CCPC & ICPC 趣题选讲

周雨扬

北京大学

2021 年 1 月 22 日

题外话

题目选自近几年的 CCPC & ICPC 题目

如果觉得很简单，请不要剧透答案，给其他同学一些思考的时间，~~说不定你做法假了~~。

题外话

题目选自近几年的 CCPC & ICPC 题目

如果觉得很简单，请不要剧透答案，给其他同学一些思考的时间，~~说不定你做法假了~~。

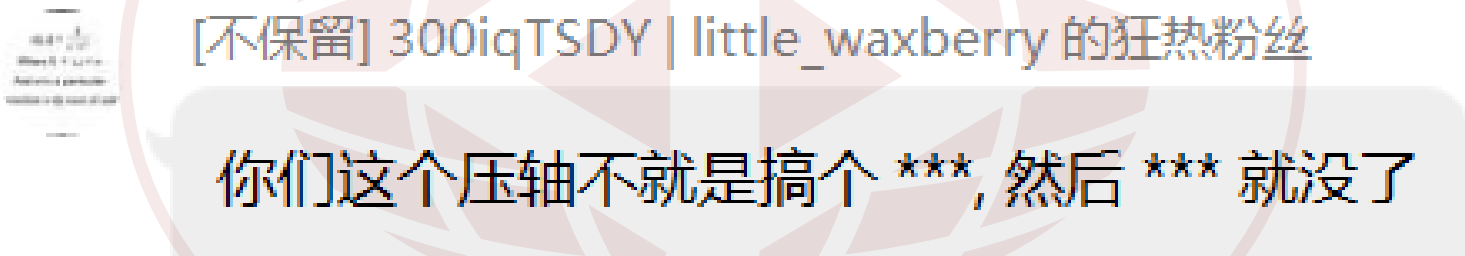

Figure: 当然这种形式的没啥问题

网络节点

给定一颗包含有 n 个节点的树，下标依次为 0 至 $n-1$。
你需要实现两个过程：

- 过程 1：你需要给每一个节点分配一个 0 至 k 的互不相同的编号。在结束过程 1 后，交互库调用 Q 次过程 2.
- 过程 2：交互库会给你 Q 次询问，每次询问会给出起点 S 的编号，终点 T 的编号和所有与 S 相邻的节点的编号。你需要找到与 S 相邻的节点 x，使得以 S 为根时，T 位于以 x 为根的子树中。你需要返回 x 的编号。

注意在执行过程 2 你不能利用过程 1 中的信息。换而言之，过程 2 的程序不知道树的结构，也不知道每一个标号具体对应到哪一个点。

数据范围

子任务 1(5 分): $k = 10^3$，不存在度数大于 2 的节点。

子任务 2(8 分): $k = 10^3$，第 i 条边连接 $i+1$ 和 $\lfloor i/2 \rfloor$

子任务 3(16 分): $k = 10^6$，至多有一个度数大于 2 的节点。

子任务 4(10 分): $n \leq 8, k = 10^9$。

子任务 5(61 分): $k = 10^9$。假设所有测试点中编号最大值为 m，你的得分如下：

$m > 10^9$	0
$2000 < m \leq 10^9$	$50 \log_{5\times 10^5}(10^9/m)$
$1000 < m \leq 2000$	50
$m \leq 1000$	61

部分分

子任务 1 是一条链，只需要按照链的顺序依次编号为 $0 \sim n-1$ 即可。

部分分

子任务 1 是一条链，只需要按照链的顺序依次编号为 $0 \sim n-1$ 即可。

子任务 2 树的结构已知，因此把第 i 个点编号为 i，每一次遍历整棵树即可。

部分分

子任务 1 是一条链，只需要按照链的顺序依次编号为 $0 \sim n-1$ 即可。

子任务 2 树的结构已知，因此把第 i 个点编号为 i，每一次遍历整棵树即可。

子任务 3 类似于菊花，删去中间点之后变成不超过 n 个链。

将中间点删除之后，一条链可以按照 $1000\times$ 链编号 $+$ 链上节点的编号编号，核心点单独编号。查询分情况讨论即可。

部分分

我们考虑将节点编号控制在 $O(n^2)$ 级别。

不妨以节点 0, 记录每一个节点在 dfs 序列上的位置，记入栈序为 dfn，出栈序为 ed。此时我们设当前节点的编号为 $dfn \times 1000 + ed$

部分分

我们考虑将节点编号控制在 $O(n^2)$ 级别。

不妨以节点 0, 记录每一个节点在 dfs 序列上的位置，记入栈序为 dfn，出栈序为 ed。此时我们设当前节点的编号为 $dfn \times 1000 + ed$

此时我们总可以通过判断两个节点的编号判断祖先关系。

如果 T 在 S 子树内，则直接找到对应子树，否则找到 S 的祖先即可。

节点编号级别 $O(n^2)$, 可以得到 50.32 分。

标准算法

考虑将节点规模优化至 $O(n)$.

对于整棵树来说，对于奇数层节点我们在入栈时候加入序列，偶数层节点我们在出栈时加入序列。这里我们认为根节点深度为 0。一个节点的编号为其在序列中的位置。

标准算法

考虑将节点规模优化至 $O(n)$.

对于整棵树来说，对于奇数层节点我们在入栈时候加入序列，偶数层节点我们在出栈时加入序列。这里我们认为根节点深度为 0。一个节点的编号为其在序列中的位置。

不难发现，如果一个节点为奇数层节点，其相邻节点编号均比其大；偶数层节点，其相邻节点编号均比其小。因此我们只需要判断大小关系即可判断层数。

标准算法

不妨考虑奇数层的例子。将相邻节点从小至大排序后，编号最大的节点为其父亲节点，其余为其子节点。因而我们可以判断出所有位于 S 子树内的节点编号。

标准算法

不妨考虑奇数层的例子。将相邻节点从小至大排序后，编号最大的节点为其父亲节点，其余为其子节点。因而我们可以判断出所有位于 S 子树内的节点编号。

同时，子节点的编号将子树编号划分成了若干个子树，因而我们可以判断 T 处于哪一个节点的子树内。否则其显然处于父节点方向，返回父节点即可。

标准算法

不妨考虑奇数层的例子。将相邻节点从小至大排序后，编号最大的节点为其父亲节点，其余为其子节点。因而我们可以判断出所有位于 S 子树内的节点编号。

同时，子节点的编号将子树编号划分成了若干个子树，因而我们可以判断 T 处于哪一个节点的子树内。否则其显然处于父节点方向，返回父节点即可。

偶数层的分析过程类似，这里留给选手自己思考。

标准算法

不妨考虑奇数层的例子。将相邻节点从小至大排序后，编号最大的节点为其父亲节点，其余为其子节点。因而我们可以判断出所有位于 S 子树内的节点编号。

同时，子节点的编号将子树编号划分成了若干个子树，因而我们可以判断 T 处于哪一个节点的子树内。否则其显然处于父节点方向，返回父节点即可。

偶数层的分析过程类似，这里留给选手自己思考。

因而我们只需要 n 个编号即可完成寻址，节点编号最大值为 999。

期望得分 100。

数蘑菇

给定 n 个元素，下标为 0 至 $n-1$。每个元素属于 A 或者 B，已知编号为 0 的元素属于 A。

每一次你可以选定若干个互不相同元素，将其排成一列后向交互器询问，交互器会返回这个序列中有多少对相邻的元素，使得他们的种类互不相同。

你需要使用不超过 20000 次操作询问出属于 A 的元素的个数，同时你在交互器中询问的元素个数和不能超过 100000。

数据范围

如果询问调用不合法会得到 0 分，否则假设你询问次数最大值为 m，你的得分如下表：

$20000 < m$	0
$10010 < m \leq 20000$	10
$904 < m \leq 10010$	25
$226 < m \leq 904$	$22600/m$
$226 \leq m$	100

