第 11 届 CCPC 中国大学生程序设计竞赛哈尔滨站简要题解

哈尔滨工业大学

November 11, 2025

前言

- **预计难度**：AGL $<$ BIJM $<$ CDFH $<$ E。
- **封榜前难度**：AGKLIJBMCFDE。

简要题意

令 $f(k)$ 为一个长为 n 的序列 a 的任选 k 个数之和的 $\gcd$（可以重复选）。
求 $f(k)$ 的最大值以及取得该最大值最小的 k。

简要题意

令 $f(k)$ 为一个长为 n 的序列 a 的任选 k 个数之和的 $\gcd$（可以重复选）。
求 $f(k)$ 的最大值以及取得该最大值最小的 k。

- 考虑直接计算 $f(k)$，可以发现 $f(k) = \gcd(ka_1, a_2 - a_1, \ldots, a_n - a_1)$。

简要题意

令 $f(k)$ 为一个长为 n 的序列 a 的任选 k 个数之和的 $\gcd$（可以重复选）。
求 $f(k)$ 的最大值以及取得该最大值最小的 k。

- 考虑直接计算 $f(k)$，可以发现 $f(k) = \gcd(ka_1, a_2 - a_1, \ldots, a_n - a_1)$。
- 因此，当 a 中所有元素相同时答案为无穷。否则答案就为 $ans = \gcd(a_2 - a_1, \ldots, a_n - a_1)$，最小的 k 为 $\mathrm{lcm}(a_1, ans)/a_1$。

简要题意

令 $f(k)$ 为一个长为 n 的序列 a 的任选 k 个数之和的 gcd（可以重复选）。
求 $f(k)$ 的最大值以及取得该最大值最小的 k。

- 考虑直接计算 $f(k)$，可以发现 $f(k) = \gcd(ka_1, a_2 - a_1, \ldots, a_n - a_1)$。
- 因此，当 a 中所有元素相同时答案为无穷。否则答案就为 $ans = \gcd(a_2 - a_1, \ldots, a_n - a_1)$，最小的 k 为 $\mathrm{lcm}(a_1, ans)/a_1$。
- $f(k)$ 值的由来：令 $t = f(k)$，显然 $t \mid \gcd(a_2 - a_1, a_3 - a_1, \ldots, a_n - a_1)$。同时显然 $t \mid ka_1$，故得证。

简要题意

令 $f(k)$ 为一个长为 n 的序列 a 的任选 k 个数之和的 gcd（可以重复选）。
求 $f(k)$ 的最大值以及取得该最大值最小的 k。

- 考虑直接计算 $f(k)$，可以发现 $f(k) = \gcd(ka_1, a_2 - a_1, \ldots, a_n - a_1)$。
- 因此，当 a 中所有元素相同时答案为无穷。否则答案就为 $ans = \gcd(a_2 - a_1, \ldots, a_n - a_1)$，最小的 k 为 $\mathrm{lcm}(a_1, ans)/a_1$。
- $f(k)$ 值的由来：令 $t = f(k)$，显然 $t \mid \gcd(a_2 - a_1, a_3 - a_1, \ldots, a_n - a_1)$。同时显然 $t \mid ka_1$，故得证。
- 充分性可由 $\gcd(a, b) = \gcd(a, a + b)$ 得到。

简要题意

令 $f(k)$ 为一个长为 n 的序列 a 的任选 k 个数之和的 gcd（可以重复选）。
求 $f(k)$ 的最大值以及取得该最大值最小的 k。

- 考虑直接计算 $f(k)$，可以发现 $f(k) = \gcd(ka_1, a_2 - a_1, \ldots, a_n - a_1)$。
- 因此，当 a 中所有元素相同时答案为无穷。否则答案就为 $ans = \gcd(a_2 - a_1, \ldots, a_n - a_1)$，最小的 k 为 $\mathrm{lcm}(a_1, ans)/a_1$。
- $f(k)$ 值的由来：令 $t = f(k)$，显然 $t \mid \gcd(a_2 - a_1, a_3 - a_1, \ldots, a_n - a_1)$。同时显然 $t \mid ka_1$，故得证。
- 充分性可由 $\gcd(a, b) = \gcd(a, a + b)$ 得到。

时间复杂度为 $O(n \log n)$。

简要题意

给定一个 $n \times m$ 雪堆，可以无代价向右方或下方铲雪。或花 1 代价无条件铲雪或加雪。求将所有位置都抹平的最小代价。

简要题意

给定一个 $n \times m$ 雪堆，可以无代价向右方或下方铲雪。或花 1 代价无条件铲雪或加雪。求将所有位置都抹平的最小代价。

- 从左上到右下进行贪心，对于一堆雪在保证其右端能够抹平的前提下尽量向下铲雪。

简要题意

给定一个 $n \times m$ 雪堆，可以无代价向右方或下方铲雪。或花 1 代价无条件铲雪或加雪。求将所有位置都抹平的最小代价。

- 从左上到右下进行贪心，对于一堆雪在保证其右端能够抹平的前提下尽量向下铲雪。
- 一行雪能够抹平的充要条件是，它们的高度和为 0，且任意前缀和大于等于 0。

简要题意

给定一个 $n \times m$ 雪堆，可以无代价向右方或下方铲雪。或花 1 代价无条件铲雪或加雪。求将所有位置都抹平的最小代价。

- 从左上到右下进行贪心，对于一堆雪在保证其右端能够抹平的前提下尽量向下铲雪。
- 一行雪能够抹平的充要条件是，它们的高度和为 0，且任意前缀和大于等于 0。
- 因此对行维护前缀和的后缀最小值即可。如果这个值小于 0，则需要补雪，否则可以将大于 0 的部分转移给下方。

