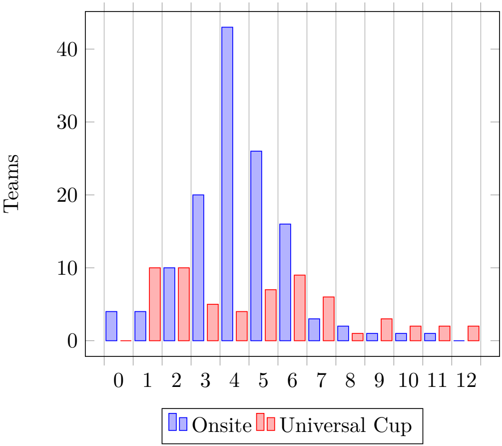
CCPC Final 2022 Tutorial

Qingyu

May 21, 2023

Statistics



Statistics - Onsite

冠: 11 题 @ 1497
杯: 8 题 @ 1206
金: 6 题 @ 810
银: 5 题 @ 676
铜: 4 题 @ 510

Statements

给定一张图，将其划分成两棵有根树，第一棵树每个节点的编号小于父亲的编号，第二棵树每个节点的编号大于父亲的编号，求方案数，取模 $998\,244\,353$。

Statements

给定一张图，将其划分成两棵有根树，第一棵树每个节点的编号小于父亲的编号，第二棵树每个节点的编号大于父亲的编号，求方案数，取模 $998\,244\,353$。

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

Statements

给定一张图，将其划分成两棵有根树，第一棵树每个节点的编号小于父亲的编号，第二棵树每个节点的编号大于父亲的编号，求方案数，取模 $998\,244\,353$。

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

Solved by 55 teams. Submissions: 284. (Onsite + UCup)

Solution

树可以看成给每个点确定一个父亲，那么原题可以看成给 $[1, n-1]$ 的每个点找一个更大的父亲，$[2, n]$ 找一个更小的父亲。

Solution

树可以看成给每个点确定一个父亲，那么原题可以看成给 $[1, n-1]$ 的每个点找一个更大的父亲，$[2, n]$ 找一个更小的父亲。

一条连接 $x, y(x \leq y)$ 的边可成为 x 在第一棵树上到父亲的边，或者成为 y 在第二棵树上到父亲的边。整个题目可以看成每条边匹配一棵树上的某一个点的问题。

Solution

树可以看成给每个点确定一个父亲，那么原题可以看成给 $[1, n-1]$ 的每个点找一个更大的父亲，$[2, n]$ 找一个更小的父亲。

一条连接 $x, y(x \leq y)$ 的边可成为 x 在第一棵树上到父亲的边，或者成为 y 在第二棵树上到父亲的边。整个题目可以看成每条边匹配一棵树上的某一个点的问题。

总共有 $2n-2$ 个需要确定父亲的点，且输入中有 $2n-2$ 条边。可以发现有解当且仅当是一个基环树森林，即每一个连通块边数 $=$ 点数，此时的答案为 2^C。

Solution

树可以看成给每个点确定一个父亲，那么原题可以看成给 $[1, n-1]$ 的每个点找一个更大的父亲，$[2, n]$ 找一个更小的父亲。

一条连接 $x, y(x \leq y)$ 的边可成为 x 在第一棵树上到父亲的边，或者成为 y 在第二棵树上到父亲的边。整个题目可以看成每条边匹配一棵树上的某一个点的问题。

总共有 $2n-2$ 个需要确定父亲的点，且输入中有 $2n-2$ 条边。可以发现有解当且仅当是一个基环树森林，即每一个连通块边数 $=$ 点数，此时的答案为 2^C。

总复杂度 $O(n)$。

Solution

树可以看成给每个点确定一个父亲，那么原题可以看成给 $[1, n-1]$ 的每个点找一个更大的父亲，$[2, n]$ 找一个更小的父亲。

一条连接 $x, y(x \le y)$ 的边可成为 x 在第一棵树上到父亲的边，或者成为 y 在第二棵树上到父亲的边。整个题目可以看成每条边匹配一棵树上的某一个点的问题。

总共有 $2n-2$ 个需要确定父亲的点，且输入中有 $2n-2$ 条边。可以发现有解当且仅当是一个基环树森林，即每一个连通块边数 $=$ 点数，此时的答案为 2^C。

总复杂度 $O(n)$。

Fun fact：本题曾被郭雨豪同学计划用作集训队互测题。

Statements

给定一个路径压缩并查集的起始状态，问是否能到达另一个状态，并构造方案。

Statements

给定一个路径压缩并查集的起始状态，问是否能到达另一个状态，并构造方案。

$1 \le n \le 1\,000, \sum n^2 \le 5 \times 10^6$

Statements

给定一个路径压缩并查集的起始状态，问是否能到达另一个状态，并构造方案。

$1 \leq n \leq 1\,000, \sum n^2 \leq 5 \times 10^6$

Solved by 8 teams. Submissions: 30. (Onsite + UCup)

Solution

最终成环/初始状态联通的点在最终不联通 $\Longrightarrow$ 无解.

Solution

最终成环/初始状态联通的点在最终不联通 $\Longrightarrow$ 无解.
设初始有 k 棵树 $T_1, T_2, \cdots, T_k$，根分别为 $r_1, r_2, \cdots, r_k$。

Solution

最终成环/初始状态联通的点在最终不联通 $\Longrightarrow$ 无解.
设初始有 k 棵树 $T_1, T_2, \cdots, T_k$，根分别为 $r_1, r_2, \cdots, r_k$。

Solution

最终成环/初始状态联通的点在最终不联通 $\Longrightarrow$ 无解.

设初始有 k 棵树 $T_1, T_2, \cdots, T_k$，根分别为 $r_1, r_2, \cdots, r_k$。

假设最终状态只有一棵树组成，考虑由这 k 个根在最终状态构成的树的结构。

Solution

最终成环/初始状态联通的点在最终不联通 $\Longrightarrow$ 无解.