部分分

由于我们已知 0 属于 A, 我们可以通过 $x1$ 0 $x2$ 询问出 $x1, x2$ 有多少个 A 的蘑菇。因而总询问次数为 $n/2$，可以得到 10 分。

部分分

如果我们已知有 n 个相同的蘑菇，假设下标依次为 $a_1, a_2, \cdots$，则我们总可以通过一次形如 $a_1\ b_1\ a_2\ b_2\ \cdots$ 询问问出 $b_1, b_2, \cdots$ 个元素中有多少个属于 A 的蘑菇。

部分分

如果我们已知有 n 个相同的蘑菇，假设下标依次为 $a_1, a_2, \cdots$，则我们总可以通过一次形如 $a_1\ b_1\ a_2\ b_2\ \cdots$ 询问问出 $b_1, b_2, \cdots$ 个元素中有多少个属于 A 的蘑菇。

同时，我们总可以通过一次询问，唯一确定一个蘑菇的种类。

因而我们总可以通过 $2S$ 次操作找到一个大小为 S 的集合，使得所有元素种类相同。

部分分

如果我们已知有 n 个相同的蘑菇，假设下标依次为 $a_1, a_2, \cdots$，则我们总可以通过一次形如 $a_1\ b_1\ a_2\ b_2\ \cdots$ 询问问出 $b_1, b_2, \cdots$ 个元素中有多少个属于 A 的蘑菇。

同时，我们总可以通过一次询问，唯一确定一个蘑菇的种类。

因而我们总可以通过 $2S$ 次操作找到一个大小为 S 的集合，使得所有元素种类相同。取 $S = \sqrt{n}$，则可以通过 $2.82\sqrt{n}$ 次询问解决，可以得到 56 分。

部分分

观察形如 $a_1\ b_1\ a_2\ b_2\ \cdots$ 的询问。不难发现，b_n 的类型总可以通过返回值奇偶性确定。

因此我们一边询问一边更新集合，每次选出个数较多的做询问。

部分分

观察形如 $a_1\ b_1\ a_2\ b_2\ \cdots$ 的询问。不难发现，b_n 的类型总可以通过返回值奇偶性确定。

因此我们一边询问一边更新集合，每次选出个数较多的做询问。

询问次数可以优化至 $2\sqrt{n} = 282$, 可以得到 80.32 分。

部分分

同时如果我们询问 0 $x1$ 0 $x2$, 我们可以同时确定这两个元素的类型。

将其与之前边询问边确定的算法结合，询问次数可以得到 $1.75\sqrt{n} = 245$, 可以得到 92.24 分。

部分分

同时如果我们询问 0 $x1$ 0 $x2$, 我们可以同时确定这两个元素的类型。

将其与之前边询问边确定的算法结合，询问次数可以得到 $1.75\sqrt{n} = 245$, 可以得到 92.24 分。

虽然 245 理论上是可以严格卡到的，但是稍微手操一下，可以卡到 244 次，也就是 92.62 分。

~~142857cs 他没有卡进这一次操作，然后以 0.38 分的差距被区分到了第七。如果他卡进去了，中国队就有四个前三了:)~~

标准算法

我们考虑一个类似的问题：

- 有一个长度为 n 的 01 序列 A，你每一次可以询问任意一个子集的和，使用尽量少的操作询问出每一个元素的值。

标准算法

在那一道题目中，假设我们存在一种使用 a 次询问问出 b 个元素的方案。假设元素集合为 $B1$，第 i 次询问的子集为 $S1_i$。

同时我们假设存在另外一个大小为 b 的集合，假设元素集合为 $B2$，第 i 次询问的子集为 $S2_i$。

标准算法

在那一道题目中，假设我们存在一种使用 a 次询问问出 b 个元素的方案。假设元素集合为 $B1$ ，第 i 次询问的子集为 $S1_i$。

同时我们假设存在另外一个大小为 b 的集合，假设元素集合为 $B2$ ，第 i 次询问的子集为 $S2_i$。

我们额外增加一次询问，询问的集合为 $B2$。同时我们对于原先方案的第 i 次询问，我们构造新的询问 $S1_i + S2_i$ 和 $S1_i + (B2/S2_i)$。

则将两次询问的和相加后，减去询问 $B2$ 的和，则我们可以得到 2 倍的 $S1_i$ 的元素和。因而我们可以额外根据奇偶性确定额外一个元素的值。

标准算法

我们通过一次询问 $0, a_0, ..., 0, a_n, 0$ 的操作，询问出 $a_0, a_1, ..., a_n$ 中存在着多少个属于 B 的元素。这样子我们就实现了一个简单的询问操作。

假设初始时 $a = b$，则我们可以利用 $2a + 1$ 次询问问出 $3a$ 个元素的值。加上利用询问过程中末尾的 0，利用每一次询问额外确定一个元素，因此假设我们存在 n 个相同类型元素，我们就可以在 $2n + 1$ 次操作中确定出 $5n + 1$ 个元素。

标准算法

我们通过一次询问 $0, a_0, ..., 0, a_n, 0$ 的操作，询问出 $a_0, a_1, ..., a_n$ 中存在着多少个属于 B 的元素。这样子我们就实现了一个简单的询问操作。

假设初始时 $a = b$，则我们可以利用 $2a + 1$ 次询问问出 $3a$ 个元素的值。加上利用询问过程中末尾的 0，利用每一次询问额外确定一个元素，因此假设我们存在 n 个相同类型元素，我们就可以在 $2n + 1$ 次操作中确定出 $5n + 1$ 个元素。

如果阈值得当可以实现 227 次询问的上界，~~然后当场通过的 William Lin 应该使用的就是较为巧妙的该种算法，但是我在 loj 上就死活卡不进去~~。

标准算法

我们也可以反复调用这个算法来降低询问次数。在朱震霆学长的这一篇博客中，利用这种思路将交互次数优化到了 203 次。

博客链接 https://codeforces.com/blog/entry/82924。

J. Abstract Painting

给定 n 个圆，第 i 个圆心为 $(0, x_i)$, 半径为 r_i , 且 $0 \le x_i \le K$, $1 \le r_i \le 5$。

你需要插入若干个圆，第 i 个圆心为 $(0, X_i)$, 半径为 R_i，且 $0 \le X_i \le K$,$1 \le R_i \le 5$。注意你可以不插入任何圆。

有多少种方案，使得插入结束后，图中任意两个不同的圆之间交点个数不超过 1。答案对 998244353 取模。

$1 \le K \le 1000, 1 \le n \le 5000$

标准做法

一个方案合法当且仅当 $[x_i - r_i, x_i + r_i]$ 这些区间互相之间只有严格包含和相离关系。且没有完全相同的线段。

换而言之如果存在两个区间 l_1, l_2，且 l_1 长度不小于 l_2，则必满足 l_2 为 l_1 的严格子区间，或者 l_2 与 l_1 的交只有至多一个点。

标准做法

一个方案合法当且仅当 $[x_i - r_i, x_i + r_i]$ 这些区间互相之间只有严格包含和相离关系。且没有完全相同的线段。

换而言之如果存在两个区间 l_1, l_2，且 l_1 长度不小于 l_2，则必满足 l_2 为 l_1 的严格子区间，或者 l_2 与 l_1 的交只有至多一个点。

