杭电多校题解

HDU MultiSchool Solution

2024 年 8 月 12 日

Table of contents

D

cats 的重力拼图

- 首先，特判 $n = 1$ 和 $m = 1$ 的情况，答案为 $n \cdot m$。

cats 的重力拼图

- 首先，特判 $n = 1$ 和 $m = 1$ 的情况，答案为 $n \cdot m$。
- 然后，先不考虑 142857 次操作的限制。容易发现物块到达重力拼图的四个角后将只能沿着拼图的最外圈移动。所以物块一定可以经过最外圈的所有 $(n + m - 2) \cdot 2$ 个方格。同时，由于执行两次方向垂直的操作后物块将到达四个角之一，物块在到达四个角之前只能执行一组左右或者一组上下以使物块初始位置所在的一行或者一列被额外经过。然后再经过一次垂直的操作到达四个角之一，再经过 4 次操作经过最外圈的所有方格。

cats 的重力拼图

- 所以可以进行分类讨论，如果 $a \in \{1, n\}$ 且 $b \in \{1, m\}$，则无法额外经过任何方格，答案为 $(n + m - 2) \cdot 2$。如果 $a \in \{1, n\}$ 且 $1 < b < m$，则可以额外经过一行 $m - 2$ 个方格，$(n + m - 2) \cdot 2 + m - 2$。如果 $1 < a < n$ 且 $b \in \{1, m\}$，则可以额外经过一列 $n - 2$ 个方格，$(n + m - 2) \cdot 2 + n - 2$。如果 $1 < a < n$ 且 $1 < b < m$，则既可以额外经过一行，也可以额外经过一列。答案为 $(n + m - 2) \cdot 2 + \max(n - 2, m - 2)$。

cats 的重力拼图

- 所以可以进行分类讨论，如果 $a \in \{1, n\}$ 且 $b \in \{1, m\}$，则无法额外经过任何方格，答案为 $(n + m - 2) \cdot 2$。如果 $a \in \{1, n\}$ 且 $1 < b < m$，则可以额外经过一行 $m - 2$ 个方格，$(n + m - 2) \cdot 2 + m - 2$。如果 $1 < a < n$ 且 $b \in \{1, m\}$，则可以额外经过一列 $n - 2$ 个方格，$(n + m - 2) \cdot 2 + n - 2$。如果 $1 < a < n$ 且 $1 < b < m$，则既可以额外经过一行，也可以额外经过一列。答案为 $(n + m - 2) \cdot 2 + \max(n - 2, m - 2)$。
- 由于经过以上过程最多需要 $2 + 1 + 4 = 7$ 次操作，不超过 142857 的限制。所以可以认为限制对答案没有影响，上述讨论的答案即为最终答案。

cats 的重力拼图

- 所以可以进行分类讨论，如果 $a \in \{1, n\}$ 且 $b \in \{1, m\}$，则无法额外经过任何方格，答案为 $(n + m - 2) \cdot 2$。如果 $a \in \{1, n\}$ 且 $1 < b < m$，则可以额外经过一行 $m - 2$ 个方格，$(n + m - 2) \cdot 2 + m - 2$。如果 $1 < a < n$ 且 $b \in \{1, m\}$，则可以额外经过一列 $n - 2$ 个方格，$(n + m - 2) \cdot 2 + n - 2$。如果 $1 < a < n$ 且 $1 < b < m$，则既可以额外经过一行，也可以额外经过一列。答案为 $(n + m - 2) \cdot 2 + \max(n - 2, m - 2)$。
- 由于经过以上过程最多需要 $2 + 1 + 4 = 7$ 次操作，不超过 142857 的限制。所以可以认为限制对答案没有影响，上述讨论的答案即为最终答案。
- 对于每组测试数据，总时间复杂度为 $O(1)$，空间复杂度为 $O(1)$。

G

cats 的 k-xor

- 首先特判 $a+b=c$ 为 -1，$a+b<c$ 为 0。对于 $a+b>c$ 的情况：

cats 的 k-xor

- 首先特判 $a+b=c$ 为 -1，$a+b<c$ 为 0。对于 $a+b>c$ 的情况：
- 解法 1：暴力枚举所有 $k \le \sqrt{a+b}$ 的 k，检查是否符合要求。当 $k > \sqrt{a+b}$ 时，由于只有最低位会发生进位，k 只能为 $a+b-c$。检查是否符合要求即可。

cats 的 k-xor

- 首先特判 $a+b=c$ 为 -1，$a+b<c$ 为 0。对于 $a+b>c$ 的情况：
- 解法 1：暴力枚举所有 $k \le \sqrt{a+b}$ 的 k，检查是否符合要求。当 $k > \sqrt{a+b}$ 时，由于只有最低位会发生进位，k 只能为 $a+b-c$。检查是否符合要求即可。
- 对于每组测试数据，总时间复杂度为 $O(\sqrt{a+b} \cdot \log(a+b))$，空间复杂度为 $O(1)$。

cats 的 k-xor

- 首先特判 $a+b=c$ 为 -1，$a+b<c$ 为 0。对于 $a+b>c$ 的情况：
- 解法 1：暴力枚举所有 $k \le \sqrt{a+b}$ 的 k，检查是否符合要求。当 $k > \sqrt{a+b}$ 时，由于只有最低位会发生进位，k 只能为 $a+b-c$。检查是否符合要求即可。
- 对于每组测试数据，总时间复杂度为 $O(\sqrt{a+b} \cdot \log(a+b))$，空间复杂度为 $O(1)$。
- 解法 2：由于 $a+b-c$ 一定是 k 的倍数。检查 $a+b-c$ 的所有因子是否符合要求即可。

