可。

Statements

对于一张图，告诉你任意两个点之间的最大流，保证 ≤ 3，构造任意合法的原图，可能无解。

$n \leq 300, \sum n^2 \leq 9 \times 10^6$

Solution

对于有解的情况，显然我们有如下的限制条件：

- $\forall i, j : A_{i,j} = A_{j,i}$
- $\forall i : A_{i,i} = 0$
- $\forall i, j, k : A_{i,j} = A_{j,k} = w \Rightarrow A_{i,k} \geq w$

在此基础上，题目相当于给定了图中边三连通分量、边双连通分量、连通块的划分方案，要求我们构造原图。

Solution

首先，0 边我们可以直接忽略，因为每个联通块显然是独立的。

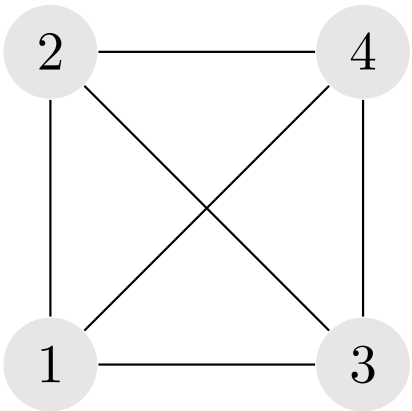
其次，1 边我们也可以直接忽略，我们可以将每个 ≥ 2 的团构造完成后使用一条链将它们串起来。