考虑从左到右维护扫描线，记录前 10 个端点哪一些是合法的。

插入一条 $[x, y]$ 的线段会使得 $[x+1, y-1]$ 非法，打标记即可。

时间复杂度 $O(n \times 2^{10})$

I. Invaluable Assets

给定常数 C，对于任意正整数 i，你都有无穷多个大小为 i 的物品。每一个大小为 i 物品的质量为 $i^2 + C$。

有 n 个无穷大的背包，第 i 个背包已有了体积为 a_i 的物品。

Q 次询问，第 i 次询问给出一个 k，询问如果填充物品使得每一个背包的物品体积不小于 a_i，新增物品质量和最小为多少。

$n, Q \leq 10^5, C \leq 10^4, a_i, k$ 在 $[0, 10^9]$ 等概率随机生成。

标准算法

如果物品可以取到任意非负实数体积，则我们选择的每一个物品体积均相同，且每个物品体积会在所有合法方案中，选择 $\frac{i^2+C}{i}$ 值最小的那一种。

设 $f(x)=\frac{x^2+C}{x}$，分析该函数的导数 $1-\frac{C}{x^2}$，可以推导出该函数为严格凸函数，且恰好在 $\sqrt{C}$ 处取到最小值 $2\sqrt{C}$。

标准算法

如果物品可以取到任意非负实数体积，则我们选择的每一个物品体积均相同，且每个物品体积会在所有合法方案中，选择 $\frac{i^2+C}{i}$ 值最小的那一种。

设 $f(x)=\frac{x^2+C}{x}$，分析该函数的导数 $1-\frac{C}{x^2}$，可以推导出该函数为严格凸函数，且恰好在 $\sqrt{C}$ 处取到最小值 $2\sqrt{C}$。

据此分析，设 $\theta=\mathbf{argmin}_{x\in\mathbb{N}+}f(x)$，若有多个则选取任意一个。则有 θ 数量级为 $O(\sqrt{n})$

同时有推导出 $f_1,f_2,\cdots,f_\theta$ 是一个不增的序列，$f_\theta,f_{\theta+1},\cdots$ 是一个不降的序列，且有：$\forall i\geq 1, f_{i+2}-f_{i+1}>f_{i+1}-f_i$。

标准算法

我们不妨假设存在参数 N, 使得对于任意正整数体积的背包，总存在一种最优方案，选取的体积不是 θ 的物品体积和不超过 N。

标准算法

我们不妨假设存在参数 N, 使得对于任意正整数体积的背包，总存在一种最优方案，选取的体积不是 θ 的物品体积和不超过 N。

同时，我们可以证明参数 N 是 $O(C)$ 级别的。

这里我们发现，如果存在一个方案存在两个体积为 x, y 的物品，且满足 $x+1 \leq y-1$。此时根据凸函数的性质，两物品体积修改为 $x+1, y-1$ 答案一定不会变劣。因而对于任意最优解，最多只有两种体积的物品，且体积差不超过 1。

标准算法

同时，如果一个方案中体积非 θ 的体积和超过 $3C$, 则我们总可以通过调整物品个数优化答案。具体的，如果个数超过 $3\sqrt{n}$，则减少物品个数，否则增加物品个数。

标准算法

同时，如果一个方案中体积非 θ 的体积和超过 $3C$, 则我们总可以通过调整物品个数优化答案。具体的，如果个数超过 $3\sqrt{n}$，则减少物品个数，否则增加物品个数。

因此，任意体积大于 $3C$ 的方案中至少有一个物品体积 θ。因而如果存在 $x > 3C$，使得 $f(x+\theta) \neq f(x) + \theta^2 + C$，则总可以导出矛盾的结论。

标准算法

同时，如果一个方案中体积非 θ 的体积和超过 $3C$, 则我们总可以通过调整物品个数优化答案。具体的，如果个数超过 $3\sqrt{n}$，则减少物品个数，否则增加物品个数。

因此，任意体积大于 $3C$ 的方案中至少有一个物品体积 θ。因而如果存在 $x > 3C$，使得 $f(x+\theta) \neq f(x) + \theta^2 + C$，则总可以导出矛盾的结论。

这提示我们可以预处理出体积 $\leq 3C$ 的背包，之后部分可以按照模 θ 的同余类看成是 θ 个互不相同的一次函数。

标准算法

由于物品 a_i, k 随机，因而每一次询问填充体积小于 $3C$ 的期望个数为 $\frac{3C\times 10^5}{10^9}$。小范围暴力求物品体积小于 $3\sqrt{C}$ 的背包即可。

其余的部分我们可以按照模 θ 的同余类暴力查询即可。

不同的询问之间可以通过扫描线离线解决。

标准算法

由于物品 a_i, k 随机，因而每一次询问填充体积小于 $3C$ 的期望个数为 $\frac{3C\times 10^5}{10^9}$。小范围暴力求物品体积小于 $3\sqrt{C}$ 的背包即可。

其余的部分我们可以按照模 θ 的同余类暴力查询即可。

不同的询问之间可以通过扫描线离线解决。

时间复杂度 $O(C^{1.5} + QC^{0.5} + \frac{NCQ}{10^9})$。

F. Skeleton Dynamization

给定一张 n 个点 m 条边的无向简单图 $G = \{V, E\}$。
你需要将其划分成 $k(k \geq 2)$ 张子图，满足如下条件:

- 每张子图节点数相同，均为 n'，且 n 个点都恰好属于某一张子图。这里假设第 i 张子图的第 j 个节点为 (i, j)。
- $\forall 1 \leq i < j \leq k, 1 \leq x < y \leq n'$，$(i, x)$ 和 (j, y) 有边当且仅当 $j = i + 1$ 且 $x = y$。
- $\forall 1 \leq i < j \leq k, 1 \leq x < y \leq n'$，$(i, x)$ 和 (i, y) 有边当且仅当 (j, x) 和 (j, y) 有边。

题目大意

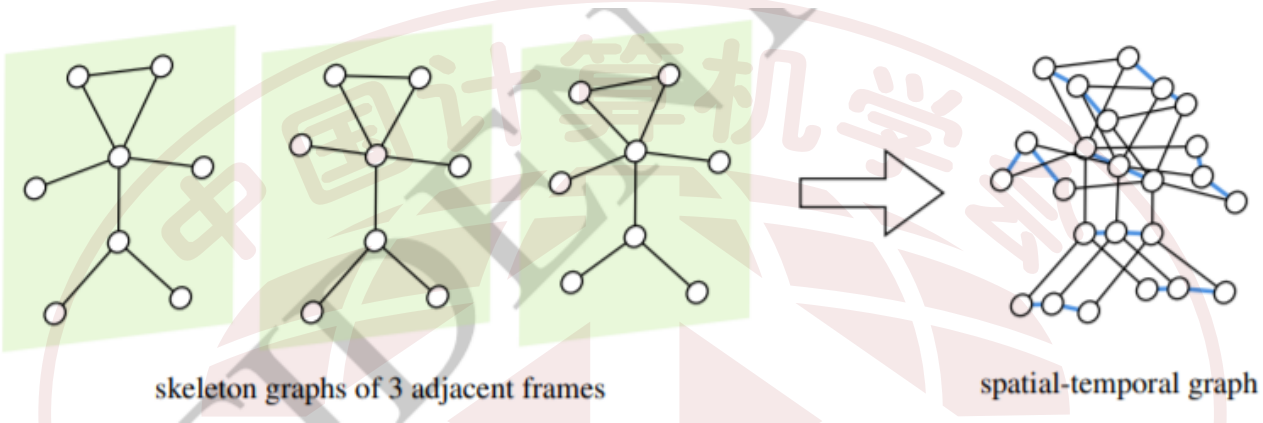

Figure: 一个简单的例子

你需要求出 G 的划分，如果有多个划分，输出 k 最大的方案。如果有多个 k 最大的方案，输出任意一个即可。

$1 \leq n \leq 100000, 1 \leq m \leq 200000$。

标准做法

考虑找到任意一条连接两张子图的边。不妨设这条边为 (x, y)。

假设我们找到的边连接的为第 $i, i+1$ 张子图，则由我们的构造，容易得出：对于所有位于子图 $1\ 2, \cdots, i$ 的点，其距离 x 更近。否则距离 y 更近，据此我们可以将图划分为两个集合，设为 S, T。