简要题意

给定一个 $n \times m$ 雪堆，可以无代价向右方或下方铲雪。或花 1 代价无条件铲雪或加雪。求将所有位置都抹平的最小代价。

- 从左上到右下进行贪心，对于一堆雪在保证其右端能够抹平的前提下尽量向下铲雪。
- 一行雪能够抹平的充要条件是，它们的高度和为 0，且任意前缀和大于等于 0。
- 因此对行维护前缀和的后缀最小值即可。如果这个值小于 0，则需要补雪，否则可以将大于 0 的部分转移给下方。

时间复杂度为 $O(nm)$ 或 $O(nm \log(nm))$，取决于实现方式。预期都能通过该题。

简要题意

给定一个包含 k 个障碍物的 $n \times m$ 的网格图。现在考虑一条从第 1 列某行出发，到达第 m 列某行的不经过任何障碍的路径。对于每个障碍，有从上方通过和从下方通过两种情况。

对于所有 2^k 种情况，求出路径的最小值。

简要题意

给定一个包含 k 个障碍物的 $n \times m$ 的网格图。现在考虑一条从第 1 列某行出发，到达第 m 列某行的不经过任何障碍的路径。对于每个障碍，有从上方通过和从下方通过两种情况。

对于所有 2^k 种情况，求出路径的最小值。

- 按题意模拟，边 bfs 边记录障碍的经过状态即可。

简要题意

给定一个包含 k 个障碍物的 $n \times m$ 的网格图。现在考虑一条从第 1 列某行出发，到达第 m 列某行的不经过任何障碍的路径。对于每个障碍，有从上方通过和从下方通过两种情况。

对于所有 2^k 种情况，求出路径的最小值。

- 按题意模拟，边 bfs 边记录障碍的经过状态即可。
- 可以采用状态压缩去记录障碍的经过状态。

简要题意

给定一个包含 k 个障碍物的 $n \times m$ 的网格图。现在考虑一条从第 1 列某行出发，到达第 m 列某行的不经过任何障碍的路径。对于每个障碍，有从上方通过和从下方通过两种情况。

对于所有 2^k 种情况，求出路径的最小值。

- 按题意模拟，边 bfs 边记录障碍的经过状态即可。
- 可以采用状态压缩去记录障碍的经过状态。

时间复杂度 $O(nm2^k)$。

简要题意

一个二维矩形内的一个质点，给定其初速度。垂直于 x 轴的边以一定速度向外扩张，垂直于 y 轴的边以一定速度向内压缩。

该质点在碰到任何边后都会使垂直于该边的速度完全反向。求质点在压缩结束后所在的位置。

简要题意

一个二维矩形内的一个质点，给定其初速度。垂直于 x 轴的边以一定速度向外扩张，垂直于 y 轴的边以一定速度向内压缩。

该质点在碰到任何边后都会使垂直于该边的速度完全反向。求质点在压缩结束后所在的位置。

- 将 v_x,v_y 分开考虑，发现 y 轴坐标的变化是显然的，并且能计算出结束时间 T_{end}。

简要题意

一个二维矩形内的一个质点，给定其初速度。垂直于 x 轴的边以一定速度向外扩张，垂直于 y 轴的边以一定速度向内压缩。

该质点在碰到任何边后都会使垂直于该边的速度完全反向。求质点在压缩结束后所在的位置。

- 将 v_x,v_y 分开考虑，发现 y 轴坐标的变化是显然的，并且能计算出结束时间 T_{end}。
- 当左右两板速度都为 0 时，可以直接计算出答案。

简要题意

一个二维矩形内的一个质点，给定其初速度。垂直于 x 轴的边以一定速度向外扩张，垂直于 y 轴的边以一定速度向内压缩。

该质点在碰到任何边后都会使垂直于该边的速度完全反向。求质点在压缩结束后所在的位置。

- 将 v_x,v_y 分开考虑，发现 y 轴坐标的变化是显然的，并且能计算出结束时间 T_{end}。
- 当左右两板速度都为 0 时，可以直接计算出答案。
- 否则小球只会在左右方向碰撞 $O(\log(T_{end}))$ 的次数，因为每次小球从上一次碰撞到下一次碰撞经过的路程会至少增加 $1/20$。

简要题意

一个二维矩形内的一个质点，给定其初速度。垂直于 x 轴的边以一定速度向外扩张，垂直于 y 轴的边以一定速度向内压缩。

该质点在碰到任何边后都会使垂直于该边的速度完全反向。求质点在压缩结束后所在的位置。

- 将 v_x,v_y 分开考虑，发现 y 轴坐标的变化是显然的，并且能计算出结束时间 T_{end}。
- 当左右两板速度都为 0 时，可以直接计算出答案。
- 否则小球只会在左右方向碰撞 $O(\log(T_{end}))$ 的次数，因为每次小球从上一次碰撞到下一次碰撞经过的路程会至少增加 $1/20$。
- 因此暴力模拟左右碰撞即可。

简要题意

一个二维矩形内的一个质点，给定其初速度。垂直于 x 轴的边以一定速度向外扩张，垂直于 y 轴的边以一定速度向内压缩。

该质点在碰到任何边后都会使垂直于该边的速度完全反向。求质点在压缩结束后所在的位置。

- 将 v_x,v_y 分开考虑，发现 y 轴坐标的变化是显然的，并且能计算出结束时间 T_{end}。
- 当左右两板速度都为 0 时，可以直接计算出答案。
- 否则小球只会在左右方向碰撞 $O(\log(T_{end}))$ 的次数，因为每次小球从上一次碰撞到下一次碰撞经过的路程会至少增加 $1/20$。
- 因此暴力模拟左右碰撞即可。

