Constructive Algorithms

Charlie

2023 年 7 月 8 日

引言

在近年的算法竞赛中，构造题的出现越来越频繁。不同于传统的计数、最优化等问题，构造题只要求选手给出一组满足约束条件的解，而不需要统计解的数量，或是寻找一组“最优”的解。然而，由于其模型繁多，涉及图论、数论、字符串等各领域，且常常难以发现，要解决起来并不容易。本课件对构造题中较常出现的一些解题思路进行了介绍，并给出了例题和讲解，希望对读者有所启发，在解决构造题时能更加得心应手。

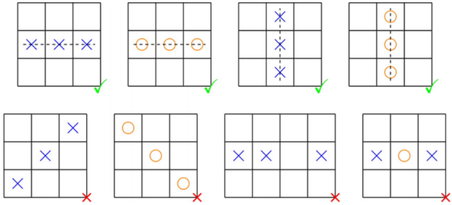
抽屉原理

- 抽屉原理, 或称为鸽巢原理, 是组合数学中一个非常重要的原理。通常的表述是, 若将 n 件物品放入 k 个抽屉, 则其中一定有一个抽屉包含至少 $\lceil \frac{n}{k} \rceil$ 件物品, 也一定有一个抽屉包含至多 $\lfloor \frac{n}{k} \rfloor$ 件物品。
- 在一些构造题中, 常常会要求构造一个权值至少为 (或不超过) 某一个数的方案。很多时候, 可以考虑找出若干个可行的方案, 使得它们的权值之和是定值。假设找出了 k 个可行方案, 其总权值和为 n, 由抽屉原理, 这些方案中最小的权值一定不超过 $\lfloor \frac{n}{k} \rfloor$, 最大的权值至少为 $\lceil \frac{n}{k} \rceil$ 。

CF1450C2 Errich-Tac-Toe

Statement

- 给定一张 n 行 n 列的棋盘，每个格子可能是空的或包含一个标志，标志有 X 和 O 两种。如果有三个相同的标志排列在一行或一列上的三个连续的位置，则称这个棋盘是一个胜局，否则称其为平局。



- 例如，上图第一行的局面都是胜局，而第二行的局面都是平局。
- 在一次操作中，你可以将一个 X 改成 O，或将一个 O 改成 X。
- 设棋盘中标志的总数为 k，你需要用不超过 $\lfloor \frac{k}{3} \rfloor$ 次操作把给定的局面变成平局。
- $1 \leq n \leq 300$。

- 不妨将行列都用 $0, 1, \ldots, n-1$ 编号, 将第 r 行第 c 列的格子记为 (r, c) 。我们将所有格子分成 3 类, 其中第 $i (0 \le i < 3)$ 类包含所有满足 $r + c \equiv i \bmod 3$ 的格子 (r, c) 。不难发现, 在一行或一列上的连续三个格子包含第 $0, 1, 2$ 类格子各一个。

- 不妨将行列都用 $0, 1, \ldots, n-1$ 编号, 将第 r 行第 c 列的格子记为 (r, c) 。我们将所有格子分成 3 类, 其中第 $i (0 \le i < 3)$ 类包含所有满足 $r + c \equiv i \bmod 3$ 的格子 (r, c) 。不难发现, 在一行或一列上的连续三个格子包含第 $0, 1, 2$ 类格子各一个。
- 由此, 不难想到以下的几种操作方案:

- 不妨将行列都用 $0, 1, \ldots, n-1$ 编号，将第 r 行第 c 列的格子记为 (r, c) 。我们将所有格子分成 3 类，其中第 $i(0 \le i < 3)$ 类包含所有满足 $r + c \equiv i \bmod 3$ 的格子 (r, c) 。不难发现，在一行或一列上的连续三个格子包含第 $0, 1, 2$ 类格子各一个。
- 由此，不难想到以下的几种操作方案:
-
  - 将第 0 类格子上的 x 都改成 0，将第 1 类格子上的 0 都改成 x 。