标准做法

考虑找到任意一条连接两张子图的边。不妨设这条边为 (x, y)。

假设我们找到的边连接的为第 $i, i+1$ 张子图，则由我们的构造，容易得出：对于所有位于子图 $1\ 2, \cdots, i$ 的点，其距离 x 更近。否则距离 y 更近，据此我们可以将图划分为两个集合，设为 S, T。

如果一个 T 中的节点和 S 中结点直接相连，则我们可以直接确定其为第 $i+1$ 张子图的点。因而我们可以将这些节点从 T 中移除。对于 S 也有类似的结论。

因而我们很容易就可以得到每一张子图的编号，以及对应的节点集合。

标准做法

同时，我们容易根据连接不同子图的边，得到相邻两张子图节点之间的对应关系，因而只需要简单的检查两张图的全等关系即可。

因而单次检查的复杂度为 $O(n+m)$。

标准做法

同时，我们容易根据连接不同子图的边，得到相邻两张子图节点之间的对应关系，因而只需要简单的检查两张图的全等关系即可。

因而单次检查的复杂度为 $O(n+m)$。

同时由于这张图至少有 $O(\sqrt{m})$ 个节点，根据鸽巢原理，容易得：这张图片一定有一个节点度数小于 $2\sqrt{m}$。设为 x。

由于我们强制层数 ≥ 2，因而其一定有一条边连接不同层的节点，因而只需要检查 x 相邻的所有边即可。

时间复杂度 $O((n+m)\sqrt{m})$

L. Lost Temple

给定一个 $10^9 \times 10^9$ 的网格图，左上角坐标为 $(0, 0)$。

初始时有 n 组石柱子，第 i 组石柱子包含了坐标为 $(i, l_i), (i, r_i)$ 的石柱，除此之外没有认可石柱。

由于风雨侵袭，每一年，如果一个石柱存在一个存在相邻边的空格子，则他会在这一年结束的时候崩塌，石柱所在的这个格子也会变为空格子。

对于每一组石柱，询问在哪一年，这一组石柱全部崩塌。

$\Sigma n \leq 1.2 \times 10^7, 1 \leq l_i \leq r_i \leq 5 \times 10^8$。

L. Lost Temple

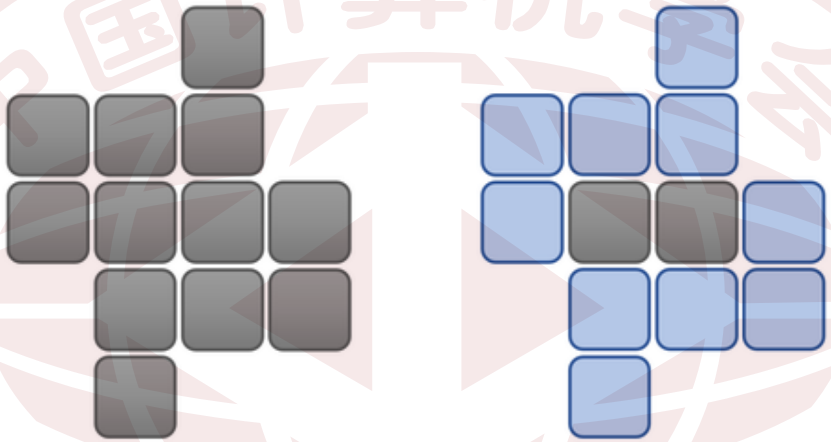

Figure: 石柱的崩塌过程

标准做法

考虑给定的一个石柱 (x, y)，如何计算其崩塌时间。

如果存在一个空格子 (x', y')，则其显然会在 $\max\{|x - x'|, |y - y'|\}$ 年内崩塌。

标准做法

考虑给定的一个石柱 (x, y)，如何计算其崩塌时间。

如果存在一个空格子 (x', y')，则其显然会在 $\max\{|x - x'|, |y - y'|\}$ 年内崩塌。

在这种情况下设第 i 组的答案为 ans_i，则显然有：$ans_i - 1 \leq ans_{i+1} \leq ans_i + 1$。因而 $i - ans_i, i + ans_i$，也就是极大合法的左右端点均单调增。

不妨设 $(i, l_i - 1), (i, r_i + 1)$ 为上下关键点，考虑用单调栈分别维护左右侧的上下关键点的四个单调队列。队列维护的是 x, y 两轴的坐标的和差，用于计算当前石柱没有崩塌的区间的上下边界。

标准做法

我们同时维护队列左右两侧 4 个方向的单调队列，同时记录当前极大合法区间左右端点 L, R。

从 $i-1$ 转移到 i 的时候，右端点最多向右移动 2，因而这些零散的部分可以通过暴力询问求得。

标准做法

我们同时维护队列左右两侧 4 个方向的单调队列，同时记录当前极大合法区间左右端点 L, R。

从 $i-1$ 转移到 i 的时候，右端点最多向右移动 2，因而这些零散的部分可以通过暴力询问求得。

每一次中点的移动，合法区间左右端点的移动，相当于从若干个队列中删去起始元素，加入末尾元素。检验合法性等价于求最小值，因而直接维护 4 个单调队列即可。

时间复杂度 $O(n)$。

H. Rice Arrangement

有一个可以作 n 个人的圆形餐桌，桌上有 k 个人，k 碗饭。第 i 个人坐在 a_i 个位置，第 i 盘菜正对着 b_i 个位置。

你是服务员，每一次可以化一单位的时间将桌子顺时针或逆时针旋转 1 单位，使得原来面对第 i 个位置的菜面对第 $(i+1) \bmod n/(i+n-1) \bmod n$ 个人。

在每一次旋转前和旋转结束后，如果一个人面对一碗饭，它可以不花时间取走这盘饭。每个人需要恰好一碗饭。

假设在场所有人都想要最小化旋转花费的时间，输出最小花费时间。

$n \leq 10^9, k \leq 2000$

标准做法

定义顺时针方向为左。如果第 i 个人在顺时针转了若干圈拿到了饭，则连接一条顺时针方向连接 i 所在位置和饭所在位置的环上区间。否则按照逆时针连接 i 所在位置和饭所在位置的环上区间。

