杭电多校题解

2025 年 7 月 24 日

A. cats 学乗法

A. cats 学乘法

- 数组 a 中所有数的乘积大于 0 的充要条件是数组 a 中没有 0，且负数的个数为偶数。

A. cats 学乘法

- 数组 a 中所有数的乘积大于 0 的充要条件是数组 a 中没有 0，且负数的个数为偶数。
- 如果初始时数组 a 中有 0，我们需要把所有 0 变为 1 或 -1。由于操作最后一个 0 时，我们可以根据此时负数个数的奇偶性选择变为 1 或者 -1，所以不需要考虑初始时负数的个数，答案为数组中 0 的个数。

A. cats 学乘法

- 数组 a 中所有数的乘积大于 0 的充要条件是数组 a 中没有 0，且负数的个数为偶数。
- 如果初始时数组 a 中有 0，我们需要把所有 0 变为 1 或 -1。由于操作最后一个 0 时，我们可以根据此时负数个数的奇偶性选择变为 1 或者 -1，所以不需要考虑初始时负数的个数，答案为数组中 0 的个数。
- 如果初始时数组 a 中没有 0，如果此时负数个数已经是偶数，则不需要操作，答案为 0。如果此时负数个数为奇数，我们需要将一个正数或者负数切换符号改变奇偶性，答案为数组 a 中绝对值最小的数的绝对值 $+1$。

A. cats 学乘法

- 数组 a 中所有数的乘积大于 0 的充要条件是数组 a 中没有 0，且负数的个数为偶数。
- 如果初始时数组 a 中有 0，我们需要把所有 0 变为 1 或 -1。由于操作最后一个 0 时，我们可以根据此时负数个数的奇偶性选择变为 1 或者 -1，所以不需要考虑初始时负数的个数，答案为数组中 0 的个数。
- 如果初始时数组 a 中没有 0，如果此时负数个数已经是偶数，则不需要操作，答案为 0。如果此时负数个数为奇数，我们需要将一个正数或者负数切换符号改变奇偶性，答案为数组 a 中绝对值最小的数的绝对值 $+1$。
- 总时间复杂度为 $O(n)$。

B. 隧道挖掘

B. 隧道挖掘

- 分 $k_1 < k_2$ 和 $k_1 > k_2$ 两种情况讨论（$k_1 = k_2$ 可以在两种情况中计算出不单独考虑)。

B. 隧道挖掘

- 分 $k_1 < k_2$ 和 $k_1 > k_2$ 两种情况讨论（$k_1 = k_2$ 可以在两种情况中计算出不单独考虑）。
- 先考虑较为简单的 $k_2 < k_1$ 的情况，此时所有长度小于 k_1 的必须要第二种设备挖掘，而第二种设备要能挖掘完整条隧道的要求的这条隧道的长度为 k_2 的倍数，所以此时 k_2 只能是所有小于 k_1 的隧道长度公因数，取最优的情况就是取最大公因数。

B. 隧道挖掘

- 分 $k_1 < k_2$ 和 $k_1 > k_2$ 两种情况讨论（$k_1 = k_2$ 可以在两种情况中计算出不单独考虑）。
- 先考虑较为简单的 $k_2 < k_1$ 的情况，此时所有长度小于 k_1 的必须要第二种设备挖掘，而第二种设备要能挖掘完整条隧道的要求的这条隧道的长度为 k_2 的倍数，所以此时 k_2 只能是所有小于 k_1 的隧道长度公因数，取最优的情况就是取最大公因数。
- 对于隧道长度大于等于 k_1 的隧道，使用第一种设备一定更优，所以全部使用第一种设备，一条隧道的代价就是除以 k_1 后向上取整的值，这部分可以通过前缀和优化后快速计算区间和，在 $O(n \log n)$ 的复杂度计算出。

B. 隧道挖掘

- 分 $k_1 < k_2$ 和 $k_1 > k_2$ 两种情况讨论（$k_1 = k_2$ 可以在两种情况中计算出不单独考虑）。
- 先考虑较为简单的 $k_2 < k_1$ 的情况，此时所有长度小于 k_1 的必须要第二种设备挖掘，而第二种设备要能挖掘完整条隧道的要求的这条隧道的长度为 k_2 的倍数，所以此时 k_2 只能是所有小于 k_1 的隧道长度公因数，取最优的情况就是取最大公因数。
- 对于隧道长度大于等于 k_1 的隧道，使用第一种设备一定更优，所以全部使用第一种设备，一条隧道的代价就是除以 k_1 后向上取整的值，这部分可以通过前缀和优化后快速计算区间和，在 $O(n \log n)$ 的复杂度计算出。
- 下面考虑 $k_1 < k_2$ 的情况，此时所有小于 k_2 的必须用 k_1 解决，同时 k_1 必须要小于等于最小的隧道长度，所以 k_1 就取隧道长度的最小值。此时对于长度为 l 的隧道，优先使用第二种设备显然是更优的，但是会剩下 $l \bmod k_2$ 的部分需要使用第一种设备，总代价为

$$\lfloor \frac{l}{k_2} \rfloor + \lceil \frac{l \bmod k_2}{k_1} \rceil$$

B. 隧道挖掘

- 分 $k_1 < k_2$ 和 $k_1 > k_2$ 两种情况讨论（$k_1 = k_2$ 可以在两种情况中计算出不单独考虑）。
- 先考虑较为简单的 $k_2 < k_1$ 的情况，此时所有长度小于 k_1 的必须要第二种设备挖掘，而第二种设备要能挖掘完整条隧道的要求的这条隧道的长度为 k_2 的倍数，所以此时 k_2 只能是所有小于 k_1 的隧道长度公因数，取最优的情况就是取最大公因数。
- 对于隧道长度大于等于 k_1 的隧道，使用第一种设备一定更优，所以全部使用第一种设备，一条隧道的代价就是除以 k_1 后向上取整的值，这部分可以通过前缀和优化后快速计算区间和，在 $O(n \log n)$ 的复杂度计算出。
- 下面考虑 $k_1 < k_2$ 的情况，此时所有小于 k_2 的必须用 k_1 解决，同时 k_1 必须要小于等于最小的隧道长度，所以 k_1 就取隧道长度的最小值。此时对于长度为 l 的隧道，优先使用第二种设备显然是更优的，但是会剩下 $l \bmod k_2$ 的部分需要使用第一种设备，总代价为

$$\lfloor \frac{l}{k_2} \rfloor + \lceil \frac{l \bmod k_2}{k_1} \rceil$$

- 其中 $\lfloor \frac{l}{k_2} \rfloor$ 是使用第二种设备挖掘的隧道数，计算方式和上面类似。关键在于 $\lceil \frac{l \bmod k_2}{k_1} \rceil$ 这一部分。

B. 隧道挖掘

- 考虑在 $[pk_2, (p+1)k_2)$ 这段区间下，其中 p 为非负整数。长度对于 k_2 的余数为 $0, 1, 2, \ldots, k_2 - 1$，余数除以 k_1 向上取整后的结果为

$$0, \underbrace{1, 1, \ldots, 1}_{k_1}, \underbrace{2, 2, \ldots, 2}_{k_1}, \ldots$$

B. 隧道挖掘

- 考虑在 $[pk_2, (p+1)k_2)$ 这段区间下，其中 p 为非负整数。长度对于 k_2 的余数为 $0, 1, 2, \ldots, k_2 - 1$，余数除以 k_1 向上取整后的结果为

$$0, \underbrace{1, 1, \ldots, 1}_{k_1}, \underbrace{2, 2, \ldots, 2}_{k_1}, \ldots$$

