杂题选讲

徐骁扬

GDCPC 2024 图

简要题意

给定一个 n 个点，m 条边的无向图，记 $k = \left\lceil \frac{m}{n-1} \right\rceil$，问能否在图中找到两个不同的点 u, v，使得它们之间存在 k 条边不相交的路径。

如果可以，找到这样的一对 u, v 并给出构造。

$n \le 10^5$，$m \le 2 \times 10^5$。

GDCPC 2024 图

首先考虑 $k = \left\lceil \dfrac{m}{n-1} \right\rceil$ 的含义，将 m 条边至多 $n-1$ 个分成一组，则边将会分成至少 k 组。

而 $n-1$ 条边分一组，容易让人联想到生成树，而对于一个生成树，其上的每一对点 u, v 之间有且仅有一条路径。

GDCPC 2024 图

同时维护 k 张图，依次尝试加入每一条边 (a_i, b_i)：
从第一张图中开始尝试，如果当前这张图中 a_i, b_i 已经联通了，那么就枚举下一张图；否则将这条边加入当前的图中。如果 k 张图中均联通了 a_i, b_i，说明已经能够得到一组答案，当前这条边可以不加入了。
使用并查集维护连通性，时间复杂度为 $O(mk\alpha(n))$ 的，在 m 很大，n 很小的时候并不能接受这个复杂度。

GDCPC 2024 图

根据上面的构造方法，如果在第 j 张图中 a_i, b_i 已经联通了，那么就说明第 $1 \sim j$ 张图中 a_i, b_i 都已经被联通了。所以可以通过二分来确定最早的 a_i, b_i 没有联通的图。

时间复杂度 $O(m \log k\alpha(n))$。

AGC016E Poor Turkeys

简要题意

有 n 只火鸡和 m 个人，每一个人依次操作，其中第 i 个人的操作时会指定两只火鸡 x_i, y_i：

- 如果 x_i 和 y_i 都还活着，则第 i 个人会随机挑一只吃掉。
- 如果 x_i 和 y_i 中只有一只还活着，则第 i 个人会吃掉活着的那只。
- 如果 x_i 和 y_i 都被吃掉了，则不进行任何操作。

问有多少对二元组 (i, j)，满足，最终**有可能**第 i 只火鸡和第 j 只火鸡都活着。

$n \leq 400$，$m \leq 10^5$。

AGC016E Poor Turkeys

先考虑如何对于一对于一只火鸡 i，需要如何才能被保留。

如果正序考虑操作，在两只火鸡都没有被选择的情况下，并不确定要选择哪一只火鸡。因此可以尝试逆序处理所有的操作。

从后往前看每一个人，如果遇到了一个人选择的两只火鸡为 x 和 i，由于希望将 i 保留，那么这个人必然是选择吃掉火鸡 x。因此，在这个人以前，x 和 i 都不能被吃掉，只有保留了第 x 只火鸡，才能通过吃掉第 x 只火鸡来保留第 i 只火鸡。

如果遇到一个人选的两只火鸡 x 和 y 都需要保留，就意味着 i 一定会被吃掉。

若 i 不是一定会被吃掉的，可以得到一个集合 S_i，表示有哪些火鸡需要保留到某一个特定时刻再被吃掉（或者就是第 i 只火鸡）。

AGC016E Poor Turkeys

现在考虑两只火鸡 i 和 j 能否同时保留。如果不行，说明在保留其他火鸡的问题上出现了冲突。

对于一只火鸡 x，如果同时有 $x \in S_i$ 和 $x \in S_j$，讨论其中可能的情况：

1. 如果 $x = i$ 或 j，说明会出现要保留 i 就必须吃掉 j 的情况，则 i, j 必然无法同时保留。
2. 如果 x 在 i 的构造中吃它的人和在 j 中不同，那么 x 只能被其中编号更小的人吃掉，编号更大的那一侧无法使得 i/j 不被吃掉。
3. 如果 x 在 i 的构造中吃它的人和在 j 中的相同，为 (x, y)，那么意味着 x 被吃掉时为了保留 y 不被吃，显然有 $y \in S_i$ 且 $y \in S_j$。对于 y 递归分类讨论的过程直至变成上面两种情况。

AGC016E Poor Turkeys

根据上页的讨论，只要存在 x 有 $x \in S_i \land x \in S_j$，那么就意味着 i, j 无法同时保留。因此 i, j 可以同时保留的必要条件是 $S_i \cap S_j = \emptyset$。
同时，如果有 $S_i \cap S_j = \emptyset$，意味着在 i 和 j 前面的逆序扫描中不存在冲突，可以通过求解 S_i 过程中的决策来得到一组构造。所以 i, j 可以同时保留和 $S_i \cap S_j = \emptyset$ 是互为充要的。
枚举每一对 i, j，检查是否满足 $S_i \cap S_j = \emptyset$。总时间复杂度 $O(nm + n^3)$。