不难发现存在区间互相包含的情况。

如果区间包含，我们总可以通过调整得到一个更优的答案。

标准做法

不妨将 a,b 从小到大排序，根据上面描述，一定存在一种 b 的循环位移，使得最优解中 a_i 的人拿到了 b_i 的饭。

否则容易得到一对相交区间，矛盾。

标准做法

不妨将 a,b 从小到大排序，根据上面描述，一定存在一种 b 的循环位移，使得最优解中 a_i 的人拿到了 b_i 的饭。

否则容易得到一对相交区间，矛盾。

因此我们可以确定每一对匹配需要顺时针转动的次数。而选择逆时针匹配的肯定是顺时针转动次数最大的那几对。因而排序后贪心即可。

时间复杂度 $O(k^2 \log k)$

K. Traveling Merchant

给定一张 n 个点 m 条边的无向无重边自环的简单图，每一个点有颜色 0/1。保证 1 号点初始颜色为 0。

你现在在 1 号点，每当你离开节点 x，节点颜色会改变。

你希望你经过节点颜色依次为 $01010101\cdots$。

判断是否存在一条从 1 出发的无限长的满足你要求的路径。

$n, m \leq 200000$。

标准做法

不妨考虑任意合法的序列。如果存在一个节点 x 序列之中出现了两次以上，则显而易见存在一个合法解。我们直接沿着这个环不断走就是一种合法方案。

标准做法

不妨考虑任意合法的序列。如果存在一个节点 x 序列之中出现了两次以上，则显而易见存在一个合法解。我们直接沿着这个环不断走就是一种合法方案。

不妨假设最终方案里第一个出现两次的点为 x，第二次进入节点 x 之前所在的节点为 y。则显而易见在初始状态中 x, y 同色。而 y 之前序列所有相邻元素异色。

标准做法

不妨考虑任意合法的序列。如果存在一个节点 x 序列之中出现了两次以上，则显而易见存在一个合法解。我们直接沿着这个环不断走就是一种合法方案。

不妨假设最终方案里第一个出现两次的点为 x，第二次进入节点 x 之前所在的节点为 y。则显而易见在初始状态中 x, y 同色。而 y 之前序列所有相邻元素异色。

这里不妨假设第一次进入环的边为 (x, y)，他从 y 走向了 x。则如果存在一条 $1 \to x \to y$ 的不经过重复节点的颜色交错路径，则我们总可以构造出一组合法答案。

同时，如果存在合法解，根据上面推导过程则显而易见存在一组满足条件的 (x, y)。

标准做法

提取出所有连接异色点的边。假设我们求出了点双连通分量，并据此求出了原图的圆方树 T。

则存在 $1 \to x \to y$ 的不经过重复节点的交错路径当且仅当 T 在以 1 为根的时候，满足以下两种中的至少一种：

- y 在 x 子树内。
- x 父节点为方点，且 y 在 x 父亲的子树内。

标准做法

提取出所有连接异色点的边。假设我们求出了点双连通分量，并据此求出了原图的圆方树 T。

则存在 $1 \to x \to y$ 的不经过重复节点的交错路径当且仅当 T 在以 1 为根的时候，满足以下两种中的至少一种：

- y 在 x 子树内。
- x 父节点为方点，且 y 在 x 父亲的子树内。

求点双，预处理父子关系均可以 $O(n)$，单次判断可以 $O(1)$，因而该算法总时间复杂度 $O(n)$。

标准做法

提取出所有连接异色点的边。假设我们求出了点双连通分量，并据此求出了原图的圆方树 T。

则存在 $1 \to x \to y$ 的不经过重复节点的交错路径当且仅当 T 在以 1 为根的时候，满足以下两种中的至少一种：

- y 在 x 子树内。
- x 父节点为方点，且 y 在 x 父亲的子树内。

求点双，预处理父子关系均可以 $O(n)$，单次判断可以 $O(1)$，因而该算法总时间复杂度 $O(n)$。

~~记得别写错点双。~~

H. Rikka with Storehouse

给定一颗包含 2^n-1 个节点的满二叉树，节点编号为 $1\sim 2^n-1$。$\forall 2\le i<2^n-1$, 第 i 个节点父节点为 $\lceil \frac{i}{2} \rceil$。

所有叶子节点有一个固定的权值 v_i，其余权值未定。

你需要确定所有非叶子节点的权值 v_i，使得 $\sum (v_i-v_{\lfloor i/2 \rfloor})^2$ 最小，输出答案对 998244353 取模的结果。

同时会有 Q 次单点修改，你也需要求出修改后上式最小值对 998244353 取模的结果。

$n\le 18, Q\le 200000$

标准做法

这里尝试归纳证明，假设 i 号节点为非叶子节点，$v_i = t$，i 子树内非叶子节点权值任意的情况下子树内权值最小值为 $f_i(t)$，则 $f_i(t)$ 总为一个关于 t 的二次多项式。

标准做法

这里尝试归纳证明，假设 i 号节点为非叶子节点，$v_i = t$，i 子树内非叶子节点权值任意的情况下子树内权值最小值为 $f_i(t)$，则 $f_i(t)$ 总为一个关于 t 的二次多项式。

对于非叶子节点 x，如果两个儿子均为叶节点，则上式显然成立。

否则我们对左子树右子树分别考虑。设左子树对应函数为 $f(t) = at^2 + bt + c$。此时显然有 $g(t) = \min ax^2 + bx + c + (t - x)^2 = \min(a+1)x^2 + (b-2t)x + c + t^2$。

通过求导我们容易得出在 t 固定时,x 极值点为 $\frac{(2t-b)}{2(a+1)}$。

将极值点权值代入，则 $g(t)$ 也为关于 t 的二次多项式。

标准做法

因而对于树上每一个节点，我们直接维护该多项式对应的系数即可。

容易验证全过程中确定的二次多项式 ax^2+bx+c 中，a 在模意义下恒不为 0。

时间复杂度 $O((n+m)\log n\log 998244353)$。

标准做法

因而对于树上每一个节点，我们直接维护该多项式对应的系数即可。

容易验证全过程中确定的二次多项式 ax^2+bx+c 中，a 在模意义下恒不为 0。

时间复杂度 $O((n+m)\log n\log 998244353)$。

注意系数 a 取值仅和树结构相关，与 v 无关，因而可以省去修改时求逆部分，复杂度 $O((n+m)(\log n+\log 998244353))$。

H. Harmonious Rectangle

给定一个 $n \times m$ 的棋盘 A，你可以给棋盘染三种颜色种的任意一种。假设给第 i 行第 j 列的格子染的颜色数为 $A_{i,j}$, 行列均从 1 开始编号。

定义一种染色方案是好的，当且仅当: 存在 $1 \le p < q \le n, 1 \le r < s \le m$, 使得以下两个条件中至少满足一个:

- $A_{p,r} = A_{p,s}$ 且 $A_{q,r} = A_{q,s}$
- $A_{p,r} = A_{q,r}$ 且 $A_{p,s} = A_{q,s}$

求好的染色方案数，对 998244353 取模。

标准解法

这里不妨假设 $n \leq m$.

标准解法

这里不妨假设 $n \leq m$.

如果 $n = 1$, 则显而易见所有方案均不合法。

否则不妨假设 $p = 1, q = 2$. 考虑对于所有 $i \in [1, m]$ 形成的有序对 $(a_{p,i}, a_{q,i})$。

标准解法

这里不妨假设 $n \le m$.