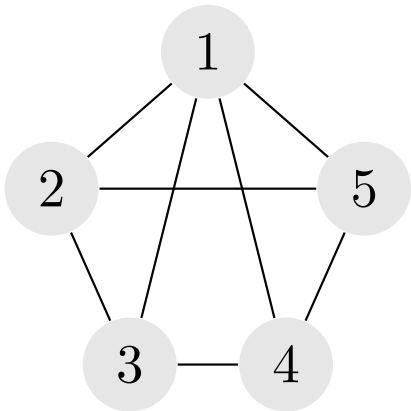
因此，原问题可以 reduce 到只存在 2 与 3 的边，构造方案。

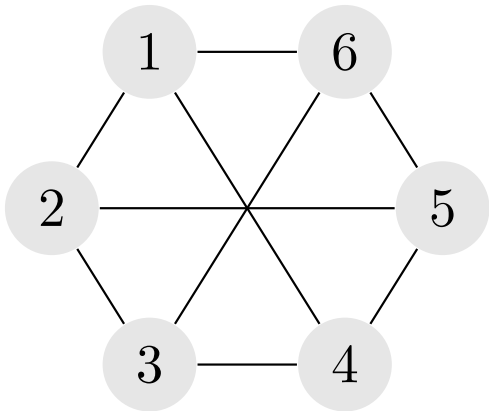
Solution

一个基本的思路：对于每个边三连通分量，我们分别构造，最后将它们连成一个大环。

怎么构造呢？注意到如果一个边三连通分量的大小 ≥ 4，那么我们总是可以将其直接构造，具体构造如下：

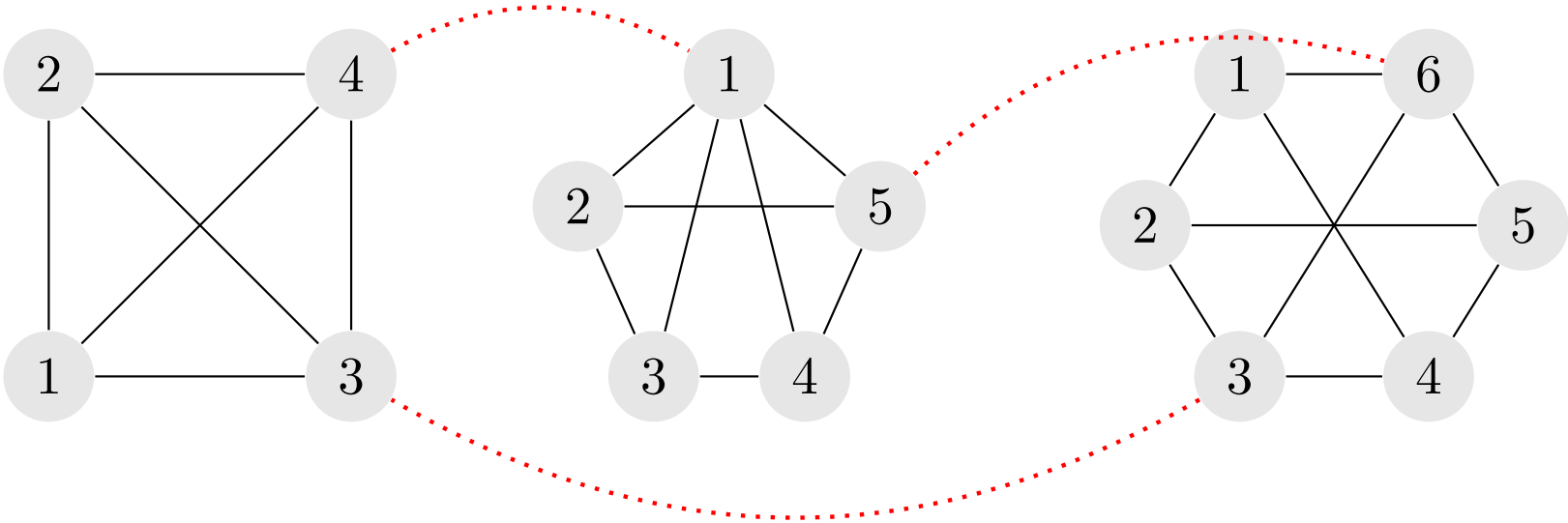