- 不妨将行列都用 $0, 1, \ldots, n-1$ 编号，将第 r 行第 c 列的格子记为 (r, c) 。我们将所有格子分成 3 类，其中第 $i(0 \le i < 3)$ 类包含所有满足 $r + c \equiv i \bmod 3$ 的格子 (r, c) 。不难发现，在一行或一列上的连续三个格子包含第 $0, 1, 2$ 类格子各一个。
- 由此，不难想到以下的几种操作方案:
-
  - 将第 0 类格子上的 x 都改成 0 ，将第 1 类格子上的 0 都改成 x 。
  - 将第 1 类格子上的 x 都改成 0 ，将第 2 类格子上的 0 都改成 x 。

- 不妨将行列都用 $0,1,\ldots,n-1$ 编号, 将第 r 行第 c 列的格子记为 (r,c) 。我们将所有格子分成 3 类, 其中第 $i(0 \le i < 3)$ 类包含所有满足 $r+c \equiv i \bmod 3$ 的格子 (r,c) 。不难发现, 在一行或一列上的连续三个格子包含第 $0,1,2$ 类格子各一个。
- 由此, 不难想到以下的几种操作方案:
-
  - 将第 0 类格子上的 x 都改成 0 , 将第 1 类格子上的 0 都改成 x 。
  - 将第 1 类格子上的 x 都改成 0 , 将第 2 类格子上的 0 都改成 x 。
  - 将第 2 类格子上的 x 都改成 0 , 将第 0 类格子上的 0 都改成 X 。

- 不妨将行列都用 $0, 1, \ldots, n-1$ 编号, 将第 r 行第 c 列的格子记为 (r, c) 。我们将所有格子分成 3 类, 其中第 $i (0 \le i < 3)$ 类包含所有满足 $r + c \equiv i \bmod 3$ 的格子 (r, c) 。不难发现, 在一行或一列上的连续三个格子包含第 $0, 1, 2$ 类格子各一个。
- 由此, 不难想到以下的几种操作方案:
-
  - 将第 0 类格子上的 x 都改成 0 , 将第 1 类格子上的 0 都改成 x 。
  - 将第 1 类格子上的 x 都改成 0 , 将第 2 类格子上的 0 都改成 x 。
  - 将第 2 类格子上的 x 都改成 0 , 将第 0 类格子上的 0 都改成 X 。
- 显然这三种操作方案都能使得局面变成平局, 而它们的操作次数的总和恰好是棋盘中标志的总数 k, 因此其中操作次数最少的方案的操作次数一定不超过 $\lfloor \frac{k}{3} \rfloor$ 。

- 扫雷地图是一张 n 行 m 列的网格，其中每个格子是地雷或空地。每个空地会显示一个数字代表与它相邻的雷的数量（两个格子相邻当且仅当它们共用一个顶点或一条边，不在边界上的格子与恰好 8 个格子相邻）。
- 在一次操作中，你可以将一个地雷改成空地，或将空地改成地雷。
- 给定两张扫雷地图 A, B，你需要对 A 进行不超过 $\lfloor \frac{nm}{2} \rfloor$ 次操作，使得 A 所有空地上的数字之和等于 B 所有空地上的数字之和。
- $1 \leq n, m \leq 1000$。

- 注意到一张地图所有空地上的数字之和等于相邻的（地雷，空地）的对数。这就意味着，如果将一张地图的所有地雷改成空地，所有空地改成地雷，其所有空地上的数字和不变。

- 注意到一张地图所有空地上的数字之和等于相邻的（地雷，空地）的对数。这就意味着，如果将一张地图的所有地雷改成空地，所有空地改成地雷，其所有空地上的数字和不变。
- 由此，显然有以下两种方案：

- 注意到一张地图所有空地上的数字之和等于相邻的（地雷，空地）的对数。这就意味着，如果将一张地图的所有地雷改成空地，所有空地改成地雷，其所有空地上的数字和不变。
- 由此，显然有以下两种方案：
-
    - 将 A 改成 B。