如果 $n = 1$, 则显而易见所有方案均不合法。

否则不妨假设 $p = 1, q = 2$. 考虑对于所有 $i \in [1, m]$ 形成的有序对 $(a_{p,i}, a_{q,i})$。

考虑如下 7 个等价类:

$$\begin{array}{ll} \{(1,2)\} & \{(1,3)\}\ \{(2,1)\} \\ \{(2,3)\} & \{(3,1)\}\ \{(3,2)\} \\ \{(1,1),(2,2),(3,3)\} & \end{array} \tag{1}$$

标准解法

不难发现，如果出现了两个属于同一等价类的有序对，则其合法。

因而当 $m > 7$ 时候，所有方案都合法。

标准解法

不难发现，如果出现了两个属于同一等价类的有序对，则其合法。

因而当 $m > 7$ 时候，所有方案都合法。

这也提示我们不合法方案非常少。

因而打出 $m \leq 7$ 时候的表即可。

G. Go

给定一个 $n \times n$ 的围棋棋盘，棋盘上有若干个黑子和白子。

定义一个白子连通块为一个极大的仅仅包含白子的四连通块。这个连通块的气为与这个联通块联通的没有棋子的点个数。

对于棋盘上的所有棋子，如果将其颜色反色，有多少个白子所在的白子连通块气为 0。

$2 \leq n \leq 1000$

G. Go

给定一个 $n \times n$ 的围棋棋盘，棋盘上有若干个黑子和白子。

定义一个白子连通块为一个极大的仅仅包含白子的四连通块。这个连通块的气为与这个联通块联通的没有棋子的点个数。

对于棋盘上的所有棋子，如果将其颜色反色，有多少个白子所在的白子连通块气为 0。

$2 \leq n \leq 1000$

$O(n^2 \log^2 n)$ 无法通过。

不动脑子的做法

本质是个动态连通性，因此用线段树 + 资瓷撤回的并查集直接维护即可。

时间复杂度 $O(n^2 \log^2 n)$。

不动脑子的做法

本质是个动态连通性，因此用线段树 + 资瓷撤回的并查集直接维护即可。

时间复杂度 $O(n^2 \log^2 n)$。

当场会超时。

~~记得特判空棋盘~~

标准解法

不妨按照翻转棋子的颜色，分成黑子和白子讨论。

标准解法

不妨按照翻转棋子的颜色，分成黑子和白子讨论。

当黑子翻转为白子时候，原有的白子连通块不会改变，因而可以预处理出来每个白子连通块节点数，气的个数。

将黑子反色后，等价于插入了一个白子，只需要判断直接与其相连的不超过 4 个连通块信息即可。

标准解法

不妨按照翻转棋子的颜色，分成黑子和白子讨论。

当黑子翻转为白子时候，原有的白子连通块不会改变，因而可以预处理出来每个白子连通块节点数，气的个数。

将黑子反色后，等价于插入了一个白子，只需要判断直接与其相连的不超过 4 个连通块信息即可。

如果采用广搜，时间复杂度 $O(n^2)$。

标准解法

当白子翻转为黑子时候，考虑将相邻的两个白子连边后，求出这一张无向图的点双连通分量。

因而，这种情况下等价于从原图中删除一个节点，判断新生成的若干个连通块是否有气。

标准解法

当白子翻转为黑子时候，考虑将相邻的两个白子连边后，求出这一张无向图的点双连通分量。

因而，这种情况下等价于从原图中删除一个节点，判断新生成的若干个连通块是否有气。

建立出圆方树后，删去该节点得到的树上连通块就是最终我们得到的连通块。

因而维护子树节点个数，以及子树气的数量即可。

标准解法

当白子翻转为黑子时候，考虑将相邻的两个白子连边后，求出这一张无向图的点双连通分量。

因而，这种情况下等价于从原图中删除一个节点，判断新生成的若干个连通块是否有气。

建立出圆方树后，删去该节点得到的树上连通块就是最终我们得到的连通块。

因而维护子树节点个数，以及子树气的数量即可。

总时间复杂度 $O(n^2)$

E. Tree Paths

给定一个 n 个节点的树。

询问有多少条树链，使得其编号重排后，为一个 $[a, b]$ 的重排列。

$n \leq 50000$, 15 秒。

E. Tree Paths

给定一个 n 个节点的树。

询问有多少条树链，使得其编号重排后，为一个 $[a, b]$ 的重排列。

$n \leq 50000$, 15 秒。

~~其实完全可以出成 $n \leq 10^6$ 的~~。

标准做法

如果是解决连通块问题的话，可以通过简单的扫描线解决。

具体的，我们总可以计算左端点固定时候的子图点数 -边数的值。其为一个连通块当且仅当上式值为 1。证明略去。

对右端点做扫描线，而修改操作仅有线段树加，询问最小值以及最小值数量。

总复杂度 $O(n \log n)$。

标准做法

我们称一个区间 $[l, r]$ 对应的点形成了一条广义链，当且仅当点集在原树上形成的虚树为一条链。

不难发现，如果我们已知 $[l, r]$ 为一条广义链，则所有 $[l, r]$ 的连续非空子区间也是一条广义链。

标准做法

我们称一个区间 $[l, r]$ 对应的点形成了一条广义链，当且仅当点集在原树上形成的虚树为一条链。

不难发现，如果我们已知 $[l, r]$ 为一条广义链，则所有 $[l, r]$ 的连续非空子区间也是一条广义链。

因此在扫描线的时候，我们同时维护一个极大的广义链区间 $[l', r]$。此时对于 $x \geq l$，若 $[x', r]$ 为连通块，则显然其为一条链。

在上面问题的基础上，加入区间询问最小值和数量，以及询问一个集合是否为广义链即可。

时间复杂度 $O(n \log n)$

G. Sheep

你有 n 只羊坐落在一个数轴上。第 i 只羊时刻 t 在数轴上的位置为 $f_i(t) = a_i t + b_i$。你是羊倌，你的移动方式与羊类似，时刻 t 在数轴上的位置为 $g(t) = At + B$。

第 i 只羊的孤单度为

$$[\min_{t\in[0,T]}(f_i(t) - g(t))]^2 + [\max_{t\in[0,T]}(f_i(t) - g(t))]^2$$

找到任意 A, B，使得所有羊孤单度最大值最小。

$n, |a_i|, |b_i| \le 10^5, T \le 100$。

标准做法

考虑一次函数 $g_i(t) = f_i(t) - g(t)$ 的极大极小值。不难发现极大极小一定有一个在 $t = 0$ 取到，另外一个在 $t = T$ 取到。

标准做法

考虑一次函数 $g_i(t) = f_i(t) - g(t)$ 的极大极小值。不难发现极大极小一定有一个在 $t = 0$ 取到，另外一个在 $t = T$ 取到。

据此，我们可以采用 $f_i(0)$ 和 $f_i(T)$ 作为两维的坐标，以平面上的一个点描述一条直线，而根据题目描述，两条直线之间的距离 (孤单度) 就是两个对应点欧氏距离的平方。

标准做法

考虑一次函数 $g_i(t) = f_i(t) - g(t)$ 的极大极小值。不难发现极大极小一定有一个在 $t = 0$ 取到，另外一个在 $t = T$ 取到。

据此，我们可以采用 $f_i(0)$ 和 $f_i(T)$ 作为两维的坐标，以平面上的一个点描述一条直线，而根据题目描述，两条直线之间的距离 (孤单度) 就是两个对应点欧氏距离的平方。

反演之后问题转化为简单的最小圆覆盖问题，其有 $O(n)$ 做法。

时间复杂度 $O(n)$。

A. Arithmetic

给定 a, b, p, k。保证 p 是质数。

询问最小的 x，使得 $a^x + b^x$ 是 p^k 的倍数。

$1 \le a, b, p \le 10^4, p^k \le 10^{18}$

标准做法

如果 a, b 存在不和 p 互质的数字，可以通过简单特判解决。

标准做法

如果 a, b 存在不和 p 互质的数字，可以通过简单特判解决。

否则两侧同除 a^x, 则 $1 + \left(\frac{b}{a}\right)^x \equiv 0 (\mathrm{mod}\ p^k)$, 也就是 $\left(\frac{b}{a}\right)^x \equiv p^k - 1 (\mathrm{mod}\ p^k)$.