- 所以考虑对于每一个后缀维护假设当前位置是乘 0，接下来 k_1 个为乘 1，再接下来 k_1 个为乘 2，依此类推。这部分的计算可以通过后缀和 $O(n)$ 计算。

B. 隧道挖掘

- 考虑在 $[pk_2, (p+1)k_2)$ 这段区间下，其中 p 为非负整数。长度对于 k_2 的余数为 $0, 1, 2, \ldots, k_2 - 1$，余数除以 k_1 向上取整后的结果为

$$0, \underbrace{1, 1, \ldots, 1}_{k_1}, \underbrace{2, 2, \ldots, 2}_{k_1}, \ldots$$

- 所以考虑对于每一个后缀维护假设当前位置是乘 0，接下来 k_1 个为乘 1，再接下来 k_1 个为乘 2，依此类推。这部分的计算可以通过后缀和 $O(n)$ 计算。
- 得到这个值后可以计算在这个数组中差分得到一个区间乘以这个值的和，也就可以计算这段区间的总代价。而枚举所有这些区间的复杂度为 $O(n \log n)$。

B. 隧道挖掘

- 考虑在 $[pk_2, (p+1)k_2)$ 这段区间下，其中 p 为非负整数。长度对于 k_2 的余数为 $0, 1, 2, \ldots, k_2 - 1$，余数除以 k_1 向上取整后的结果为

$$0, \underbrace{1, 1, \ldots, 1}_{k_1}, \underbrace{2, 2, \ldots, 2}_{k_1}, \ldots$$

- 所以考虑对于每一个后缀维护假设当前位置是乘 0，接下来 k_1 个为乘 1，再接下来 k_1 个为乘 2，依此类推。这部分的计算可以通过后缀和 $O(n)$ 计算。
- 得到这个值后可以计算在这个数组中差分得到一个区间乘以这个值的和，也就可以计算这段区间的总代价。而枚举所有这些区间的复杂度为 $O(n \log n)$。
- 综上，可以在 $O(n \log n)$ 的时间复杂度内完成。

C. 纸船效应

C. 纸船效应

- 设 r_i 表示左端点为 i 的区间向右最远的合法区间的右端点，则任意两个区间 $[i, r_i]$ 和 $[j, r_j]$ 一定无交或包含。

C. 纸船效应

- 设 r_i 表示左端点为 i 的区间向右最远的合法区间的右端点，则任意两个区间 $[i, r_i]$ 和 $[j, r_j]$ 一定无交或包含。
- 证明：假设存在 $i < j \leq r_i < r_j$，那么从 j 开始的区间吃到 r_i 时，其生命值一定小于从 i 开始的区间吃到 r_i，所以一定无法继续向右吃，矛盾。

C. 纸船效应

- 设 r_i 表示左端点为 i 的区间向右最远的合法区间的右端点，则任意两个区间 $[i, r_i]$ 和 $[j, r_j]$ 一定无交或包含。
- 证明：假设存在 $i < j \le r_i < r_j$，那么从 j 开始的区间吃到 r_i 时，其生命值一定小于从 i 开始的区间吃到 r_i，所以一定无法继续向右吃，矛盾。
- 考虑用区间 dp 解决问题：设 dp_{l,r,p_1,p_2,p_3,p_4} 表示从 l 开始的区间，目前吃到了 r，l 和 $r+1$ 左/右侧的药水使用状态分别为 p_1, p_2, p_3, p_4 时的总贡献。此时可以通过数组 a 的前缀和以及数组 h 在 $O(1)$ 时间内判断这个区间能否继续向右扩展区间。

C. 纸船效应

- 设 r_i 表示左端点为 i 的区间向右最远的合法区间的右端点，则任意两个区间 $[i, r_i]$ 和 $[j, r_j]$ 一定无交或包含。
- 证明：假设存在 $i < j \le r_i < r_j$，那么从 j 开始的区间吃到 r_i 时，其生命值一定小于从 i 开始的区间吃到 r_i，所以一定无法继续向右吃，矛盾。
- 考虑用区间 dp 解决问题：设 dp_{l,r,p_1,p_2,p_3,p_4} 表示从 l 开始的区间，目前吃到了 r，l 和 $r+1$ 左/右侧的药水使用状态分别为 p_1, p_2, p_3, p_4 时的总贡献。此时可以通过数组 a 的前缀和以及数组 h 在 $O(1)$ 时间内判断这个区间能否继续向右扩展区间。
- dp 的转移可以通过拼接两个区间 $dp_{l,mid,p_1,p_2,p_3,p_4}$ 和 $dp_{mid+1,r,p_3,p_4,p_5,p_6}$ 实现。此时要求 $dp_{l,mid,p_1,p_2,p_3,p_4}$ 一定可以向右扩展，而 $dp_{mid+1,r,p_3,p_4,p_5,p_6}$ 一定不能向右扩展（否则从 $mid+1$ 开始的区间还没扩展完全就被其他区间吃掉而无法继续扩展）。为方便转移，可以在数组最右侧放一个生命值为 inf 的怪物，让所有区间在此停止扩展。

C. 纸船效应

- 设 r_i 表示左端点为 i 的区间向右最远的合法区间的右端点，则任意两个区间 $[i, r_i]$ 和 $[j, r_j]$ 一定无交或包含。
- 证明：假设存在 $i < j \le r_i < r_j$，那么从 j 开始的区间吃到 r_i 时，其生命值一定小于从 i 开始的区间吃到 r_i，所以一定无法继续向右吃，矛盾。
- 考虑用区间 dp 解决问题：设 dp_{l,r,p_1,p_2,p_3,p_4} 表示从 l 开始的区间，目前吃到了 r，l 和 $r+1$ 左/右侧的药水使用状态分别为 p_1, p_2, p_3, p_4 时的总贡献。此时可以通过数组 a 的前缀和以及数组 h 在 $O(1)$ 时间内判断这个区间能否继续向右扩展区间。
- dp 的转移可以通过拼接两个区间 $dp_{l,mid,p_1,p_2,p_3,p_4}$ 和 $dp_{mid+1,r,p_3,p_4,p_5,p_6}$ 实现。此时要求 $dp_{l,mid,p_1,p_2,p_3,p_4}$ 一定可以向右扩展，而 $dp_{mid+1,r,p_3,p_4,p_5,p_6}$ 一定不能向右扩展（否则从 $mid+1$ 开始的区间还没扩展完全就被其他区间吃掉而无法继续扩展）。为方便转移，可以在数组最右侧放一个生命值为 inf 的怪物，让所有区间在此停止扩展。
- 最后数组一定被划分为若干无法继续向右扩展的区间。最后用一次普通的 dp 整合所有区间即可得到最终答案。

C. 纸船效应

- 设 r_i 表示左端点为 i 的区间向右最远的合法区间的右端点，则任意两个区间 $[i, r_i]$ 和 $[j, r_j]$ 一定无交或包含。
- 证明：假设存在 $i < j \le r_i < r_j$，那么从 j 开始的区间吃到 r_i 时，其生命值一定小于从 i 开始的区间吃到 r_i，所以一定无法继续向右吃，矛盾。
- 考虑用区间 dp 解决问题：设 dp_{l,r,p_1,p_2,p_3,p_4} 表示从 l 开始的区间，目前吃到了 r，l 和 $r+1$ 左/右侧的药水使用状态分别为 p_1, p_2, p_3, p_4 时的总贡献。此时可以通过数组 a 的前缀和以及数组 h 在 $O(1)$ 时间内判断这个区间能否继续向右扩展区间。
- dp 的转移可以通过拼接两个区间 $dp_{l,mid,p_1,p_2,p_3,p_4}$ 和 $dp_{mid+1,r,p_3,p_4,p_5,p_6}$ 实现。此时要求 $dp_{l,mid,p_1,p_2,p_3,p_4}$ 一定可以向右扩展，而 $dp_{mid+1,r,p_3,p_4,p_5,p_6}$ 一定不能向右扩展（否则从 $mid+1$ 开始的区间还没扩展完全就被其他区间吃掉而无法继续扩展）。为方便转移，可以在数组最右侧放一个生命值为 inf 的怪物，让所有区间在此停止扩展。
- 最后数组一定被划分为若干无法继续向右扩展的区间。最后用一次普通的 dp 整合所有区间即可得到最终答案。
- 总时间复杂度为 $O(n^3)$。

