# CCPC 北京 2023

E·Space, gyh, Itst, JohnVictor, liuzhangfeiabc, SpiritualKhorosho[†]

2023 年 12 月 23 日

清华大学学生算法协会

除命题人外, 感谢 caeious, ix35, Mys.C.K., rsy 等人参与题目准备工作.

† 按照字典顺序排列.

## Part I (讲者 *liuzhangfeiabc*)

K 报数 IV

F 最小环

I 勿蹧宠物

## Part II (讲者 *Mys.C.K.*)

A 游戏

B 替换

H 哈密顿

Part III (讲者 *E.Space*)

E 广播

J 图

C 史莱姆工厂

Part IV (讲者 *JohnVictor*)

G 【模板】线段树 1

D 三染色

# Part I (讲者 *liuzhangfeiabc*)

# K 报数 IV by *liuzhangfeiabc*

设 $f(n)$ 为 $n$ 的十进制各位数字之和. 设 $f_k(n) = f(f(\cdots f(n)))$（复合 $k$ 次）. 给定正整数 $N, k, m$，求有多少 $n \in [1, N]$ 满足 $f_k(n) = m$.

$N \le 10^{1000}, k, m \le 10^9$.

# 题解

不超过 $10^{1000}$ 的数中，$f(n)$ 最大为 $f(999\ldots9)$（ 1000 个 9）$=9000$.

不超过 9000 的数中，$f(n)$ 最大为 $f(8999)=35$.

不超过 35 的数中，$f(n)$ 最大为 $f(29)=11$.

不超过 11 的数中，$f(n)$ 最大为 9.

因此，任何不超过 $10^{1000}$ 的数最多作用 4 次 $f$ 就会变成一位数，以后就再也不会变了.

因此 $m$ 和 $k$ 的数据范围是唬人的.

# 题解

对于原问题，我们可以把作用 $k$ 次 $f$ 的过程拆成两部分:

- 先作用一次 $f$，得到一个不超过 $9000$ 的数.
- 再作用 $k-1$ 次 $f$ 得到 $m$.

我们直接预处理出 $1\ldots9000$ 的所有数进行 $0\ldots4$ 次 $f$ 作用之后变成什么.

然后对每个 $1\ldots9000$ 中的 $x$，计算 $1\ldots N$ 中有多少个数的数位之和是 $x$.

- 直接大力数位 dp: $dp(i,x,0/1)$ 表示前 $i$ 位，数位之和为 $x$，是否卡上界的数的个数即可.

然后对每个 $1\ldots9000$ 中的 $x$ 检查一下：$x$ 作用 $k-1$ 次 $f$ 之后是否等于 $m$，是的话就加上 $dp(x)$ 的贡献.

# F 最小环 by *ltst*

求一个 $n$ 个点、$m$ 条边、边带正整数权值的弱联通图的最小环.

$n \le 3 \times 10^5, m - n \le 1500$.

# 题解

注意到当一个节点的入度或出度为 0 时，这个点一定不会出现在最小环内，我们可以把这样的点删掉.

同时，如果一个节点 $v$ 的入度和出度都恰好为 1，那么我们可以直接把 $u \to v$ 和 $v \to w$ 缩成一条边 $u \to w$，边权为原本的两条边权和. 由于经过 $v$ 必然会走 $u \to v \to w$，这样不会改变最小环的存在性和长度.

注意到这样做完之后每个节点的入度和出度的和至少为三，而每条边只贡献一个入度和一个出度，因此设最终得到的图的点数为 $n'$，边数为 $m'$，我们有 $2m' \geq 3n'$.

又由于图弱联通，我们每次做以上操作删除一个点时都至少删除了一条边. 因此 $m' - n' \leq 1500$，得到 $n' \leq 3000, m' \leq 4500$.

因此我们在最终得到的图上枚举每个节点跑一遍最短路并更新最小环答案，复杂度为 $O(n'm' \log m')$.

给定 $s_1, \cdots, s_N$，求满足以下条件的下标序列 $i_1, i_2, \cdots, i_k$（$k$ 为任意正整数）的数量，其中 $+$ 表示字符串的连接：

- $s_{i_1} + s_{i_2} + \cdots + s_{i_k}$ 是长度为 $L$ 的回文串。

保证 $1 \le N \le 333$，$1 \le L \le 1000$，$\sum_{i=1}^{N} |s_i| \le 600$。

**题目名称是什么意思？**
使用样例 3 给出的单词，可以还原出一个有名的英文回文句子，而题目名称是这个句子的中文翻译。除此之外，题目名称的第一个字和第四个字、第二个字和第三个字的拼音（不考虑声调）是相同的。

# 分析

在做字符串匹配相关的题目时，有一类比较常见的策略是记当前匹配到了哪一位，并且这一位对应自动机中的哪个结点。

本题需要满足整个串是回文串，因此不好单向维护匹配情况。为了解决这个问题，我们不妨从两侧往内同时匹配：记 $f(i,\cdots)$ 表示当前处理到第 $i$ 位及第 $(L-i+1)$ 位，匹配情况为 $\cdots$ 时对应的方案数。

# 分析

在做字符串匹配相关的题目时，有一类比较常见的策略是记当前匹配到了哪一位，并且这一位对应自动机中的哪个结点。

本题需要满足整个串是回文串，因此不好单向维护匹配情况。为了解决这个问题，我们不妨从两侧往内同时匹配：记 $f(i,\cdots)$ 表示当前处理到第 $i$ 位及第 $(L-i+1)$ 位，匹配情况为 $\cdots$ 时对应的方案数。