一个自然的想法是直接将所有的边三串在一起，就做完了。

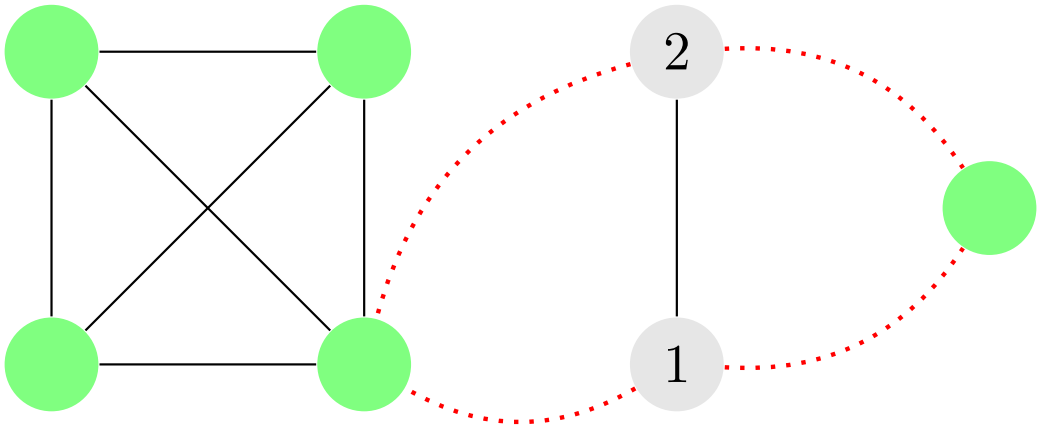
Solution



- 若一个边三 ≤ 3，那么这个边三无法通过这种构造得到。
- 若一个边双内边三的数量 ≤ 2，无法保证直接能够连成环。

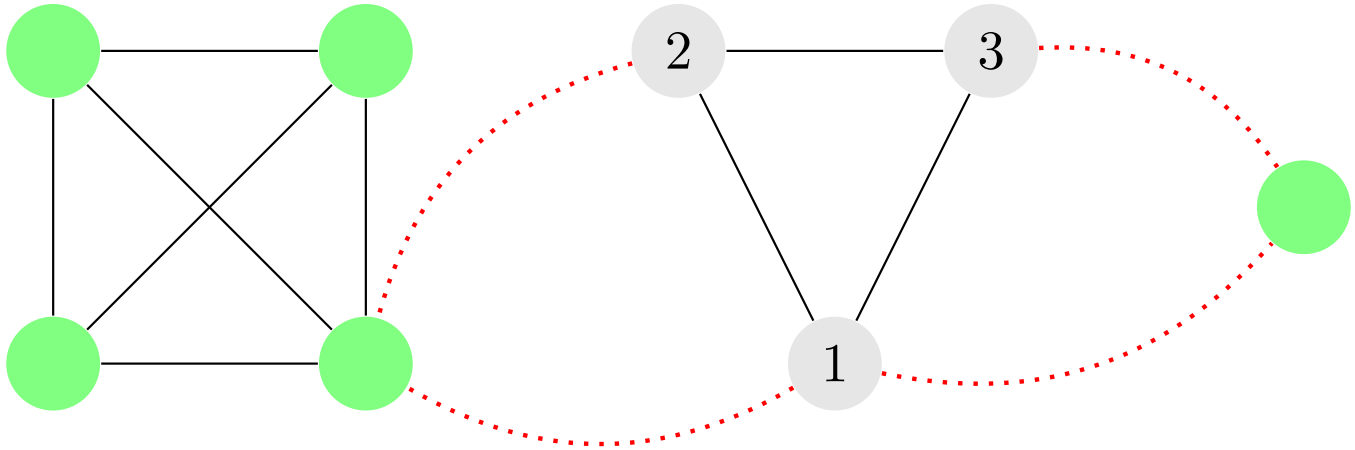
Solution

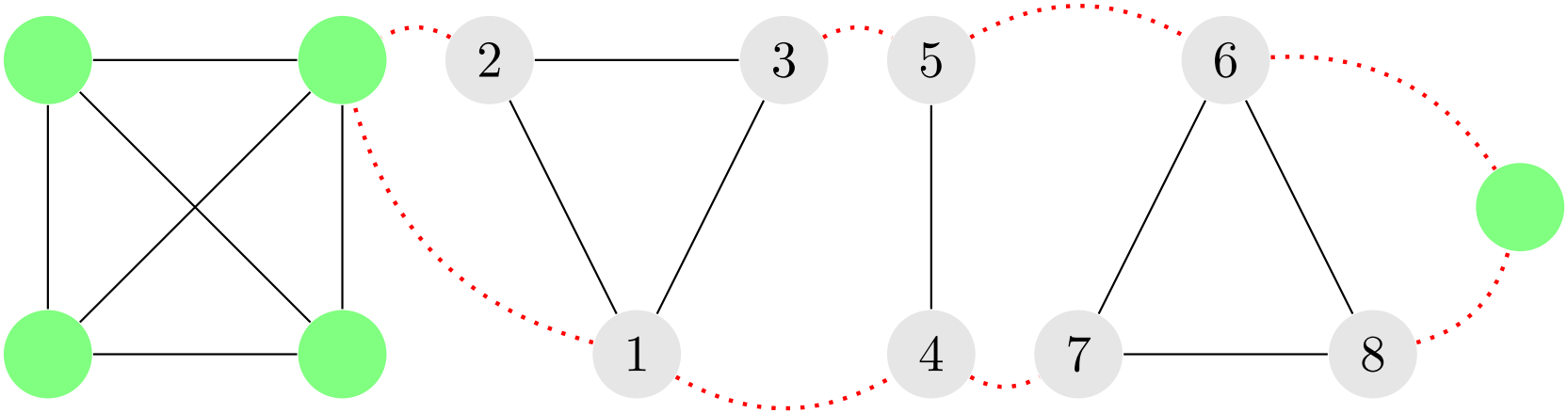
对于大小大于 3 或恰好等于 1 的边三，它们不需要外部的帮助，因此可以直接完成。