CEOI 2023 Balance

简要题意

有 N 台评测机和 T 道题目，在每台评测机上都有 $S = 2^k$ 个提交还未评测，每个提交对应一个题目，一个题目可能有多个提交。在接下来的 S 单位时间内，每 1 单位时间每一个评测机都评测一个提交。

为了评测的稳定，对于每一道题目，任意两个单位时间内，被评测的该题的提交数量之差不能超过 1。构造一组满足条件的评测方式。

$1 \le N, S, T \le 10^5$，$NS \le 5 \times 10^5$。

CEOI 2023 Balance

首先考虑 $S = 2$ 如何解决。
每一个评测机都只会评测两个提交 a_i, b_i，唯一能够做的操作就是交换着两次提交的顺序。
评测题目数量之差不能超过 1 这个限制并不是好处理，尝试将这个限制转化一下。假如已经有了一组构造，对于在所有评测机中总提交次数为奇数的题目 x，添加一台评测机评测 x 和 0，其中 0 是添加的一个 “题目”。并且令 x 的那个提交在两个时刻中原本 x 的评测数量较小的那个时刻评测。
此时 $0 \sim T$ 均满足两个时刻的评测的该题的提交数量**相同**，这样容易处理。

CEOI 2023 Balance

发现二元组 (a_i, b_i) 和无向边的形式类似，而确定这两个点的顺序就是对边进行定向：如果第 1 个时刻评测 a_i，则将这条边定向为 $a_i \to b_i$；否则将这条边定向为 $b_i \to a_i$。
而每道题两个时刻的的评测数量相同，就意味着图上这个点的入度和出度相同。这等价于构造出了原图的一条欧拉回路（如果图不联通则是对于每一个连通块的欧拉回路）。由于每一个点的度数都为偶数，所以欧拉回路一定存在，可以直接 dfs 进行构造。

欧拉回路求解

从任意点开始进行 dfs，可以重复遍历同一个点，但是每一条边只能遍历依次。在从一条边回溯的时候，将这条边及其方向记录下来。则按照这个顺序记录得到的就是一条欧拉回路。

时间复杂度为 $O(|V|+|E|)$ 其中 V 和 E 分别是无向图的点集和边集。在这道题目中复杂度为 $O(NS)$。

CEOI 2023 Balance

已经解决了 $S = 2$ 的情况，尝试能否将这个做法套用到 $S = 2^k$ 上。
将前 2^{k-1} 个时刻和后 2^{k-1} 分成两组，可以通过 $S = 2$ 的算法，让每一个题目在前 2^{k-1} 个时刻评测的次数和后 2^{k-1} 个时刻评测的次数之差 ≤ 1。
在有了这个条件的基础之上，如果对于这两个 $S' = 2^{k-1}$ 的规模的子问题，都能够在组内使得任意两个时刻评测次数差 ≤ 1，那么就能够证明，此时组间任意两个时刻评测次数差也 ≤ 1。
因此，可以直接分治处理，计算可知时间复杂度为 $O(NS \log S)$。

CF1956E Nene vs. Monsters

简要题意

有一个长度为 n 的序列 $\{a_n\}$，进行如下操作 10^{100} 次:
依次令 $i = 1, 2 \dots n$，将 $a_{i \bmod n+1}$ 变成 $\max(0, a_{i \bmod n+1} - a_i)$。
问最终有哪些 i 使得 $a_i \neq 0$。
$1 \leq n \leq 2 \times 10^5$，$0 \leq a_i \leq 10^9$。

CF1956E Nene vs. Monsters

由于 10^{100} 远大于值域 V 的大小，所以最终不会存在相邻的两个 a_i 都不等于 0。
而出现了 $a_i = 0$，则 $< i$ 和 $> i$ 的两个部分就互不影响了，在相邻两个 0 之间的距离比较短，那么可以直接 $O(1)$ 判断出会剩余哪些数。
如果能够通过一定的操作之后，使得极长的非零 a 连续段比较短，就可以解决问题了。

CF1956E Nene vs. Monsters

假设 $a_{i-1}=0$，$a_i=1$，计算 $a_j(j>i)$ 在 t 轮操作之后仍然非零至少需要是多少：

$$a_{i+1}\geq t+1,\ a_{i+2}\geq \sum_{i=1}^{t} i+1=\frac{t(t+1)}{2}+1,$$

$$a_{i+3}\geq \sum_{i=1}^{t}(\frac{i(i-1)}{2}+1)+1=O(t^3)。$$

由此可知，如果 t 轮之后，仍然存在长度 ≥ 4 的极长非零段，则第四个数的初始值是 $O(t^3)$ 量级的。这也就意味着，在按照题意进行 $O(\sqrt[3]{V})$ 轮操作之后，就不会存在长度 >3 的连续非零段了。