单组时间复杂度 $O(\log(T_{end}))$。

简要题意

给定一个黑白六边形网格图，每次可以选择一个六边形并翻转与其相邻的六边形的颜色（不翻转自己）。要求判断一个网格图是否能通过若干次操作变成另一个。

简要题意

给定一个黑白六边形网格图，每次可以选择一个六边形并翻转与其相邻的六边形的颜色（不翻转自己）。要求判断一个网格图是否能通过若干次操作变成另一个。

- 结论：答案为 YES 当且仅当两图每个 x 坐标（y 坐标、z 坐标）相同的黑色格点数奇偶性都相同。

简要题意

给定一个黑白六边形网格图，每次可以选择一个六边形并翻转与其相邻的六边形的颜色（不翻转自己）。要求判断一个网格图是否能通过若干次操作变成另一个。

- 结论：答案为 YES 当且仅当两图每个 x 坐标（y 坐标、z 坐标）相同的黑色格点数奇偶性都相同。
- 必要性：显然单次操作，不会改变某个 x 坐标（y 坐标、z 坐标）相同的黑色格点数的奇偶性（由图可知每个坐标恰好翻转两个格点）。

简要题意

给定一个黑白六边形网格图，每次可以选择一个六边形并翻转与其相邻的六边形的颜色（不翻转自己）。要求判断一个网格图是否能通过若干次操作变成另一个。

- 结论：答案为 YES 当且仅当两图每个 x 坐标（y 坐标、z 坐标）相同的黑色格点数奇偶性都相同。
- 必要性：显然单次操作，不会改变某个 x 坐标（y 坐标、z 坐标）相同的黑色格点数的奇偶性（由图可知每个坐标恰好翻转两个格点）。
- 充分性：问题本质等价于将某个网格图变为全白（可以通过类似异或操作达成）。

简要题意

给定一个黑白六边形网格图，每次可以选择一个六边形并翻转与其相邻的六边形的颜色（不翻转自己）。要求判断一个网格图是否能通过若干次操作变成另一个。

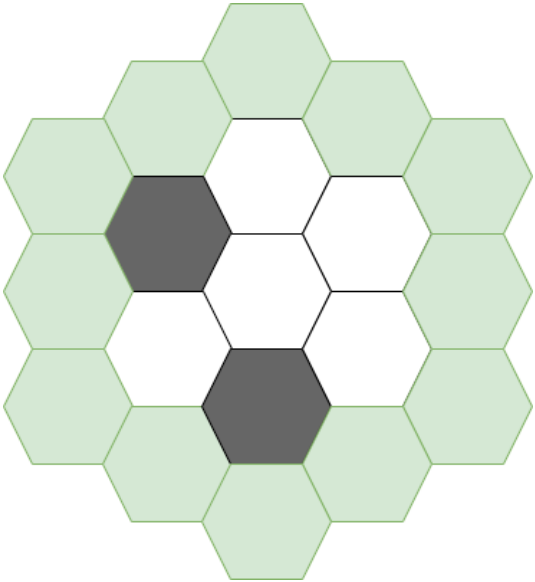
- 结论：答案为 YES 当且仅当两图每个 x 坐标（y 坐标、z 坐标）相同的黑色格点数奇偶性都相同。
- 必要性：显然单次操作，不会改变某个 x 坐标（y 坐标、z 坐标）相同的黑色格点数的奇偶性（由图可知每个坐标恰好翻转两个格点）。
- 充分性：问题本质等价于将某个网格图变为全白（可以通过类似异或操作达成）。
- 考虑从外向内一圈一圈地构造，每次最外圈（x 或 y 或 z 取得最大值）的六边形格点也会构成一个六边形。

简要题意

给定一个黑白六边形网格图，每次可以选择一个六边形并翻转与其相邻的六边形的颜色（不翻转自己）。要求判断一个网格图是否能通过若干次操作变成另一个。

- 结论：答案为 YES 当且仅当两图每个 x 坐标（y 坐标、z 坐标）相同的黑色格点数奇偶性都相同。
- 必要性：显然单次操作，不会改变某个 x 坐标（y 坐标、z 坐标）相同的黑色格点数的奇偶性（由图可知每个坐标恰好翻转两个格点）。
- 充分性：问题本质等价于将某个网格图变为全白（可以通过类似异或操作达成）。
- 考虑从外向内一圈一圈地构造，每次最外圈（x 或 y 或 z 取得最大值）的六边形格点也会构成一个六边形。
- 一种平凡的构造方法是，我们先翻转六个角上的黑色格子，再独立处理六个边上的黑色格点。由上述条件可知，每条边上的黑色格点数量都是偶数，因此一定能全部翻转为白色。

  时间复杂度 $O(n \log n)$。



简要题意

给定一个只有 ovw 的字符串 s，其中 w 会被视作 vv。求该字符串的最长回文子串。

简要题意

给定一个只有 ovw 的字符串 s，其中 w 会被视作 vv。求该字符串的最长回文子串。

- 一个合理的想法是，先将所有 w 变成两个 v（设为字符串 s'），然后做 manacher。

简要题意

给定一个只有 ovw 的字符串 s，其中 w 会被视作 vv。求该字符串的最长回文子串。

- 一个合理的想法是，先将所有 w 变成两个 v（设为字符串 s'），然后做 manacher。
- 该思路存在两个问题：
  - s' 最长的合法子串反过来不一定对应 s 最长的合法子串；
  - s' 的某个子串不一定能对应为 s 里的一个合法的字符串，例如 wov 在 s' 里为 vvov，最长回文串是 vov，但它不代表 s 的任何子串。

简要题意

给定一个只有 ovw 的字符串 s，其中 w 会被视作 vv。求该字符串的最长回文子串。

- 一个合理的想法是，先将所有 w 变成两个 v（设为字符串 s'），然后做 manacher。
- 该思路存在两个问题：
  - s' 最长的合法子串反过来不一定对应 s 最长的合法子串；
  - s' 的某个子串不一定能对应为 s 里的一个合法的字符串，例如 wov 在 s' 里为 vvov，最长回文串是 vov，但它不代表 s 的任何子串。
- 前者可以枚举每个回文中心求最长半径解决。