cats 的 k-xor

- 首先特判 $a+b=c$ 为 -1，$a+b<c$ 为 0。对于 $a+b>c$ 的情况：
- 解法 1：暴力枚举所有 $k \le \sqrt{a+b}$ 的 k，检查是否符合要求。当 $k > \sqrt{a+b}$ 时，由于只有最低位会发生进位，k 只能为 $a+b-c$。检查是否符合要求即可。
- 对于每组测试数据，总时间复杂度为 $O(\sqrt{a+b} \cdot \log(a+b))$，空间复杂度为 $O(1)$。
- 解法 2：由于 $a+b-c$ 一定是 k 的倍数。检查 $a+b-c$ 的所有因子是否符合要求即可。
- 对于每组测试数据，总时间复杂度为 $O(\sqrt{a+b-c} + d(a+b-c) \cdot \log(a+b))$，其中 $d(x)$ 表示 x 的因数个数，空间复杂度为 $O(1)$。

cats 的 k-xor

- 首先特判 $a+b=c$ 为 -1，$a+b<c$ 为 0。对于 $a+b>c$ 的情况：
- 解法 1：暴力枚举所有 $k \le \sqrt{a+b}$ 的 k，检查是否符合要求。当 $k > \sqrt{a+b}$ 时，由于只有最低位会发生进位，k 只能为 $a+b-c$。检查是否符合要求即可。
- 对于每组测试数据，总时间复杂度为 $O(\sqrt{a+b} \cdot \log(a+b))$，空间复杂度为 $O(1)$。
- 解法 2：由于 $a+b-c$ 一定是 k 的倍数。检查 $a+b-c$ 的所有因子是否符合要求即可。
- 对于每组测试数据，总时间复杂度为 $O(\sqrt{a+b-c} + d(a+b-c) \cdot \log(a+b))$，其中 $d(x)$ 表示 x 的因数个数，空间复杂度为 $O(1)$。
- bonus：改变数据范围限制为 $T=1$，$0 \le a,b,c \le 10^{36}$，解决此问题。

L

cats 的电脑中毒

- 我们将两个 01 串间的距离定义为两个字符串不一样的位数，一个节点被感染的时间即为其与 S_1, S_2, S_3 距离的最小值，我们需要找到一个字符串最大化这个最小值。

cats 的电脑中毒

- 我们将两个 01 串间的距离定义为两个字符串不一样的位数，一个节点被感染的时间即为其与 S_1, S_2, S_3 距离的最小值，我们需要找到一个字符串最大化这个最小值。
- 考虑对字符串每一位分别考虑，将所有位分为 4 类：

  1. $S_{1,i} = S_{2,i} = S_{3,i}$。此时一定贪心的让构造的字符串这一位和这三个字符串不一样，答案直接加 1。
  2. $S_{2,i} = S_{3,i} \neq S_{1,i}$。设这种位置共有 x 个。
  3. $S_{1,i} = S_{3,i} \neq S_{2,i}$。设这种位置共有 y 个。
  4. $S_{1,i} = S_{2,i} \neq S_{3,i}$。设这种位置共有 z 个。

cats 的电脑中毒

- 我们将两个 01 串间的距离定义为两个字符串不一样的位数，一个节点被感染的时间即为其与 S_1, S_2, S_3 距离的最小值，我们需要找到一个字符串最大化这个最小值。
- 考虑对字符串每一位分别考虑，将所有位分为 4 类：

  1. $S_{1,i} = S_{2,i} = S_{3,i}$。此时一定贪心的让构造的字符串这一位和这三个字符串不一样，答案直接加 1。
  2. $S_{2,i} = S_{3,i} \neq S_{1,i}$。设这种位置共有 x 个。
  3. $S_{1,i} = S_{3,i} \neq S_{2,i}$。设这种位置共有 y 个。
  4. $S_{1,i} = S_{2,i} \neq S_{3,i}$。设这种位置共有 z 个。

- 由于交换这三个字符串对答案没有影响，不妨设 $x \leq y \leq z$。

- 我们将两个 01 串间的距离定义为两个字符串不一样的位数，一个节点被感染的时间即为其与 S_1, S_2, S_3 距离的最小值，我们需要找到一个字符串最大化这个最小值。
- 考虑对字符串每一位分别考虑，将所有位分为 4 类：

  1. $S_{1,i} = S_{2,i} = S_{3,i}$。此时一定贪心的让构造的字符串这一位和这三个字符串不一样，答案直接加 1。
  2. $S_{2,i} = S_{3,i} \neq S_{1,i}$。设这种位置共有 x 个。
  3. $S_{1,i} = S_{3,i} \neq S_{2,i}$。设这种位置共有 y 个。
  4. $S_{1,i} = S_{2,i} \neq S_{3,i}$。设这种位置共有 z 个。

- 由于交换这三个字符串对答案没有影响，不妨设 $x \leq y \leq z$。
- 我们将构造的字符串与 S_1, S_2, S_3 的距离分别称为 d_1, d_2, d_3，则可以看做，将 d_1, d_2, d_3 初始化为 0，进行 x 次操作使 d_2, d_3 增大 1，或使 d_1 增大 1，其余两种操作同理。

cats 的电脑中毒

- 我们先假设每种操作都让两个数增大 1，则 d_1, d_2, d_3 分别为 $y+z$，$x+z$，$x+y$。而反悔其中的一个操作相当于给其中两个数减 1，给另外一个数加 1。显然最多反悔其中一种操作，因为反悔两种不同的操作各一次会使 d_1, d_2, d_3 其中一者减 2，而其余两者无法增加，一定不优。