标准做法

如果 a, b 存在不和 p 互质的数字，可以通过简单特判解决。

否则两侧同除 a^x, 则 $1 + \left(\frac{b}{a}\right)^x \equiv 0 (\mathrm{mod}\ p^k)$, 也就是 $\left(\frac{b}{a}\right)^x \equiv p^k - 1 (\mathrm{mod}\ p^k)$.

不妨设 k 为最小的正整数，使得 $\left(\frac{b}{a}\right)^x \equiv 1 (\mathrm{mod}\ p^k)$。此时如果 k 为奇数无解，否则答案为 $k/2$。

正确性由构造过程容易证得。

I. Ranks

给定一个 $n \times n$ 的 01 矩阵 $A_{i,j}$。

询问对于矩阵的每一个元素 $A_{i,j}$ 翻转后，模二意义下的线性基大小是否改变，若改变，变大还是变小

$n \leq 1500$。

标准做法

对于矩阵 A，我们称其为最简的，当且仅当:

- 设 A 第 i 行的向量为 a_i
- 假设 a_i 的编号最小的非零元素下标为 b_i，如果为 0 向量则 $b_i = n + 1$
- $\forall i, b_i = n + 1$ 或者矩阵 A 的第 b_i 列有且仅有 1。
- b_i 单调不下降。

标准做法

对于矩阵 A，我们称其为最简的，当且仅当:

- 设 A 第 i 行的向量为 a_i
- 假设 a_i 的编号最小的非零元素下标为 b_i，如果为 0 向量则 $b_i = n + 1$
- $\forall i, b_i = n + 1$ 或者矩阵 A 的第 b_i 列有且仅有 1。
- b_i 单调不下降。

我们总可以通过初等行变换得到唯一的最简矩阵 B。

类似于矩阵求逆过程，我们同时维护一个矩阵 C，记录 C 的每一个行向量是由 A 中的哪一些行向量组成的。

标准做法

对于一次单点修改，我们尝试尽量少的修改 B 对应的元素。

具体的，假设我们修改的是第 i 行的元素，假设 k 是最大的正整数，使得 $C_{k,i}$ 为 1。显而易见 k 存在。

标准做法

对于一次单点修改，我们尝试尽量少的修改 B 对应的元素。

具体的，假设我们修改的是第 i 行的元素，假设 k 是最大的正整数，使得 $C_{k,i}$ 为 1。显而易见 k 存在。

则 $\forall j < k$，若 $C_{j,i}$ 为 1，则将 j 行异或上 k 行的值。结束后删去第 i 行。这里可以证明经过这个变换得到的新矩阵 B' 仍然为最简矩阵。