简要题意

给定一个只有 ovw 的字符串 s，其中 w 会被视作 vv。求该字符串的最长回文子串。

- 一个合理的想法是，先将所有 w 变成两个 v（设为字符串 s'），然后做 manacher。
- 该思路存在两个问题：
  - s' 最长的合法子串反过来不一定对应 s 最长的合法子串；
  - s' 的某个子串不一定能对应为 s 里的一个合法的字符串，例如 wov 在 s' 里为 vvov，最长回文串是 vov，但它不代表 s 的任何子串。
- 前者可以枚举每个回文中心求最长半径解决。
- 后者我们可以考虑不断删去两端的 w，直到遇到两端的第一个非 w 的字符即是该回文中心对应的**在 s 中存在的**最长回文子串。

- 因此我们枚举 s' 中的每个回文中心和最长回文半径，并通过不断缩 w 找到最长的在 s 中存在的回文子串即可。

- 因此我们枚举 s' 中的每个回文中心和最长回文半径，并通过不断缩 w 找到最长的在 s 中存在的回文子串即可。

时间复杂度 $O(|s|)$。

简要题意

给定一个 W_{lim}，要求构造一个物品数量，物品体积，物品价值尽量小的 01 背包，使得 2 到 W_{lim} 的背包 W 都无法通过贪心得到正确解。

简要题意

给定一个 W_{lim}，要求构造一个物品数量，物品体积，物品价值尽量小的 01 背包，使得 2 到 W_{lim} 的背包 W 都无法通过贪心得到正确解。

- **唯一的构造方法**（不妨假设 $n = W_{lim}$）：
  - 物品的体积：$1, n, n-1, \ldots, 2$。
  - 物品的价值：$n, n^2-1, n^2-n-2, \ldots, n+1$。

简要题意

给定一个 W_{lim}，要求构造一个物品数量，物品体积，物品价值尽量小的 01 背包，使得 2 到 W_{lim} 的背包 W 都无法通过贪心得到正确解。

- **唯一的构造方法**（不妨假设 $n = W_{lim}$）：
  - 物品的体积：$1, n, n-1, \ldots, 2$。
  - 物品的价值：$n, n^2-1, n^2-n-2, \ldots, n+1$。
- 首先证明排序后物品的体积一定是 $1, n, n-1, \ldots, 2$。

简要题意

给定一个 W_{lim}，要求构造一个物品数量，物品体积，物品价值尽量小的 01 背包，使得 2 到 W_{lim} 的背包 W 都无法通过贪心得到正确解。

- **唯一的构造方法**（不妨假设 $n = W_{lim}$）：
  - 物品的体积：$1, n, n-1, \ldots, 2$。
  - 物品的价值：$n, n^2-1, n^2-n-2, \ldots, n+1$。
- 首先证明排序后物品的体积一定是 $1, n, n-1, \ldots, 2$。
- $n = 2$ 时显然唯一的方案 $1, 2$（$1, 2, 1$ 后面的 1 无用）。

简要题意

给定一个 W_{lim}，要求构造一个物品数量，物品体积，物品价值尽量小的 01 背包，使得 2 到 W_{lim} 的背包 W 都无法通过贪心得到正确解。

- **唯一的构造方法**（不妨假设 $n = W_{lim}$）：
  - 物品的体积：$1, n, n-1, \ldots, 2$。
  - 物品的价值：$n, n^2-1, n^2-n-2, \ldots, n+1$。
- 首先证明排序后物品的体积一定是 $1, n, n-1, \ldots, 2$。
- $n = 2$ 时显然唯一的方案 $1, 2$（$1, 2, 1$ 后面的 1 无用）。
- 接下来考虑归纳，假设对于 $n-1$ 唯一的方案是 $1, n-1, \ldots, 2$。此时 n 显然能通过贪心得到最优解。

简要题意

给定一个 W_{lim}，要求构造一个物品数量，物品体积，物品价值尽量小的 01 背包，使得 2 到 W_{lim} 的背包 W 都无法通过贪心得到正确解。

- **唯一的构造方法**（不妨假设 $n = W_{lim}$）：
  - 物品的体积：$1, n, n-1, \ldots, 2$。
  - 物品的价值：$n, n^2-1, n^2-n-2, \ldots, n+1$。
- 首先证明排序后物品的体积一定是 $1, n, n-1, \ldots, 2$。
- $n = 2$ 时显然唯一的方案 $1, 2$（$1, 2, 1$ 后面的 1 无用）。
- 接下来考虑归纳，假设对于 $n-1$ 唯一的方案是 $1, n-1, \ldots, 2$。此时 n 显然能通过贪心得到最优解。
- 由于唯一性，我们无法插入体积小于等于 $n-1$ 的物品，故只能在 1 后插入 n 变为 $1, n, n-1, \ldots, 2$。

- 接下来考虑物品的价值。物品的价值要满足：
  - $v_1 > v_n/n > v_{n-1}/(n-1) > \ldots > v_2/2$。
  - $\forall 1 < i < n, v_i + v_1 < v_{i+1},\ v_1 < v_2$。

- 接下来考虑物品的价值。物品的价值要满足：
  - $v_1 > v_n/n > v_{n-1}/(n-1) > \ldots > v_2/2$。
  - $\forall 1 < i < n, v_i + v_1 < v_{i+1}$，$v_1 < v_2$。
- 可以发现，在后者成立以及 1 在最前面的前提下，前者显然成立。所以我们就是要找一个最小的 v_1 使得 $(v_n = v_1 + 1 + (n-2) \cdot (v_1 + 1)) < n \cdot v_1$。
  即 $v_1 > n - 1$。故 $v_1 = n$，也就可以推得 $v_2, v_3, \ldots, v_n$ 的值。