cats 的电脑中毒

- 我们先假设每种操作都让两个数增大 1，则 d_1, d_2, d_3 分别为 $y+z$，$x+z$，$x+y$。而反悔其中的一个操作相当于给其中两个数减 1，给另外一个数加 1。显然最多反悔其中一种操作，因为反悔两种不同的操作各一次会使 d_1, d_2, d_3 其中一者减 2，而其余两者无法增加，一定不优。
- 由于 $y+z \geq x+z \geq x+y$，为了使三者最小值最大，可以贪心的反悔 $\lfloor \frac{z-y}{2} \rfloor$ 次使 d_1, d_2 增大 1 的操作让 $\min(d_1, d_2, d_3)$ 达到最大，最大值即为 $x+y+\lfloor \frac{z-y}{2} \rfloor = x+\lfloor \frac{y+z}{2} \rfloor$。

cats 的电脑中毒

- 我们先假设每种操作都让两个数增大 1，则 d_1, d_2, d_3 分别为 $y+z$，$x+z$，$x+y$。而反悔其中的一个操作相当于给其中两个数减 1，给另外一个数加 1。显然最多反悔其中一种操作，因为反悔两种不同的操作各一次会使 d_1, d_2, d_3 其中一者减 2，而其余两者无法增加，一定不优。
- 由于 $y+z \geq x+z \geq x+y$，为了使三者最小值最大，可以贪心的反悔 $\lfloor \frac{z-y}{2} \rfloor$ 次使 d_1, d_2 增大 1 的操作让 $\min(d_1, d_2, d_3)$ 达到最大，最大值即为 $x+y+\lfloor \frac{z-y}{2} \rfloor = x+\lfloor \frac{y+z}{2} \rfloor$。
- 枚举字符串的每一位，计算出直接使答案加 1 的次数和 x, y, z 的值即可得到答案。

cats 的电脑中毒

- 我们先假设每种操作都让两个数增大 1，则 d_1, d_2, d_3 分别为 $y+z$，$x+z$，$x+y$。而反悔其中的一个操作相当于给其中两个数减 1，给另外一个数加 1。显然最多反悔其中一种操作，因为反悔两种不同的操作各一次会使 d_1, d_2, d_3 其中一者减 2，而其余两者无法增加，一定不优。
- 由于 $y+z \geq x+z \geq x+y$，为了使三者最小值最大，可以贪心的反悔 $\lfloor \frac{z-y}{2} \rfloor$ 次使 d_1, d_2 增大 1 的操作让 $\min(d_1, d_2, d_3)$ 达到最大，最大值即为 $x+y+\lfloor \frac{z-y}{2} \rfloor = x+\lfloor \frac{y+z}{2} \rfloor$。
- 枚举字符串的每一位，计算出直接使答案加 1 的次数和 x, y, z 的值即可得到答案。
- 对于每组测试数据，总时间复杂度为 $O(n)$，空间复杂度为 $O(n)$。

E

cats 的二分答案

- 题目相当于求有多少的 n 在二分过程中 $mid > n$ 的次数不超过 k。

cats 的二分答案

- 题目相当于求有多少的 n 在二分过程中 $mid > n$ 的次数不超过 k。
- 首先，二分的过程不会超过 $\log_2 n$ 次（在这里实际上限为 60 次），故有意义的 k 只有 $0 \sim 60$。同时可以证明在 k 次二分后二分的区间长度（$r - l + 1$）只有最多两种，且差最多为 1。那么就可以对于两种长度的区间都维护一个 $f_{i,j}$ 表示在 i 次二分后，使用了 j 次越界的访问的方案数。暴力转移即可。总复杂度 $O(\log^2 r)$，空间复杂度 $O(\log^2 r)$。

cats 的二分答案

- 题目相当于求有多少的 n 在二分过程中 $mid > n$ 的次数不超过 k。
- 首先，二分的过程不会超过 $\log_2 n$ 次（在这里实际上限为 60 次），故有意义的 k 只有 $0 \sim 60$。同时可以证明在 k 次二分后二分的区间长度（$r-l+1$）只有最多两种，且差最多为 1。那么就可以对于两种长度的区间都维护一个 $f_{i,j}$ 表示在 i 次二分后，使用了 j 次越界的访问的方案数。暴力转移即可。总复杂度 $O(\log^2 r)$，空间复杂度 $O(\log^2 r)$。
- 开始时 $T \le 10^5$，不过在验题时有很多队使用了 `map`，最终决定放宽限制，让使用 `map` 的 $O(\log^2 r \log\log r)$ 也能通过本题。

H

cats 的数据结构

- 给出一棵树，树上每个节点有两个权值 a, b，构造一个字典序最小的方案满足只有两个点满足祖先关系时才能有二维偏序关系，求字典序最小的构造方案。

cats 的数据结构

- 给出一棵树，树上每个节点有两个权值 a, b，构造一个字典序最小的方案满足只有两个点满足祖先关系时才能有二维偏序关系，求字典序最小的构造方案。
- 下面给出 a_i, b_i 为排列的情况，对于任意构造离散化后与排列的构造等价，且离散化后字典序必然不增。

cats 的数据结构

- 给出一棵树，树上每个节点有两个权值 a, b，构造一个字典序最小的方案满足只有两个点满足祖先关系时才能有二维偏序关系，求字典序最小的构造方案。
- 下面给出 a_i, b_i 为排列的情况，对于任意构造离散化后与排列的构造等价，且离散化后字典序必然不增。
- 下面证明 a_i, b_i 分别对应两个 DFS 序 $dfn1_i, dfn_2$ 满足 $a_i = n + 1 - dfn1_i, b_i = n + 1 - dfn2_i$。