通过 $f(i,\cdots)$ 转移到 $f(i+1,\cdots)$ 时，需要枚举下一个字母 $\alpha$，让第 $i$ 位对应的指向自动机 $q_l$ 结点的指针找对应 $\alpha$ 的正向转移的边，第 $(L-i+1)$ 位对应的指向自动机 $q_r$ 结点的指针找对应 $\alpha$ 的反向转移的边。正向转移可以使用 Trie 树简单维护，但是反向转移比较麻烦。

# 分析

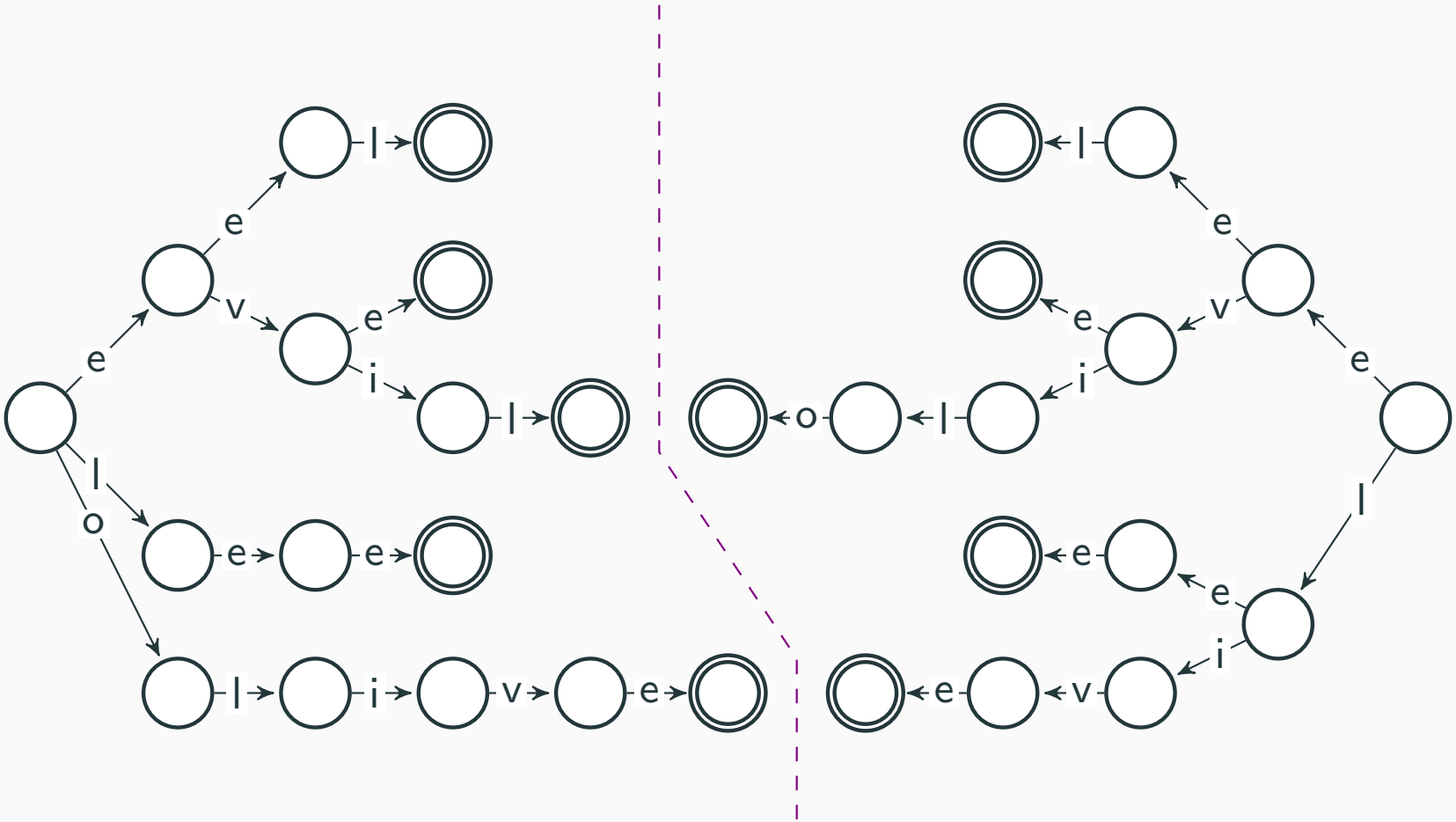
在做字符串匹配相关的题目时，有一类比较常见的策略是记当前匹配到了哪一位，并且这一位对应自动机中的哪个结点。

本题需要满足整个串是回文串，因此不好单向维护匹配情况。为了解决这个问题，我们不妨从两侧往内同时匹配：记 $f(i,\cdots)$ 表示当前处理到第 $i$ 位及第 $(L-i+1)$ 位，匹配情况为 $\cdots$ 时对应的方案数。

通过 $f(i,\cdots)$ 转移到 $f(i+1,\cdots)$ 时，需要枚举下一个字母 $\alpha$，让第 $i$ 位对应的指向自动机 $q_l$ 结点的指针找对应 $\alpha$ 的正向转移的边，第 $(L-i+1)$ 位对应的指向自动机 $q_r$ 结点的指针找对应 $\alpha$ 的反向转移的边。正向转移可以使用 Trie 树简单维护，但是反向转移比较麻烦。