简要题意

给定一个大小为 n 的正三角形网格图的。每个方向的路径方向相同，求图中有多少个连通的正三角形。

- 假设一个正三角形是由右斜方向上长度为 i，左斜方向上长度为 j，水平方向上长度为 k 的三条路径相交构成的。

简要题意

给定一个大小为 n 的正三角形网格图的。每个方向的路径方向相同，求图中有多少个连通的正三角形。

- 假设一个正三角形是由右斜方向上长度为 i，左斜方向上长度为 j，水平方向上长度为 k 的三条路径相交构成的。
- 这三条路径能确定一个正三角形的充要条件为：
  - $i+j \geq n, j+k \geq n, i+k \geq n$。（否则两个路径根本没有交）
  - $i+j+k \neq 2n$。（如果小于 $2n$ 是正着的三角形，大于 $2n$ 是倒着的正三角形，而等于 $2n$ 会退化为一个点）。

简要题意

给定一个大小为 n 的正三角形网格图的。每个方向的路径方向相同，求图中有多少个连通的正三角形。

- 假设一个正三角形是由右斜方向上长度为 i，左斜方向上长度为 j，水平方向上长度为 k 的三条路径相交构成的。
- 这三条路径能确定一个正三角形的充要条件为：
  - $i+j \geq n, j+k \geq n, i+k \geq n$。（否则两个路径根本没有交）
  - $i+j+k \neq 2n$。（如果小于 $2n$ 是正着的三角形，大于 $2n$ 是倒着的正三角形，而等于 $2n$ 会退化为一个点）。
- 可以先忽略第二个条件，计算第一个条件的方案数量。可以考虑钦定 $i = \min(i, j, k)$ 并枚举 i。此时我们知道 j, k 的下界，可以直接通过维护后缀和算出方案数量。

简要题意

给定一个大小为 n 的正三角形网格图的。每个方向的路径方向相同，求图中有多少个连通的正三角形。

- 假设一个正三角形是由右斜方向上长度为 i，左斜方向上长度为 j，水平方向上长度为 k 的三条路径相交构成的。
- 这三条路径能确定一个正三角形的充要条件为：
  - $i+j \geq n, j+k \geq n, i+k \geq n$。（否则两个路径根本没有交）
  - $i+j+k \neq 2n$。（如果小于 $2n$ 是正着的三角形，大于 $2n$ 是倒着的正三角形，而等于 $2n$ 会退化为一个点）。
- 可以先忽略第二个条件，计算第一个条件的方案数量。可以考虑钦定 $i = \min(i,j,k)$ 并枚举 i。此时我们知道 j,k 的下界，可以直接通过维护后缀和算出方案数量。
- 再考虑减去第二个条件的答案数量。第二个条件满足时第一个条件显然满足，所以可以使用 FFT 去计算。

简要题意

给定一个大小为 n 的正三角形网格图的。每个方向的路径方向相同，求图中有多少个连通的正三角形。

- 假设一个正三角形是由右斜方向上长度为 i，左斜方向上长度为 j，水平方向上长度为 k 的三条路径相交构成的。
- 这三条路径能确定一个正三角形的充要条件为：
  - $i+j \geq n, j+k \geq n, i+k \geq n$。（否则两个路径根本没有交）
  - $i+j+k \neq 2n$。（如果小于 $2n$ 是正着的三角形，大于 $2n$ 是倒着的正三角形，而等于 $2n$ 会退化为一个点）。
- 可以先忽略第二个条件，计算第一个条件的方案数量。可以考虑钦定 $i = \min(i,j,k)$ 并枚举 i。此时我们知道 j,k 的下界，可以直接通过维护后缀和算出方案数量。
- 再考虑减去第二个条件的答案数量。第二个条件满足时第一个条件显然满足，所以可以使用 FFT 去计算。

时间复杂度 $O(n \log n)$。

时间复杂度 $O(n \log n)$。

Bonus：How $O(n)$?
（提示：现在的思路难以避免 FFT 条件，因此应该去统计不合法的数量）

- 在最基础的版本中，可以认为操作没有非包含的相交关系（除了端点相交）。

- 在最基础的版本中，可以认为操作没有非包含的相交关系（除了端点相交）。
- 结论：最优的代价为 $\sum_{i=1}^{n} \max(a_i - a_{i-1}, 0)b_i + \max(a_i - a_{i+1}, 0)c_i$。

- 在最基础的版本中，可以认为操作没有非包含的相交关系（除了端点相交）。
- 结论：最优的代价为 $\sum_{i=1}^{n} \max(a_i - a_{i-1}, 0)b_i + \max(a_i - a_{i+1}, 0)c_i$。
- 其实非常好理解，因为以 i 为左端点的区间必须至少得有 $\max(a_i - a_{i-1}, 0)$ 个，右端点同理，而刚好可以构造出这种方案，所以这就是最优解。

- 在最基础的版本中，可以认为操作没有非包含的相交关系（除了端点相交）。
- 结论：最优的代价为 $\sum_{i=1}^{n} \max(a_i - a_{i-1}, 0)b_i + \max(a_i - a_{i+1}, 0)c_i$。
- 其实非常好理解，因为以 i 为左端点的区间必须至少得有 $\max(a_i - a_{i-1}, 0)$ 个，右端点同理，而刚好可以构造出这种方案，所以这就是最优解。
- 第二个问题的答案其实跟第一个问题的答案是一样的，因为一旦有 $\max(a_i - a_{i-1}, 0) > d_i / \max(a_i - a_{i+1}, 0) > e_i$ 那一直加这个点就会使得答案是无穷大，否则肯定不加。