标准做法

现在我们只需要检查向量 a 能否由这个集合内元素线性表出。

由于 B' 为最简矩阵，因而我们我们只需要检查 a 为零向量, 或者 B' 中存在于 A 全等的行向量即可。除去这两种情况全部不能线性表出。

标准做法

现在我们只需要检查向量 a 能否由这个集合内元素线性表出。

由于 B' 为最简矩阵，因而我们我们只需要检查 a 为零向量, 或者 B' 中存在于 A 全等的行向量即可。除去这两种情况全部不能线性表出。

证明可以分成 B' 是否存在最高位与 a 相等的向量。如果不存在 a 本身无法被表出，否则设 B' 中与 a 最高位相等的向量 c，则 a^c 无法被线性表出。

标准做法

现在我们只需要检查向量 a 能否由这个集合内元素线性表出。

由于 B' 为最简矩阵，因而我们我们只需要检查 a 为零向量, 或者 B' 中存在于 A 全等的行向量即可。除去这两种情况全部不能线性表出。

证明可以分成 B' 是否存在最高位与 a 相等的向量。如果不存在 a 本身无法被表出，否则设 B' 中与 a 最高位相等的向量 c，则 a^c 无法被线性表出。

全程可以采用 bitset 优化，时间复杂度 $O(\frac{n^3}{w})$。

H. Four-Coloring

给定一张包含 n 个点 m 条边的平面图。

这张图非常特殊，第 i 个点在平面直角坐标系下的坐标为 (x_i, y_i)。

如果节点 i, j 之间有连边，则至少满足如下三个条件中的一个：

- $x_i = x_j$
- $y_i = y_j$
- $|x_i - x_j| = |y_i - y_j|$

你需要给每个节点染上四种颜色中的任意一种，使得对于任意一条边，其连接两个端点异色，由于四色定理，其显然有解，输出任一合法方案即可。

$n \leq 10000$。

提示

考虑解决五色染色。

标准做法

考虑 x 坐标最小的节点。如有多个只考虑 y 坐标最小的。

显而易见，这一个节点的度数显然不超过 4。因而在删去这个节点后，不论剩余节点如何染色，总存在当前节点的合法一种颜色。

直接递归解决这个子问题，时间复杂度 $O(n)$。

标准做法

考虑 x 坐标最小的节点。如有多个只考虑 y 坐标最小的。

显而易见，这一个节点的度数显然不超过 4。因而在删去这个节点后，不论剩余节点如何染色，总存在当前节点的合法一种颜色。

直接递归解决这个子问题，时间复杂度 $O(n)$。

但是如果只有四种颜色，则删去这个节点后，可能会出现四个相邻节点颜色互不相同的情况。

标准做法

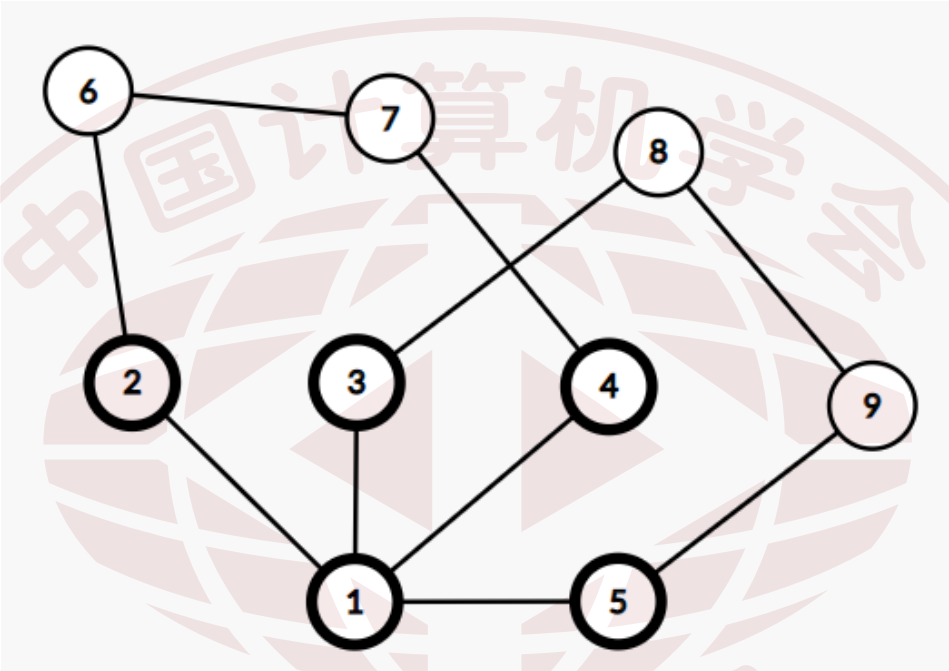

Figure: 一个简陋的示意图

假设节点 $2,3,4,5$ 的颜色分别为 c_2, c_3, c_4, c_5。

标准做法

考虑从节点 2 出发的颜色为 $c_2, c_4, \cdots$ 的增广路，以及 3 出发的颜色为 $c_3, c_5, \cdots$ 的增广路。

标准做法

考虑从节点 2 出发的颜色为 $c_2, c_4, \cdots$ 的增广路，以及 3 出发的颜色为 $c_3, c_5, \cdots$ 的增广路。

如果存在一条从 2 出发的增广路，其末尾节点不为 4，则我们沿着增广路将节点反色为 c_4, c_2，并且将当前节点染色为 c_2 即可。

对于节点 3，我们也有类似的结论。

标准做法

考虑从节点 2 出发的颜色为 $c_2, c_4, \cdots$ 的增广路，以及 3 出发的颜色为 $c_3, c_5, \cdots$ 的增广路。

如果存在一条从 2 出发的增广路，其末尾节点不为 4，则我们沿着增广路将节点反色为 c_4, c_2，并且将当前节点染色为 c_2 即可。

对于节点 3，我们也有类似的结论。

同时，如果以 2 出发增广路末尾节点为 4，且以 3 出发增广路末尾节点为 5，则由平面图性质知，两条增广路一定在某一结点相交。而两条增广路颜色集合交集为空，矛盾。

单次插入节点复杂度为 $O(n)$，总复杂度 $O(n^2)$。

五色定理

由于 n 个节点平面图的边数严格 $\leq 3n-5$。[1] 因而恒存在一个节点，其度数不超过 5。

类似的我们考虑将这个节点删除后的子问题，假设我们已经找到了一组合法解。

[1]:https://en.wikipedia.org/wiki/Planar_graph

五色定理

由于 n 个节点平面图的边数严格 $\leq 3n-5$。[1] 因而恒存在一个节点，其度数不超过 5。

类似的我们考虑将这个节点删除后的子问题，假设我们已经找到了一组合法解。

如果存在任意合法颜色，直接填入即可。

否则有这个节点相邻的 5 个点颜色互不相同。可以证明存在两种互不相同颜色 c_1, c_2, 使得从当前节点出发，颜色为 $c_1, c_2, \cdots$ 的增广路，使得末尾节点与当前节点不相邻，此时直接增广即可。

证明过程与上面类似，利用平面图的性质导出两条包含不同颜色的增广路相交导出矛盾即可。

[1]:https://en.wikipedia.org/wiki/Planar_graph

B. Bulbasaur

有一张 $n \times k$ 个节点的图，节点总共分为 n 层，每层 k 个。为了方便描述，之后我们称第 i 层的第 j 个点为 (i, j)。

这张图非常特殊，$\forall i \in [1, n-1]$，有且仅有连接从第 i 层出发，终止于第 $i+1$ 层的单向的流量为 1 的边。

假设 $f(i, j)$ 为将 $(i, 1), \cdots, (i, k)$ 设为网络流中的源点,$(j, 1), \cdots, (j, k)$ 设为网络流中的汇点，且每个节点入流量和出流量均不超过 1 的情况下的最大流。

询问 $\sum_{i=1}^{n} \sum_{j=i+1}^{n} f(i, j)$。

$n \leq 40000, k \leq 9$。

老师，这题我很眼熟!

这题去年 wxh010910 的确讲过!

老师，这题我很眼熟!

这题去年 wxh010910 的确讲过!

他去年讲的是一个非多项式做法。

这里讲一个多项式算法。

如果有好兄弟讲一下基于 Hall 定理的非多项式算法如果时间有多的话讲完多项式算法可以来讲一下。

多项式算法

不妨考虑先求出 $\sum f(1,i)$ 的值。

多项式算法

不妨考虑先求出 $\sum f(1,i)$ 的值。

由于每条汇点到源点的路径长度均相同，不妨考虑如下算法:

- 按照正常的网络流记录反向边。
- 每一次增广的时候增广到层数最大的点，也就是 (i,j) 中 i 最大的点
- 按照找到的路径进行暴力增广

多项式算法

不妨考虑先求出 $\sum f(1,i)$ 的值。

由于每条汇点到源点的路径长度均相同，不妨考虑如下算法:

- 按照正常的网络流记录反向边。
- 每一次增广的时候增广到层数最大的点，也就是 (i,j) 中 i 最大的点
- 按照找到的路径进行暴力增广

此时答案即为有效的增广路径长度。

不妨假设 $f(1,x_i)=g_i$, 则上述算法给出的合法增广路总时恰好有 g_i 条经过了形如 (x_i,j) 的节点。这是有网络流的性质所保证的。

这样子我们就可以在 $O(nk^3)$ 的时间复杂度内计算出单起点答案。

多项式算法

不妨考虑将 $\sum f(1,i)$ 的增广路继续沿用到 $\sum(2,i)$ 的计算中。

多项式算法

不妨考虑将 $\sum f(1,i)$ 的增广路继续沿用到 $\sum(2,i)$ 的计算中。

我们在求出 $\sum f(1,i)$ 后，将 $(1,1),\cdots,(1,k)$ 直接从原图中删除。根据我们建图的性质其不会影响答案。

同时对于同一层的节点，根据网络流的性质，由于到汇点增广路长度均相同，因而其增广顺序不会影响最终的增广路长度和。

因而直接删除 $(1,1),\cdots,(1,k)$ 之后，剩余的残量网络可以被认为是对 $(2,1),\cdots,(2,k)$ 的节点按照上述算法增广到一半的结果。

多项式算法

不妨考虑将 $\sum f(1,i)$ 的增广路继续沿用到 $\sum(2,i)$ 的计算中。

我们在求出 $\sum f(1,i)$ 后，将 $(1,1),\cdots,(1,k)$ 直接从原图中删除。根据我们建图的性质其不会影响答案。

同时对于同一层的节点，根据网络流的性质，由于到汇点增广路长度均相同，因而其增广顺序不会影响最终的增广路长度和。

因而直接删除 $(1,1),\cdots,(1,k)$ 之后，剩余的残量网络可以被认为是对 $(2,1),\cdots,(2,k)$ 的节点按照上述算法增广到一半的结果。

此时我们直接继续对 $(2,1),\cdots,(2,k)$ 按照上述算法暴力增广即可。

这个算法复杂度是什么？

复杂度分析

如果我们找到了一条从 (p, q) 到 (r, s) 的极长增广路，则所有位于 $p \sim r$ 层的节点至多在这次 bfs 中入队一次，因而这样一次增广的复杂度为 $O((r-p+1)k^2)$。

复杂度分析

如果我们找到了一条从 (p,q) 到 (r,s) 的极长增广路，则所有位于 $p\sim r$ 层的节点至多在这次 bfs 中入队一次，因而这样一次增广的复杂度为 $O((r-p+1)k^2)$。

在该过程中我们直接继承了上一层的状态。由于我们强制每一个节点出入度均不超过 1，而每一次 (p,q) 到 (r,s) 的极长增广路会增加 $r-p+1$ 个被使用的节点，因此每一次增广 $r-p+1$ 的和不超过 nk。

据此我们可以证明上面算法的时间复杂度为 $O(nk^3)$。

撒花

祝各位选手在冬令营中取得理想的成绩！