D. 传送排序

D. 传送排序

- 考虑所有没有被传送的数，它们一定构成原来的序列中的一个单调递增的子序列，所有不在子序列中的数则需要被操作。每一段（数值上）连续的，需要被操作的数可以使用同一个传送门传送。总操作次数即为所有被操作的数的总数与所有被操作的数组成的连续段数之和。

D. 传送排序

- 考虑所有没有被传送的数，它们一定构成原来的序列中的一个单调递增的子序列，所有不在子序列中的数则需要被操作。每一段（数值上）连续的，需要被操作的数可以使用同一个传送门传送。总操作次数即为所有被操作的数的总数与所有被操作的数组成的连续段数之和。
- 设 dp_j 表示以 p_j 为结尾的子序列需要操作次数的最小值，考虑子序列中前一个数 p_i，如果 $p_i+1=p_j$，则不需要插入连续段，更新为 $dp_j=\min(dp_j,dp_i)$。如果 $p_i+1<p_j$，则需要插入一个含有 p_j-p_i-1 个数的连续段。则有 $dp_j=\min(dp_j,dp_i+p_j-p_i)$。

D. 传送排序

- 考虑所有没有被传送的数，它们一定构成原来的序列中的一个单调递增的子序列，所有不在子序列中的数则需要被操作。每一段（数值上）连续的，需要被操作的数可以使用同一个传送门传送。总操作次数即为所有被操作的数的总数与所有被操作的数组成的连续段数之和。
- 设 dp_j 表示以 p_j 为结尾的子序列需要操作次数的最小值，考虑子序列中前一个数 p_i，如果 $p_i+1=p_j$，则不需要插入连续段，更新为 $dp_j=\min(dp_j,dp_i)$。如果 $p_i+1<p_j$，则需要插入一个含有 p_j-p_i-1 个数的连续段。则有 $dp_j=\min(dp_j,dp_i+p_j-p_i)$。
- 对于数组两侧的需要被操作的数的连续段，可以通过在排列 p 开头放入一个 0，末尾放入一个 $n+1$ 解决。第一种转移可以直接 $O(n)$ 做，第二种转移可以用树状数组快速维护 p_j 所有满足 $i<j$，$p_i+1<p_j$ 的位置 dp_i-p_i 的 $\min$ 实现。

D. 传送排序

- 考虑所有没有被传送的数，它们一定构成原来的序列中的一个单调递增的子序列，所有不在子序列中的数则需要被操作。每一段（数值上）连续的，需要被操作的数可以使用同一个传送门传送。总操作次数即为所有被操作的数的总数与所有被操作的数组成的连续段数之和。
- 设 dp_j 表示以 p_j 为结尾的子序列需要操作次数的最小值，考虑子序列中前一个数 p_i，如果 $p_i+1=p_j$，则不需要插入连续段，更新为 $dp_j=\min(dp_j,dp_i)$。如果 $p_i+1<p_j$，则需要插入一个含有 p_j-p_i-1 个数的连续段。则有 $dp_j=\min(dp_j,dp_i+p_j-p_i)$。
- 对于数组两侧的需要被操作的数的连续段，可以通过在排列 p 开头放入一个 0，末尾放入一个 $n+1$ 解决。第一种转移可以直接 $O(n)$ 做，第二种转移可以用树状数组快速维护 p_j 所有满足 $i<j$，$p_i+1<p_j$ 的位置 dp_i-p_i 的 $\min$ 实现。
- 总时间复杂度为 $O(n\log n)$。

E. 钥匙迷宫

E. 钥匙迷宫

- 考虑每一对钥匙和锁，将锁所在的节点删掉，你只能从钥匙所在的连通块出发。考虑所有钥匙和锁的限制，你可以选择出发的点一定是一个完全由钥匙组成的连通块。否则，假如两个可以作为出发节点钥匙之间存在锁，则删掉这个锁，两侧最多有一侧和钥匙连通，矛盾。

E. 钥匙迷宫

- 考虑每一对钥匙和锁，将锁所在的节点删掉，你只能从钥匙所在的连通块出发。考虑所有钥匙和锁的限制，你可以选择出发的点一定是一个完全由钥匙组成的连通块。否则，假如两个可以作为出发节点钥匙之间存在锁，则删掉这个锁，两侧最多有一侧和钥匙连通，矛盾。
- 考虑找到这个连通块，只考虑对于一对钥匙和锁，你可以出发的节点一定是以锁为根，某个和锁直接相邻的节点的子树。可以通过树上倍增快速找到这个直接相邻的节点，同时用 dfs 序子树剖分和前缀和快速标记所有不满足要求的节点。

E. 钥匙迷宫

- 考虑每一对钥匙和锁，将锁所在的节点删掉，你只能从钥匙所在的连通块出发。考虑所有钥匙和锁的限制，你可以选择出发的点一定是一个完全由钥匙组成的连通块。否则，假如两个可以作为出发节点钥匙之间存在锁，则删掉这个锁，两侧最多有一侧和钥匙连通，矛盾。
- 考虑找到这个连通块，只考虑对于一对钥匙和锁，你可以出发的节点一定是以锁为根，某个和锁直接相邻的节点的子树。可以通过树上倍增快速找到这个直接相邻的节点，同时用 dfs 序子树剖分和前缀和快速标记所有不满足要求的节点。
- 找到连通块后（如果满足要求的连通块不存在则显然无解，全输出 0），还需要进行一次 bfs 搜索判断这个连通块是否符合要求。因为上面的转化是必要不充分的。如果 bfs 确定从这个连通块出发符合要求，则输出连通块内所有点作为答案，否则全输出 0。

E. 钥匙迷宫

- 考虑每一对钥匙和锁，将锁所在的节点删掉，你只能从钥匙所在的连通块出发。考虑所有钥匙和锁的限制，你可以选择出发的点一定是一个完全由钥匙组成的连通块。否则，假如两个可以作为出发节点钥匙之间存在锁，则删掉这个锁，两侧最多有一侧和钥匙连通，矛盾。
- 考虑找到这个连通块，只考虑对于一对钥匙和锁，你可以出发的节点一定是以锁为根，某个和锁直接相邻的节点的子树。可以通过树上倍增快速找到这个直接相邻的节点，同时用 dfs 序子树剖分和前缀和快速标记所有不满足要求的节点。
- 找到连通块后（如果满足要求的连通块不存在则显然无解，全输出 0），还需要进行一次 bfs 搜索判断这个连通块是否符合要求。因为上面的转化是必要不充分的。如果 bfs 确定从这个连通块出发符合要求，则输出连通块内所有点作为答案，否则全输出 0。
- 总时间复杂度为 $O(n \log n)$。