cats 的数据结构

- 给出一棵树，树上每个节点有两个权值 a, b，构造一个字典序最小的方案满足只有两个点满足祖先关系时才能有二维偏序关系，求字典序最小的构造方案。
- 下面给出 a_i, b_i 为排列的情况，对于任意构造离散化后与排列的构造等价，且离散化后字典序必然不增。
- 下面证明 a_i, b_i 分别对应两个 DFS 序 $dfn1_i, dfn_2$ 满足 $a_i = n + 1 - dfn1_i, b_i = n + 1 - dfn2_i$。
- 考虑节点 x, y, z，若 x 是 y 的祖先，z 在以 x 为根的子树外。如果 $a_x > a_z > a_y$ 那么根据 z 在 x, y 的子树外可得 $b_y > b_z > b_x$，且 z 不为 x 祖先，与 x 是 y 祖先矛盾。可得 a_i, b_i 分别对应两个 DFS 序 $dfn1_i, dfn_2$ 满足 $a_i = n + 1 - dfn1_i, b_i = n + 1 - dfn2_i$ 且遍历方式中对于每个节点的子节点的遍历顺序相反（这样可以保证当 a, b 不会在非祖先关系下形成偏序关系）。

cats 的数据结构

- 由上可得，可以贪心地考虑 a_i 的值，让编号更小的节点得到一个较大的 dfn，从而有更小的字典序。所以在遍历 a_i 对应的 DFS 序时按子树中编号最小的节点从大到小遍历即可。确定了 a_i 的遍历顺序同样可以得到 b_i 的遍历顺序。总复杂度 $O(n \log n)$。

F

cats 的最小生成树

- 解法 1：对 n 个点开 $\lfloor \frac{m}{n-1} \rfloor$ 层并查集，从下到上第 i 层并查集表示第 i 次删除最小生成树前图的连通状态。

cats 的最小生成树

- 解法 1：对 n 个点开 $\lfloor \frac{m}{n-1} \rfloor$ 层并查集，从下到上第 i 层并查集表示第 i 次删除最小生成树前图的连通状态。
- 从小到大枚举所有边维护并查集，每次连接 (u, v) 时，二分找到最低层 u 和 v 不连通的并查集（如果存在），将其合并即可。边被删除的操作次数即为这个层数。

cats 的最小生成树

- 解法 1：对 n 个点开 $\lfloor \frac{m}{n-1} \rfloor$ 层并查集，从下到上第 i 层并查集表示第 i 次删除最小生成树前图的连通状态。
- 从小到大枚举所有边维护并查集，每次连接 (u, v) 时，二分找到最低层 u 和 v 不连通的并查集（如果存在），将其合并即可。边被删除的操作次数即为这个层数。
- 插入全部 m 个边后根据并查集中最高的所有 n 个点被合并为一个点的层数确定实际操作次数，将所有连边层数超过实际操作次数的边的删除次数设为 -1 即可。

cats 的最小生成树

- 解法 1：对 n 个点开 $\lfloor \frac{m}{n-1} \rfloor$ 层并查集，从下到上第 i 层并查集表示第 i 次删除最小生成树前图的连通状态。
- 从小到大枚举所有边维护并查集，每次连接 (u, v) 时，二分找到最低层 u 和 v 不连通的并查集（如果存在），将其合并即可。边被删除的操作次数即为这个层数。
- 插入全部 m 个边后根据并查集中最高的所有 n 个点被合并为一个点的层数确定实际操作次数，将所有连边层数超过实际操作次数的边的删除次数设为 -1 即可。
- 对于每组测试数据，总时间复杂度为 $O(m \cdot \log(m) \cdot \alpha(n))$。总空间复杂度为 $O(m)$。

cats 的最小生成树

- 解法 2：当图中存在重边时，只有其中权值最低的会对生成树产生影响。可以用 vector 维护每一对点间所有边的权值，每次取出每一对点之间权值最小的边，用并查集求出最小生成树。总边数不超过 $\min(m, n^2)$，删除次数不超过 $\lfloor \frac{m}{n-1} \rfloor$。

cats 的最小生成树

- 解法 2：当图中存在重边时，只有其中权值最低的会对生成树产生影响。可以用 vector 维护每一对点间所有边的权值，每次取出每一对点之间权值最小的边，用并查集求出最小生成树。总边数不超过 $\min(m, n^2)$，删除次数不超过 $\lfloor \frac{m}{n-1} \rfloor$。
- 对于每组测试数据，总时间复杂度为 $O(\min(m, n^2) \cdot \frac{m}{n} \cdot \alpha(n)) = O(min(\frac{m^2}{n}, m \cdot n) \cdot \alpha(n))$。取 $n = \sqrt{m}$ 时上述式子达到最大，即为 $O(m \cdot \sqrt{m} \cdot \alpha(m))$。总空间复杂度为 $O(m)$。

C

cats 的飞机坠毁

- 首先把每个飞机看成一个括号，向左飞的飞机看做)，向右飞的飞机看做(，那么所有飞机飞行方向的每一种可能的决定情况都可以看做一个括号序列，而发生匹配的括号就是一对发生坠毁的飞机。定义一个极短合法括号序列为序列最左侧的(与最右侧的)匹配的合法括号序列，设一个括号序列中所有极短合法括号子串长度的最大值为 d，则最后一次飞机坠毁发生的时间即为 $\frac{d-1}{2}$。

cats 的飞机坠毁

- 首先把每个飞机看成一个括号，向左飞的飞机看做)，向右飞的飞机看做(，那么所有飞机飞行方向的每一种可能的决定情况都可以看做一个括号序列，而发生匹配的括号就是一对发生坠毁的飞机。定义一个极短合法括号序列为序列最左侧的(与最右侧的)匹配的合法括号序列，设一个括号序列中所有极短合法括号子串长度的最大值为 d，则最后一次飞机坠毁发生的时间即为 $\frac{d-1}{2}$。
- 首先对于每个区间求出它是一个极短合法括号子串概率。这可以通过区间 dp 或者枚举左端点 l，向右 dp 前缀未匹配的(个数两种方式在 $O(n^3)$ 时间内求出。