- 在最基础的版本中，可以认为操作没有非包含的相交关系（除了端点相交）。
- 结论：最优的代价为 $\sum_{i=1}^{n} \max(a_i - a_{i-1}, 0)b_i + \max(a_i - a_{i+1}, 0)c_i$。
- 其实非常好理解，因为以 i 为左端点的区间必须至少得有 $\max(a_i - a_{i-1}, 0)$ 个，右端点同理，而刚好可以构造出这种方案，所以这就是最优解。
- 第二个问题的答案其实跟第一个问题的答案是一样的，因为一旦有 $\max(a_i - a_{i-1}, 0) > d_i / \max(a_i - a_{i+1}, 0) > e_i$ 那一直加这个点就会使得答案是无穷大，否则肯定不加。
- 于是问题就非常好计数了，设 $f_{i,j}$ 表示确定了 $a[i,n], b[i+1,n], c[i,n], d[i+1,n], e[i,n]$ 且 $a_i = j$ 的合法的方案数，$g_{i,j}$ 表示所有合法方案对应的权值和。转移使用前缀和优化即可做到 $O(nV)$。

简要题意

给定两个长为 n 的递增数组 l, r，以及一个固定的常数 k。

q 次询问，每次询问给定一个区间 L, R 以及常数 D，询问能否将这些人分为 k 组，使得每组内部存在一种安排 $x_i \in [l_i, r_i]$ 的方式，使得组内 $|x_i - x_j|(i \neq j)$ 的最小值不小于 D。

简要题意

给定两个长为 n 的递增数组 l, r，以及一个固定的常数 k。

q 次询问，每次询问给定一个区间 L, R 以及常数 D，询问能否将这些人分为 k 组，使得每组内部存在一种安排 $x_i \in [l_i, r_i]$ 的方式，使得组内 $|x_i - x_j|(i \neq j)$ 的最小值不小于 D。

- 假设 $l_i = r_i$。可以发现分组方式一定是 $[L, L+k, L+2k, \ldots], [L+1, L+1+k, L+1+2k \ldots], \ldots$。即 $\bmod k$ 相同的位置安排在同一组。

简要题意

给定两个长为 n 的递增数组 l, r，以及一个固定的常数 k。

q 次询问，每次询问给定一个区间 L, R 以及常数 D，询问能否将这些人分为 k 组，使得每组内部存在一种安排 $x_i \in [l_i, r_i]$ 的方式，使得组内 $|x_i - x_j|(i \neq j)$ 的最小值不小于 D。

- 假设 $l_i = r_i$。可以发现分组方式一定是 $[L, L+k, L+2k, \ldots], [L+1, L+1+k, L+1+2k \ldots], \ldots$。即 $\bmod k$ 相同的位置安排在同一组。
- 假设 $k = 1$，可以发现问题相当于判断 $\forall i < j, (r_j - l_i)/(j - i) \geq D$。

简要题意

给定两个长为 n 的递增数组 l, r，以及一个固定的常数 k。

q 次询问，每次询问给定一个区间 L, R 以及常数 D，询问能否将这些人分为 k 组，使得每组内部存在一种安排 $x_i \in [l_i, r_i]$ 的方式，使得组内 $|x_i - x_j|(i \neq j)$ 的最小值不小于 D。

- 假设 $l_i = r_i$。可以发现分组方式一定是 $[L, L+k, L+2k, \ldots], [L+1, L+1+k, L+1+2k \ldots], \ldots$。即 $\bmod k$ 相同的位置安排在同一组。
- 假设 $k = 1$，可以发现问题相当于判断 $\forall i < j, (r_j - l_i)/(j - i) \geq D$。
- 综上，实际上每次询问就是判断区间内 $mod k$ 相同的所有 $i < j$，是否满足 $r_j - jD \geq l_i - iD$。这个形式有多种可以维护的数据结构，下面只讲 std 的实现。

- 考虑一个类似线段树的静态分治结构。对于代表区间 $[l, r]$ 的节点 x，先预处理 val_x 表示 $[l, r]$ 内合法最大的 D。

- 考虑一个类似线段树的静态分治结构。对于代表区间 $[l,r]$ 的节点 x，先预处理 val_x 表示 $[l,r]$ 内合法最大的 D。
- val_x 可以通过先与 val_{ls_x} 和 val_{rs_x} 取 $\min$，然后对 $[l,mid]$ 和 $[mid+1,r]$ 分别构建凸包与进行询问求得。

- 考虑一个类似线段树的静态分治结构。对于代表区间 $[l,r]$ 的节点 x，先预处理 val_x 表示 $[l,r]$ 内合法最大的 D。
- val_x 可以通过先与 val_{ls_x} 和 val_{rs_x} 取 $\min$，然后对 $[l,mid]$ 和 $[mid+1,r]$ 分别构建凸包与进行询问求得。
- 此部分若在凸包求切线时使用二分则时间复杂度为 $O(n\log n\log D)$，若使用双指针则为 $O(n\log n)$。

- 考虑一个类似线段树的静态分治结构。对于代表区间 $[l,r]$ 的节点 x，先预处理 val_x 表示 $[l,r]$ 内合法最大的 D。
- val_x 可以通过先与 val_{ls_x} 和 val_{rs_x} 取 $\min$，然后对 $[l,mid]$ 和 $[mid+1,r]$ 分别构建凸包与进行询问求得。
- 此部分若在凸包求切线时使用二分则时间复杂度为 $O(n\log n\log D)$，若使用双指针则为 $O(n\log n)$。
- 对于询问 $[L,R]$，我们可以类似于线段树询问一样，将所有询问插入到横跨中点的节点上 $(l\le L\le mid<R\le r)$。当递归到某个整区间时直接返回 val_x。