对于大小为 2 或 3 的边三，如果这个边双内存在至少两个可以直接完成的边三，那么可以通过任取两个这样的边三辅助来构造出这些边三。

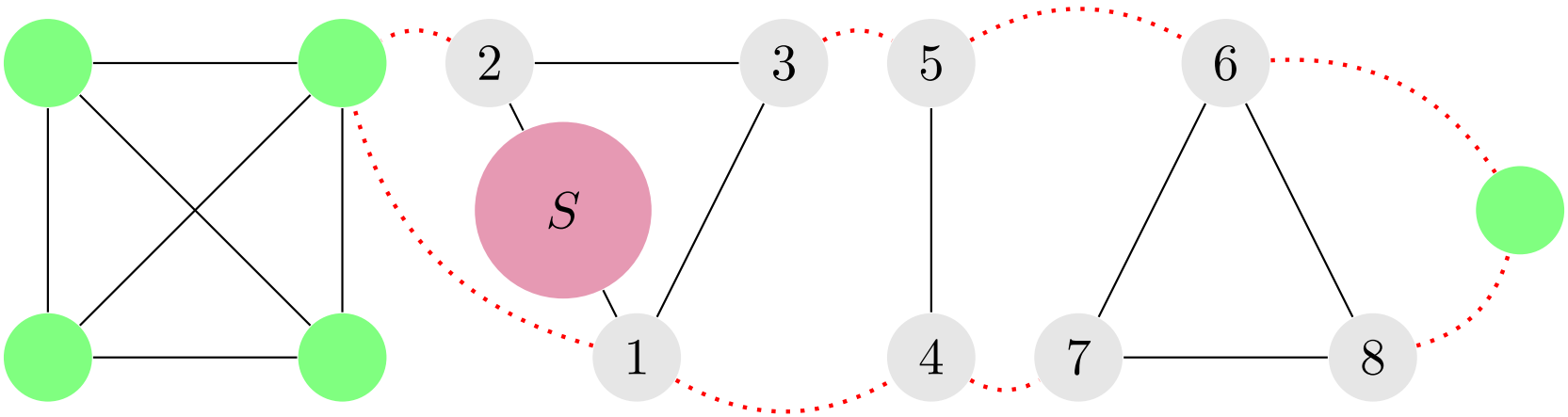
Solution





Solution

这个时候，对于其他的边三，我们可以直接塞到 $1-2$ 的边中来串起来。

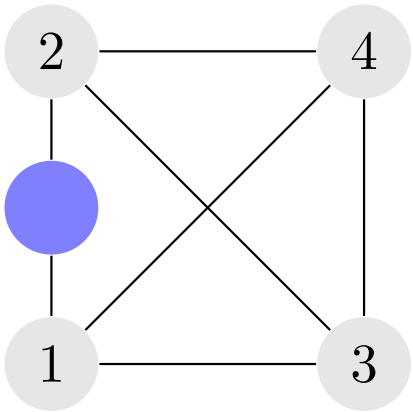


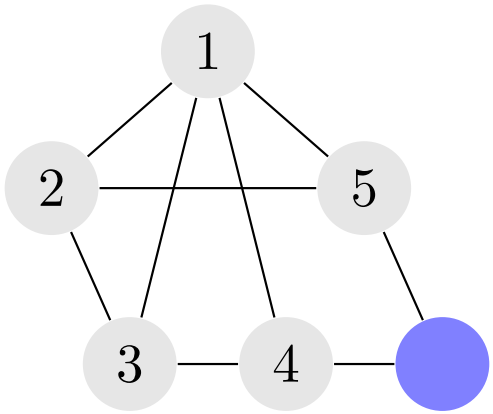
Solution

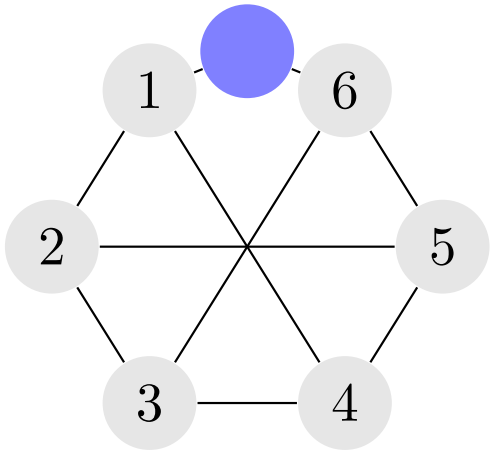
一个特例是，如果这个边双中只有两个边三，那么需要注意如何将这两个边三串起来。

特例中的特例为某一个边三为单点。

对于前者，另一个边三的 size 必须至少为 4，否则容易验证其无解。此时，我们修改对另一个边三点构造，在某一条边中多塞入一个点。







Statements

给一个 n 个点的简单多边形和一个固定的起点，q 组询问多边形内部，从这个起点出发的最短路。

$1 \leq n, q \leq 5 \times 10^3$

Solution

考虑一根皮筋，一端固定到起点，另一端固定到多边形的边上。我们能保证起点到皮筋上的点都是最短路。

我们把端点在多边形边界上绕一圈，这样就能求出到所有点的最短路。

在移动的过程中，中间会遇到行进路线前和后的两种凸起把皮筋绷紧，我们可用一个栈去维护这些点，并且考虑移动的过程中会碰到新的点还是原来的点消失。

每个点入栈出栈一次，所以一共会碰到 $O(n)$ 个事件，每次找下个事件，需要枚举所有点找到碰到的新点，时间复杂度为 $O(n^2)$ 的。

Solution

在绷紧点不变的情况下，皮筋末端划过的三角形经过的询问点都可以回答。

这个过程实际上可以导出一个多边形的三角剖分，对于每个询问，我们只需要定位到它在哪个三角形内就可以快速回答。

在本题中不需要快速点定位，所以使用暴力即可，这部分时间复杂度为 $O(nq)$，总复杂度 $O(n^2 + nq)$。

事实上，在文献 [Guibas-Hershberger ť89] 中描述了一个，可以 $O(n + q \log n)$ 解决简单多边形内部任意两点最短路的算法。但是实现过于复杂，大家可以自行参考。

Statements

给定一张 n 个点 m 条边的无向连通图，和若干个数，你需要把数字填入到图中，使得