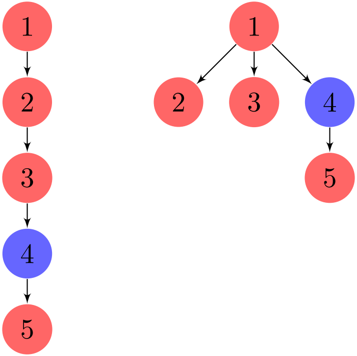
设初始有 k 棵树 $T_1, T_2, \cdots, T_k$，根分别为 $r_1, r_2, \cdots, r_k$。

假设最终状态只有一棵树组成，考虑由这 k 个根在最终状态构成的树的结构。

如果 x 是 y 的父亲，那么必定是某一步 y 直接合并到 x。

Solution

因为 find 操作只会将具有祖先后代关系的一组点的祖先后代关系破坏，而不会增加新的祖先后代关系。



Solution

所以如果最终时刻有限制 x 必须是 y 的祖先，那么在任意时刻该限制都需要满足。

Solution

所以如果最终时刻有限制 x 必须是 y 的祖先，那么在任意时刻该限制都需要满足。

接下来考虑，如果最终某个 r_x 的子树内存在点 u，使得 u 在初始时位于另一棵子树 r_y 中，那么在某一时刻包含 r_y 是 r_x 的祖先。

Solution

所以如果最终时刻有限制 x 必须是 y 的祖先，那么在任意时刻该限制都需要满足。

接下来考虑，如果最终某个 r_x 的子树内存在点 u，使得 u 在初始时位于另一棵子树 r_y 中，那么在某一时刻包含 r_y 是 r_x 的祖先。

因此我们可以得到若干形如 r_x 必须是 r_y 祖先的限制。

Solution

所以如果最终时刻有限制 x 必须是 y 的祖先，那么在任意时刻该限制都需要满足。

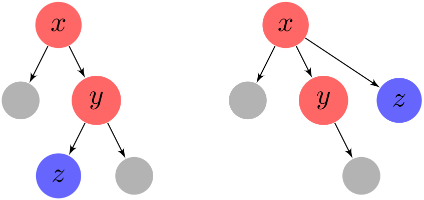
接下来考虑，如果最终某个 r_x 的子树内存在点 u，使得 u 在初始时位于另一棵子树 r_y 中，那么在某一时刻包含 r_y 是 r_x 的祖先。

因此我们可以得到若干形如 r_x 必须是 r_y 祖先的限制。

如果限制成环，显然无解。否则跑出任意一组拓扑序，按照拓扑序来构造方案。

Solution

注意到如果 x 是根，y 是 x 的某一个儿子，z 是 y 的某一个儿子。那么我们可以通过操作 z 来将 z 所在子树提到 x 的儿子，而其他结构没有影响。



Solution

同时，注意到，对于初始时非根节点 t，t 与 $fa[t]$ 具有父子关系，当且仅当 t 没有被操作过。

Solution

同时，注意到，对于初始时非根节点 t，t 与 $fa[t]$ 具有父子关系，当且仅当 t 没有被操作过。

注意到我们可以通过初始状态中 x 与 x 的父亲在最终是否具有直接父子关系来判定是否要在某个时刻操作 x。

Solution

同时，注意到，对于初始时非根节点 t，t 与 $fa[t]$ 具有父子关系，当且仅当 t 没有被操作过。

注意到我们可以通过初始状态中 x 与 x 的父亲在最终是否具有直接父子关系来判定是否要在某个时刻操作 x。

因此我们可以知道在初始时需要对哪些点先进行 find。在初始时便进行这样的操作一定是最优的。

Solution

同时，注意到，对于初始时非根节点 t，t 与 $fa[t]$ 具有父子关系，当且仅当 t 没有被操作过。

注意到我们可以通过初始状态中 x 与 x 的父亲在最终是否具有直接父子关系来判定是否要在某个时刻操作 x。

因此我们可以知道在初始时需要对哪些点先进行 find。在初始时便进行这样的操作一定是最优的。

因此我们直接按照拓扑序依次复原即可。由于做到一个点时至多进行 n 次操作，因此操作次数（即使是粗略的）上界为 n^2+n。

Statements

给定一棵树中以每个点为起点的 DFS 序，复原这棵树。

Statements

给定一棵树中以每个点为起点的 DFS 序，复原这棵树。

$1 \leq n \leq 1\,000, \sum n^2 \leq 2 \times 10^6$

Statements

给定一棵树中以每个点为起点的 DFS 序，复原这棵树。
$1 \le n \le 1\,000, \sum n^2 \le 2 \times 10^6$
Solved by 135 teams. Submissions: 323. (Onsite + UCup)

Solution

Lemma I

DFS 序中最后一个点一定是叶子节点。

Solution

Lemma I

DFS 序中最后一个点一定是叶子节点。

Lemma II

DFS 序中第二个点一定与第一个点直接相邻。

Solution

Lemma I

DFS 序中最后一个点一定是叶子节点。

Lemma II

DFS 序中第二个点一定与第一个点直接相邻。

于是我们不停的删除叶子节点即可。

Solution

Lemma I

DFS 序中最后一个点一定是叶子节点。

Lemma II

DFS 序中第二个点一定与第一个点直接相邻。

于是我们不停的删除叶子节点即可。

Fun fact: DFS Order 是 EC-Final 2021 的题，DFS Order 2 是 Jinan Regional 2022 的题。

Statements

维护一个 012 串 s，支持区间加 1 模 3，或区间询问能否通过删除两个相邻相等元素删成空串。

Statements

维护一个 012 串 s，支持区间加 1 模 3，或区间询问能否通过删除两个相邻相等元素删成空串。

Solved by 11 teams. Submissions: 27. (Onsite + UCup)

Solution

构造三个随机可逆矩阵 M_0, M_1, M_2。

Solution

构造三个随机可逆矩阵 M_0, M_1, M_2。

对于每个 i，如果 $i \bmod 2 = 0$，则令 $A_i = M_{s_i}$，否则 $A_i = M_{s_i}^{-1}$。

Solution

构造三个随机可逆矩阵 M_0, M_1, M_2。

对于每个 i，如果 $i \bmod 2 = 0$，则令 $A_i = M_{s_i}$，否则 $A_i = M_{s_i}^{-1}$。

一个区间可以消空 $\implies \prod_{i=l}^{r} A_i = I$。

Solution

构造三个随机可逆矩阵 M_0, M_1, M_2。

对于每个 i，如果 $i \bmod 2 = 0$，则令 $A_i = M_{s_i}$，否则 $A_i = M_{s_i}^{-1}$。

一个区间可以消空 $\Longrightarrow \prod_{i=l}^{r} A_i = I$。

可以使用线段树维护区间加 1 模 3 与区间矩阵乘积，时间复杂度 $O(n \log n \cdot k^3)$。

Solution

Instead, you should consider the following two matrices.

$$\begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix} \& \begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix}$$

To prove that they form a free group you can use the ping-pong lemma. For a worked, and very readable, proof using (essentially) this method, see John Meier's book *Graphs, Groups and Trees*, Section 3.1.3. The idea is to consider how these matrices and their inverses act on a coordinate (a, b), and then extend this to regions. Finally, one supposes that a non-empty, reduced word w is trivial and applies this to a region X, but it is seen that this yields a different region, $wX \neq X$, so w cannot be trivial.