cats 的飞机坠毁

- 然后对于每一种可能的 d，求出序列中所有极短合法括号子串长度均不超过 d 的概率。由于整个括号序列一定由若干极短合法括号子串和一些单独的未匹配的括号组成，而所有未匹配的括号一定一个前缀为)，后缀为(。所以可以将 i 个括号，其中未匹配括号是否已经从) 转变为 (做一个时间复杂度为 $O(n^2)$ 的 dp。由于 d 总共只有 $O(n)$ 种，所以这一部分的时间复杂度为 $O(n^3)$。

cats 的飞机坠毁

- 然后对于每一种可能的 d，求出序列中所有极短合法括号子串长度均不超过 d 的概率。由于整个括号序列一定由若干极短合法括号子串和一些单独的未匹配的括号组成，而所有未匹配的括号一定一个前缀为)，后缀为(。所以可以将 i 个括号，其中未匹配括号是否已经从) 转变为 (做一个时间复杂度为 $O(n^2)$ 的 dp。由于 d 总共只有 $O(n)$ 种，所以这一部分的时间复杂度为 $O(n^3)$。
- 最后，对上述求出的结果进行差分，即可得到对于每个 d，序列中所有极短合法括号子串长度最大值为 d 的概率。直接计算期望即可。

cats 的飞机坠毁

- 然后对于每一种可能的 d，求出序列中所有极短合法括号子串长度均不超过 d 的概率。由于整个括号序列一定由若干极短合法括号子串和一些单独的未匹配的括号组成，而所有未匹配的括号一定一个前缀为)，后缀为(。所以可以将 i 个括号，其中未匹配括号是否已经从) 转变为 (做一个时间复杂度为 $O(n^2)$ 的 dp。由于 d 总共只有 $O(n)$ 种，所以这一部分的时间复杂度为 $O(n^3)$。
- 最后，对上述求出的结果进行差分，即可得到对于每个 d，序列中所有极短合法括号子串长度最大值为 d 的概率。直接计算期望即可。
- 对于每组测试数据，总时间复杂度为 $O(n^3)$，空间复杂度为 $O(n^2)$。

K

cats 的集合 2

- Hint 1：异或运算具有封闭性。对于自然数 x 和 k，如果 $x \in [0, 2^k)$，那么 x 异或上 $[0, 2^k)$ 中的任意一个数，得到的结果仍然在 $[0, 2^k)$ 中。

cats 的集合 2

- Hint 1：异或运算具有封闭性。对于自然数 x 和 k，如果 $x \in [0, 2^k)$，那么 x 异或上 $[0, 2^k)$ 中的任意一个数，得到的结果仍然在 $[0, 2^k)$ 中。
- Hint 2：异或运算满足消去律。对于自然数 x, a, b 和 k，如果 $x, a, b \in [0, 2^k)$，且 $x \operatorname{xor} a = x \operatorname{xor} b$，那么一定有 $a = b$。

cats 的集合 2

- Hint 1：异或运算具有封闭性。对于自然数 x 和 k，如果 $x \in [0, 2^k)$，那么 x 异或上 $[0, 2^k)$ 中的任意一个数，得到的结果仍然在 $[0, 2^k)$ 中。
- Hint 2：异或运算满足消去律。对于自然数 x, a, b 和 k，如果 $x, a, b \in [0, 2^k)$，且 $x \operatorname{xor} a = x \operatorname{xor} b$，那么一定有 $a = b$。
- Hint 3：给定自然数 x 和 k，如果 $x \in [0, 2^k)$，那么 $[0, 2^k)$ 中的每个数异或上 x 后得到的这些数各不相同且均在 $[0, 2^k)$ 之间。

cats 的集合 2

- 如果每个数的取值范围很小，那么直接做异或卷积就可以解决这个问题，考虑如何利用每个数的取值范围连续来加速卷积的过程。假设我们现在有两个数，取值范围分别是 $[0, a_1)$ 和 $[0, a_2)$ 。先考虑第一个数，可以将长度为 $a_1 + 1$ 的 $[0, a_1)$ 拆分成最多 $\log_2(a_1 + 1)$ 个极大的长度为 2 的幂次取值范围。例如当 $a_1 = 7$ 的时候，可以分成 $[0, 4)$ ，$[4, 6)$ 和 $[6, 7)$ 这三段。同理，$[0, a_2)$ 也可以被拆分成若干个这样的段。

cats 的集合 2

- 对于拆分出的任意两个段，它们的异或卷积是可以快速求出的。假设两个段的长度分别为 2^{k_1} 和 2^{k_2} 且 $\max(k_1, k_2) = k_1$，那么第一个段中所有的数从第 k_1 位开始都是一样。例如 $[4, 6)$ 的二进制表示是 $100, 101$，从第 1 位开始都是 10。第二个段同理，所有数从第 k_2 位开始是一样的，所以说这两个段进行异或卷积后从较高的位，也就是第 k_1 位开始，所有的位都是可以确定的。剩余的位恰好构成了从 $[0, 2^{k_1})$ 的这一段，所以这两个段异或卷积后得到了一个长度为 2^{k_1} 的段，且段中的每个数出现了 2^{k_2} 次。

cats 的集合 2

- 这样之后，我们可以快速得到两个段的异或卷积，直接模拟这个过程是 $O(\log^2 V)$ 的，其中 V 是值域。枚举卷积过程中出现过最长的段是 2^k，在转移的过程中只取出长度小于等于 2^k 的段，注意这样做从 $k+1$ 位开始才是确定的，因为第 k 位是可以改变的。

cats 的集合 2

- 这样之后，我们可以快速得到两个段的异或卷积，直接模拟这个过程是 $O(\log^2 V)$ 的，其中 V 是值域。枚举卷积过程中出现过最长的段是 2^k，在转移的过程中只取出长度小于等于 2^k 的段，注意这样做从 $k+1$ 位开始才是确定的，因为第 k 位是可以改变的。
- 利用前缀和优化动态规划即可在 $O(\log V)$ 的时间复杂度下完成整个集合的异或卷积，注意需要容斥一下保证至少出现过一次长度为 2^k 的段。对于一组给定的 $[L, R]$，只需要对于每一个长度为 2^k 的段 $[A, A+2^k)$ 和它取一个交集即可。总时间复杂度为 $O(n \log V)$。