$$\sum_{(u,v)\in E} |a_u - a_v| \leq \max_{1\leq i\leq n}\{a_i\} - \min_{1\leq i\leq n}\{a_i\}$$

$1 \leq n \leq 5 \times 10^5, 1 \leq m \leq 10^6$

Solution

首先可以注意到，上式满足的条件为每个双连通分量必须填相同的数字，并且所有 a_i 相同的块，按从小到大的顺序形成了一条链的结构。

直觉上，如果有个环上面的数字不同，那么这些会贡献至少两倍极差，所以一个环上的数字一定相同。如果没有形成链的结构，那么在分叉的地方一定也会有些数字贡献了至少两倍极差，严谨的证明见后。

可以先把每个双连通分量缩起来，得到了一个树形结构。要在树上找一条有向的路径，把所有数沿着路径填入。

Solution

首先将 a_i 从小到大排序，记 c 为多少对相邻的 a_i 不同。

一条有向边 (u, v) 是好的，当且仅当把它分成两部分，前后的大小分别为 $k, n-k$，且序列 a 中，第 k 大和第 $k+1$ 大的数字不同。

直觉上，相当于把前 k 大填入到其中一边，另外的填入到另一边。

那么一条路径是好的，也就是数字能沿着这条路径从小到大填入，当且仅当上面有 c 条好的边。

可以使用树形 DP，记录好边最多的路径，来求出这条路径。

时间复杂度 $O(n \log n + m)$。

Solution

证明如下：

- 记 $A_{\leq i}$ 表示所有权值不超过 i 的点的点集，$A_{\geq i}, A_{=i}$ 同理，$\mathrm{cut}(S, T)$ 表示一端在 S，一端在 T 的无向边。
- 左式等于 $\sum_i \mathrm{cut}(A_{\leq i}, A_{\geq i+1})$，因为图连通，所以当 $\min(A) \leq i \leq \max(A) - 1$ 时，cut 一定大于 0。
- 也就是左式一定大于等于 $\max(A) - \min(A)$，等号在每个 cut 都等于 1 取到。
- 也就是对于所有 i，小于等于 i 的点和大于 i 的点分别是个连通块，并且中间只有一条边相连。

Solution

证明如下：

- 也就是对于所有 i，小于等于 i 的点和大于 i 的点分别是个连通块，并且中间只有一条边相连。
- 这样的边一定得是割边，所以每个双连通分量一定相同。
- 并且考虑非空的 $A_{=i}$，与 $A_{<i}$，$A_{>i}$ 之间各由一条边相连，所以把所有权值相同的点缩起来之后，形成一条链。