F. 缺失的子序列

F. 缺失的子序列

- 我们把排列看做平面上 n 个点 (i, p_i)。

F. 缺失的子序列

- 我们把排列看做平面上 n 个点 (i, p_i)。
- 定义一个排列为可分割的，当且仅当存在一条平行 x 轴的直线和一条平行 y 轴的直线，可以把排列分成左上-右下两部分或者右上-左下两部分(每一部分不能为空)。

F. 缺失的子序列

- 我们把排列看做平面上 n 个点 (i, p_i)。
- 定义一个排列为可分割的，当且仅当存在一条平行 x 轴的直线和一条平行 y 轴的直线，可以把排列分成左上-右下两部分或者右上-左下两部分(每一部分不能为空)。
- 定义一个排列为无限可分割的，当且仅当其在分割后得到的每一部分都是可分割的，不断的递归直到每一部分只有一个点为止。

F. 缺失的子序列

- 我们把排列看做平面上 n 个点 (i, p_i)。
- 定义一个排列为可分割的，当且仅当存在一条平行 x 轴的直线和一条平行 y 轴的直线，可以把排列分成左上-右下两部分或者右上-左下两部分(每一部分不能为空)。
- 定义一个排列为无限可分割的，当且仅当其在分割后得到的每一部分都是可分割的，不断的递归直到每一部分只有一个点为止。
- 结论：排列不存在 2413 子序列和 3142 子序列，等价于排列无限可分割。

F. 缺失的子序列

- 充分性：

F. 缺失的子序列

- 充分性：
- 只需要证明所有不含 2413 和 3142 的长度大于 1 的排列一定可分割即可。

F. 缺失的子序列

- 充分性：
- 只需要证明所有不含 2413 和 3142 的长度大于 1 的排列一定可分割即可。
- 假设存在不含 2413 和 3142 的排列 p 不可分割。因为 2413 和 3142 对称，这里不妨设 $p_1 < p_n$。那么可知 $p_1 > 1$，$p_n < n$。

F. 缺失的子序列

- 充分性：
- 只需要证明所有不含 2413 和 3142 的长度大于 1 的排列一定可分割即可。
- 假设存在不含 2413 和 3142 的排列 p 不可分割。因为 2413 和 3142 对称，这里不妨设 $p_1 < p_n$。那么可知 $p_1 > 1$，$p_n < n$。
- 我们找到所有 $a_1, a_2, \ldots, a_n$ 满足 p_{a_k} 是 $p_{a_k}, p_{a_k+1}, \ldots, p_n$ 中最小的数的位置（即后缀 $\min$ 位置）。

F. 缺失的子序列

- 充分性：
- 只需要证明所有不含 2413 和 3142 的长度大于 1 的排列一定可分割即可。
- 假设存在不含 2413 和 3142 的排列 p 不可分割。因为 2413 和 3142 对称，这里不妨设 $p_1 < p_n$。那么可知 $p_1 > 1$，$p_n < n$。
- 我们找到所有 $a_1, a_2, \ldots, a_n$ 满足 p_{a_k} 是 $p_{a_k}, p_{a_k+1}, \ldots, p_n$ 中最小的数的位置（即后缀 $\min$ 位置）。
- 由于 $p_{a_i} < p_{a_i+1}$，$p_{a_1} = 1 < p_1 < p_{a_k} = p_n$。所以一定存在数组 a 中两个相邻的数 x, y，使得 $p_x < p_1 < p_y$。

F. 缺失的子序列

- 充分性：
- 只需要证明所有不含 2413 和 3142 的长度大于 1 的排列一定可分割即可。
- 假设存在不含 2413 和 3142 的排列 p 不可分割。因为 2413 和 3142 对称，这里不妨设 $p_1 < p_n$。那么可知 $p_1 > 1$，$p_n < n$。
- 我们找到所有 $a_1, a_2, \ldots, a_n$ 满足 p_{a_k} 是 $p_{a_k}, p_{a_k+1}, \ldots, p_n$ 中最小的数的位置（即后缀 $\min$ 位置）。
- 由于 $p_{a_i} < p_{a_i+1}$，$p_{a_1} = 1 < p_1 < p_{a_k} = p_n$。所以一定存在数组 a 中两个相邻的数 x, y，使得 $p_x < p_1 < p_y$。
- 此时考虑 (x, p_x) 和 (y, p_y) 的位置。由于后缀 $\min$ 的性质，(x, p_y) 右下方的区域内一定没有任何点。

F. 缺失的子序列

- 充分性：
- 只需要证明所有不含 2413 和 3142 的长度大于 1 的排列一定可分割即可。
- 假设存在不含 2413 和 3142 的排列 p 不可分割。因为 2413 和 3142 对称，这里不妨设 $p_1 < p_n$。那么可知 $p_1 > 1$，$p_n < n$。
- 我们找到所有 $a_1, a_2, \ldots, a_n$ 满足 p_{a_k} 是 $p_{a_k}, p_{a_k+1}, \ldots, p_n$ 中最小的数的位置（即后缀 $\min$ 位置）。
- 由于 $p_{a_i} < p_{a_i+1}$，$p_{a_1} = 1 < p_1 < p_{a_k} = p_n$。所以一定存在数组 a 中两个相邻的数 x, y，使得 $p_x < p_1 < p_y$。
- 此时考虑 (x, p_x) 和 (y, p_y) 的位置。由于后缀 $\min$ 的性质，(x, p_y) 右下方的区域内一定没有任何点。
- 然后观察 p_1, p_x 和 p_y 的位置。由于不存在 2413 子序列，又有 (x, p_y) 左上方的区域内没有任何点。

F. 缺失的子序列

- 充分性：
- 只需要证明所有不含 2413 和 3142 的长度大于 1 的排列一定可分割即可。
- 假设存在不含 2413 和 3142 的排列 p 不可分割。因为 2413 和 3142 对称，这里不妨设 $p_1 < p_n$。那么可知 $p_1 > 1$，$p_n < n$。
- 我们找到所有 $a_1, a_2, \ldots, a_n$ 满足 p_{a_k} 是 $p_{a_k}, p_{a_k+1}, \ldots, p_n$ 中最小的数的位置（即后缀 $\min$ 位置）。
- 由于 $p_{a_i} < p_{a_i+1}$，$p_{a_1} = 1 < p_1 < p_{a_k} = p_n$。所以一定存在数组 a 中两个相邻的数 x, y，使得 $p_x < p_1 < p_y$。
- 此时考虑 (x, p_x) 和 (y, p_y) 的位置。由于后缀 $\min$ 的性质，(x, p_y) 右下方的区域内一定没有任何点。
- 然后观察 p_1, p_x 和 p_y 的位置。由于不存在 2413 子序列，又有 (x, p_y) 左上方的区域内没有任何点。
- 于是可以沿着这条横线和竖线把排列切开，矛盾。

F. 缺失的子序列

- 必要性：

F. 缺失的子序列

- 必要性：
- 对于一个可分割的数组，显然 2413 和 3142 必须位于分割结果的同侧。所以有 2413 或 3142 的数组一定存在一个长度大于等于 4 的极小不可分割单元。

F. 缺失的子序列

- 必要性：
- 对于一个可分割的数组，显然 2413 和 3142 必须位于分割结果的同侧。所以有 2413 或 3142 的数组一定存在一个长度大于等于 4 的极小不可分割单元。
- 于是可以对矩阵进行 dp。设 $dp_{x,y,d}$ 为左下角为 (x,y)，边长为 d 的正方形作为一个分割的方案数，枚举所有 $O(d)$ 种分割方案转移即可。