注意到可以反向建立 Trie 树，这样就方便反向找了。

# 正向 Trie 及反向 Trie



# 做法

记 $f\left(i, q_l, q_r\right)$ 表示当前处理到第 $i$ 位及第 $(L-i+1)$ 位，第 $i$ 位对应正向 Trie 树上的结点 $q_l$，第 $(L-i+1)$ 位对应反向 Trie 树上的结点 $q_r$。转移时枚举第 $(i+1)$ 位及第 $(L-i)$ 位的字母 $\alpha$，如果 $q_l$ 和 $q_r$ 均有 $\alpha$ 的转移边，则相应转移给 $f\left(i+1, q_l', q_r'\right)$。

需要注意在 $i=L/2$ 处特判 $q_l$ 和 $q_r$ 能否合并：如果是奇数，则两个结点需要能通过同一个字母走到同一个单词的相同位置；如果是偶数，则有可能两边都恰好走到单词末尾，也有可能为同一个单词的相邻位置。

# 做法

标程的处理方法是记 $f\left(i,q_l,q_r\right)$ 表示已知两侧分别匹配到 $q_l$ 及 $q_r$ 时，第 $i$ 至 $(L-i+1)$ 位的合法方案数。可以先根据单词表计算 $f\left(\lceil L/2\rceil,\cdot,\cdot\right)=0/1$，再根据 $f(i+1,\cdot,\cdot)$ 往外扩展计算 $f(i,\cdot,\cdot)$。直接用循环计算 DP 可以滚动数组，不用担心被卡空间。

- 当然，本题的空间限制最后开到了 1024MiB，只要不开两个 $500\times600^2$ 的 int 数组都不会被卡。如果你想写记忆化搜索，用 char 记每个状态是否被访问过也是可以通过本题的。

复杂度看上去是 $O\left(L\left(\sum_{i=1}^{N}|s_i|\right)^2|\Sigma|\right)$，但是实际上状态不会很满。比较优秀的写法是枚举 $q_l$ 及 $q_r$ 转移边，这样可以做到严格 $O\left(L\left(\sum_{i=1}^{N}|s_i|\right)^2\right)$，可以轻松通过本题。

# $N$ 的范围？

$$1 \times 26 + 2 \times (333 - 26) = 640 > 600.$$

$$\frac{600 - 1 \times 26}{2} = 287.$$

# Part II (讲者 *Mys.C.K.*)

# A 游戏 by *ltst*

给定一棵 $n$ 个节点的树，双方在树上博弈.

初始 1 号节点有一个棋子。先手每次禁掉一条边，后手每次选择一条没被禁掉的边将棋子沿着它移动。后手移到度数恰好为 1 的节点时后手获胜，不能移动时先手获胜.

问双方绝顶聪明时谁获胜.

$1 \le n \le 10^5$

# 题解

首先需要注意到 $n=1$ 时先手胜，因为没有节点度数是 $1$.

# 题解

首先需要注意到 $n=1$ 时先手胜，因为没有节点度数是 $1$.

我们先获得一些比较直观的想法:

- 对于后手来说，走回头路是不优的，因为这样给了先手更多的机会禁边。所以我们可以把树看成以 $1$ 为根的外向树，后手只会往远离 $1$ 的方向移动棋子.

# 题解

首先需要注意到 $n=1$ 时先手胜，因为没有节点度数是 $1$.

我们先获得一些比较直观的想法:

- 对于后手来说，走回头路是不优的，因为这样给了先手更多的机会禁边。所以我们可以把树看成以 $1$ 为根的外向树，后手只会往远离 $1$ 的方向移动棋子.
- 对于先手来说，每次禁与棋子相连的一条边一定是更优的.

# 题解

首先需要注意到 $n=1$ 时先手胜，因为没有节点度数是 $1$.

我们先获得一些比较直观的想法:

- 对于后手来说，走回头路是不优的，因为这样给了先手更多的机会禁边。所以我们可以把树看成以 $1$ 为根的外向树，后手只会往远离 $1$ 的方向移动棋子.
- 对于先手来说，每次禁与棋子相连的一条边一定是更优的.
- 如果按照后手的策略每次只会往远离 $1$ 的方向走，那么先手每次也一定会禁掉当前棋子所在节点和一个儿子的边.

# 题解

将这个问题看成外向树之后，自然产生了递归的结构:

# 题解

将这个问题看成外向树之后，自然产生了递归的结构:

- 如果有 $\geq 2$ 个子树是获胜的，那么后手直接钻进任意一个获胜的子树内就可以获胜;

# 题解

将这个问题看成外向树之后，自然产生了递归的结构:

- 如果有 $\geq 2$ 个子树是获胜的，那么后手直接钻进任意一个获胜的子树内就可以获胜;
- 而如果只有 $\leq 1$ 个子树获胜，那么先手把这个子树禁掉，后手就没救了.

# 题解

将这个问题看成外向树之后，自然产生了递归的结构:

- 如果有 $\geq 2$ 个子树是获胜的，那么后手直接钻进任意一个获胜的子树内就可以获胜;
- 而如果只有 $\leq 1$ 个子树获胜，那么先手把这个子树禁掉，后手就没救了.

而将外向树换成一般的无向树不会改变胜负情况:

# 题解

将这个问题看成外向树之后，自然产生了递归的结构:

- 如果有 $\geq 2$ 个子树是获胜的，那么后手直接钻进任意一个获胜的子树内就可以获胜;
- 而如果只有 $\leq 1$ 个子树获胜，那么先手把这个子树禁掉，后手就没救了.

而将外向树换成一般的无向树不会改变胜负情况:

- 我们只需要考虑先手获胜的情况会不会发生改变.

# 题解

将这个问题看成外向树之后，自然产生了递归的结构:

- 如果有 $\geq 2$ 个子树是获胜的，那么后手直接钻进任意一个获胜的子树内就可以获胜;
- 而如果只有 $\leq 1$ 个子树获胜，那么先手把这个子树禁掉，后手就没救了.