CF1956E Nene vs. Monsters

假设对于一个长度为 3 的非零段，三个数依次为 a_i, a_{i+1}, a_{i+2}。由于 a_i 不可能非零了，所以必然有 $a_{i+1} = 0$。令 $t = \left\lfloor \frac{a_{i+1}}{a_i} \right\rfloor$，则在 a_{i+1} 变为 0 之前，它会让 a_{i+2} 减少 $ta_{i+1} - \frac{t(t+1)}{2}a_i$。判断这个数和 a_{i+2} 的大小即可确定 a_{i+2} 是否会变为 0。

总时间复杂度 $O(n\sqrt[3]{V})$。

PA 2024 Desant 3

简要题意

有一个长度为 n 的 01 序列 $\{x_i\}$，有 m 次操作，每次操作会选择两个数 a_i，b_i。如果 $x_{a_i}=1$ 且 $x_{b_i}=0$，则交换 a_i 和 b_i。

对于 $k=1\sim n$，问对于所有包含 k 个 1 的 $\binom{n}{k}$ 种可能的初始序列，有多少种在 m 次操作之后 k 个 1 构成了一个连续的区间。答案对 2 取模。

$2\le n\le 35$，$m\le 1000$。

PA 2024 Desant 3

发现这题有答案对 2 取模，这会在统计的过程中带来一定的便利，如果能够确定两个部分对于答案的贡献相同（方案一一对应），那么就可以直接忽略掉这一部分的贡献。

根据操作的定义：如果有 $x_{a_i}=1$，$x_{b_i}=0$，则在交换之后，有 $x_{a_i}=0$，$x_{b_i}=1$；同时如果原本就有 $x_{a_i}=0$ 且 $x_{b_i}=1$，此时不会交换。这两种情况在操作之后等价，对答案的贡献相同。

因此，如果出现了 x_{a_i} 和 x_{b_i} 交换的操作，且 $(x_{a_i},x_{b_i})=(1,0)$ 和 $(0,1)$ 的两种情况均没有被忽略，那么就可以直接将这两种情况忽略掉。

PA 2024 Desant 3

从前往后依次枚举每个操作，并记录每一个 x_i 当前的状态：

- 如果 x_{a_i} 和 x_{b_i} 没有被确定值，则分成 $x_{a_i}=x_{b_i}=0$ 和 $x_{a_i}=x_{b_i}=1$ 分别处理。
- 如果 x_{a_i} 和 x_{b_i} 均被确定值，按照题意模拟修改。
- 如果 x_{a_i} 和 x_{b_i} 有一个被确定值且为 $x_{a_i}=1$ 或 $x_{b_i}=0$，发现两种情况均等价于交换 x_{a_i} 和 x_{b_i}。
- 如果 x_{a_i} 和 x_{b_i} 有一个被确定值且为 $x_{a_i}=0$ 或 $x_{b_i}=1$，必然不会交换。

在扫描完所有的操作之后，会得到每一个 x 的状态，可以通过枚举 k 计算出这种局面对于答案的贡献。

PA 2024 Desant 3

对于上页的第 1 种情况，会产生两个分支，同时确定两个 x_i 的值；其余的情况都不会产生分支。因此最多只有 $\left\lfloor \frac{n}{2} \right\rfloor$ 次产生分支的情况。

因此，直接使用搜索，最多只会有 $O(2^{\frac{n}{2}})$ 个可能的分支，直接使用搜索实现的时间复杂度是 $O(2^{\frac{n}{2}}(n+m))$。

JOISC 2022 京都观光

简要题意

有一个 H 行 W 列的网格，从左上角到右下角依次编号为 $(1,1)$ 到 (H,W)。有一个人希望从左上角走到右下角，其中：

- 从 (i,c) 走到 $(i,c+1)$ 的代价为 A_i。
- 从 (c,i) 走到 $(c+1,i)$ 的代价为 B_i。

问从 $(1,1)$ 走到 (H,W) 的最小代价是多少。

$1 \le H, W \le 10^5$，$1 \le A_i, B_i \le 10^9$。

JOISC 2022 京都观光

行走的路径是一条折线，因此，问题的重点就在于什么情况下会选择转弯。例如考虑 l, r 两行，以及 x, y, z 三列，什么时候会选择路径 $(l, x) \to (l, y) \to (r, y) \to (r, z)$，而不是 $(l, x) \to (r, x) \to (r, z)$ 或 $(l, x) \to (l, z) \to (r, z)$。

分别计算三种情况的代价：

- $(l, x) \to (l, y) \to (r, y) \to (r, z)$：$A_l(y - x) + A_r(z - y) + B_y(r - l)$。
- $(l, x) \to (r, x) \to (r, z)$：$B_x(r - l) + A_r(z - x)$。
- $(l, x) \to (l, z) \to (r, z)$：$A_l(z - x) + B_z(r - l)$。

JOISC 2022 京都观光

求解不等式组 $\begin{cases} A_l(y-x)+A_r(z-y)+B_y(r-l)\le B_x(r-l)+A_r(z-x) \\ A_l(y-x)+A_r(z-y)+B_y(r-l)\le A_l(z-x)+B_z(r-l) \end{cases}$