F. 缺失的子序列

- 必要性：
- 对于一个可分割的数组，显然 2413 和 3142 必须位于分割结果的同侧。所以有 2413 或 3142 的数组一定存在一个长度大于等于 4 的极小不可分割单元。
- 于是可以对矩阵进行 dp。设 $dp_{x,y,d}$ 为左下角为 (x,y)，边长为 d 的正方形作为一个分割的方案数，枚举所有 $O(d)$ 种分割方案转移即可。
- 为了防止多次左上-右下切割和多次左下-右上切割时转移重复，可以要求进行左上-右下切割后，其两部分中有一部分一定无法继续左上-右下切割。另一个方向同理。

F. 缺失的子序列

- 必要性：
- 对于一个可分割的数组，显然 2413 和 3142 必须位于分割结果的同侧。所以有 2413 或 3142 的数组一定存在一个长度大于等于 4 的极小不可分割单元。
- 于是可以对矩阵进行 dp。设 $dp_{x,y,d}$ 为左下角为 (x,y)，边长为 d 的正方形作为一个分割的方案数，枚举所有 $O(d)$ 种分割方案转移即可。
- 为了防止多次左上-右下切割和多次左下-右上切割时转移重复，可以要求进行左上-右下切割后，其两部分中有一部分一定无法继续左上-右下切割。另一个方向同理。
- 总时间复杂度为 $O(n^4)$。

G. 喵喵题

G. 喵喵题

- 结论：你取的 1 一定是一个区间中所有的 1。

G. 喵喵题

- 结论：你取的 1 一定是一个区间中所有的 1。
- 如果你对复杂的数学证明不感兴趣，只想知道怎么想到这一点：

G. 喵喵题

- 结论：你取的 1 一定是一个区间中所有的 1。
- 如果你对复杂的数学证明不感兴趣，只想知道怎么想到这一点：
- 你抛弃 1 的原因一定是为了不让自己的过牌速度下降。所以说固定你选的最后一个 1（也就是固定需要取的总长度），你一定是尽可能更晚的选择保留 1 而降低速度。

- 结论：你取的 1 一定是一个区间中所有的 1。
- 如果你对复杂的数学证明不感兴趣，只想知道怎么想到这一点：
- 你抛弃 1 的原因一定是为了不让自己的过牌速度下降。所以说固定你选的最后一个 1（也就是固定需要取的总长度），你一定是尽可能更晚的选择保留 1 而降低速度。
- 如果你不满意上面的证明，想听听更严谨的证明：

G. 喵喵题

- 结论：你取的 1 一定是一个区间中所有的 1。
- 如果你对复杂的数学证明不感兴趣，只想知道怎么想到这一点：
- 你抛弃 1 的原因一定是为了不让自己的过牌速度下降。所以说固定你选的最后一个 1（也就是固定需要取的总长度），你一定是尽可能更晚的选择保留 1 而降低速度。
- 如果你不满意上面的证明，想听听更严谨的证明：
- 将字符串中除了你最后保留的 k 个 1，其余 1 全部改为 0，这样你在选择好保留的 1 的集合后，取数的方案就是唯一的了。然后，只考虑最后一个 1 的位置相同的方案。

G. 喵喵题

- 对于一个选择 1 的方案 S，定义 $f(S,i)$ 表示方案 S 取了 i 次后取走的前缀长度，$g(S,i)$ 表示方案 S 前 i 个数中 1 的个数。设 T 表示选择最靠后的 1 的方案。这里用数学归纳法证明对于任意的方案 S 和 $i \geq 0$，都有 $f(T,i) \geq f(S,i)$。

G. 喵喵题

- 对于一个选择 1 的方案 S，定义 $f(S,i)$ 表示方案 S 取了 i 次后取走的前缀长度，$g(S,i)$ 表示方案 S 前 i 个数中 1 的个数。设 T 表示选择最靠后的 1 的方案。这里用数学归纳法证明对于任意的方案 S 和 $i \geq 0$，都有 $f(T,i) \geq f(S,i)$。
- 当 $i=0$ 时，显然有 $f(T,i)=f(S,i)=0$。对于任意的 $i \geq 0$，如果 $f(T,i) \geq f(S,i)$。由于 $g(T,f(T,i)) \leq g(T,f(S,i))+f(T,i)-f(S,i) \leq g(S,f(S,i))+f(T,i)-f(S,i)$。所以 $f(T,i+1)=f(T,i)+k-g(T,f(T,i)) \geq f(S,i)+k-g(S,f(S+1))=f(S,i+1)$。数学归纳证毕。

G. 喵喵题

- 回到这个题。我们可以枚举选择的包含 k 个 1 的区间，暴力求解每个区间的答案。因为当前可以取的长度超过根号 n 时，一定取不超过根号 n 次，长度不超过根号 n 时，每次找到跳到下一个 1 快速跳过去，也不超过根号 n 次。所以我们可以得到一个 $O(nsqrt(n))$ 的做法。但事实上跑得非常快，但 cats 也证不出更低的时间复杂度或造出数据卡满上界，不过时间限制仍然是按照单组 $nsqrt(n)$ 定的。

G. 喵喵题

- 回到这个题。我们可以枚举选择的包含 k 个 1 的区间，暴力求解每个区间的答案。因为当前可以取的长度超过根号 n 时，一定取不超过根号 n 次，长度不超过根号 n 时，每次找到跳到下一个 1 快速跳过去，也不超过根号 n 次。所以我们可以得到一个 $O(nsqrt(n))$ 的做法。但事实上跑得非常快，但 cats 也证不出更低的时间复杂度或造出数据卡满上界，不过时间限制仍然是按照单组 $nsqrt(n)$ 定的。
- 考虑快速跳到下一个 1 的过程，其实节省的都是全部为 0 的区间。但是除了第一段外，不同次查询中，这样的全 0 区间一定出现在不同的 1 中间，这样额外多出的区间总数不超过 $O(nlogn)$。所以如果只快速跳过第一段，剩下的全部暴力，也可以得到不超过 $O(nsqrt(n))$ 的复杂度。但如果不快速跳过第一段，则会被全部为 1 的字符串和 $k=2$ 卡到 $O(n^2)$。

G. 喵喵题

- 回到这个题。我们可以枚举选择的包含 k 个 1 的区间，暴力求解每个区间的答案。因为当前可以取的长度超过根号 n 时，一定取不超过根号 n 次，长度不超过根号 n 时，每次找到跳到下一个 1 快速跳过去，也不超过根号 n 次。所以我们可以得到一个 $O(nsqrt(n))$ 的做法。但事实上跑得非常快，但 cats 也证不出更低的时间复杂度或造出数据卡满上界，不过时间限制仍然是按照单组 $nsqrt(n)$ 定的。
- 考虑快速跳到下一个 1 的过程，其实节省的都是全部为 0 的区间。但是除了第一段外，不同次查询中，这样的全 0 区间一定出现在不同的 1 中间，这样额外多出的区间总数不超过 $O(nlogn)$。所以如果只快速跳过第一段，剩下的全部暴力，也可以得到不超过 $O(nsqrt(n))$ 的复杂度。但如果不快速跳过第一段，则会被全部为 1 的字符串和 $k=2$ 卡到 $O(n^2)$。
- 总时间复杂度为 $O(n\sqrt{n})$。

H. cats 的 max

H. cats 的 max

- 首先如果选择的行数大于等于 m，则一定可以选择出每一列的最大值。这样选择一定是最大的。

H. cats 的 max

- 首先如果选择的行数大于等于 m，则一定可以选择出每一列的最大值。这样选择一定是最大的。
- 下面考虑选择的行数小于 m 的情况。

H. cats 的 max

- 首先如果选择的行数大于等于 m，则一定可以选择出每一列的最大值。这样选择一定是最大的。
- 下面考虑选择的行数小于 m 的情况。
- 因为 m 很小，所以可以 $n2^m$ 枚举每一行的所有子集，假设这个子集在最后是答案中成为最大值的部分。那么最后的答案一定可以拆成不超过 k 个不同的子集的并。