而将外向树换成一般的无向树不会改变胜负情况:

- 我们只需要考虑先手获胜的情况会不会发生改变.
- 因为每个子树后手都会输，所以后手就算能回头钻进别的子树也没法赢.

# 题解

将这个问题看成外向树之后，自然产生了递归的结构:

- 如果有 $\geq 2$ 个子树是获胜的，那么后手直接钻进任意一个获胜的子树内就可以获胜;
- 而如果只有 $\leq 1$ 个子树获胜，那么先手把这个子树禁掉，后手就没救了.

而将外向树换成一般的无向树不会改变胜负情况:

- 我们只需要考虑先手获胜的情况会不会发生改变.
- 因为每个子树后手都会输，所以后手就算能回头钻进别的子树也没法赢.

因此 DFS 计算以 $1$ 为根时每个子树的获胜情况即可.

# 题解

将这个问题看成外向树之后，自然产生了递归的结构:

- 如果有 $\geq 2$ 个子树是获胜的，那么后手直接钻进任意一个获胜的子树内就可以获胜;
- 而如果只有 $\leq 1$ 个子树获胜，那么先手把这个子树禁掉，后手就没救了.

而将外向树换成一般的无向树不会改变胜负情况:

- 我们只需要考虑先手获胜的情况会不会发生改变.
- 因为每个子树后手都会输，所以后手就算能回头钻进别的子树也没法赢.

因此 DFS 计算以 $1$ 为根时每个子树的获胜情况即可.

复杂度 $O(n)$，可以轻松通过.

# B 替换 by *ltst*

给定字符串 $s$，字符集为 01？. 对于每个 $1 \le k \le |s|$，定义字符串 $t_k$ 为以下 01 字符串:

- 如果 $s_i$ 不是 ?，则 $t_{k,i} = s_i$.
- 否则 $t_{k,i} = t_{k,i-k}$，你需要递归地求出 $t_{k,i-k}$；如果 $i-k$ 越界则为 0.

你需要输出每个 $t_k$ 中的 1 的数量.

$1 \le |s| \le 10^5$.

# 题解

按照下标顺序得到字符串 $t_k$ 的每个字符，得到 $O(n^2)$ 做法.

# 题解

按照下标顺序得到字符串 $t_k$ 的每个字符，得到 $O(n^2)$ 做法.

注意到字符集是 `01`，故考虑压位，每次求出 $t_k$ 的 $\omega = 64$ 个字符.

# 题解

按照下标顺序得到字符串 $t_k$ 的每个字符，得到 $O(n^2)$ 做法.

注意到字符集是 01，故考虑压位，每次求出 $t_k$ 的 $\omega = 64$ 个字符.

$k \le \omega$ 的情况暴力，$k \ge \omega$ 的情况 $t_{k,i} \dots t_{k,i+\omega}$ 依赖的字符为 $t_{k,i-k} \dots t_{k,i-k+\omega}$，这是提前求好的. 因此我们需要做以下操作得到这段字符:

# 题解

按照下标顺序得到字符串 $t_k$ 的每个字符，得到 $O(n^2)$ 做法.

注意到字符集是 `01`，故考虑压位，每次求出 $t_k$ 的 $\omega = 64$ 个字符.

$k \le \omega$ 的情况暴力，$k \ge \omega$ 的情况 $t_{k,i} \dots t_{k,i+\omega}$ 依赖的字符为 $t_{k,i-k} \dots t_{k,i-k+\omega}$，这是提前求好的. 因此我们需要做以下操作得到这段字符:

- 得到 $t_{k,i-k} \dots t_{k,i-k+\omega}$ 的二进制表示;

# 题解

按照下标顺序得到字符串 $t_k$ 的每个字符，得到 $O(n^2)$ 做法.

注意到字符集是 `01`，故考虑压位，每次求出 $t_k$ 的 $\omega = 64$ 个字符.

$k \le \omega$ 的情况暴力，$k \ge \omega$ 的情况 $t_{k,i} \dots t_{k,i+\omega}$ 依赖的字符为 $t_{k,i-k} \dots t_{k,i-k+\omega}$，这是提前求好的. 因此我们需要做以下操作得到这段字符:

- 得到 $t_{k,i-k} \dots t_{k,i-k+\omega}$ 的二进制表示;
- 得到 $s_{k,i} \dots s_{k,i+\omega}$ 中 `?` 的位置和 `1` 的位置的二进制表示.

# 题解

按照下标顺序得到字符串 $t_k$ 的每个字符，得到 $O(n^2)$ 做法.

注意到字符集是 `01`，故考虑压位，每次求出 $t_k$ 的 $\omega = 64$ 个字符.

$k \le \omega$ 的情况暴力，$k \ge \omega$ 的情况 $t_{k,i} \dots t_{k,i+\omega}$ 依赖的字符为 $t_{k,i-k} \dots t_{k,i-k+\omega}$，这是提前求好的. 因此我们需要做以下操作得到这段字符:

- 得到 $t_{k,i-k} \dots t_{k,i-k+\omega}$ 的二进制表示;
- 得到 $s_{k,i} \dots s_{k,i+\omega}$ 中 `?` 的位置和 `1` 的位置的二进制表示.
- 对于 `?` 的位置使用 $t_{k,i-k} \dots t_{k,i-k+\omega}$，对于非 `?` 的位置使用 $s_{k,i} \dots s_{k,i+\omega}$，这容易在得到二进制表示后使用位运算实现.