使用 $ax + b$ 与 $a^{-1}x - ba^{-1}$？

Solution

Instead, you should consider the following two matrices.

$$\begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix} \& \begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix}$$

To prove that they form a free group you can use the ping-pong lemma. For a worked, and very readable, proof using (essentially) this method, see John Meier's book *Graphs, Groups and Trees*, Section 3.1.3. The idea is to consider how these matrices and their inverses act on a coordinate (a, b), and then extend this to regions. Finally, one supposes that a non-empty, reduced word w is trivial and applies this to a region X, but it is seen that this yields a different region, $wX \neq X$, so w cannot be trivial.

使用 $ax + b$ 与 $a^{-1}x - ba^{-1}$?
02021210202121

Solution

Instead, you should consider the following two matrices.

$$\begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix} \& \begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix}$$

To prove that they form a free group you can use the ping-pong lemma. For a worked, and very readable, proof using (essentially) this method, see John Meier's book *Graphs, Groups and Trees*, Section 3.1.3. The idea is to consider how these matrices and their inverses act on a coordinate (a, b), and then extend this to regions. Finally, one supposes that a non-empty, reduced word w is trivial and applies this to a region X, but it is seen that this yields a different region, $wX \neq X$, so w cannot be trivial.

使用 $ax + b$ 与 $a^{-1}x - ba^{-1}$？

02021210202121

Fun fact： Flower's Land 是 Petrozavodsk Summer 2022: Qingyu, flower and their friends' Contest 的题。

Statements

给定一个包含 c,p,? 的串，问有多少子串可以通过将 ? 填写成 c 或 p 使其形如 $\mathrm{c}^{2t}\mathrm{pc}^{t}$

Statements

给定一个包含 c,p,? 的串，问有多少子串可以通过将 ? 填写成 c 或 p 使其形如 $\mathrm{c}^{2t}\mathrm{pc}^{t}$

Solved by 185 teams. Submissions: 277. (Onsite + UCup)

Solution

考虑枚举 p 的位置 i，二分出向左与向右有多少个非 p 的字符 d_l, d_r，贡献即为 $\min(\left\lfloor \frac{d_l}{2} \right\rfloor, d_r)$。

Solution

考虑枚举 p 的位置 i，二分出向左与向右有多少个非 p 的字符 d_l, d_r，贡献即为 $\min(\left\lfloor \frac{d_l}{2} \right\rfloor, d_r)$。

时间复杂度为 $O(n \log n)$。

Solution

考虑枚举 p 的位置 i，二分出向左与向右有多少个非 p 的字符 d_l, d_r，贡献即为 $\min(\left\lfloor \frac{d_l}{2} \right\rfloor, d_r)$。

时间复杂度为 $O(n \log n)$。

也可以预处理出 d_l, d_r 来做到 $O(n)$，但没必要。

Statements

在给定的图中，问是否对任意 Alice 和 Bob 的起点，Alice 都可以追逐到 Bob。

Statements

在给定的图中，问是否对任意 Alice 和 Bob 的起点，Alice 都可以追逐到 Bob。

Solved by 175 teams. Submissions: 386. (Onsite + UCup)

Solutions

考虑 B 必胜的充要条件。

Solutions

考虑 B 必胜的充要条件。

假设存在两个点 $i, i+1$，使得它们的链（L_2）上距离大于 2，那么这种方案 B 一定不是必胜的。因为这就表示 i 和 $i+1$ 在链上的邻居不相交。只要让一开始 A 的位置在 i，每一次 A 移动到 $i, i+1$ 中和 B 当前所在点不相邻的点即可，B 永远赢不了。

另一方面，如果对于任意相邻点 $i, i+1$，它们的链上距离都小于等于 2，那么这种方案一定是 B 必胜，只需要 B 每次往 A 的方向移动一步即可。

Solutions

考虑 B 必胜的充要条件。

假设存在两个点 $i, i+1$，使得它们的链（L_2）上距离大于 2，那么这种方案 B 一定不是必胜的。因为这就表示 i 和 $i+1$ 在链上的邻居不相交。只要让一开始 A 的位置在 i，每一次 A 移动到 $i, i+1$ 中和 B 当前所在点不相邻的点即可，B 永远赢不了。

另一方面，如果对于任意相邻点 $i, i+1$，它们的链上距离都小于等于 2，那么这种方案一定是 B 必胜，只需要 B 每次往 A 的方向移动一步即可。

因此我们给出了一种方案合法的充要条件：所有点对 $(i, i+1)$ 的链上距离小于等于 2，判断一下即可。

Statements

将一个字符串的所有不同子串建出一张 DAG，一个点 u 向另一个点 v 连边当且仅当 u 代表的串可以通过将 u 的第一个或最后一个字符删除得到。

给定 DAG，复原出字典序最小的原串。

Statements

将一个字符串的所有不同子串建出一张 DAG，一个点 u 向另一个点 v 连边当且仅当 u 代表的串可以通过将 u 的第一个或最后一个字符删除得到。

给定 DAG，复原出字典序最小的原串。

Solved by 4 teams. Submissions: 15. (Onsite + UCup)