- 注意到一张地图所有空地上的数字之和等于相邻的（地雷，空地）的对数。这就意味着，如果将一张地图的所有地雷改成空地，所有空地改成地雷，其所有空地上的数字和不变。
- 由此，显然有以下两种方案：
-
    - 将 A 改成 B。
    - 将 A 改成与 B 恰好相反，即若 B 的某个格子是地雷，则 A 对应的格子是空地，反之亦然。

- 注意到一张地图所有空地上的数字之和等于相邻的（地雷，空地）的对数。这就意味着，如果将一张地图的所有地雷改成空地，所有空地改成地雷，其所有空地上的数字和不变。
- 由此，显然有以下两种方案：
-
  - 将 A 改成 B。
  - 将 A 改成与 B 恰好相反，即若 B 的某个格子是地雷，则 A 对应的格子是空地，反之亦然。
- 由于每个格子只会在恰好一种方案中被修改, 这两种方案的操作次数之和应为 nm 。因此取其中较少的一种，操作次数不超过 $\left\lfloor \frac{nm}{2} \right\rfloor$ 。

DFS 树

- 在解决一些图上的构造问题时, DFS 树往往有非常大的帮助。
- 一张图的 DFS 树是在对其进行深度优先遍历时, 所形成的树结构。建立了 DFS 树后, 图上的边可以分成四类:
  - **树边**即每个点到其所有孩子结点的边, 也即每个点第一次被访问时经过的边。
  - **前向边**是每个点到其后代的边, 不包括树边。
  - **后向边**是每个点到其祖先的边。
  - 其余边称为**横叉边**。
- 其中, 前向边、后向边、横叉边统称为**非树边**。
- 在构造题中, 通常我们用到的是无向图的 DFS 树。如果我们将每条边按照第一次经过时的方向进行定向, 则无向图的 DFS 树满足所有非树边都是后向边。这个性质在解题过程中有非常大的作用。

- 给定一张 n 个点 m 条边的无向连通图，以及一个整数 k，你需要：
  - 找到一个恰好 $\left\lceil \frac{k}{2} \right\rceil$ 个点的独立集，
  - 或者找到一个长度不超过 k 的简单环。
- $3 \leq k \leq n \leq 10^5, n-1 \leq m \leq 2 \times 10^5$。

- 记 $l = \lceil \frac{k}{2} \rceil$ 。

- 记 $l = \left\lceil \frac{k}{2} \right\rceil$ 。
- 建出图的 DFS 树, 考虑每条非树边 (u, v) (正如上文所说, 它一定是后向边), 如果 $|\mathrm{dep}_u - \mathrm{dep}_v| < k$, 则取 (u, v) 加上 v 到 u 的树上路径即为一个长度不超过 k 的简单环。

- 记 $l = \lceil \frac{k}{2} \rceil$ 。
- 建出图的 DFS 树, 考虑每条非树边 (u, v) (正如上文所说, 它一定是后向边), 如果 $|\mathrm{dep}_u - \mathrm{dep}_v| < k$, 则取 (u, v) 加上 v 到 u 的树上路径即为一个长度不超过 k 的简单环。
- 否则, 考虑两种情况:

- 记 $l = \lceil \frac{k}{2} \rceil$ 。
- 建出图的 DFS 树，考虑每条非树边 (u, v) (正如上文所说，它一定是后向边)，如果 $|\mathrm{dep}_u - \mathrm{dep}_v| < k$，则取 (u, v) 加上 v 到 u 的树上路径即为一个长度不超过 k 的简单环。
- 否则，考虑两种情况:
  - 若 $m = n - 1$，即图是一棵树。把所有点按照深度的奇偶性分成两个集合，取其中较大的一个集合，即为大小至少为 $\lceil \frac{n}{2} \rceil \geq l$ 的独立集，取其中任意 l 个点即为所求。