# 题解

如果使用一个 long long 存储形如 $t_{k,j\omega} \dots t_{k,(j+1)\omega-1}$ 的一段字符，那么第一个部分只与两个 long long 有关; 如果每次我们取 $i$ 为 $\omega$ 的倍数，那么第二个部分可以直接预处理.

# 题解

如果使用一个 long long 存储形如 $t_{k,j\omega} \dots t_{k,(j+1)\omega-1}$ 的一段字符，那么第一个部分只与两个 long long 有关; 如果每次我们取 $i$ 为 $\omega$ 的倍数，那么第二个部分可以直接预处理.

这样得到复杂度为 $O\left(\frac{n^2}{\omega}\right)$ 的做法，应该可以通过.

# H 哈密顿 by *ltst*

给出 $n$ 个二元组 $(a_i, b_i)$.

考虑 $n$ 个节点的带权有向完全图 $G$，其中 $i \to j$ 边权为 $|a_i - b_j|$.

求 $G$ 的一条哈密顿回路使得其经过的边的边权和最大，并给出这个最大值.

$n \le 10^5, a_i, b_i \le 10^9$.

# 题解

我们需要一些绝对值的和最大.

# 题解

我们需要一些绝对值的和最大.

由于 $|a_i - b_j| = \max(a_i - b_j, b_j - a_i)$，我们可以在找哈密顿回路的基础上，要求给每个 $a_i$ 和 $b_j$ 一个正负号，使得每条 $i \to j$ 的边上的 $a_i$ 和 $b_j$ 一正一负，并要求按照符号加权的和尽可能大. 这个最大值就是我们想求的原问题的最大值.

# 题解

我们需要一些绝对值的和最大.

由于 $|a_i - b_j| = \max(a_i - b_j, b_j - a_i)$，我们可以在找哈密顿回路的基础上，要求给每个 $a_i$ 和 $b_j$ 一个正负号，使得每条 $i \to j$ 的边上的 $a_i$ 和 $b_j$ 一正一负，并要求按照符号加权的和尽可能大. 这个最大值就是我们想求的原问题的最大值.

由于哈密顿路上每个点入度和出度都是 1，所以最终这个标符号的方案就是在 $2n$ 个元素中 $n$ 个正 $n$ 个负. 因此我们实际上需要求的是一个给 $2n$ 个元素 $n$ 个正 $n$ 个负的方案，使得和尽可能大，且能够找到一条回路满足上面的条件（$a_i$ 和 $b_j$ 一正一负）.

# 题解

我们需要一些绝对值的和最大.

由于 $|a_i - b_j| = \max(a_i - b_j, b_j - a_i)$，我们可以在找哈密顿回路的基础上，要求给每个 $a_i$ 和 $b_j$ 一个正负号，使得每条 $i \to j$ 的边上的 $a_i$ 和 $b_j$ 一正一负，并要求按照符号加权的和尽可能大. 这个最大值就是我们想求的原问题的最大值.

由于哈密顿路上每个点入度和出度都是 1，所以最终这个标符号的方案就是在 $2n$ 个元素中 $n$ 个正 $n$ 个负. 因此我们实际上需要求的是一个给 $2n$ 个元素 $n$ 个正 $n$ 个负的方案，使得和尽可能大，且能够找到一条回路满足上面的条件（$a_i$ 和 $b_j$ 一正一负）.

如果我们扔掉回路的限制，那么这个最大的和的方案是简单的: 按照大小排序，取较大的 $n$ 个为正、较小的 $n$ 个为负. 但这样可能不存在哈密顿回路，我们需要仔细考虑这个哈密顿回路的条件.

# 题解

考察怎样的正负号标记会违背回路的限制.

# 题解

考察怎样的正负号标记会违背回路的限制.

注意到回路的连边需要要求 $a_i$ 中标为负（正）的必须和 $b_j$ 中标为正（负）的连边. 那么如果所有 $b_i$ 为负的 $i$ 对应 $a_i$ 都为正，那么由于 $b_i$ 为负、$a_i$ 为正的 $i$ 必然不能和 $b_j$ 为正、$a_j$ 为负的 $j$ 连边，那么当两种情况都存在的时候就没法连成大环.

# 题解

考察怎样的正负号标记会违背回路的限制.

注意到回路的连边需要要求 $a_i$ 中标为负（正）的必须和 $b_j$ 中标为正（负）的连边. 那么如果所有 $b_i$ 为负的 $i$ 对应 $a_i$ 都为正，那么由于 $b_i$ 为负、$a_i$ 为正的 $i$ 必然不能和 $b_j$ 为正、$a_j$ 为负的 $j$ 连边，那么当两种情况都存在的时候就没法连成大环.

而如果只存在一种情况，或者存在 $a_i, b_i$ 都为正的 $i$，那么可以通过这样的 $i$ 把两个部分连接在一起，从而可以构造出哈密顿回路.

# 题解

考察怎样的正负号标记会违背回路的限制.

注意到回路的连边需要要求 $a_i$ 中标为负（正）的必须和 $b_j$ 中标为正（负）的连边. 那么如果所有 $b_i$ 为负的 $i$ 对应 $a_i$ 都为正，那么由于 $b_i$ 为负、$a_i$ 为正的 $i$ 必然不能和 $b_j$ 为正、$a_j$ 为负的 $j$ 连边，那么当两种情况都存在的时候就没法连成大环.

而如果只存在一种情况，或者存在 $a_i, b_i$ 都为正的 $i$，那么可以通过这样的 $i$ 把两个部分连接在一起，从而可以构造出哈密顿回路.

因此不存在哈密顿回路的条件为:

# 题解

考察怎样的正负号标记会违背回路的限制.