有:

$$\frac{B_y-B_x}{y-x}\le\frac{A_r-A_l}{r-l}\le\frac{B_z-B_y}{z-y}$$

发现如果是从 y 列转弯，就需要上述不等式成立，而上述不等式可能成立的前提条件为 $\frac{B_y-B_x}{y-x}\le\frac{B_z-B_y}{z-y}$。这一个对于 B 单独的斜率型的限制，说明了，只有所有 (i,B_i) 构成的下凸包上的点对应的列上，才有可能纵向移动。A 序列同理。

JOISC 2022 京都观光

使用单调栈对 A 和 B 建立凸包，而根据上页的不等式可知，实际移动的路线会选择斜率更小一侧移动，而在 A 和 B 的凸包上，斜率是分别单调递增的。对于 A 和 B 同时维护两个指针，使用类似归并的方式处理，就可以得到移动的路径，进而计算出答案。时间复杂度 $O(H+W)$。

AGC057D Sum Avoidance

简要题意

有一个正整数 S，称正整数列 $A=\{a_1,a_2,\dots a_N\}$ 是好的，当且仅当：

1. 对于所有 $i=1,2\dots N$，$a_i\in[1,S)$。
2. 不存在非负整数列 $\{x_1,x_2\dots x_N\}$，使得 $\sum\limits_{i=1}^{N}x_ia_i=S$。

问在所有满足条件的数列 A 中，N 最大其次字典序最小的数列 A 中的第 K 小的元素是多少。

$S,K\le 10^{18}$。

AGC057D Sum Avoidance

首先考虑求解 N 的值。

如果 $a \in A$，则必然有 $(S-a) \notin A$。因此将 $[1, S)$ 中和为 S 的数两两匹配，每一对数中只能够选择一个。因此有 $N \leq \left\lfloor \dfrac{S-1}{2} \right\rfloor$。

同时这个上界也是可以取到的，具体的，取每一对数中较大的那个，也就是所有 $> \dfrac{S}{2}$ 的数。此时，任意两个数的和都 $> S$，满足第二条限制。

AGC057D Sum Avoidance

再考虑如何最小化字典序。

从小到大枚举 t，如果将 t 加入 A 中仍然合法，就直接加入。可以证明这样的方法仍然能够满足 N 最大：

如果 t 不能加入 A，说明对于当前已经加入 $A=\{a_1,a_2,\dots a_{N'}\}$，不存在非负整数列 $\{x_1,x_2\dots x_{N'}\}$ 使得 $\sum\limits_{i=1}^{N'} x_ia_i=t$。

证明考虑反证法，如果存在一组 $\{x\}$，使得 $\sum\limits_{i=1}^{N'} x_ia_i=t$，由于 t 不能加入，说明存在非负整数列 $\{x'_1,x'_2\dots x'_N,y\}(y\neq 0)$，使得 $\sum\limits_{i=1}^{N'} x'_ia_i+yt=S$，将前式带入可以得到 $\sum\limits_{i=1}^{N'}(x_i+yx'_i)a_i=S$。这与当前的 A 合法矛盾。

AGC057D Sum Avoidance

从此以后加入的元素都 $> t$，因此无论如何 A 中元素无法组成 t，说明将 $S - t$ 加入 A 中将是合法的。
这也就是说明，这样贪心处理所有 $t \leq \frac{S}{2}$ 的正整数，即可直接唯一确定 $> \frac{S}{2}$ 部分的选择，使得 A 数列的大小达到理论最大值。
但是由于 S 太大了，需要尝试加速上面贪心的过程。

AGC057D Sum Avoidance

对于 A 中的第一个元素，其应当是最小的满足 $t \nmid S$ 的正整数 d。而由于 $\text{lcm}(1, 2, \dots 50) > 10^{18}$，所以有 $d < 50$。

仿照上页的证明方式，我们可以证明，对于任意 $a, b \in A$，若 $a + b < S$，则有 $a + b \in S$。特别地，取 $b = d$，能够说明只需要对于每一个 $r = 0, 1 \dots d - 1$，确定最小的正整数 $a \bmod d = r$，使得有 $a \in S$。

AGC057D Sum Avoidance

设 f_i 表示当前已经确定了最小的满足 $x_0 \bmod d = i$ 且能够加入 A 中的 x_0 是多少。

那么初始时就有 $f_i = \begin{cases} d & , i = 0 \\ +\infty & , i \neq 0 \end{cases}$。

考虑当前能够新加入的最小的 v 是多少：假设一组方案中选择了 i 个 v $(i = 1, 2 \dots d-1)$，且希望 v 能够合法，就必然有 $f_{(S-iv) \bmod d} + iv > S$。

移项后得到：$iv > S - f_{(S-iv) \bmod d}$，即 $v \geq \left\lfloor \frac{S - f_{(S-iv) \bmod d}}{i} \right\rfloor + 1$。那么也就有 $v \geq \max_{i=1}^{d-1}(\left\lfloor \frac{S - f_{(S-iv) \bmod d}}{i} \right\rfloor + 1)$。