Statements

这是一道交互题。

交互库有一张联通简单无向图，但你不知道点数和边数。

初始时，你位于 1 号点。你在任意时刻可以知道你直到你所在的点的编号与度数，但你不知道每条出边连向哪一个点。

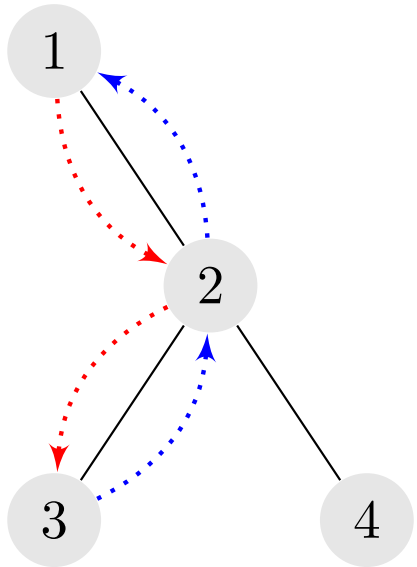
每次你可以选择一个点的一条出边走过去。

使用不超过 $2m + 2n$ 步遍历整张图的所有边至少一次。

Solution

考虑如下朴素地遍历整张图的算法：

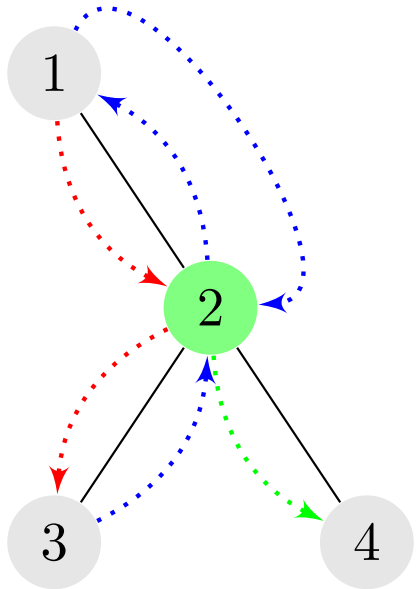
- 对于当前点 x，按照编号从小到大的顺序枚举出边。
- 如果此前已经走过了这条边，跳过。否则，走这条边。



问题：中途可能会漏掉一个点

Solution

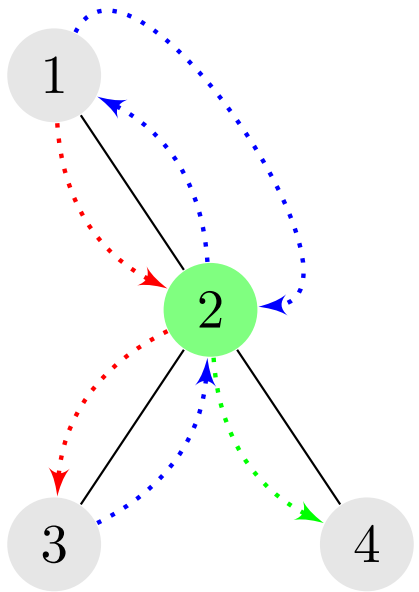
走过 $u \to v$ 时，在 v 的视角下不知道 $v \to u$ 对应的编号。
如果发现走到一个已经经过的点了，立刻返回走回来。
这样相当于，每条边最坏情况下会被来回走两次。