- 记 $l=\lceil \frac{k}{2} \rceil$ 。
- 建出图的 DFS 树，考虑每条非树边 (u,v) (正如上文所说，它一定是后向边)，如果 $|\mathrm{dep}_u-\mathrm{dep}_v|<k$，则取 (u,v) 加上 v 到 u 的树上路径即为一个长度不超过 k 的简单环。
- 否则，考虑两种情况:
  - 若 $m=n-1$，即图是一棵树。把所有点按照深度的奇偶性分成两个集合，取其中较大的一个集合，即为大小至少为 $\lceil \frac{n}{2} \rceil \geq l$ 的独立集，取其中任意 l 个点即为所求。
  - 若 $m>n-1$，这时 DFS 树上存在非树边，但不满足 $|\mathrm{dep}_u-\mathrm{dep}_v|<k$，意味着 DFS 树的深度至少为 k，且任意一对深度差在 $[2,k)$ 中的点都不存在边相连。设深度最大的点为 x，取 x 以及 x 的 $2,4,\ldots,2l-2$ 级祖先，即为所求的独立集。

- 记 $l = \lceil \frac{k}{2} \rceil$ 。
- 建出图的 DFS 树, 考虑每条非树边 (u, v) (正如上文所说, 它一定是后向边), 如果 $|\mathrm{dep}_u - \mathrm{dep}_v| < k$, 则取 (u, v) 加上 v 到 u 的树上路径即为一个长度不超过 k 的简单环。
- 否则, 考虑两种情况:
  - 若 $m = n - 1$, 即图是一棵树。把所有点按照深度的奇偶性分成两个集合, 取其中较大的一个集合, 即为大小至少为 $\lceil \frac{n}{2} \rceil \geq l$ 的独立集, 取其中任意 l 个点即为所求。
  - 若 $m > n - 1$, 这时 DFS 树上存在非树边, 但不满足 $|\mathrm{dep}_u - \mathrm{dep}_v| < k$, 意味着 DFS 树的深度至少为 k, 且任意一对深度差在 $[2, k)$ 中的点都不存在边相连。设深度最大的点为 x, 取 x 以及 x 的 $2, 4, \ldots, 2l - 2$ 级祖先, 即为所求的独立集。
- 至此, 我们仅用一次 DFS, 在 $O(n + m)$ 的时间内解决了此题。

- 给定一张 n 个点 m 条边的无向连通图, 以及三个整数 a, b, c, 满足 $a+b+c=n$ 。
- 你需要将 n 个顶点分成三个集合 A, B, C, 大小分别为 a, b, c, 使得其中至少两个集合是连通的 (集合中的任意两个点能只经过该集合内的点互相到达)。有可能无解。
- $3 \leq n \leq 10^5, 2 \leq m \leq 2 \times 10^5$。

- 不妨设 $a \le b \le c$, 则只需要集合 A, B 连通即可。假设是 A, C 连通, 我们可以通过将 C 中的一些顶点加入 B 中, 使得 C 仍然连通且大小变成 b, 因此仍然是合法的解。 B, C 连通的情况同理。

- 不妨设 $a \le b \le c$, 则只需要集合 A, B 连通即可。假设是 A, C 连通, 我们可以通过将 C 中的一些顶点加入 B 中, 使得 C 仍然连通且大小变成 b, 因此仍然是合法的解。 B, C 连通的情况同理。
- 这时我们遇到了一些困难, 因为有可能无解, 而题目中并未给出、我们也并未发现有解的条件, 非常难以下手。因此我们不妨先来考察图是一棵树的情况。

- 不妨设 $a \le b \le c$, 则只需要集合 A, B 连通即可。假设是 A, C 连通，我们可以通过将 C 中的一些顶点加入 B 中，使得 C 仍然连通且大小变成 b, 因此仍然是合法的解。 B, C 连通的情况同理。
- 这时我们遇到了一些困难，因为有可能无解，而题目中并未给出、我们也并未发现有解的条件，非常难以下手。因此我们不妨先来考察图是一棵树的情况。
- 当图是一棵树时，集合 A, B 都是其中的子树，因此一定存在一条边，使得 A, B 处于边的两侧。显然，我们只需要找到一条边，使得其两侧的较小和较大的子树大小分别不小于 a, b 即可。