Solutions

为了方便，下文有时不区分“点”和“点代表的串”。例如“点的长度”表示“这个点代表的串的长度”。

Solutions

为了方便，下文有时不区分“点”和“点代表的串”。例如“点的长度”表示“这个点代表的串的长度”。

首先，入度为 0 的的点一定代表单个字符，但是每个点代表哪个字符是无法区分的，下面我们任意为这些点分配一个字符。

Solutions

为了方便，下文有时不区分“点”和“点代表的串”。例如“点的长度”表示“这个点代表的串的长度”。

首先，入度为 0 的的点一定代表单个字符，但是每个点代表哪个字符是无法区分的，下面我们任意为这些点分配一个字符。

然后，可以通过这些点推出其它每个点的长度，包括原串的长度（原串是唯一的出度为 0 的点）。

Solutions

为了方便，下文有时不区分“点”和“点代表的串”。例如“点的长度”表示“这个点代表的串的长度”。

首先，入度为 0 的的点一定代表单个字符，但是每个点代表哪个字符是无法区分的，下面我们任意为这些点分配一个字符。

然后，可以通过这些点推出其它每个点的长度，包括原串的长度（原串是唯一的出度为 0 的点）。

通过简单模拟，可以猜测：每个点代表的串是比较唯一的。

Solutions

为了方便，下文有时不区分“点”和“点代表的串”。例如“点的长度”表示“这个点代表的串的长度”。

首先，入度为 0 的的点一定代表单个字符，但是每个点代表哪个字符是无法区分的，下面我们任意为这些点分配一个字符。

然后，可以通过这些点推出其它每个点的长度，包括原串的长度（原串是唯一的出度为 0 的点）。

通过简单模拟，可以猜测：每个点代表的串是比较唯一的。

形式化地讲，对于每个点 u 存在一个串 $S(u)$，使得它实际代表的串一定是 $S(u)$ 或者 $rev(S(u))$。这里 $rev(s)$ 代表 s 翻转后的串。

Solutions

下面来归纳地证明一下这个结论。

Solutions

下面来归纳地证明一下这个结论。

每个点的入度至多为 2，分别代表了 $S(u)$ 去掉首尾之后的串。设 $L(s)$ 表示 s 去掉结尾得到的串，$R(s)$ 表示 s 去掉开头得到的串。

Solutions

下面来归纳地证明一下这个结论。

每个点的入度至多为 2，分别代表了 $S(u)$ 去掉首尾之后的串。设 $L(s)$ 表示 s 去掉结尾得到的串，$R(s)$ 表示 s 去掉开头得到的串。

设 $G(u)$ 表示 u 的入点集合。

Solutions

下面来归纳地证明一下这个结论。

每个点的入度至多为 2，分别代表了 $S(u)$ 去掉首尾之后的串。设 $L(s)$ 表示 s 去掉结尾得到的串，$R(s)$ 表示 s 去掉开头得到的串。

设 $G(u)$ 表示 u 的入点集合。

一般情况下，点 $|G(u)|$ 应该为 2，且它的两个入点有恰好一个公共的入点。

Solutions

先来分析一些不符合上述性质的特殊情况。

Solutions

先来分析一些不符合上述性质的特殊情况。

如果 $|G(u)| = 1$，说明 $L(S(u)) = R(S(u))$，也就是 $S(u)$ 只由一种字符构成，我们可以求出这种字符。称这类点为第一类点。第一类点的入点一定也是第一类点。

Solutions

先来分析一些不符合上述性质的特殊情况。

如果 $|G(u)| = 1$，说明 $L(S(u)) = R(S(u))$，也就是 $S(u)$ 只由一种字符构成，我们可以求出这种字符。称这类点为第一类点。第一类点的入点一定也是第一类点。

如果 $G(u) = \{v_1, v_2\}$，但 $G(v_1) = G(v_2)$，此时 $S(u)$ 一定是由两种不同字符交替得到的。我们可以求出这两种字符，但只通过能到达 u 的点是无法区分这两种字符的。

Solutions

先来分析一些不符合上述性质的特殊情况。

如果 $|G(u)|=1$，说明 $L(S(u))=R(S(u))$，也就是 $S(u)$ 只由一种字符构成，我们可以求出这种字符。称这类点为第一类点。第一类点的入点一定也是第一类点。

如果 $G(u)=\{v_1,v_2\}$，但 $G(v_1)=G(v_2)$，此时 $S(u)$ 一定是由两种不同字符交替得到的。我们可以求出这两种字符，但只通过能到达 u 的点是无法区分这两种字符的。

这类点代表的串应该是 $S(u)$ 或 $flip(S(u))$，这里 $flip(s)$ 表示交换 s 中的两种字符。注意：如果 $|S(u)|$ 为奇数，那么 $rev(S(u))\neq flip(S(u))$。

Solutions

先来分析一些不符合上述性质的特殊情况。

如果 $|G(u)| = 1$，说明 $L(S(u)) = R(S(u))$，也就是 $S(u)$ 只由一种字符构成，我们可以求出这种字符。称这类点为第一类点。第一类点的入点一定也是第一类点。

如果 $G(u) = \{v_1, v_2\}$，但 $G(v_1) = G(v_2)$，此时 $S(u)$ 一定是由两种不同字符交替得到的。我们可以求出这两种字符，但只通过能到达 u 的点是无法区分这两种字符的。

这类点代表的串应该是 $S(u)$ 或 $\mathit{flip}(S(u))$，这里 $\mathit{flip}(s)$ 表示交换 s 中的两种字符。注意：如果 $|S(u)|$ 为奇数，那么 $\mathit{rev}(S(u)) \neq \mathit{flip}(S(u))$。

称这类点为第二类点。第二类点的入点一定也是第二类点。

Solutions

先来分析一些不符合上述性质的特殊情况。

如果 $|G(u)|=1$，说明 $L(S(u))=R(S(u))$，也就是 $S(u)$ 只由一种字符构成，我们可以求出这种字符。称这类点为第一类点。第一类点的入点一定也是第一类点。

如果 $G(u)=\{v_1,v_2\}$，但 $G(v_1)=G(v_2)$，此时 $S(u)$ 一定是由两种不同字符交替得到的。我们可以求出这两种字符，但只通过能到达 u 的点是无法区分这两种字符的。

这类点代表的串应该是 $S(u)$ 或 $flip(S(u))$，这里 $flip(s)$ 表示交换 s 中的两种字符。注意：如果 $|S(u)|$ 为奇数，那么 $rev(S(u))\neq flip(S(u))$。