H. cats 的 max

- 首先如果选择的行数大于等于 m，则一定可以选择出每一列的最大值。这样选择一定是最大的。
- 下面考虑选择的行数小于 m 的情况。
- 因为 m 很小，所以可以 $n2^m$ 枚举每一行的所有子集，假设这个子集在最后是答案中成为最大值的部分。那么最后的答案一定可以拆成不超过 k 个不同的子集的并。
- 所以只需要对于 2^m 种子集记录每一种选择的最优解，最后就是合并 k 次即可，一次合并就是对于每一种集合需要再拆分为两个集合，取拆分的两个集合的权值的和的最大值，这部分可以 $O(m3^m)$ 计算。

H. cats 的 max

- 首先如果选择的行数大于等于 m，则一定可以选择出每一列的最大值。这样选择一定是最大的。
- 下面考虑选择的行数小于 m 的情况。
- 因为 m 很小，所以可以 $n2^m$ 枚举每一行的所有子集，假设这个子集在最后是答案中成为最大值的部分。那么最后的答案一定可以拆成不超过 k 个不同的子集的并。
- 所以只需要对于 2^m 种子集记录每一种选择的最优解，最后就是合并 k 次即可，一次合并就是对于每一种集合需要再拆分为两个集合，取拆分的两个集合的权值的和的最大值，这部分可以 $O(m3^m)$ 计算。
- 综上，总时间复杂度为 $O(n2^m + m3^m)$。

I. 对撞器

I. 对撞器

- 特判 $n <= 2$ 的情况答案为 0。

I. 对撞器

- 特判 $n <= 2$ 的情况答案为 0。
- 此时考虑数组中最大的数。设这个数为 mx，如果 mx 是第一个数或最后一个数，则可通过不断消除其旁边的数让答案取到 $(n-2)mx$。由于有贡献的消除不超过 $n-2$ 次，且每次贡献不超过 mx，所以这一定是最优的操作方法。

I. 对撞器

- 特判 $n <= 2$ 的情况答案为 0。
- 此时考虑数组中最大的数。设这个数为 mx，如果 mx 是第一个数或最后一个数，则可通过不断消除其旁边的数让答案取到 $(n-2)mx$。由于有贡献的消除不超过 $n-2$ 次，且每次贡献不超过 mx，所以这一定是最优的操作方法。
- 如果 mx 不是第一个数或最后一个数，则可通过不断消除其旁边除 a_1 和 a_n 以外的数，产生 $(n-3)mx$ 的贡献，得到数组 a_1, mx, a_n。最后消除 mx，产生 $max(a_1, a_n)$ 的贡献，总答案为 $(n-3)mx + max(a_1, a_n)$。由于最后一次有贡献的消除一定是 a_1 和 a_n 对撞（因为提前消除 a_1 和 a_n 没有贡献一定亏损），所以最后一次操作贡献一定是 $max(a_1, a_n)$。同时其余所有操作贡献不可能超过 mx，所以这一定是最优的操作方法。

I. 对撞器

- 特判 $n <= 2$ 的情况答案为 0。
- 此时考虑数组中最大的数。设这个数为 mx，如果 mx 是第一个数或最后一个数，则可通过不断消除其旁边的数让答案取到 $(n-2)mx$。由于有贡献的消除不超过 $n-2$ 次，且每次贡献不超过 mx，所以这一定是最优的操作方法。
- 如果 mx 不是第一个数或最后一个数，则可通过不断消除其旁边除 a_1 和 a_n 以外的数，产生 $(n-3)mx$ 的贡献，得到数组 a_1, mx, a_n。最后消除 mx，产生 $max(a_1, a_n)$ 的贡献，总答案为 $(n-3)mx + max(a_1, a_n)$。由于最后一次有贡献的消除一定是 a_1 和 a_n 对撞（因为提前消除 a_1 和 a_n 没有贡献一定亏损），所以最后一次操作贡献一定是 $max(a_1, a_n)$。同时其余所有操作贡献不可能超过 mx，所以这一定是最优的操作方法。
- 综上，如果 mx 是 a_1 或 a_n 则答案为 $(n-2)mx$，否则为 $(n-3)mx + max(a_1, a_n)$。

I. 对撞器

- 特判 $n <= 2$ 的情况答案为 0。
- 此时考虑数组中最大的数。设这个数为 mx，如果 mx 是第一个数或最后一个数，则可通过不断消除其旁边的数让答案取到 $(n-2)mx$。由于有贡献的消除不超过 $n-2$ 次，且每次贡献不超过 mx，所以这一定是最优的操作方法。
- 如果 mx 不是第一个数或最后一个数，则可通过不断消除其旁边除 a_1 和 a_n 以外的数，产生 $(n-3)mx$ 的贡献，得到数组 a_1, mx, a_n。最后消除 mx，产生 $max(a_1, a_n)$ 的贡献，总答案为 $(n-3)mx + max(a_1, a_n)$。由于最后一次有贡献的消除一定是 a_1 和 a_n 对撞（因为提前消除 a_1 和 a_n 没有贡献一定亏损），所以最后一次操作贡献一定是 $max(a_1, a_n)$。同时其余所有操作贡献不可能超过 mx，所以这一定是最优的操作方法。
- 综上，如果 mx 是 a_1 或 a_n 则答案为 $(n-2)mx$，否则为 $(n-3)mx + max(a_1, a_n)$。
- 总时间复杂度为 $O(n)$。

J. 群体狂乱

J. 群体狂乱

- 观察 1：每个随从主动只能发起一次攻击其实是没有意义的。因为两个随从对撞必有一个死亡，我们让会死亡的随从主动发起攻击，即可保证存活的随从都是时刻可发起攻击的。所以此问题等价于所有随从可以任意相互攻击。

J. 群体狂乱

- 观察 1：每个随从主动只能发起一次攻击其实是没有意义的。因为两个随从对撞必有一个死亡，我们让会死亡的随从主动发起攻击，即可保证存活的随从都是时刻可发起攻击的。所以此问题等价于所有随从可以任意相互攻击。
- 观察 2：对于任意的一个随从的集合，一定可以通过相互攻击让集合中剩余的随从不超过 1 个。

J. 群体狂乱

- 观察 1：每个随从主动只能发起一次攻击其实是没有意义的。因为两个随从对撞必有一个死亡，我们让会死亡的随从主动发起攻击，即可保证存活的随从都是时刻可发起攻击的。所以此问题等价于所有随从可以任意相互攻击。
- 观察 2：对于任意的一个随从的集合，一定可以通过相互攻击让集合中剩余的随从不超过 1 个。
- 情况讨论 1：如果数组 a 中最大值的个数为偶数，则我们先让所有非最大值互相攻击到剩余不超过 1 个，再让其（如果存在）攻击一个最大值的随从。最后让最大值的随从两两互相对撞。此时一定有解。

J. 群体狂乱

- 观察 1：每个随从主动只能发起一次攻击其实是没有意义的。因为两个随从对撞必有一个死亡，我们让会死亡的随从主动发起攻击，即可保证存活的随从都是时刻可发起攻击的。所以此问题等价于所有随从可以任意相互攻击。
- 观察 2：对于任意的一个随从的集合，一定可以通过相互攻击让集合中剩余的随从不超过 1 个。
- 情况讨论 1：如果数组 a 中最大值的个数为偶数，则我们先让所有非最大值互相攻击到剩余不超过 1 个，再让其（如果存在）攻击一个最大值的随从。最后让最大值的随从两两互相对撞。此时一定有解。
- 情况讨论 2：如果数组 a 中最大值的个数为奇数。如果此时所有其他随从 a 总和小于最大值的随从，那么一定无法·改变最大值个数的奇偶性，一定无解。