总步数 $4m$，无法通过。

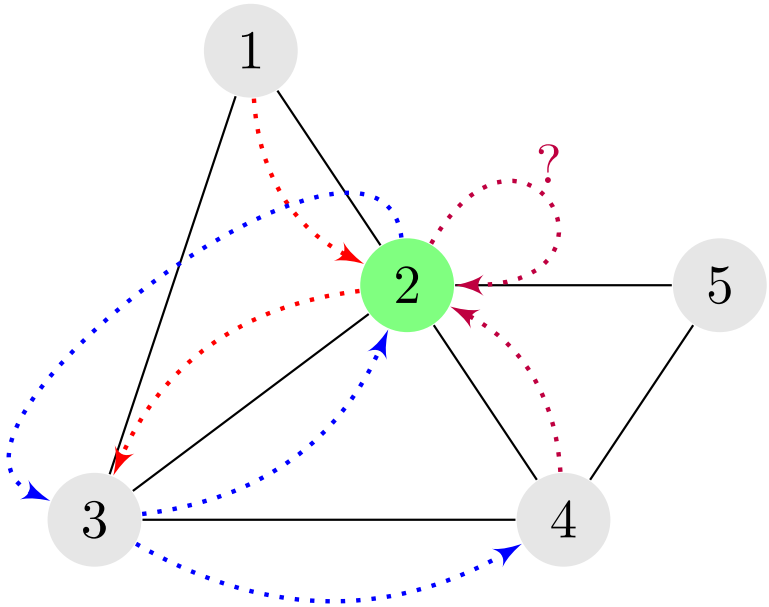
Solution

走过 $u \to v$ 时，在 v 的视角下不知道 $v \to u$ 对应的编号。
如果发现走到一个已经经过的点了，立刻返回走回来。
这样相当于，每条边最坏情况下会被来回走两次。