称这类点为第二类点。第二类点的入点一定也是第二类点。

特别地，我们认为单个字符是第一类点，长度为 2 的串是第二类点。

Solutions

称其余的点为第三类点，这类点代表的串为 $S(u)$ 或 $rev(S(u))$ 的形式。

Solutions

称其余的点为第三类点，这类点代表的串为 $S(u)$ 或 $rev(S(u))$ 的形式。

对于第三类点，在维护 $S(u)$ 的同时，还要维护 u 的入点分别代表了 $S(u)$ 的哪个部分。形式化地讲，对某个 $v \in G(u)$，我们需要维护如下形式的信息：

Solutions

称其余的点为第三类点，这类点代表的串为 $S(u)$ 或 $rev(S(u))$ 的形式。

对于第三类点，在维护 $S(u)$ 的同时，还要维护 u 的入点分别代表了 $S(u)$ 的哪个部分。形式化地讲，对某个 $v \in G(u)$，我们需要维护如下形式的信息：

【$S(v)$ 或 $rev(S(v))$】代表了【$L(u)$ 或 $R(u)$】。

Solutions

对于一个第三类点 u，设 $G(u) = v_1, v_2$。

Solutions

对于一个第三类点 u，设 $G(u) = v_1, v_2$。

情况 1：如果 v_1, v_2 都不是第二类点，那么可以通过 $G(v_1)$ 和 $G(v_2)$ 的公共点算出点 u 的信息。

Solutions

对于一个第三类点 u，设 $G(u) = v_1, v_2$。

情况 1：如果 v_1, v_2 都不是第二类点，那么可以通过 $G(v_1)$ 和 $G(v_2)$ 的公共点算出点 u 的信息。

情况 2：否则，v_1, v_2 一定恰有一个是第二类点，假设为 v_1，那么 v_2 是第三类点，$L(S(v_2))$ 和 $R(S(v_2))$ 一定恰有一个是可以匹配 $S(v_1)$ 的交替串。

Solutions

对于一个第三类点 u，设 $G(u)=v_1,v_2$。

情况 1：如果 v_1,v_2 都不是第二类点，那么可以通过 $G(v_1)$ 和 $G(v_2)$ 的公共点算出点 u 的信息。

情况 2：否则，v_1,v_2 一定恰有一个是第二类点，假设为 v_1，那么 v_2 是第三类点，$L(S(v_2))$ 和 $R(S(v_2))$ 一定恰有一个是可以匹配 $S(v_1)$ 的交替串。

如果 $|S(v_2)|>3$，那么 $L(v_2)$ 和 $R(v_2)$ 一定恰有一个交替串，v_2 本身也是这种情况 2 的形式，可以维护 v_2 的哪边是交替串，是如何交替的。

Solutions

对于一个第三类点 u，设 $G(u) = v_1, v_2$。

情况 1：如果 v_1, v_2 都不是第二类点，那么可以通过 $G(v_1)$ 和 $G(v_2)$ 的公共点算出点 u 的信息。

情况 2：否则，v_1, v_2 一定恰有一个是第二类点，假设为 v_1，那么 v_2 是第三类点，$L(S(v_2))$ 和 $R(S(v_2))$ 一定恰有一个是可以匹配 $S(v_1)$ 的交替串。

如果 $|S(v_2)| > 3$，那么 $L(v_2)$ 和 $R(v_2)$ 一定恰有一个交替串，v_2 本身也是这种情况 2 的形式，可以维护 v_2 的哪边是交替串，是如何交替的。

如果 $|S(v_2)| = 3$，直接维护 v_2 代表的串，暴力检查即可。

Solutions

对于一个第三类点 u，设 $G(u) = v_1, v_2$。

情况 1：如果 v_1, v_2 都不是第二类点，那么可以通过 $G(v_1)$ 和 $G(v_2)$ 的公共点算出点 u 的信息。

情况 2：否则，v_1, v_2 一定恰有一个是第二类点，假设为 v_1，那么 v_2 是第三类点，$L(S(v_2))$ 和 $R(S(v_2))$ 一定恰有一个是可以匹配 $S(v_1)$ 的交替串。

如果 $|S(v_2)| > 3$，那么 $L(v_2)$ 和 $R(v_2)$ 一定恰有一个交替串，v_2 本身也是这种情况 2 的形式，可以维护 v_2 的哪边是交替串，是如何交替的。

如果 $|S(v_2)| = 3$，直接维护 v_2 代表的串，暴力检查即可。

这样只用维护 $O(n)$ 的信息量即可。

Solutions

对于一个第三类点 u，设 $G(u) = v_1, v_2$。

情况 1：如果 v_1, v_2 都不是第二类点，那么可以通过 $G(v_1)$ 和 $G(v_2)$ 的公共点算出点 u 的信息。

情况 2：否则，v_1, v_2 一定恰有一个是第二类点，假设为 v_1，那么 v_2 是第三类点，$L(S(v_2))$ 和 $R(S(v_2))$ 一定恰有一个是可以匹配 $S(v_1)$ 的交替串。

如果 $|S(v_2)| > 3$，那么 $L(v_2)$ 和 $R(v_2)$ 一定恰有一个交替串，v_2 本身也是这种情况 2 的形式，可以维护 v_2 的哪边是交替串，是如何交替的。

如果 $|S(v_2)| = 3$，直接维护 v_2 代表的串，暴力检查即可。

这样只用维护 $O(n)$ 的信息量即可。

最后可以得到原串的两种可能的情况，分别修改字符集使得字典序最小，就可以得到答案了。总复杂度 $O(n)$。

Statements

给个二分图，划分成若干个子图，使得每个子图是长度不超过 3 的链，要求孤立点最少，然后最小化 3 的链的个数。

Statements

给个二分图，划分成若干个子图，使得每个子图是长度不超过 3 的链，要求孤立点最少，然后最小化 3 的链的个数。

$n_1, n_2 \leq 10^5, m \leq 2 \times 10^5$

Statements

给个二分图，划分成若干个子图，使得每个子图是长度不超过 3 的链，要求孤立点最少，然后最小化 3 的链的个数。

$n_1, n_2 \leq 10^5, m \leq 2 \times 10^5$

Solved by 18 teams. Submissions: 95. (Onsite + UCup)