- 考虑一个类似线段树的静态分治结构。对于代表区间 $[l,r]$ 的节点 x，先预处理 val_x 表示 $[l,r]$ 内合法最大的 D。
- val_x 可以通过先与 val_{ls_x} 和 val_{rs_x} 取 $\min$，然后对 $[l,mid]$ 和 $[mid+1,r]$ 分别构建凸包与进行询问求得。
- 此部分若在凸包求切线时使用二分则时间复杂度为 $O(n\log n\log D)$，若使用双指针则为 $O(n\log n)$。
- 对于询问 $[L,R]$，我们可以类似于线段树询问一样，将所有询问插入到横跨中点的节点上 $(l\le L\le mid<R\le r)$。当递归到某个整区间时直接返回 val_x。
- 而对于插在某个节点的询问，由于 D 固定，可以分别求 $[l,mid]$ 的凸包和 $[mid+1,r]$ 的凸包并询问 $k=D$ 求极值。再对极值进行比较。

- 考虑一个类似线段树的静态分治结构。对于代表区间 $[l,r]$ 的节点 x，先预处理 val_x 表示 $[l,r]$ 内合法最大的 D。
- val_x 可以通过先与 val_{ls_x} 和 val_{rs_x} 取 $\min$，然后对 $[l,mid]$ 和 $[mid+1,r]$ 分别构建凸包与进行询问求得。
- 此部分若在凸包求切线时使用二分则时间复杂度为 $O(n\log n\log D)$，若使用双指针则为 $O(n\log n)$。
- 对于询问 $[L,R]$，我们可以类似于线段树询问一样，将所有询问插入到横跨中点的节点上 $(l\le L\le mid<R\le r)$。当递归到某个整区间时直接返回 val_x。
- 而对于插在某个节点的询问，由于 D 固定，可以分别求 $[l,mid]$ 的凸包和 $[mid+1,r]$ 的凸包并询问 $k=D$ 求极值。再对极值进行比较。
- 这个过程每个询问会插到 $\log n$ 个节点上，每个节点上都会有一次二分求极值。故该部分时间复杂度为 $O(qk\log^2 n)$。

- 考虑一个类似线段树的静态分治结构。对于代表区间 $[l,r]$ 的节点 x，先预处理 val_x 表示 $[l,r]$ 内合法最大的 D。
- val_x 可以通过先与 val_{ls_x} 和 val_{rs_x} 取 $\min$，然后对 $[l,mid]$ 和 $[mid+1,r]$ 分别构建凸包与进行询问求得。
- 此部分若在凸包求切线时使用二分则时间复杂度为 $O(n\log n\log D)$，若使用双指针则为 $O(n\log n)$。
- 对于询问 $[L,R]$，我们可以类似于线段树询问一样，将所有询问插入到横跨中点的节点上 $(l\le L\le mid<R\le r)$。当递归到某个整区间时直接返回 val_x。
- 而对于插在某个节点的询问，由于 D 固定，可以分别求 $[l,mid]$ 的凸包和 $[mid+1,r]$ 的凸包并询问 $k=D$ 求极值。再对极值进行比较。
- 这个过程每个询问会插到 $\log n$ 个节点上，每个节点上都会有一次二分求极值。故该部分时间复杂度为 $O(qk\log^2 n)$。

总时间复杂度为 $O(n\log n+qk\log^2 n)$ 或 $O(n\log n\log W+qk\log^2 n)$。实际上时间相对宽松，只要是 $\log$ 做法应该都能过。

简要题意

给定一个只有 1,2 的数组，求相邻按位或操作下的本质不同序列数量。**保证相同数字段长度单调不降**。

简要题意

给定一个只有 1,2 的数组，求相邻按位或操作下的本质不同序列数量。**保证相同数字段长度单调不降**。

- 考虑子序列自动机。即对于一个序列，我们希望找一个最小前缀使得通过操作后能变成该序列。并考虑在该序列之后加一个数之后新的最小前缀。

简要题意

给定一个只有 1,2 的数组，求相邻按位或操作下的本质不同序列数量。**保证相同数字段长度单调不降**。

- 考虑子序列自动机。即对于一个序列，我们希望找一个最小前缀使得通过操作后能变成该序列。并考虑在该序列之后加一个数之后新的最小前缀。
- 如果当前序列中没有 3，则转移很固定。如果当前是 1（或 2）的末尾，则不能填 1（或 2），否则填 1 转移到下一位，填 2 或 3 转移到下一段开头。

简要题意

给定一个只有 1,2 的数组，求相邻按位或操作下的本质不同序列数量。**保证相同数字段长度单调不降**。

- 考虑子序列自动机。即对于一个序列，我们希望找一个最小前缀使得通过操作后能变成该序列。并考虑在该序列之后加一个数之后新的最小前缀。
- 如果当前序列中没有 3，则转移很固定。如果当前是 1（或 2）的末尾，则不能填 1（或 2），否则填 1 转移到下一位，填 2 或 3 转移到下一段开头。
- 如果当前序列有 3，由于 3 可以删去 3 后面的任意长度序列，所以如果当前是 1（或 2）的末尾，也是可以填 1 的，这会转移到下一段 1 的相同位置或下一个位置。

简要题意

给定一个只有 1,2 的数组，求相邻按位或操作下的本质不同序列数量。**保证相同数字段长度单调不降**。

- 考虑子序列自动机。即对于一个序列，我们希望找一个最小前缀使得通过操作后能变成该序列。并考虑在该序列之后加一个数之后新的最小前缀。
- 如果当前序列中没有 3，则转移很固定。如果当前是 1（或 2）的末尾，则不能填 1（或 2），否则填 1 转移到下一位，填 2 或 3 转移到下一段开头。
- 如果当前序列有 3，由于 3 可以删去 3 后面的任意长度序列，所以如果当前是 1（或 2）的末尾，也是可以填 1 的，这会转移到下一段 1 的相同位置或下一个位置。
- 由于序列长度不降，这个转移是合法的。