J. 群体狂乱

- 观察 1：每个随从主动只能发起一次攻击其实是没有意义的。因为两个随从对撞必有一个死亡，我们让会死亡的随从主动发起攻击，即可保证存活的随从都是时刻可发起攻击的。所以此问题等价于所有随从可以任意相互攻击。
- 观察 2：对于任意的一个随从的集合，一定可以通过相互攻击让集合中剩余的随从不超过 1 个。
- 情况讨论 1：如果数组 a 中最大值的个数为偶数，则我们先让所有非最大值互相攻击到剩余不超过 1 个，再让其（如果存在）攻击一个最大值的随从。最后让最大值的随从两两互相对撞。此时一定有解。
- 情况讨论 2：如果数组 a 中最大值的个数为奇数。如果此时所有其他随从 a 总和小于最大值的随从，那么一定无法·改变最大值个数的奇偶性，一定无解。
- 情况讨论 3：如果数组 a 中最大值的个数为大于等于 3，且此时所有其他随从 a 总和大于等于最大值的随从，那么可以让所有其他随从攻击一个 a 为最大值的随从，直到其死亡。此时则进入情况 1，一定有解。

- 剩余未讨论的情况只有 a 中最大值只出现一次，且此时所有其他随从 a 总和大于等于最大值的随从，后续将不再强调这两个条件。

- 剩余未讨论的情况只有 a 中最大值只出现一次，且此时所有其他随从 a 总和大于等于最大值的随从，后续将不再强调这两个条件。
- 设此时 a 最大的随从的 a 为 m_1，a 次大的随从的 a 为 m_2。m_2 在数组 a 中共出现 c 次。

J. 群体狂乱

- 剩余未讨论的情况只有 a 中最大值只出现一次，且此时所有其他随从 a 总和大于等于最大值的随从，后续将不再强调这两个条件。
- 设此时 a 最大的随从的 a 为 m_1，a 次大的随从的 a 为 m_2。m_2 在数组 a 中共出现 c 次。
- 情况讨论 4：如果 $c \cdot m_2 \leq m_1$，那么我们可以在非 m_1 和 m_2 的随从中，不断让随从攻击 m_1，直到其生命值不超过 $c \cdot m_2$。此时其生命值一定大于 $(c-1) \cdot m_2$。让剩余的非 m_1，m_2 随从对撞到剩余不超过 1 个，攻击任意一个 m_2，最后让所有 m_2 攻击 m_1 即可消灭所有随从，此时有解。

J. 群体狂乱

- 剩余未讨论的情况只有 a 中最大值只出现一次，且此时所有其他随从 a 总和大于等于最大值的随从，后续将不再强调这两个条件。
- 设此时 a 最大的随从的 a 为 m_1，a 次大的随从的 a 为 m_2。m_2 在数组 a 中共出现 c 次。
- 情况讨论 4：如果 $c \cdot m_2 \leq m_1$，那么我们可以在非 m_1 和 m_2 的随从中，不断让随从攻击 m_1，直到其生命值不超过 $c \cdot m_2$。此时其生命值一定大于 $(c-1) \cdot m_2$。让剩余的非 m_1，m_2 随从对撞到剩余不超过 1 个，攻击任意一个 m_2，最后让所有 m_2 攻击 m_1 即可消灭所有随从，此时有解。
- 剩余的情况一定有 $c \cdot m_2 > m_1$。设 k 为 $\lceil \frac{m1}{m2} \rceil$。

- 情况讨论 5：如果 $c-k$ 为偶数，让 k 个 m_2 撞击 m_1。此时进入情况 1，一定有解。

J. 群体狂乱

- 情况讨论 5：如果 $c-k$ 为偶数，让 k 个 m_2 撞击 m_1。此时进入情况 1，一定有解。
- 情况讨论 6：如果 $c-k$ 为奇数，且此时所有其他随从的 a 总和超过 $m_1-(k-1)\cdot m_2$，让其他随从撞击 m_1 直到其生命值小于 $k\cdot m_2$ 为止。此时生命值一定大于等于 $(k-1)\cdot m_2$。此时让所有其他随从对撞到一个后攻击任意 m_2。最后让 $k-1$ 个 m_2 攻击 m_1，剩余 m_2 两两对撞，此时一定有解。

- 情况讨论 5：如果 $c-k$ 为偶数，让 k 个 m_2 撞击 m_1。此时进入情况 1，一定有解。
- 情况讨论 6：如果 $c-k$ 为奇数，且此时所有其他随从的 a 总和超过 $m_1-(k-1)\cdot m_2$，让其他随从撞击 m_1 直到其生命值小于 $k\cdot m_2$ 为止。此时生命值一定大于等于 $(k-1)\cdot m_2$。此时让所有其他随从对撞到一个后攻击任意 m_2。最后让 $k-1$ 个 m_2 攻击 m_1，剩余 m_2 两两对撞，此时一定有解。
- 情况讨论 7：如果 $c-k$ 为奇数，且此时所有其他随从的 a 总和不超过 $m_1-(k-1)\cdot m_2$，则其他随从无论如何都无法影响 m_1 与 m_2 间的攻击。此时一定无解。

- 情况讨论 5：如果 $c-k$ 为偶数，让 k 个 m_2 撞击 m_1。此时进入情况 1，一定有解。
- 情况讨论 6：如果 $c-k$ 为奇数，且此时所有其他随从的 a 总和超过 $m_1-(k-1)\cdot m_2$，让其他随从撞击 m_1 直到其生命值小于 $k\cdot m_2$ 为止。此时生命值一定大于等于 $(k-1)\cdot m_2$。此时让所有其他随从对撞到一个后攻击任意 m_2。最后让 $k-1$ 个 m_2 攻击 m_1，剩余 m_2 两两对撞，此时一定有解。
- 情况讨论 7：如果 $c-k$ 为奇数，且此时所有其他随从的 a 总和不超过 $m_1-(k-1)\cdot m_2$，则其他随从无论如何都无法影响 m_1 与 m_2 间的攻击。此时一定无解。
- 综上，所有情况讨论完毕。

J. 群体狂乱

- 情况讨论 5：如果 $c-k$ 为偶数，让 k 个 m_2 撞击 m_1。此时进入情况 1，一定有解。
- 情况讨论 6：如果 $c-k$ 为奇数，且此时所有其他随从的 a 总和超过 $m_1-(k-1)\cdot m_2$，让其他随从撞击 m_1 直到其生命值小于 $k\cdot m_2$ 为止。此时生命值一定大于等于 $(k-1)\cdot m_2$。此时让所有其他随从对撞到一个后攻击任意 m_2。最后让 $k-1$ 个 m_2 攻击 m_1，剩余 m_2 两两对撞，此时一定有解。
- 情况讨论 7：如果 $c-k$ 为奇数，且此时所有其他随从的 a 总和不超过 $m_1-(k-1)\cdot m_2$，则其他随从无论如何都无法影响 m_1 与 m_2 间的攻击。此时一定无解。
- 综上，所有情况讨论完毕。
- 总时间复杂度为 $O(n)$。