Solution

首先考虑孤立点最少，注意到任意大于 1 的环或者链，一定能拆成若干个长度为 2 或者 3 的链。

Solution

首先考虑孤立点最少，注意到任意大于 1 的环或者链，一定能拆成若干个长度为 2 或者 3 的链。

所以孤立点最少，只要跑一个每个点流量不超过 2 的流。希望尽量多的点有非 0 的流量，那么可以考虑最小费用流。每个点向对应的源汇连两条流量为 1 的边，其中一条费用为 $-M$ （一个足够小的数即可），那么一定会优先增广这个费用小的。

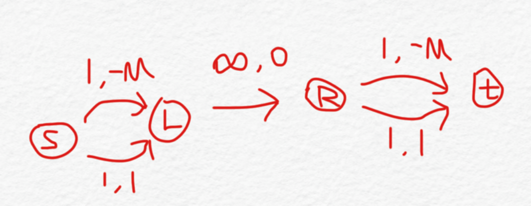
Solution

接下来，我们希望 2 度点尽量少，因为 2 度点等价于一条长度为 3 的链，所以对于剩下一条边，我们可以给一个 1 的费用。

Solution

接下来，我们希望 2 度点尽量少，因为 2 度点等价于一条长度为 3 的链，所以对于剩下一条边，我们可以给一个 1 的费用。

因为费用最小，所以会优先走所有 $-M$ 的边，然后尽量少走 1 的边。



Solution

直接使用常见的费用流做法是会超时的。

Solution

直接使用常见的费用流做法是会超时的。

注意到增广路的长度只可能是 $-2M, -M+1$，所以在求完最短路径图之后，使用 Dinic 等正确的最大流算法多路增广即可。

Solution

直接使用常见的费用流做法是会超时的。

注意到增广路的长度只可能是 $-2M, -M+1$，所以在求完最短路径图之后，使用 Dinic 等正确的最大流算法多路增广即可。

时间复杂度 $O(m\sqrt{n})$。

Solution

直接使用常见的费用流做法是会超时的。

注意到增广路的长度只可能是 $-2M, -M+1$, 所以在求完最短路径图之后，使用 Dinic 等正确的最大流算法多路增广即可。

时间复杂度 $O(m\sqrt{n})$。

这个题也存在一些不基于费用流的理解方法，最后都等价于若干次在这个图上的最大流问题。

Statements

一个串，每次操作会把极长的 $0/1$ 段减 1。
q 次询问，每次询问一个区间做 k 次操作后剩余几个字符。

Statements

一个串，每次操作会把极长的 $0/1$ 段减 1。
q 次询问，每次询问一个区间做 k 次操作后剩余几个字符。
$n, q \leq 5 \times 10^5$

Statements

一个串，每次操作会把极长的 0/1 段减 1。
q 次询问，每次询问一个区间做 k 次操作后剩余几个字符。
$n, q \leq 5 \times 10^5$
Solved by 7 teams. Submissions: 26. (Onsite + UCup)

Solutions

注意到区间询问和原串进行操作的关系

Solutions

注意到区间询问和原串进行操作的关系

实际上只差一个可能删掉的第一位，当我们对原串进行操作的时候，本来应该删除的 l 位置可能不会删除

Solutions

注意到区间询问和原串进行操作的关系

实际上只差一个可能删掉的第一位，当我们对原串进行操作的时候，本来应该删除的 l 位置可能不会删除

在维护原串进行 n 次操作的同时，对于所有操作维护当前次数的操作，目前的首位应该到了哪里，初始时为 l_i

Solutions

注意到区间询问和原串进行操作的关系

实际上只差一个可能删掉的第一位，当我们对原串进行操作的时候，本来应该删除的 l 位置可能不会删除

在维护原串进行 n 次操作的同时，对于所有操作维护当前次数的操作，目前的首位应该到了哪里，初始时为 l_i

当对原串操作一次,l_i 的变化则应该为要么这个地方已经被在原串的操作中删掉了，就往后移一位，否则本来就应该删掉还是往后移一位

Solutions

注意到区间询问和原串进行操作的关系

实际上只差一个可能删掉的第一位，当我们对原串进行操作的时候，本来应该删除的 l 位置可能不会删除

在维护原串进行 n 次操作的同时，对于所有操作维护当前次数的操作，目前的首位应该到了哪里，初始时为 l_i

当对原串操作一次,l_i 的变化则应该为要么这个地方已经被在原串的操作中删掉了，就往后移一位，否则本来就应该删掉还是往后移一位

Solutions

注意到标号是动态的，我们需要动态维护标号，原序列的标号很好维护，使用树状数组即可

Solutions

注意到标号是动态的，我们需要动态维护标号，原序列的标号很好维护，使用树状数组即可

但是操作序列的 l_i 实际上指的是当前剩下的数中的第几个，当我们要删除的时候，需要对标号大于一定值的数减去一，注意到初始到最后的标号相对顺序不发生改变，排序后使用线段树维护即可

Solutions

注意到标号是动态的，我们需要动态维护标号，原序列的标号很好维护，使用树状数组即可

但是操作序列的 l_i 实际上指的是当前剩下的数中的第几个，当我们要删除的时候，需要对标号大于一定值的数减去一，注意到初始到最后的标号相对顺序不发生改变，排序后使用线段树维护即可

而最后求答案只需要使用 r_i 的还剩下的前驱的当前标号减去维护的 l_i 当前首位的标号即可

Statements

给定三个非负整数 X, Y, K，你可以进行加一、减一、异或一个不超过 k 的非负整数，问 X 变成 Y 的最小操作次数。

$X, Y, K \le 10^{18}$

Statements

给定三个非负整数 X, Y, K，你可以进行加一、减一、异或一个不超过 k 的非负整数，问 X 变成 Y 的最小操作次数。

$X, Y, K \leq 10^{18}$

Solved by 149 teams. Submissions: 561. (Onsite + UCup)