后半部分取 max 的式子只与 $v \bmod d$ 的值有关，因此对于还没有确定最小值的 f_r，计算出在 $v \bmod d = r$ 的时候 v 的最小值 v_r。在所有的 v_r 中选择最小的那一个更新。

AGC057D Sum Avoidance

更新时，首先有 $f_r \leftarrow v_r$。然后需要进行一个类似于同余最短路的操作，对于所有 i,j 进行 $f_{(i+rj) \bmod d} \leftarrow \min(f_i + v_r j)$ 的操作。
由于每一个 $f_r \leftarrow v_r$ 只会进行一次，所以这一部分的复杂度为 $O(d^3)$。
对于找到第 K 小的，可以通过二分答案来确定。这一部分的时间复杂度为 $O(d \log n)$。

OOI 2025 The arithmetic exercise

简要题意

有 n 个初始为 0 的数 a_i，以及一个长度为 m 的序列 x_i。对于 $i = 1, 2 \dots m$，需要依次选择一个下标 j_i，令 $a_{j_i} \leftarrow x_i - a_{j_i}$。问操作完之后 $\sum_{i=1}^{n} a_i$ 最大能够是多少。

$1 \le n, m \le 3 \times 10^5$，$|x_i| \le 10^9$。

OOI 2025 The arithmetic exercise

考虑所有选择 $j_i = k$ 的 x_i 取出，则有 $x'_1, x'_2, \dots x'_t$。若 t 为奇数，则有 $a_k = x'_1 - x'_2 + \dots - x'_{t-1} + x_t$；若 t 为偶数，则有 $a_k = -x'_1 + x'_2 - \dots - x'_{t-1} + x_t$。
发现倒序考虑整个 x' 序列，则有 $a_k = x'_t - x'_{t-1} + x'_{t-2} - \dots$，这个结构是一正一负，并且和 t 的奇偶性，也就是选择的次数无关。
而最终需要最大化求和，那么就是要给每一个 x 分配正负号，最大化 x 的和。

OOI 2025 The arithmetic exercise

将整个 x 序列翻转，设计 DP 状态 $f_{i,j}$，表示处理了前 i 位，当前有 j 的 a 的下一个 x 取负号的情况下，给 $x_1 \sim x_i$ 分配符号后的和最大为多少。

有转移 $f_{i,j} = \max(f_{i-1,j-1} + x_i, f_{i-1,j+1} - x_i)$，修改一下式子之后，有 $f_{i,j} = \max(f_{i-1,j-1} + 2x_i, f_{i-1,j+1}) - x_i$。初值为 $f_{0,j} = \begin{cases} 0 & ,j = 0 \\ -\infty & ,j \neq 0 \end{cases}$

可以将每一次转移的 $-x_i$ 全部后移，最后再将答案减 $\sum\limits_{i=1}^{m} x_i$。因此就会有新的转移 $f_{i,j} = \max(f_{i-1,j-1} + 2x_i, f_{i-1,j+1})$。

OOI 2025 The arithmetic exercise

对于上面的 DP，在 i,j 奇偶性不同的时候，DP 值必然为 $-\infty$，所以可以对于状态进行一次压缩：令 $f'_{i,j} = \begin{cases} f_{i,2j+1} & ,i \text{ is odd} \\ f_{i,2j} & ,i \text{ is even} \end{cases}$。

在 i 为奇数的时候，有转移 $f'_{i,j} = \max(f'_{i-1,j} + 2x_i, f'_{i-1,j+1})$；在 i 为偶数的时候，有转移 $f'_{i,j} = \max(f'_{i-1,j-1} + 2x_i, f'_{i-1,j})$。

这个转移方程结构很简单，是经典的 slope trick 的形式。

OOI 2025 The arithmetic exercise

具体的，可以证明 $f'_{i,j}$ 是关于 j 上凸的，即 $\Delta_{i,j} = f'_{i,j} - f'_{i,j-1}$ 是单调递减的。
在 i 为偶数的转移中，有
$f'_{i,j} = \max(f'_{i-1,j-1} + 2x_i, f'_{i-1,j}) = f'_{i-1,j-1} + \max(2x_i, \Delta_{i,j})$。存在某个阈值 k，使得在 $j \le k$ 的时候，会使用 $\Delta_{i,j}$ 转移；在 $j > k$ 的时候，会使用 $2x_i$ 转移。
f'_i 的差分数组相对于 f'_{i-1} 的，相当于是将 $2x_i$ 插入到了 Δ_i 序列中第一个比它小的 $\Delta_{i,j}$ 之前。
发现差分数组比原数组更有性质，不妨尝试直接维护差分数组和 $f'_{i,0}$。而因为差分数组是单调的，所以可以直接维护当前差分数组构成的可重集合 S，例如使用 STL 中的 multiset。