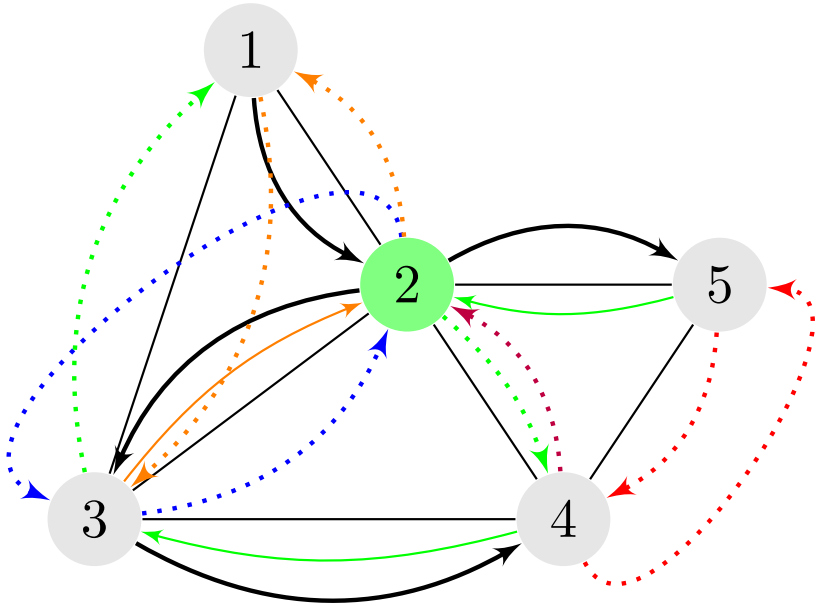
Solution

问题：走过去可能会不知道怎么走回来



Solution

在 DFS 的过程建出生成树，维护所有的树边与每个点的父亲。



在走出边时，如果走非树边回到了一个已经经过的点，那么我们 resume 那个对应点的 DFS 过程。否则，如果所有出边都以

Solution

这样，对于非树边，我们前后只会行走经过一次；对于树边，我们至多前后经过两次。

因此总步数不超过 $2m + 2(n-1) \leq 2(n+m)$。

在实现时，我们可能会多次 for 一个点的出边。如果朴素实现，时间复杂度是 $O(n^2)$ 的。可以使用类似当前弧优化的操作来实现到 $O(n+m)$

Statements

给两个 m 进制下的两个数，你需要重排数位，使得加起来进位尽量多。允许有前导 0。

$1 \leq m \leq 5 \times 10^5$

Solution

首先可以数位较少的那个数补 0，使得两个数数位相同。通过后面的构造可以发现，我们一定能把这些 0 调整到高位，从而不会影响答案。

问题转化为你要找一对数，它们加起来大于等于 m，然后找尽量多对数，它们加起来大于等于 $m-1$。

Solution

我们可以贪心找尽量多对数字，满足它们加起来大于等于 $m-1$。如果在贪心的过程中，已经存在一对数大于等于 m，那么贪心的结果 M 就是答案。

在贪心的过程中，如果保证第一个数从小到大，并且优先匹配第二个数里较大的数位。如果这么做，找不到一对数大于等于 m，那么一定找不到等于 M 的方案。

考虑调整，如果存在两个数加起来大于等于 m，那么可以强制让这两个数匹配，答案也就是 $M-1$。

否则，所有数对加起来小于 m，一定无法进位，答案为 0。

Statements

给你一个 01 串，每次可以选一个子串，把它当成三进制数，然后去掉前导 0 转为二进制，构造从 S 变成 T 的方案。

$1 \leq |S|, |T| \leq 64$，要求步数不超过 512，中间过程长度不超过 128。

Solution

首先特判一下 1，只有 1 能变成它，并且也无法变成别的数。可以猜想别的情况一定有解。

观察一下，有一些实用的操作：$10 \to 11$，$11 \to 100$，$100 \to 1001$，$0 \cdots 0x \to x$。

可以发现有一个环 $10 \to 11 \to 100 \to 10$。

根据这些操作，我们的想法是先把 S 变成某种简单的中间状态，然后从这个简单的中间状态变为 T。

Solution

考虑 11 作为中间状态。那么我们根据上述的可行操作，可以这么构造：

- 把原串调整为 $111\cdots11$ 的形式（$|S|-1$ 步），从前往后构造。
- 把原串调整为 $100\cdots01$ 的形式（$|S|-1$ 步）。
- 把原串调整为 11 的形式（1 步）。

Solution

然后考虑把 11 变成目标串 T。

- 把原串调整成和目标串长度相同的 $111\cdots11$ 的形式 $(3|T|-3$ 步$)$：$11 \to 100 \to 110 \to 111$。
- 把原串调整为目标串 $(2|T|-2$ 步$)$，从后往前构造。

这样就可以在 $O(|S|+|T|)$ 步内完成构造。其中部分步骤可以优化，但是步数限制给的比较宽松，所以大部分线性步数的做法都能通过此题。

Statements

有一个长度为 L 的环，你要从 x 位置出发。同时有 n 个出行需求，你每次只能带一个人，并且只能一次性把一个人带到目的地。

问最少多少时间能把所有人带到目的地。有 q 组关于不同的起始位置的询问。

$1 \leq n \leq 2 \times 10^5, 1 \leq q \leq 10^6$

Solution

首先考虑单组询问，首先我们可以二分答案，然后可以求出，对于每个位置，你在 i 到 $i+1$ 移动了多少次。

把每个出行需求当成一条 $s \to t$ 的有向边，可以根据上面的计算出，除去这些出行需求，你可以额外在 i 到 $i+1$ 移动多少次。

同理，如果你能在 i 到 $i+1$ 额外移动多少次，那么就连若干条 i 到 $i+1$ 的边即可。

存在一种满足的方案，当且仅当整个图存在一条欧拉路径。

根据上述构造，需要满足 i 到 $i+1$ 的移动次数能覆盖所有出行需求对应的次数，且图中所有边连通即可。根据上述构造能保证度数一定是满足条件的。

Solution

考虑多组询问，首先将所有点离散化，考虑离散化之后每个点的答案。

根据上述的能覆盖初始的出行需求，和整个图连通这两个条件，我们分别考虑答案取较大值。覆盖初始的出行需求，这部分简单扫描线即可。

关于图连通，我们先将初始的出行需求对应的边加入，然后令 i 到 $i+1$ 的边权为这条边第一次出现的时间。

问题转化成了求最小瓶颈生成树，也就是问最小的时间，满足将不超过这个时间的边加入，整个图连通。

Solution

我们直接维护最小生成树即可，同时在扫描线的时候，当起点每往前一次，只会有一条边的边权变小，我们可以用 LCT 维护最小生成树。

时间复杂度为 $O((n+q)\log n)$。

这个题也有一些双指针加可撤销并查集，或者一些类似于基于离线的动态图连通性分治做法可以做到差不多的复杂度。