注意到回路的连边需要要求 $a_i$ 中标为负（正）的必须和 $b_j$ 中标为正（负）的连边. 那么如果所有 $b_i$ 为负的 $i$ 对应 $a_i$ 都为正，那么由于 $b_i$ 为负、$a_i$ 为正的 $i$ 必然不能和 $b_j$ 为正、$a_j$ 为负的 $j$ 连边，那么当两种情况都存在的时候就没法连成大环.

而如果只存在一种情况，或者存在 $a_i, b_i$ 都为正的 $i$，那么可以通过这样的 $i$ 把两个部分连接在一起，从而可以构造出哈密顿回路.

因此不存在哈密顿回路的条件为:

- 存在 $b_i$ 为负、$a_i$ 为正的 $i$ 以及 $b_j$ 为正、$a_j$ 为负的 $j$;

# 题解

考察怎样的正负号标记会违背回路的限制.

注意到回路的连边需要要求 $a_i$ 中标为负（正）的必须和 $b_j$ 中标为正（负）的连边. 那么如果所有 $b_i$ 为负的 $i$ 对应 $a_i$ 都为正，那么由于 $b_i$ 为负、$a_i$ 为正的 $i$ 必然不能和 $b_j$ 为正、$a_j$ 为负的 $j$ 连边，那么当两种情况都存在的时候就没法连成大环.

而如果只存在一种情况，或者存在 $a_i, b_i$ 都为正的 $i$，那么可以通过这样的 $i$ 把两个部分连接在一起，从而可以构造出哈密顿回路.

因此不存在哈密顿回路的条件为:

- 存在 $b_i$ 为负、$a_i$ 为正的 $i$ 以及 $b_j$ 为正、$a_j$ 为负的 $j$;
- 不存在 $a_k, b_k$ 均为正的 $k$;

# 题解

由于初始的贪心解可能不满足限制，我们考虑用尽可能小的调整得到满足限制的解.

# 题解

由于初始的贪心解可能不满足限制，我们考虑用尽可能小的调整得到满足限制的解.

首先想到的贪心调整为交换排名为 $n$ 和 $n+1$ 的两个元素的符号，但这样依然可能不满足限制. 但如果这样不满足限制的话，排名为 $n$ 和 $n+1$ 的两个元素就是某个二元组 $(a_i, b_i)$ 了. 因此进一步的调整——交换 $n$ 和 $n+2$ 或者 $n-1$ 和 $n+1$ 的符号，就都会得到合法的方案.

# 题解

由于初始的贪心解可能不满足限制，我们考虑用尽可能小的调整得到满足限制的解.

首先想到的贪心调整为交换排名为 $n$ 和 $n+1$ 的两个元素的符号，但这样依然可能不满足限制. 但如果这样不满足限制的话，排名为 $n$ 和 $n+1$ 的两个元素就是某个二元组 $(a_i, b_i)$ 了. 因此进一步的调整——交换 $n$ 和 $n+2$ 或者 $n-1$ 和 $n+1$ 的符号，就都会得到合法的方案.

比较所有四种方案的合法性并取符合条件的方案中的最优解即可.

# 题解

由于初始的贪心解可能不满足限制，我们考虑用尽可能小的调整得到满足限制的解.

首先想到的贪心调整为交换排名为 $n$ 和 $n+1$ 的两个元素的符号，但这样依然可能不满足限制. 但如果这样不满足限制的话，排名为 $n$ 和 $n+1$ 的两个元素就是某个二元组 $(a_i, b_i)$ 了. 因此进一步的调整——交换 $n$ 和 $n+2$ 或者 $n-1$ 和 $n+1$ 的符号，就都会得到合法的方案.

比较所有四种方案的合法性并取符合条件的方案中的最优解即可.

复杂度 $O(n \log n)$.

# Part III (讲者 *E.Space*)

# E 广播 by *ltst*

称两个序列 $a_1,\ldots,a_m$ 和 $b_1,\ldots,b_n$ 是可广播的当且仅当以下条件成立:

- $\forall 0 \le i \le \min(n,m)-1$，要么 $a_{m-i}=b_{n-i}$，要么 $\min(a_{m-i},b_{n-i})=1$.

给定两个正整数序列，你需要向两个序列中插入若干个 `1`，使得插入 `1` 的个数最少，同时插入完后两个序列是可广播的.

序列长度、值域 $\le 2000$.

# 题解

由于可广播需要匹配序列的两段后缀，考虑从后往前进行匹配的决策. 同时这个插入和匹配的过程跟编辑距离很像，于是考虑动态规划.

设动态规划状态 $f_{i,j}$ 表示第一个序列已经匹配了 $i,\ldots,m$、第二个序列已经匹配了 $j,\ldots,n$ 时最少插入几个额外的 1.

# 题解

三种转移:

- $a_i$ 是新插入的 1 匹配的，从 $f_{i+1,j}+1$ 转移来;
- $b_j$ 是新插入的 1 匹配的，从 $f_{i,j+1}+1$ 转移来;
- $a_i$ 和 $b_j$ 本身可以匹配（$a_i=b_j$ 或 $\min(a_i,b_j)=1$），从 $f_{i+1,j+1}$ 转移来.

初始值为 $f_{m+1,n+1}=0$，最终答案为 $\min_{\min(i,j)=1} f_{i,j}$，这是由于根据可广播的定义，两个序列有一个匹配完了之后另外一个部分的剩下的前缀就不用匹配了.

复杂度 $O(nm)$.