Solutions

给你一个数 x，支持加一，减一，xor 一个不超过 K 的数，问变成 y 的最小步数。

Solutions

给你一个数 x，支持加一，减一，xor 一个不超过 K 的数，问变成 y 的最小步数。

考虑最小的 m 满足 $2^m > K$，那么 xor 只能改变二进制后 m 位。所以把所有数按照 $[i/2^m]$ 分组。

Solutions

给你一个数 x，支持加一，减一，xor 一个不超过 K 的数，问变成 y 的最小步数。

考虑最小的 m 满足 $2^m > K$，那么 xor 只能改变二进制后 m 位。所以把所有数按照 $[i/2^m]$ 分组。

对于同一组的两个数，如果相同，需要 0 步，如果 xor 不超过 K，或者差不超过 1，需要 1 步，否则需要两步。

Solutions

给你一个数 x，支持加一，减一，xor 一个不超过 K 的数，问变成 y 的最小步数。

考虑最小的 m 满足 $2^m > K$，那么 xor 只能改变二进制后 m 位。所以把所有数按照 $[i/2^m]$ 分组。

对于同一组的两个数，如果相同，需要 0 步，如果 xor 不超过 K，或者差不超过 1，需要 1 步，否则需要两步。

对于不同组之间的数，一定是先把末尾变成 m 个 1 ，然后加一到下一组，一直到同组为止。

Solutions

给你一个数 x，支持加一，减一，xor 一个不超过 K 的数，问变成 y 的最小步数。

考虑最小的 m 满足 $2^m > K$，那么 xor 只能改变二进制后 m 位。所以把所有数按照 $[i/2^m]$ 分组。

对于同一组的两个数，如果相同，需要 0 步，如果 xor 不超过 K，或者差不超过 1，需要 1 步，否则需要两步。

对于不同组之间的数，一定是先把末尾变成 m 个 1 ，然后加一到下一组，一直到同组为止。

对于每个组，如果 $K = 2^m - 1$，那么需要 2 步，否则需要 3 步。

Statements

给定一个 2 行 n 列的数组 a。每次你可以选择两列，把两列的最大值进行交换。问能不能从一个状态交换到另一个状态。

Statements

给定一个 2 行 n 列的数组 a。每次你可以选择两列，把两列的最大值进行交换。问能不能从一个状态交换到另一个状态。
Solved by 1 team. Submissions: 16. (Onsite + UCup)

Solutions

首先我们把 $a_{1,i}$ 和 $a_{2,i}$ 中较大的数置零，对 b 也进行同样的操作。

Solutions

首先我们把 $a_{1,i}$ 和 $a_{2,i}$ 中较大的数置零，对 b 也进行同样的操作。

可以证明，进行完这样的操作后 a, b 相同，那我们总能进行一些交换使得操作前的 a, b 也相同。

Solutions

首先我们把 $a_{1,i}$ 和 $a_{2,i}$ 中较大的数置零，对 b 也进行同样的操作。

可以证明，进行完这样的操作后 a, b 相同，那我们总能进行一些交换使得操作前的 a, b 也相同。

具体来说，我们可以使得 $\min(a_{1,i}, a_{2,i})$ 第 k 小的对应 max 第 k 小的。这一步可以简单地实现。

Solutions

首先我们把 $a_{1,i}$ 和 $a_{2,i}$ 中较大的数置零，对 b 也进行同样的操作。

可以证明，进行完这样的操作后 a, b 相同，那我们总能进行一些交换使得操作前的 a, b 也相同。

具体来说，我们可以使得 $\min(a_{1,i}, a_{2,i})$ 第 k 小的对应 max 第 k 小的。这一步可以简单地实现。

那么问题就变成如何让两个数组每一位的 min 做到对应相等。

Solutions

那么保留这些位置后，让我们判掉一些显然无解的情况：

Solutions

那么保留这些位置后，让我们判掉一些显然无解的情况：

1. 由于 min 只会单调变小，若 a 的 min 小于 b 的 min 则无解；

Solutions

那么保留这些位置后，让我们判掉一些显然无解的情况：

1. 由于 min 只会单调变小，若 a 的 min 小于 b 的 min 则无解；
2. 如果两数组对应位置 min 相等，且不在同侧，显然无解；

Solutions

那么保留这些位置后，让我们判掉一些显然无解的情况：

1. 由于 min 只会单调变小，若 a 的 min 小于 b 的 min 则无解；
2. 如果两数组对应位置 min 相等，且不在同侧，显然无解；
3. 如果两数组对应位置 min 在同侧，记 a 的为 m，b 的为 m'，如果 $m' = m - 1$，或者不存在中转物品 x 使得其重量在区间 (m', m) 内且该位置 a 与 b 的 min 不等，那么无解。

Solutions

那么保留这些位置后，让我们判掉一些显然无解的情况：

1. 由于 min 只会单调变小，若 a 的 min 小于 b 的 min 则无解；
2. 如果两数组对应位置 min 相等，且不在同侧，显然无解；
3. 如果两数组对应位置 min 在同侧，记 a 的为 m，b 的为 m'，如果 $m' = m - 1$，或者不存在中转物品 x 使得其重量在区间 (m', m) 内且该位置 a 与 b 的 min 不等，那么无解。

接下来我们每次选出对应 m' 最小的 m，将其直接归位或借助中转物品两次交换归位，可以归纳的证明该操作总能一直进行直到两数组对应位置的 min 完全相同。总操作次数为 $4n$，可以通过。

Statements

构造一个取值 ± 1 的积性函数，使得 $\sum_{i=1}^{n} f(i) = k$，$k \geq 0$。

Statements

构造一个取值 ± 1 的积性函数，使得 $\sum_{i=1}^{n} f(i) = k, \ k \geq 0$。
Solved by 80 teams. Submissions: 459. (Onsite + UCup)

Solution

首先等价于我们需要使得其中 $\frac{n+k}{2}$ 个 $f(i)$ 的取值是 1。

Solution

首先等价于我们需要使得其中 $\frac{n+k}{2}$ 个 $f(i)$ 的取值是 1。

积性函数由所有的素数的取值所决定。当我们确定 $\sqrt{n}$ 以下素数的取值的时候，剩下的素数对答案的贡献是独立的。

Solution

首先等价于我们需要使得其中 $\frac{n+k}{2}$ 个 $f(i)$ 的取值是 1。

积性函数由所有的素数的取值所决定。当我们确定 $\sqrt{n}$ 以下素数的取值的时候，剩下的素数对答案的贡献是独立的。

当 $n \geq 200$ 的时候，我们可以使得 $\sqrt{n}$ 以下的素数全都是 1，对于剩下的素数就是个背包问题。由于素数分布的性质，可以贪心。

Solution

首先等价于我们需要使得其中 $\frac{n+k}{2}$ 个 $f(i)$ 的取值是 1。

积性函数由所有的素数的取值所决定。当我们确定 $\sqrt{n}$ 以下素数的取值的时候，剩下的素数对答案的贡献是独立的。

当 $n \geq 200$ 的时候，我们可以使得 $\sqrt{n}$ 以下的素数全都是 1，对于剩下的素数就是个背包问题。由于素数分布的性质，可以贪心。

当 $n \leq 200$ 的时候，如果使得 $\sqrt{n}$ 以下的素数全都是 1，可能导致 1 的个数超过一半，这个时候可以使用搜索/随机化等各种方法对小素数赋值，预处理出答案解决。