- 不妨设 $a \le b \le c$, 则只需要集合 A, B 连通即可。假设是 A, C 连通, 我们可以通过将 C 中的一些顶点加入 B 中, 使得 C 仍然连通且大小变成 b, 因此仍然是合法的解。 B, C 连通的情况同理。
- 这时我们遇到了一些困难, 因为有可能无解, 而题目中并未给出、我们也并未发现有解的条件, 非常难以下手。因此我们不妨先来考察图是一棵树的情况。
- 当图是一棵树时, 集合 A, B 都是其中的子树, 因此一定存在一条边, 使得 A, B 处于边的两侧。显然, 我们只需要找到一条边, 使得其两侧的较小和较大的子树大小分别不小于 a, b 即可。
- 注意到一条边两侧较大的子树一定包含重心, 我们可以考虑对重心进行一些分析。如果删去重心后最大的连通块大小小于 a, 则显然无解。否则, 设这个连通块的大小为 x, 由重心的性质显然有 $x \le n/2$, 因此删去这棵子树后还剩 $n - x \ge n/2$ 个点。又由于 $b \le n/2$ (因为 $b \le c$), 因此这棵子树与重心之间的边就是我们要找的边。

- 回到一般的情况, 我们建立图的 DFS 树。找到 DFS 树的重心, 设为 u, 记 u 上方的子树为 T, u 下方的子树为 $S_1, S_2, \ldots S_k$ 。考虑几种情况:

- 回到一般的情况，我们建立图的 DFS 树。找到 DFS 树的重心，设为 u，记 u 上方的子树为 T, u 下方的子树为 $S_1, S_2, \dots S_k$ 。考虑几种情况：
  - 如果 T 或某个 S_i 的大小不小于 a，则我们可以用和树一样的方法构造一组解。

- 回到一般的情况，我们建立图的 DFS 树。找到 DFS 树的重心，设为 u，记 u 上方的子树为 T, u 下方的子树为 $S_1, S_2, \ldots S_k$ 。考虑几种情况:
  - 如果 T 或某个 S_i 的大小不小于 a，则我们可以用和树一样的方法构造一组解。
  - 如果 T 和所有 S_i 的大小都小于 a，我们就需要考虑无向图 DFS 树的性质。不同的 S_i 之间是没有边相连的，同时有一些 S_i 与 T 相连。如果所有与 T 相连的 S_i 加上 T 的大小之和小于 a，则一定是无解的，因为这表示集合 A, B 都必须包含重心 u 。从 T 开始，我们依次加入与 T 相连的 S_i，直到其大小不小于 a 。设得到的点集为 X，则 X 是连通的，我们可以在其中选出 A 。同时由于 T 和所有 S_i 的大小之和都小于 a, X 的大小不超过 $2a$ 。而 $2a + b \leq a + b + c = n$，因此我们在删除 X 之后，剩余的点数至少为 $n - 2a \geq b$，我们可以在其中选出集合 B 。

- 回到一般的情况, 我们建立图的 DFS 树。找到 DFS 树的重心, 设为 u, 记 u 上方的子树为 T, u 下方的子树为 $S_1, S_2, \ldots S_k$ 。考虑几种情况:
  - 如果 T 或某个 S_i 的大小不小于 a, 则我们可以用和树一样的方法构造一组解。
  - 如果 T 和所有 S_i 的大小都小于 a, 我们就需要考虑无向图 DFS 树的性质。不同的 S_i 之间是没有边相连的, 同时有一些 S_i 与 T 相连。如果所有与 T 相连的 S_i 加上 T 的大小之和小于 a, 则一定是无解的, 因为这表示集合 A, B 都必须包含重心 u 。从 T 开始, 我们依次加入与 T 相连的 S_i, 直到其大小不小于 a 。设得到的点集为 X, 则 X 是连通的, 我们可以在其中选出 A 。同时由于 T 和所有 S_i 的大小之和都小于 a, X 的大小不超过 $2a$ 。而 $2a + b \leq a + b + c = n$, 因此我们在删除 X 之后, 剩余的点数至少为 $n - 2a \geq b$, 我们可以在其中选出集合 B 。
- 至此, 我们在 $O(n + m)$ 的时间内完成了构造。