# J 图 by *gyh*

有一个 $n$ 个点的无向正权图，对于每条边，判断其是否是某两个点最短路的必经边。

$n \leq 500$

# 题解

性质：对于一条边 $(x, y)$，如果其是某两个点最短路的必经边，那么其一定是 $x, y$ 之间最短路的必经边。

证明：如果 $(x, y)$ 不是 $x, y$ 之间的必经边，那么每次出现 $x, y$ 都可以替换为一条其他路径。

# 题解

判定 $(x, y)$ 是否为 $x, y$ 之间的唯一最短路。只需要判断是否存在 $z$ 使得 $d_{x,z} + d_{z,y} \leq w_{x,y}$。可以使用 Floyd 算法求出全源最短路，判定可以直接枚举 $z$。总复杂度 $O(n^3)$。

# C 史莱姆工厂 by *E.Space*

祖玛游戏

每发射一个球需要 $w$ 的代价（只能增加某一段的长度）.

至少要 $k$ 个连着才能消除.

消除一段长为 $i$ 的球可以得到 $p_i$ 的分数，保证 $p_i - p_{i-1} < w$

初始有 $n$ 段，求消完之后的最高分数.

$n \le 150, k \le 10$.

# 题解

区间 DP，考虑最后删哪些.

这个子序列必须满足能分成两段和 $< k$ 的部分，且整个子序列的总和必须 $\geq k$.

然后这个子序列把整个序列分成了若干段.

每一段中最后删去的不能是和这个子序列颜色相同的.

所以把这些都记到状态里就能转移了.

# 题解

记 $f(l,r)$ 为区间 $[l,r]$ 的答案.

$g_1(l,r,p_1)$ 为区间左端点为 $l$ 时，考虑了子序列在 $[l,r]$ 中的部分，且取了 $r$ 这段，第一部分的和为 $p_1$ 时，$[l,r]$ 区间内不在子序列中的所有区间的答案.

$g_2(l,r,p_2)$ 为…… 且第二部分的和为 $p_2$ 时，…… 的答案.

然后枚举下一段/最后一段转移就行.

颜色冲突就是在算 $f$ 的时候考虑一下 $l-1$ 和 $r+1$.

时间复杂度 $O(n^3k^2)$.

# Part IV (讲者 *JohnVictor*)

# G 【模板】线段树 1 by *JohnVictor*

有一个长度为 $n$ 的序列 $a_1, a_2, \cdots, a_n$，保证 $a_i$ 为奇数.

有两种操作:

1. 给定 $l, r, x$，将 $a_l, a_{l+1}, \cdots, a_r$ 加上偶数 $x$;
2. 给定 $l, r$，求 $a_l, a_{l+1}, \cdots, a_r$ 的乘积，答案对 $2^{20}$ 取模;

$n, q \leq 2 \times 10^5$.

# 题解

考虑线段树，每个节点维护多项式 $\prod(x+a_i) \bmod x^{20}$.

对于平移操作，只用将 $x$ 换为 $x+\Delta_x$ 并代入，复杂度为 $O(20^2)$。正确性是因为在 $x$ 保证偶数并且答案 $\bmod 2^{20}$ 的意义下，多项式 $\bmod x^{20}$ 并不影响结果.

对于求乘积操作，只用返回常数项.

总复杂度 $O(20^2 n \log n)$.

# D 三染色 by *JohnVictor*

回顾题意: 有一种填了 $\mathbb{Z}_3$ 颜色的 $n \times m$ 网格.

称一次演化为: 对于一个颜色为 $c$ 的格子, 如果有颜色为 $c-1$ 的相邻格子, 那么在下一轮这个格子的颜色变成 $c-1$, 否则不变.

称一个染色方案是 **好的**, 当且仅当在有限次演化之后所有格子颜色相同.

对于一个好的染色方案, 称它的权值为最小的正整数 $t$, 满足网格演化 $t$ 次后, 左上角格子颜色再也不变.

网格上有些格子已经确定颜色, 有些没有, 求好的染色方案数, 以及它们的权值和.

$2 \le n \le 5, 2 \le m \le 50$.

# 题解

注意到

$$\begin{bmatrix} 0 & 1 \\ 2 & 2 \end{bmatrix} \mapsto \begin{bmatrix} 2 & 0 \\ 1/2 & 1 \end{bmatrix},$$

可知形如这种矩阵 (或其翻转旋转, 整体加一个数) 一定会转移到自己.

# 题解

注意到

$$\begin{bmatrix} 0 & 1 \\ 2 & 2 \end{bmatrix} \mapsto \begin{bmatrix} 2 & 0 \\ 1/2 & 1 \end{bmatrix},$$

可知形如这种矩阵 (或其翻转旋转, 整体加一个数) 一定会转移到自己.

我们称这种结构 **阻塞** 了一个网格成为好的.

# 题解

注意到

$$\begin{bmatrix} 0 & 1 \\ 2 & 2 \end{bmatrix} \mapsto \begin{bmatrix} 2 & 0 \\ 1/2 & 1 \end{bmatrix},$$

可知形如这种矩阵 (或其翻转旋转, 整体加一个数) 一定会转移到自己.

我们称这种结构 **阻塞** 了一个网格成为好的.

**断言**: 不存在阻塞的网格都是好的.

# 题解

现在已知网格 $C \in \mathbb{Z}_3^{n \times m}$, 它的所有 $2 \times 2$ 子网格, 都不是阻塞结构.

我们一定可以找到一种给每个格子赋予一个 **整数** 的方案 $F \in \mathbb{Z}^{n \times m}$, 使得:

- $C_{ij} = F_{ij} \bmod 3$.
- $F$ 的相邻两项差不超过 $1$.