OOI 2025 The arithmetic exercise

对于 i 是奇数的转移，就是在和偶数进行相同的转移之后，将整个序列向 $j \to j-1$ 进行了偏移，对应到差分数组上的操作就是将最大的差分 Δ_{max} 从 S 中弹出，加到 $f'_{i,0}$ 上。
在转移的过程中，如果由于数组的长度有限制，也就是对需要的差分 S 的数量有限制。在 S 中差分的数量超过了限制之后，可以直接将最小的差分 Δ_{min} 从 S 中弹出即可。
最终时间复杂度为 $O((m+n)\log n)$。

COTS 2019 Izazov

简要题意

有一个 $N \times M$ 的黑白矩阵，构造尽可能少的矩形覆盖这个矩阵，使得：每一个黑色格子**恰好**被一个矩形覆盖，没有白色格子被矩形覆盖。

$N, M \leq 500$。

COTS 2019 Izazov

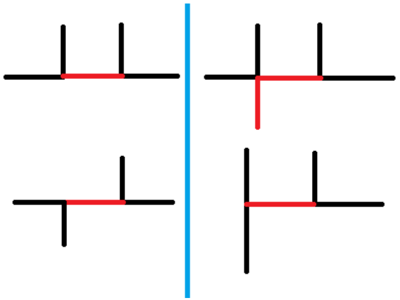
将黑色的部分看作一个大的图形，那么就是要将图形分割成尽可能少的部分，使得每一个部分都是一个矩形。
现在的主要问题是如何描述当前的图形是否已经被分割成了若干个矩形。
在所有边界均与横轴或纵轴平行的图形之中，只有矩形是仅包含 90° 内角而不包含 270° 内角的。因此，如果能够将初始图形通过若干次切割，使得其中仅包含 90° 内角，也就说明将图形分割成了若干个矩形。
同时，假设内角数量为 d，可以直接确定矩形的数量为 $\frac{d}{4}$，这也方便我们最小化矩形数量。

COTS 2019 Izazov

而对于 270° 内角的分割形式也很简单：将其分成一个 90° 内角和一个 180° 内角（等价于普通边）。

具体的分割也会有两种情况：

1. 通过一次分割将两个 270° 内角一起分割（对应右图左侧）。
2. 通过一次分割只将一个 270° 内角分割（对应右图右侧）。



COTS 2019 Izazov

上面的两种分割方式中：

1. 第一种能够减少 2 个 270° 内角，产生 2 个 90° 内角；
2. 第二种能够减少 1 个 270° 内角，产生 3 个 90° 内角。

假设原图有 N_1 个 270° 内角和 N_2 个 90° 内角，有 t 个第一种分割，那么就还有 $(N_1 - 2t)$ 个 270° 角使用的是第二种分割。

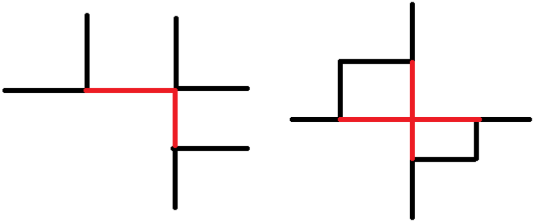
那么最终的 90° 内角数量就等于 $N_2 + 2 \times t + 3 \times (N_1 - 2t) = N_2 + 3N_1 - 4t$，最小化内角数量就是最大化 t。

所以希望能够尽可能多的构造第一种分割。而第一种分割需要对 270° 的角进行匹配，尝试从匹配的角度思考这个问题。

COTS 2019 Izazov

每一个 270° 内角，向着其可能的两条分割线方向延申，如果能够到达另一个 270° 内角，那么就说明这是一组可行的匹配。

如果两条可能的匹配的分割线线相交了（包括端点），那么这同时选择这两条分割线，会有一个分割等价于两个第二种分割。因此，如右图这些有相交的分割线是不能同时选择的。



COTS 2019 Izazov

将每一对匹配看作点，将冲突关系看作边，选取尽可能多的匹配就是选择最大独立集。
但是由于横向的分割线只会和纵向的分割线相交，所以这张图是一个二分图。
二分图最大权独立集大小 $=$ 点数-最小点覆盖大小，而最小点覆盖大小 $=$ 最大匹配大小 $=$ 最小割大小，直接建图跑网络流即可。
由于可能的匹配数和冲突数都是 $O(NM)$ 的，因此使用 Dinic 的时间复杂度即为 $O((NM)^{1.5})$。

湖北省选模拟 2025 团队分组

简要题意

给定一棵有 n 个点的树，点有点权。定义一个树上独立集的权值为点集中所有点点权的最大值。对于每一个点 i，求包含该点的所有独立集的权值之和。

$n \le 3 \times 10^5$。

湖北省选模拟 2025 团队分组

不妨认为所有点的点权都是不同的。先只考虑以某一个点 i 的点权 v_i 作为最大点权的独立集。

那么独立集的点集就是所有 $v_j \le v_i$ 的点 j 构成的集合的子集。考虑 j 会对 i 做出的贡献 $t_{i,j}$，表示包含 j 且权值为 v_i 的独立集的数量。