简要题意

给定一个只有 1,2 的数组，求相邻按位或操作下的本质不同序列数量。**保证相同数字段长度单调不降**。

- 考虑子序列自动机。即对于一个序列，我们希望找一个最小前缀使得通过操作后能变成该序列。并考虑在该序列之后加一个数之后新的最小前缀。
- 如果当前序列中没有 3，则转移很固定。如果当前是 1（或 2）的末尾，则不能填 1（或 2），否则填 1 转移到下一位，填 2 或 3 转移到下一段开头。
- 如果当前序列有 3，由于 3 可以删去 3 后面的任意长度序列，所以如果当前是 1（或 2）的末尾，也是可以填 1 的，这会转移到下一段 1 的相同位置或下一个位置。
- 由于序列长度不降，这个转移是合法的。
- 根据这个思路写 DP 即可，细节较多。

简要题意

给定一个只有 1,2 的数组，求相邻按位或操作下的本质不同序列数量。**保证相同数字段长度单调不降**。

- 考虑子序列自动机。即对于一个序列，我们希望找一个最小前缀使得通过操作后能变成该序列。并考虑在该序列之后加一个数之后新的最小前缀。
- 如果当前序列中没有 3，则转移很固定。如果当前是 1（或 2）的末尾，则不能填 1（或 2），否则填 1 转移到下一位，填 2 或 3 转移到下一段开头。
- 如果当前序列有 3，由于 3 可以删去 3 后面的任意长度序列，所以如果当前是 1（或 2）的末尾，也是可以填 1 的，这会转移到下一段 1 的相同位置或下一个位置。
- 由于序列长度不降，这个转移是合法的。
- 根据这个思路写 DP 即可，细节较多。

时间复杂度 $O(n)$。

- 用势能法处理，每个集合只与自身数量有关，与其他集合的状态无关，所以设 $f(a)$ 表示一个有 a 个元素的集合的势能函数。

- 用势能法处理，每个集合只与自身数量有关，与其他集合的状态无关，所以设 $f(a)$ 表示一个有 a 个元素的集合的势能函数。
- 设当前状态下每个集合的元素分别为 $b_1, b_2, ..., b_{n_0}$ ；所有可能的配对方案为 S ；在第 i 种配对方案操作后，每个集合的元素变化为 $b_1^i, b_2^i, ..., b_{n_i}^i$ 。

- 用势能法处理，每个集合只与自身数量有关，与其他集合的状态无关，所以设 $f(a)$ 表示一个有 a 个元素的集合的势能函数。
- 设当前状态下每个集合的元素分别为 $b_1, b_2, ..., b_{n_0}$ ；所有可能的配对方案为 S ；在第 i 种配对方案操作后，每个集合的元素变化为 $b_1^i, b_2^i, ..., b_{n_i}^i$ 。
- 则有 $1 + \sum_{i=1}^{n_0} f(b_i) = \frac{1}{|S|} \sum_{i=1}^{|S|} \sum_{j=1}^{n_i} f(b_j^i)$ ，原问题的答案即 $f(m) - \sum_{i=1}^{n} f(a_i)$。

- 用势能法处理，每个集合只与自身数量有关，与其他集合的状态无关，所以设 $f(a)$ 表示一个有 a 个元素的集合的势能函数。
- 设当前状态下每个集合的元素分别为 $b_1, b_2, ..., b_{n_0}$ ；所有可能的配对方案为 S ；在第 i 种配对方案操作后，每个集合的元素变化为 $b_1^i, b_2^i, ..., b_{n_i}^i$ 。
- 则有 $1 + \sum_{i=1}^{n_0} f(b_i) = \frac{1}{|S|} \sum_{i=1}^{|S|} \sum_{j=1}^{n_i} f(b_j^i)$ ，原问题的答案即 $f(m) - \sum_{i=1}^{n} f(a_i)$。
- 利用高斯消元求解 $f(a)$ 即可，对 $f(a)$ 列方程时只需考虑一个有 a 个元素的集合在一次操作后集合中元素的数量变化。

- min25 筛子练习题。

- min25 筛子练习题。
- 第一种做法是 dls 验题时写的，跟传统 min25 一样，搜索时最后一个质数单独处理即可。复杂度完全算不了，但是就是能过，不过这个做法超级难写。

- min25 筛子练习题。
- 第一种做法是 dls 验题时写的，跟传统 min25 一样，搜索时最后一个质数单独处理即可。复杂度完全算不了，但是就是能过，不过这个做法超级难写。
- 还是给一个有道理的做法：考虑 min25 $n^{1-\epsilon}$ 的搜索时把所有状态全部存下来，对其做高维后缀和。但是你会发现会在询问的 x 的最大质因子次数为 1 时算少。于是可以想到对 x 的最大质因子进行根号分治，考虑阈值 B，最大质因子 $p \le B$ 的每次重新做高维后缀和，否则可以转化为一维大小在 n/B 以内的二维数点问题，将大的这一维离线下来后使用一个 $O(1)$ 加，$O(\sqrt{n})$ 查询的分块维护即可，另一维的大小为 $\sqrt{n}$，所以单次查询复杂度是 $O(n^{1/4})$。

- min25 筛子练习题。
- 第一种做法是 dls 验题时写的，跟传统 min25 一样，搜索时最后一个质数单独处理即可。复杂度完全算不了，但是就是能过，不过这个做法超级难写。
- 还是给一个有道理的做法：考虑 min25 $n^{1-\epsilon}$ 的搜索时把所有状态全部存下来，对其做高维后缀和。但是你会发现会在询问的 x 的最大质因子次数为 1 时算少。于是可以想到对 x 的最大质因子进行根号分治，考虑阈值 B，最大质因子 $p \le B$ 的每次重新做高维后缀和，否则可以转化为一维大小在 n/B 以内的二维数点问题，将大的这一维离线下来后使用一个 $O(1)$ 加，$O(\sqrt{n})$ 查询的分块维护即可，另一维的大小为 $\sqrt{n}$，所以单次查询复杂度是 $O(n^{1/4})$。
- 取合适的 B 可以通过此题，时限开得算是很大了，不卡常数。