# 题解

$$C=\begin{bmatrix}0&0&2\\1&0&0\\1&1&0\end{bmatrix},F=\begin{bmatrix}0\\\\\\\end{bmatrix}$$

# 题解

$$C=\begin{bmatrix}0&0&2\\1&0&0\\1&1&0\end{bmatrix},F=\begin{bmatrix}0&0&\\&&\\&&\end{bmatrix}$$

# 题解

$$C=\begin{bmatrix}0&0&2\\1&0&0\\1&1&0\end{bmatrix},F=\begin{bmatrix}0&0&-1\\&&\\&&\end{bmatrix}$$

$F_{ij}\in F_{i(j-1)}+\{-1,0,1\}$ 保证 $F$ 被唯一确定.

# 题解

$$C=\begin{bmatrix}0&0&2\\1&0&0\\1&1&0\end{bmatrix},F=\begin{bmatrix}0&0&-1\\1&0&0\\&&\end{bmatrix}$$

$F_{ij}\in F_{i(j-1)}+\{-1,0,1\}$ 保证 $F$ 被唯一确定.

# 题解

$$C = \begin{bmatrix} 0 & 0 & 2 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix}, F = \begin{bmatrix} 0 & 0 & -1 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix}$$

$F_{ij} \in F_{i(j-1)} + \{-1, 0, 1\}$ 保证 $F$ 被唯一确定.

没有阻塞结构, 保证 $F$ 合法 (保证不用于生成 $F$ 的那部分约束被满足).

# 题解

$$C=\begin{bmatrix}0&0&2\\1&0&0\\1&1&0\end{bmatrix},F=\begin{bmatrix}0&0&-1\\1&0&0\\1&1&0\end{bmatrix}$$

# 题解

$$C=\begin{bmatrix}0&2&2\\0&0&2\\1&0&0\end{bmatrix},F=\begin{bmatrix}0&-1&-1\\0&0&-1\\1&0&0\end{bmatrix}$$

# 题解

$$C=\begin{bmatrix}2 & 2 & 2\\ 0 & 2 & 2\\ 0 & 0 & 2\end{bmatrix},F=\begin{bmatrix}-1 & -1 & -1\\ 0 & -1 & -1\\ 0 & 0 & -1\end{bmatrix}$$

# 题解

$$C=\begin{bmatrix}2 & 2 & 2\\ 2 & 2 & 2\\ 0 & 2 & 2\end{bmatrix},F=\begin{bmatrix}-1 & -1 & -1\\ -1 & -1 & -1\\ 0 & -1 & -1\end{bmatrix}$$

# 题解

$$C = \begin{bmatrix} 2 & 2 & 2 \\ 2 & 2 & 2 \\ 2 & 2 & 2 \end{bmatrix}, F = \begin{bmatrix} -1 & -1 & -1 \\ -1 & -1 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

- $C$ 上的演化可以对应于 $F$ 上的演化.
- $F$ 上的演化: 每次把格子变成自己和周围的 max.

# 题解

$$C = \begin{bmatrix} 2 & 2 & 2 \\ 2 & 2 & 2 \\ 2 & 2 & 2 \end{bmatrix}, F = \begin{bmatrix} -1 & -1 & -1 \\ -1 & -1 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

- $C$ 上的演化可以对应于 $F$ 上的演化.
- $F$ 上的演化: 每次把格子变成自己和周围的 max.
- 稳定态: 所有数稳定到 $F$ 的**最小值**.

# 题解

$$C=\begin{bmatrix}2&2&2\\2&2&2\\2&2&2\end{bmatrix},F=\begin{bmatrix}-1&-1&-1\\-1&-1&-1\\-1&-1&-1\end{bmatrix}$$

- $C$ 上的演化可以对应于 $F$ 上的演化.
- $F$ 上的演化: 每次把格子变成自己和周围的 max.
- 稳定态: 所有数稳定到 $F$ 的**最小值**.
- 权值: 左上角格子到 $F$ 的一个最小值的**最短路**.

# 题解

不妨设 $n \le m$, 按照列从小到大, 按照顺序一个个填写颜色, 维护轮廓线上 $F$ 的值.

# 题解

不妨设 $n \le m$, 按照列从小到大, 按照顺序一个个填写颜色, 维护轮廓线上 $F$ 的值.

轮廓上的 $F$ 本身的状态, 只需要知道相邻数的差, 状态数为 $O(3^n)$.

# 题解

不妨设 $n \le m$, 按照列从小到大, 按照顺序一个个填写颜色, 维护轮廓线上 $F$ 的值.

轮廓上的 $F$ 本身的状态, 只需要知道相邻数的差, 状态数为 $O(3^n)$.

但我们还需要知道左上角距离目前看到的最小值的最短距离, 这个数大小 $O(m)$.

还需要知道目前轮廓线上最小的数比最小值大多少, 这个数大小 $O(m)$.

时间复杂度 $O(nm \cdot 3^n m^2)$.

# 题解

不妨设 $n \le m$, 按照列从小到大, 按照顺序一个个填写颜色, 维护轮廓线上 $F$ 的值.

轮廓上的 $F$ 本身的状态, 只需要知道相邻数的差, 状态数为 $O(3^n)$.

~~但我们还需要知道左上角距离目前看到的最小值的最短距离, 这个数大小 $O(m)$.~~

还需要知道目前轮廓线上最小的数比最小值大多少, 这个数大小 $O(m)$.

~~时间复杂度 $O(nm \cdot 3^n m^2)$.~~

在更新最小值的时候决策这是不是最后的最小值, 如果是, 那么最短距离最多再多 $n$, 可以据此优化状态, 做到时间复杂度 $O(nm \cdot 3^n nm)$.

感谢倾听!