这个 t 乍一看并不好处理。给每一个点设一个独立的元 x_j，考虑值 $\sum\limits_{j=1}^{n} t_{i,j}x_j$。

其有实际含义“权值为 v_i 的所有独立集的点的 x_j 之和”。

如果 x_j 是一个具体的数值，那么这个问题是一个比较简单的动态 DP 问题。

湖北省选模拟 2025 团队分组

先将问题差分成“权值 $\le v_i$ 的所有独立集的点的 x_j 之和”。

从小到大扫描 v_i，那么每一次就是将一个点从可选变成了不可选。设计 DP 状态：$f_{i,0/1}$ 表示在 i 的子树内，是否选择 i 点的情况下，有多少个独立集；$g_{i,j}$ 表示在 i 的子树内，是否选择 i 点的情况下，所有独立集的点的 x_j 之和。

如果当前点可选，则 $f_{i,1} = \prod_{j \in son_i} f_{j,0}$，

$g_{i,1} = f_{i,1} \times x_i + \sum_{j \in son_i} g_{j,0} \prod_{k \in son_i \setminus \{j\}} f_{k,0}$；否则 $f_{i,1} = g_{i,1} = 0$。

对于 $f_{i,0}$ 和 $g_{i,0}$ 的转移是类似的。

湖北省选模拟 2025 团队分组

对于将一个点从可选变成不可选的修改，可以通过使用数据结构（树链剖分 $+$ 线段树，全局平衡二叉树，静态 Top-tree 等）动态 DP 来维护。时间复杂度 $O(n\log^2 n)$ 或 $O(n\log n)$。

每一次取出的最终结果便是 $g_{1,0}+g_{1,1}$ 的值。

而 g 的值就是 x 的一组线性组合，在整个运算的过程中只会对其作相加和数乘，整个转移类似于对于一张有向无环图作从起点到终点的路径计数。其中 x_j 对应每一个起点，$(g_{i,0}+g_{i,1})$ 对应每一个终点，求解起点对于每一个终点的贡献。

湖北省选模拟 2025 团队分组

但是实际上题目要解决的问题是 $\sum\limits_{i=1}^{n} t_{i,j} v_i$。

这个值，也可以对应到上页构建的那个 DAG 之中：以 v_i 为终点，以 x_i 为起点，求解每一个终点对于起点的贡献。

在 DAG 上解决这个问题的方法是逆序执行拓扑排序的过程。而在这个问题中，就是就之前所有的数据结构维护动态 DP 的过程逆序执行，反向转移所有的贡献。

时间复杂度 $O(n \log n)$ 或 $O(n \log^2 n)$。

Triangular Lamps Hard

简要题意

在一个无限大的平面直角坐标系中，每一个整点上有一盏灯，初始只有点 (X, Y) 处的灯是亮的。

现在进行若干次操作，每一次操作为选择两个数 x, y，改变 $(x, y), (x+1, y), (x, y+1)$ 的亮灭状态。最终有 N 个点 (x_i, y_i) 的灯是亮着的。问初始 (X, Y) 的位置是多少。

$N \le 10^4$，$|x_i|, |y_i| \le 10^{17}$。

Triangular Lamps Hard

假设只有 $(0,0)$ 处的灯是亮的，那么就可以通过将对 $x+y=0$ 且 (x,y) 亮着的位置进行操作将所有亮着的灯移动到 $x+y=1$ 上。

不断重复上述操作使得所有亮着的灯移动到 $x+y=k$ 上。这个转移的过程形似杨辉三角，所以 (x,y) 的灯亮当且仅当 $\binom{x+y}{x} \bmod 2 = 1$。

同时，$(0,k)$ 和 $(k,0)$ 必然亮着。

Triangular Lamps Hard

对于最终有 N 个灯亮着的局面，将所有亮着的灯调整到 $x+y=V$（V 是一个比值域范围略大的数）上。
如果能够找到最上和最下两个点亮的灯 $(x_1, V-x_1)$，$(x_2, V-x_2)$，那么初始点亮的灯就应当是 $(x_2\ V-x_1)$。
现在的问题将会转化成求解 x_1 和 x_2。

Triangular Lamps Hard

回到初始 $(0,0)$ 时灯会亮的定义：$\binom{x+y}{x} \bmod 2 = 1$，有数学结论说明，$\binom{x+y}{x} \bmod 2 = 1$ 当且仅当 x 和 y 的二进制数码不存在某一位同时为 1。

要将 (x,y) 移动到 $(x+y,0)$，那么就可以以任意顺序依次增加 y 二进制拆分中的所有 2^k，此时增加过程中经过的每一个灯也都是亮着的。

假设知道某个 $(x_0, V-x_0)$ 的灯亮着，且 x_0 不是横坐标最大的亮着的灯，那么必然存在 2^k 使得 $(x_0+2^k, V-x_0-2^k)$ 也是亮着的。从小到大枚举 2^k，检验移动的灯是否亮着，如果亮着，就令 $x \leftarrow x+2^k$。