I

cats 的凸包计算

- 暴力枚举所有可能是凸包边界的边。对于上凸壳的边，其对面积的贡献为其与其在 x 轴上投影形成的梯形的面积，对于下凸壳的边，其对面积的贡献为其与其在 x 轴上投影形成的梯形的面积的相反数。计算每条边作为凸包边界的次数，与它的贡献相乘并求和即为答案。

cats 的凸包计算

- 暴力枚举所有可能是凸包边界的边。对于上凸壳的边，其对面积的贡献为其与其在 x 轴上投影形成的梯形的面积，对于下凸壳的边，其对面积的贡献为其与其在 x 轴上投影形成的梯形的面积的相反数。计算每条边作为凸包边界的次数，与它的贡献相乘并求和即为答案。
- 对于上凸壳的每一条可能的边，它作为凸包边界存在的条件即为：
  1. 两个端点存在。
  2. 且所有其他点都在这条边所在直线下方或直线上。
  3. 所有处在直线上的点一定存在于这条边的线段上（此处我们认为凸包边上的点不为凸包的顶点）。

cats 的凸包计算

- 如果直线斜率为正，则从左到右由于前一个位置可选填入的数的集合一定是后一个位置的子集，直接从左到右确定每个位置填入的数计算方案数即可。如果斜率为负，同理可以从右到左确定。

cats 的凸包计算

- 如果直线斜率为正，则从左到右由于前一个位置可选填入的数的集合一定是后一个位置的子集，直接从左到右确定每个位置填入的数计算方案数即可。如果斜率为负，同理可以从右到左确定。
- 对于下凸壳的每一条边，也可以用一样的方法求出其出现次数。

cats 的凸包计算

- 如果直线斜率为正，则从左到右由于前一个位置可选填入的数的集合一定是后一个位置的子集，直接从左到右确定每个位置填入的数计算方案数即可。如果斜率为负，同理可以从右到左确定。
- 对于下凸壳的每一条边，也可以用一样的方法求出其出现次数。
- 对于每组测试数据，总时间复杂度为 $O(n^5)$，空间复杂度为 $O(n)$。

A

- 首先假设没有双端队列的限制，忽略 $a_i = b_i = 0$ 的题目，那么对于每一对题目 (a_i, b_i) 和 (a_j, b_j)，先做第一个题后做第二个题产生的额外收益为 $a_i \cdot b_j$。而先做第二个题后做第一个题产生的额外收益为 $a_j \cdot b_i$。所以如果 $a_i \cdot b_j - a_j \cdot b_i > 0$，即二维平面上向量 (a_i, b_i) 与 x 轴夹角比 (a_j, b_j) 小，那么先做第一个题的收益更高，反之则先做第二个题的收益更高。由此可得在没有双端队列限制的条件下，按照向量 (a_i, b_i) 与 x 轴夹角从小到大做题一定是最优的。

- 首先假设没有双端队列的限制，忽略 $a_i = b_i = 0$ 的题目，那么对于每一对题目 (a_i, b_i) 和 (a_j, b_j)，先做第一个题后做第二个题产生的额外收益为 $a_i \cdot b_j$。而先做第二个题后做第一个题产生的额外收益为 $a_j \cdot b_i$。所以如果 $a_i \cdot b_j - a_j \cdot b_i > 0$，即二维平面上向量 (a_i, b_i) 与 x 轴夹角比 (a_j, b_j) 小，那么先做第一个题的收益更高，反之则先做第二个题的收益更高。由此可得在没有双端队列限制的条件下，按照向量 (a_i, b_i) 与 x 轴夹角从小到大做题一定是最优的。
- 在后文中我们将向量夹角更小的题目称作性价比高，将夹角更大的题目称作性价比低。

- 现在考虑双端队列的限制。由于这个双端队列从左侧取出的所有题一定是一个前缀，而从右侧取出的是剩余的后缀。所以我们可以枚举这个分界点，将双端队列转化为两个栈。

- 现在考虑双端队列的限制。由于这个双端队列从左侧取出的所有题一定是一个前缀，而从右侧取出的是剩余的后缀。所以我们可以枚举这个分界点，将双端队列转化为两个栈。
- 然后，如果我们钦定某个栈中的一段连续区间中的题一定被连续取出(其中不会插入另一个栈中的题)，那么这一段连续的题可以视为合并成一个新的题。其 a 和 b 为区间中所有题目的 a 和 b 之和，并在合并时产生一个固定的额外收益。这个合并的过程不具有交换律，但具有结合律，即无关合并顺序的影响。

- 现在考虑双端队列的限制。由于这个双端队列从左侧取出的所有题一定是一个前缀，而从右侧取出的是剩余的后缀。所以我们可以枚举这个分界点，将双端队列转化为两个栈。
- 然后，如果我们钦定某个栈中的一段连续区间中的题一定被连续取出(其中不会插入另一个栈中的题)，那么这一段连续的题可以视为合并成一个新的题。其 a 和 b 为区间中所有题目的 a 和 b 之和，并在合并时产生一个固定的额外收益。这个合并的过程不具有交换律，但具有结合律，即无关合并顺序的影响。
- 接下来，如果某个栈中存在一个性价比低的题 P 在一个性价比高的题 Q 的前面，那么这两个题中间一定不会插入其他题目。