Solution

首先等价于我们需要使得其中 $\frac{n+k}{2}$ 个 $f(i)$ 的取值是 1。

积性函数由所有的素数的取值所决定。当我们确定 $\sqrt{n}$ 以下素数的取值的时候，剩下的素数对答案的贡献是独立的。

当 $n \geq 200$ 的时候，我们可以使得 $\sqrt{n}$ 以下的素数全都是 1，对于剩下的素数就是个背包问题。由于素数分布的性质，可以贪心。

当 $n \leq 200$ 的时候，如果使得 $\sqrt{n}$ 以下的素数全都是 1，可能导致 1 的个数超过一半，这个时候可以使用搜索/随机化等各种方法对小素数赋值，预处理出答案解决。

本题也存在一些不需要特判，直接从小素数到大素数调整的做法，但正确性也不一定显然。

Statements

给一个表达式，按随机的优先级进行运算，问最后结果的期望。

Statements

给一个表达式，按随机的优先级进行运算，问最后结果的期望。

$n \leq 2 \times 10^5$

Statements

给一个表达式，按随机的优先级进行运算，问最后结果的期望。

$n \leq 2 \times 10^5$

Solved by 19 teams. Submissions: 53. (Onsite + UCup)

Solution

考虑每个数最后对答案的贡献。一个数 a_i 只有在运算的右边，并且运算符是减号的时候，符号会变反。

Solution

考虑每个数最后对答案的贡献。一个数 a_i 只有在运算的右边，并且运算符是减号的时候，符号会变反。

通过观察，只需要关注 $i-1$ 之前运算符优先级的后缀最小值即可。

Solution

考虑每个数最后对答案的贡献。一个数 a_i 只有在运算的右边，并且运算符是减号的时候，符号会变反。

通过观察，只需要关注 $i-1$ 之前运算符优先级的后缀最小值即可。

考虑这些符号的位置是 $p_1, p_2, \cdots, p_k$ $(p_k = i-1)$，那么概率为 $\frac{1}{i-1} \cdot \frac{1}{i-p_1-1} \cdot \dots \cdot \frac{1}{1-p_{k-1}-1}$。

Solution

考虑每个数最后对答案的贡献。一个数 a_i 只有在运算的右边，并且运算符是减号的时候，符号会变反。

通过观察，只需要关注 $i-1$ 之前运算符优先级的后缀最小值即可。

考虑这些符号的位置是 $p_1, p_2, \cdots, p_k$ $(p_k = i-1)$，那么概率为 $\frac{1}{i-1} \cdot \frac{1}{i-p_1-1} \cdot \ldots \cdot \frac{1}{1-p_{k-1}-1}$。

所以每个位置被选中的贡献为 $sgn_j/(i-j-1)$，未被选中的贡献为 1，整体贡献大概等于 $\frac{sgn[j]+i-j-1}{i-j-1}$。

Solution

我们需要计算 $a_1 + \sum_{i=2}^{n} \frac{a_i sgn_{i-1}}{(i-1)!} \prod_{j=1}^{i-1} (sgn_j + i - j - 1)$

Solution

我们需要计算 $a_1 + \sum_{i=2}^{n} \frac{a_i sgn_{i-1}}{(i-1)!} \prod_{j=1}^{i-1}(sgn_j + i - j - 1)$

接下来考虑如何优化这个表达式的计算。有一些方法可以通过这个题：

Solution

我们需要计算 $a_1 + \sum_{i=2}^{n} \frac{a_i sgn_{i-1}}{(i-1)!} \prod_{j=1}^{i-1}(sgn_j + i - j - 1)$
接下来考虑如何优化这个表达式的计算。有一些方法可以通过这个题:

- 注意到是一段 1 到 $i-2$ 的连乘，每一项都有 1 或者 -1 的偏移。所以可以用 Method of Four Russians 的方法，按 B 分段，并且预处理每一段的所有可能，可以在 $O(n^2/B + \frac{n}{B}2^B)$ 的复杂度内解决，需要一些精细的实现。

Solution

我们需要计算 $a_1 + \sum_{i=2}^{n} \frac{a_i sgn_{i-1}}{(i-1)!} \prod_{j=1}^{i-1}(sgn_j + i - j - 1)$

接下来考虑如何优化这个表达式的计算。有一些方法可以通过这个题:

- 注意到是一段 1 到 $i-2$ 的连乘，每一项都有 1 或者 -1 的偏移。所以可以用 Method of Four Russians 的方法，按 B 分段，并且预处理每一段的所有可能，可以在 $O(n^2/B + \frac{n}{B}2^B)$ 的复杂度内解决，需要一些精细的实现。
- 记 $c_j = j + 1 - sgn_j$。连乘可以看做 $\prod(i - cj)$。可以对这个式子做离散对数，并且记负数和 0 的离散对数为 0，那么问题等价于求 $\sum \ln(i - c_j)$，可以使用卷积解决。时间复杂度 $O(n\text{次离散对数} + n\log n)$。

Solution

记 $c_j = j + 1 - sgn_j$。问题可以看成每次先对 i 求值，然后乘上一个多项式 $(x - c_i)$。

Solution

记 $c_j = j + 1 - sgn_j$。问题可以看成每次先对 i 求值，然后乘上一个多项式 $(x - c_i)$。

注意到 $P(i) = P(x) \pmod{x - i} = [x^n]x^n P(1/x)/(1/x - i)$。

Solution

记 $c_j = j + 1 - sgn_j$。问题可以看成每次先对 i 求值，然后乘上一个多项式 $(x - c_i)$。

注意到 $P(i) = P(x) \pmod{x - i} = [x^n]x^n P(1/x)/(1/x - i)$。

所以原问题等价于多项式连乘，并且求出每一项除掉一个分式并乘以一个系数的和，可以用分治 FFT 解决。时间复杂度为