Triangular Lamps Hard

目前还需要解决的问题有两个：如何找到一个亮着的灯 $(x_0, V-x_0)$，如何检验单个灯是否是亮着的。

对于后者，枚举初始的 N 个灯，计算其是否会改变当前要检查的点的状态即可。

对于前者，记 t_i 表示 $(i, V-i)$ 被初始的 N 个点改变状态的次数，如果能够找到一个区间 $[l, r]$ 使得 $(\sum\limits_{i=l}^{r} t_i) \bmod 2 = 1$，也就意味着其中存在奇数个点亮的灯。

将这个区间分成两个部分，那么其中就一个部分点亮的灯的数量是奇数，上面的和式也等于 1。利用这个性质不断二分知道只剩余一个灯，那么这个灯必然是被点亮的。

Triangular Lamps Hard

但简单的 $(\sum_{i=l}^{r} t_i) \bmod 2$ 也并不容易快速找到一个结果为 1 的。

通过数学归纳法可知，在 $(\sum_{i=0}^{n}[i \bmod 3 = r]\binom{n}{i}) \bmod 2$ 中，对于 $r = 0, 1, 2$，至少有一个的结果为 1，同时这个式子也是可以 $O(\log V)$ 计算的。

找到某一个 r 之后，对于 $(\sum_{i=l}^{r}[i \bmod 3 = r]t_i) \bmod 2$ 这个结构进行二分，总时间复杂度为 $O(n \log^2 V)$。

JOISC 2022 蚂蚁与方糖

简要题意

在一个长度为 10^9 的木条上进行实验，一共有 Q 次操作，每一次操作为在某一个位置 x_i 放 a_i 只蚂蚁或 a_i 块方糖。

对于 $k = 1, 2 \dots Q$，回答假如在第 k 次操作后，让每一只蚂蚁在距离 L 以内存在方糖的情况下，选择一块并吃掉，最多会有多少块方糖被吃掉。

$Q \le 5 \times 10^5$，$L, x_i, a_i \le 10^9$。

JOISC 2022 蚂蚁与方糖

这是一个类似于匹配的问题，存在 $O(n)$ 直接贪心匹配的方法，但并不好优化。
考虑使用 Hall 定理求解：假设左部点集为 A，右部点集为 B，$N(S)$ 表示一个和左部点集合 S 有连边的所有右部点构成的集合 $N(S)$。那么最大匹配的大小即为 $|A| - \max_{S\subseteq A}(|S| - |N|(S))$。
在题目中，S 就是对应若干段区间 $[l, r]$ 内的蚂蚁，而其对应的方糖就是若干段区间 $[l - L, r + L]$ 内的方糖。
记 A_i 表示每一个位置蚂蚁的数量，B_i 为每一个位置方糖的数量，那么答案就等于 $\sum_{i=0}^{X} A_i - \max_S \left(\sum_{[l',r']\in S} \left(\sum_{i=l'}^{r'} A_i - \sum_{i=l'-L}^{r'+L} B_i \right) \right)$。

JOISC 2022 蚂蚁与方糖

对于某一组方案 S 中的两段区间 $[l_1, r_1]$ 和 $[l_2, r_2]$（$l_1 < l_2$），必然有 $r_1 + L < l_2 - L$，否则将这两段合并成 $[l_1, r_2]$，不会减少选中的蚂蚁的数量，不会增加选中的方糖的数量。

记 $p_i = -\sum\limits_{j=1}^{i} A_j + \sum\limits_{j=1}^{i-L} B_j$，$q_i = \sum\limits_{j=1}^{i} A_j - \sum\limits_{j=1}^{i+L} B_j$。

则上式就等于 $\sum\limits_{i=0}^{X} A_i - \max\limits_{S} \left(\sum\limits_{[l',r'] \in S} (q_{r'} - p_{l'-1}) \right)$。

也就是要最大化 $-p$ 和 q 交替出现求和。使用线段树维护 $f_{0/1,0/1}$ 表示当前区间中最左边或最右边是 $-p$ 还是 q 的情况下的最大值。

JOISC 2022 蚂蚁与方糖

考虑到对于一个 A 或者 B 的修改就是对一段后缀的 p 或者 q 进行加减：
在进行 A 的操作时，是会对某个后缀的 p 减少，q 增加，对于每一个 $f_{0/1,0/1}$ 的 p 和 q 的数量差是确定的，可以直接进行对应的修改；
而在对 B 进行操作的时候，q 会比 p 多修改一段长度为 $2L$ 的区间。
前面证明了选择的区间之间的距离大于 $2L$，所以说只有 q 被修改的那一段内的 $f_{0/1,0/1}$ 至多只有一个 p 和一个 q，所以对于每一个 $f_{0/1,0/1}$ 的修改也是可以直接计算出来的。

Q&A

谢谢大家。