- 证明：假设其中插入了其他题目。如果其中插入了多个题目则可以按照上面的过程将其视为一个题目。如果这个题性价比高于 P，则将其与 P 交换则更优。如果这个题性价比低于 Q，则将其与 Q 交换则更优。而由于 P 性价比低于 Q，以上二者至少有一者满足。所以将这个题目插入 P, Q 中间一定不是最优。

- 证明：假设其中插入了其他题目。如果其中插入了多个题目则可以按照上面的过程将其视为一个题目。如果这个题性价比高于 P，则将其与 P 交换则更优。如果这个题性价比低于 Q，则将其与 Q 交换则更优。而由于 P 性价比低于 Q，以上二者至少有一者满足。所以将这个题目插入 P, Q 中间一定不是最优。
- 所以我们每遇到一个性价比低的题在一个性价比高的题的前面的结构，都可以通过题目合并的过程将其合并为一个题。

- 证明：假设其中插入了其他题目。如果其中插入了多个题目则可以按照上面的过程将其视为一个题目。如果这个题性价比高于 P，则将其与 P 交换则更优。如果这个题性价比低于 Q，则将其与 Q 交换则更优。而由于 P 性价比低于 Q，以上二者至少有一者满足。所以将这个题目插入 P, Q 中间一定不是最优。
- 所以我们每遇到一个性价比低的题在一个性价比高的题的前面的结构，都可以通过题目合并的过程将其合并为一个题。
- 所以我们可以从前往后和从后往前分别维护一个性价比从高到低的单调栈，每次插入一个题目时如果其性价比高于栈顶，就不断与栈顶题目合并直到性价比低于栈顶题目或者栈为空为止。此时对于我们枚举的每个分界位置，由于两侧的栈中的题目一定是性价比单调不增的，所以可以直接贪心按性价比从高到低取出所有题目。

- 由于两侧的栈中总共最多存在 $O(n)$ 种不同的题目，所以我们可以提前预处理出所有曾经在栈中出现的题目，按性价比从高到低排序后，维护所有当前栈中存在的题目的 $\sum_{i<j} a_i \cdot b_j$ 即为当前的额外收益。记录每个题出现和消失的时间并用树状数组维护即可。

- 由于两侧的栈中总共最多存在 $O(n)$ 种不同的题目，所以我们可以提前预处理出所有曾经在栈中出现的题目，按性价比从高到低排序后，维护所有当前栈中存在的题目的 $\sum_{i<j} a_i \cdot b_j$ 即为当前的额外收益。记录每个题出现和消失的时间并用树状数组维护即可。
- 对于每组测试数据，总时间复杂度为 $O(n \cdot \log(n))$，空间复杂度为 $O(n)$。

J

cats 的集合 1

- 维护一个集合，支持加入元素，将所有元素或/与/异或上一个值，查询集合中的元素和某个数异或的最大值。

cats 的集合 1

- 维护一个集合，支持加入元素，将所有元素或/与/异或上一个值，查询集合中的元素和某个数异或的最大值。
- 对于第一个和第五个操作可以直接建 01trie 维护，比较麻烦的是第二和第三个操作。

cats 的集合 1

- 维护一个集合，支持加入元素，将所有元素或/与/异或上一个值，查询集合中的元素和某个数异或的最大值。
- 对于第一个和第五个操作可以直接建 01trie 维护，比较麻烦的是第二和第三个操作。
- 对于或和与操作的处理方式相同，下面先只考虑或操作。

cats 的集合 1

- 维护一个集合，支持加入元素，将所有元素或/与/异或上一个值，查询集合中的元素和某个数异或的最大值。
- 对于第一个和第五个操作可以直接建 01trie 维护，比较麻烦的是第二和第三个操作。
- 对于或和与操作的处理方式相同，下面先只考虑或操作。
- 在 01trie 上将所有元素或上一个值。如果这个值在二进制下第 k 位为 1，那么相当于所有数的这一位都要变成 1，也就是在这一位上将代表 0 和 1 的两颗子树合并。如果没有加入元素的操作，这样的合并可以通过维护一个全局标记实现（某一位需要强制变为 $0/1$ 或 0 变 1 同时 1 变 0）不过在有加入元素的操作后，新的加入的元素不需要满足之前操作。于是可以将这个全局标记放置在节点上，遍历到某个节点时先处理这个节点上的标记再递归子节点完成加入元素、查询等操作。

cats 的集合 1

- 可以发现在合并两颗子树的过程中也要不断下传标记，会形成一个多层的递归。下面证明这样的递归下总的复杂度仍然合法。

- 可以发现在合并两颗子树的过程中也要不断下传标记，会形成一个多层的递归。下面证明这样的递归下总的复杂度仍然合法。
- 定义一次合并是有意义的当且仅当合并两颗非空子树，只有在有意义的合并中需要继续下传标记。每次有意义的合并会使得整棵字典树的节点个数减少 1，而只有往字典树中加入元素的操作会增加新的节点，且一次只会增加不超过 31 个，所以有意义的合并次数不超过 $31(n+m)$。而无意义的操作只会出现在下传标记和递归合并时。下传标记的次数为 1 和 5 操作遍历到的节点个数加有意义的合并个数乘二。递归合并的个数是有意义的合并个数乘二。所以综上，无意义的合并个数也是 1 和 5 操作遍历到的节点个数级别。总复杂度仍然是 $O((n+m)\log w)$。