K. 取模

K. 取模

- 特判数组 a 中出现的不同数种类数小于等于 c 的情况。如果此时数组已经满足要求则答案为 -1，否则答案为 0。

K. 取模

- 特判数组 a 中出现的不同数种类数小于等于 c 的情况。如果此时数组已经满足要求则答案为 -1，否则答案为 0。
- 对于一个特定的 k，我们按照 $\lfloor \frac{a}{k} \rfloor$ 的值不同将 0 到 m 的值域分为 $\lfloor \frac{m}{k} \rfloor + 1$ 类。由于每类中只有不超过 c 种不同的数，所以说此时最多只有 $O(\frac{mc}{k})$ 种不同的数。反之，如果数组中有 cnt 种不同的数，那么 k 不能超过 $O(\frac{mc}{cnt})$。预处理所有在数组 a 中出现的数的值和出现的次数，$O(cnt)$ 检查每一个 k 即可在 $O(mc)$ 的时间内找到所有满足条件的 k。

K. 取模

- 特判数组 a 中出现的不同数种类数小于等于 c 的情况。如果此时数组已经满足要求则答案为 -1，否则答案为 0。
- 对于一个特定的 k，我们按照 $\lfloor \frac{a}{k} \rfloor$ 的值不同将 0 到 m 的值域分为 $\lfloor \frac{m}{k} \rfloor + 1$ 类。由于每类中只有不超过 c 种不同的数，所以说此时最多只有 $O(\frac{mc}{k})$ 种不同的数。反之，如果数组中有 cnt 种不同的数，那么 k 不能超过 $O(\frac{mc}{cnt})$。预处理所有在数组 a 中出现的数的值和出现的次数，$O(cnt)$ 检查每一个 k 即可在 $O(mc)$ 的时间内找到所有满足条件的 k。
- 总时间复杂度为 $O(n + mc)$。

L. cats 的加减乘除

L. cats 的加减乘除

- 首先需要注意到填入加减的概率是一样的，对于任意一段非从头开始的由乘除构成的连续段，都可以改变前面一个符号从加变成减或者从减变成加使得两个表达式的值之和的这一段为 0，所以可以得到所有非从头开始的由加减隔开的段都是没有贡献的。

L. cats 的加减乘除

- 首先需要注意到填入加减的概率是一样的，对于任意一段非从头开始的由乘除构成的连续段，都可以改变前面一个符号从加变成减或者从减变成加使得两个表达式的值之和的这一段为 0，所以可以得到所有非从头开始的由加减隔开的段都是没有贡献的。
- 所以现在要算所有前缀中仅加入乘除的权值（这个前缀的所有情况的表达式的值的和）和方案数，其中方案数可以简单计算，为可以填符号的方案和后面 -1 的个数的阶乘的乘积。

- 下面考虑如何计算权值。如果其中没有 -1，那么长度为 k 的前缀的权值就是

$$a_1 \prod_{i=2}^{k} (a_i \frac{1}{a_i})$$

- 下面考虑如何计算权值。如果其中没有 -1，那么长度为 k 的前缀的权值就是

$$a_1 \prod_{i=2}^{k} (a_i \frac{1}{a_i})$$

- 如存在 -1，那么假设在整个排列中有 m 个数需要填入，且这 m 个数为 $b_1, b_2, \ldots, b_m$。

L. cats 的加减乘除

- 下面考虑如何计算权值。如果其中没有 -1，那么长度为 k 的前缀的权值就是

$$a_1 \prod_{i=2}^{k} (a_i \frac{1}{a_i})$$

- 如存在 -1，那么假设在整个排列中有 m 个数需要填入，且这 m 个数为 $b_1, b_2, \ldots, b_m$。
- 可以发现第一个数不能有除法，考虑第一个数不是 -1，那么前 k 个数中对于已经填的数仍然可以用上面的计算方式计算贡献，假设还需要填入 p 个数，那么权值为：

$$\prod_{i=1}^{m} (1 + (b_i + \frac{1}{b_i})x)[x^l]$$

- 下面考虑如何计算权值。如果其中没有 -1，那么长度为 k 的前缀的权值就是

$$a_1 \prod_{i=2}^{k} (a_i \frac{1}{a_i})$$

- 如存在 -1，那么假设在整个排列中有 m 个数需要填入，且这 m 个数为 $b_1, b_2, \ldots, b_m$。
- 可以发现第一个数不能有除法，考虑第一个数不是 -1，那么前 k 个数中对于已经填的数仍然可以用上面的计算方式计算贡献，假设还需要填入 p 个数，那么权值为：

$$\prod_{i=1}^{m} (1 + (b_i + \frac{1}{b_i})x)[x^l]$$

- 这部分可以通过分治 NTT 计算。

- 下面考虑第一个数也是 -1，那么需要钦定第一个数，因为这个数不能是除法，所以也可以得到权值为：

$$\sum_{i=1}^{m} \frac{xb_i}{(1+(b_i+\frac{1}{b_i})x)} \prod_{j=1}^{m}(1+(b_j+\frac{1}{b_j})x)[x^l]$$

- 下面考虑第一个数也是 -1，那么需要钦定第一个数，因为这个数不能是除法，所以也可以得到权值为：

$$\sum_{i=1}^{m} \frac{xb_i}{(1+(b_i+\frac{1}{b_i})x)} \prod_{j=1}^{m}(1+(b_j+\frac{1}{b_j})x)[x^l]$$

- 所以现在要计算的是

$$\sum_{i=1}^{m} \frac{xb_i}{(1+(b_i+\frac{1}{b_i})x)}$$

L. cats 的加减乘除

- 也就是要计算若干一次分式的和，假设要计算

$$\frac{F_1(x)}{G_1(x)}+\frac{F_2(x)}{G_2(x)}=\frac{F_1(x)G_2(x)+F_2(x)G_1(x)}{G_1(x)G_2(x)}$$

- 也就是要计算若干一次分式的和，假设要计算

$$\frac{F_1(x)}{G_1(x)}+\frac{F_2(x)}{G_2(x)}=\frac{F_1(x)G_2(x)+F_2(x)G_1(x)}{G_1(x)G_2(x)}$$

- 可以继续保留分式的形式，进行通分，得到的新的分式的分子与分母的次数也是不超过原两侧分式的次数相加，所以仍然可以分治 NTT，最后得到的分母恰好为

$$\prod_{j=1}^{m}(1+(b_j+\frac{1}{b_j})x)[x^l]$$

- 也就是要计算若干一次分式的和，假设要计算

$$\frac{F_1(x)}{G_1(x)}+\frac{F_2(x)}{G_2(x)}=\frac{F_1(x)G_2(x)+F_2(x)G_1(x)}{G_1(x)G_2(x)}$$

- 可以继续保留分式的形式，进行通分，得到的新的分式的分子与分母的次数也是不超过原两侧分式的次数相加，所以仍然可以分治 NTT，最后得到的分母恰好为

$$\prod_{j=1}^{m}(1+(b_j+\frac{1}{b_j})x)[x^l]$$

- 也就是最后要计算的多项式。

- 也就是要计算若干一次分式的和，假设要计算

$$\frac{F_1(x)}{G_1(x)}+\frac{F_2(x)}{G_2(x)}=\frac{F_1(x)G_2(x)+F_2(x)G_1(x)}{G_1(x)G_2(x)}$$

- 可以继续保留分式的形式，进行通分，得到的新的分式的分子与分母的次数也是不超过原两侧分式的次数相加，所以仍然可以分治 NTT，最后得到的分母恰好为

$$\prod_{j=1}^{m}(1+(b_j+\frac{1}{b_j})x)[x^l]$$

- 也就是最后要计算的多项式。
- 综上时间复杂度为 $O(n\log^2 n)$。