B

- 假设我们始终取球，而不是取完球之后就停止，取完第 i 种球的次数为 X_i，那么要求的就是 $\min_{i=1}^{n} X_i$ 的期望。

cats 的随机原神

- 假设我们始终取球，而不是取完球之后就停止，取完第 i 种球的次数为 X_i，那么要求的就是 $\min_{i=1}^{n} X_i$ 的期望。
- 发现最小值期望不太好求，但是最大值期望比较好求，例如 $\max_{i=1}^{n} X_i$ 的期望显然是一个定值，即所有球的总个数，因为总有一种球是要最后一次才能取完。

cats 的随机原神

- 假设我们始终取球，而不是取完球之后就停止，取完第 i 种球的次数为 X_i，那么要求的就是 $\min_{i=1}^{n} X_i$ 的期望。
- 发现最小值期望不太好求，但是最大值期望比较好求，例如 $\max_{i=1}^{n} X_i$ 的期望显然是一个定值，即所有球的总个数，因为总有一种球是要最后一次才能取完。
- 在最小值期望不好求，最大值期望好求的情况下，可以使用 $\min - \max$ 反演，假设全集 S 为 $\{X_1, \cdots, X_n\}$，有

$$\min(S) = \sum_{T \subseteq S T \neq \varnothing} (-1)^{|T|+1} max(T)$$

- 假设我们始终取球，而不是取完球之后就停止，取完第 i 种球的次数为 X_i，那么要求的就是 $\min_{i=1}^{n} X_i$ 的期望。
- 发现最小值期望不太好求，但是最大值期望比较好求，例如 $\max_{i=1}^{n} X_i$ 的期望显然是一个定值，即所有球的总个数，因为总有一种球是要最后一次才能取完。
- 在最小值期望不好求，最大值期望好求的情况下，可以使用 $\min - \max$ 反演，假设全集 S 为 $\{X_1, \cdots, X_n\}$，有

$$\min(S) = \sum_{T \subseteq S T \neq \varnothing} (-1)^{|T|+1} max(T)$$

- 由期望可以线性求和，有

$$E(\min(S)) = \sum_{T \subseteq S T \neq \varnothing} (-1)^{|T|+1} E(max(T))$$

- 对于元素个数相同的集合所得到的期望相等，我们可以按照集合内元素个数枚举求和，并记 $t_i = E(max(T)), |T| = i$。

$$E(\min(S)) = \sum_{i=1}^{n} (-1)^{i+1} \binom{n}{i} t_i$$

- 对于元素个数相同的集合所得到的期望相等，我们可以按照集合内元素个数枚举求和，并记 $t_i = E(max(T)), |T| = i$。

$$E(\min(S)) = \sum_{i=1}^{n} (-1)^{i+1} \binom{n}{i} t_i$$

- 考虑 t_i 如何计算，相当于有两种球，第一种有 mi 个，第二种有 $m(n-i)$ 个，第一种球将整个取球序列分成了 $mi+1$ 段，前 mi 个段中球个数的期望即为 t_i，有

$$t_i = \frac{mi}{mi+1} m(n-i) + mi$$

- 综上所述

$$
\begin{aligned}
ans &= \sum_{i=1}^{n}(-1)^{i+1}\binom{n}{i}t_i \\
&= \sum_{i=1}^{n}(-1)^{i+1}\binom{n}{i}(\frac{mi}{mi+1}m(n-i)+mi) \\
&= (mn+1)\sum_{i=1}^{n}(-1)^{i+1}\binom{n}{i}\frac{mi}{mi+1} \\
&= \frac{(mn+1)m^n n!}{\prod_{i=1}^{n}(mi+1)}
\end{aligned}
$$

cats 的随机原神

- 定义函数 $g(a,b)=\prod_{i=1}^{n}(ai+b)$，则有

$$ans=\frac{(mn+1)m^n g(1,0)}{g(m,1)}$$

cats 的随机原神

- 定义函数 $g(a,b)=\prod_{i=1}^{n}(ai+b)$，则有

$$ans=\frac{(mn+1)m^n g(1,0)}{g(m,1)}$$

- 现在考虑函数 g 怎么快速求，定义 $s=\lfloor\sqrt{n}\rfloor, f(x)=\prod_{i=1}^{s}(ai+b+x)$

$$\begin{aligned}g(a,b)&=\prod_{i=1}^{n}(ai+b)\\&=\prod_{i=0}^{s-1}\prod_{j=1}^{s}(aj+b+asi)\prod_{k=s^2+1}^{n}k\\&=\prod_{i=0}^{s-1}f(asi)\prod_{k=s^2+1}^{n}k\end{aligned}$$

cats 的随机原神

- 后面的一项可以暴力乘法得到，前面的一项用分治乘法得到多项式 f 后，使用多项式多点求值即可。

cats 的随机原神

- 后面的一项可以暴力乘法得到，前面的一项用分治乘法得到多项式 f 后，使用多项式多点求值即可。
- 时间复杂度为 $O(\sqrt{n}\log^2 n)$，使用类似多项式快速阶乘的做法也可以做到 $O(\sqrt{n}\log n)$，但是为了让本题尽可能简单，直接使用上述做法也可以完成。

cats 的随机原神

- 后面的一项可以暴力乘法得到，前面的一项用分治乘法得到多项式 f 后，使用多项式多点求值即可。
- 时间复杂度为 $O(\sqrt{n}\log^2 n)$，使用类似多项式快速阶乘的做法也可以做到 $O(\sqrt{n}\log n)$，但是为了让本题尽可能简单，直接使用上述做法也可以完成。
- 使用 $O(n)$ 的算法进行大力打表也可以通过，但是出题人并不会卡这个。