- 在一些构造题中，对于不同的输入，问题的结构有很大的相似性。在很多时候，这往往意味着我们的构造也具有很大的相似性，或是具有周期性。
- 这时，我们往往可以通过递归的方式，对子问题进行构造，并在子问题的构造的基础上进行一些小的调整，来得到原问题的构造。

- 有 $2n$ 个包裹, 其中有 n 个 A 类包裹, 和 n 个 B 类包裹, 初始时它们的排列如下:

```
B A B A B A ...  B A
```

- 这些包裹占据了编号为 1 到 $2n$ 的格子, 同时还有编号为 $-2n+1$ 到 0 的 $2n$ 个空格子可供使用。
- 现在要将这些包裹重新排列, 使得它们形如

```
A A ...  A B ...  B B
```

- 即, 这些包裹占据了相邻的 $2n$ 个格子 (不一定是 1 到 $2n$), 且所有的 A 类包裹在所有的 B 类包裹的左边。
- 排列过程由若干次操作组成, 在每一次操作中, 可以选择相邻的两个包裹 (不能只选择一个), 并将它们移动至某两个相邻的空格中。
- 给定 n, 找到一个最短的操作序列。
- $3 \le n \le 100$。

- 与通常的构造题不同，本题要求的是最短的操作序列，看起来难以下手。但经过一些尝试，或是对 n 较小的情况进行搜索，会发现它们的最短操作序列的长度都是 n 。

- 与通常的构造题不同，本题要求的是最短的操作序列，看起来难以下手。但经过一些尝试，或是对 n 较小的情况进行搜索，会发现它们的最短操作序列的长度都是 n 。
- 事实上，证明操作次数不少于 n 是容易的: 考虑有多少对相邻的包裹的类型相同，设这个个数为 d 。初始时 $d=0$，而在结束局面中 $d=2n-2$ 。在一次操作过程中，取出包裹时不会 d 不会增加，而在放回包裹时，假设放的位置是 t 和 $t+1$，则只可能增加 $(t-1,t)$ 和 $(t+1,t+2)$ 这两对相邻的包裹。同时，容易发现第一次操作至多使得 d 增加 1 ，因此总的操作次数不少于 $1+\lceil(2n-3)/2\rceil=n$ 。

- 与通常的构造题不同，本题要求的是最短的操作序列，看起来难以下手。但经过一些尝试，或是对 n 较小的情况进行搜索，会发现它们的最短操作序列的长度都是 n 。
- 事实上，证明操作次数不少于 n 是容易的：考虑有多少对相邻的包裹的类型相同，设这个个数为 d 。初始时 $d = 0$，而在结束局面中 $d = 2n - 2$ 。在一次操作过程中，取出包裹时不会 d 不会增加，而在放回包裹时，假设放的位置是 t 和 $t + 1$，则只可能增加 $(t - 1, t)$ 和 $(t + 1, t + 2)$ 这两对相邻的包裹。同时，容易发现第一次操作至多使得 d 增加 1 ，因此总的操作次数不少于 $1 + \lceil (2n - 3)/2 \rceil = n$ 。
- 接下来，我们就要尝试对所有 n 构造长度为 n 的操作序列。对于 n 较小 $(n \leq 7)$ 的情况，我们可以直接利用搜索求出操作序列。经过观察发现，在 $n > 3$ 的情形中，我们都是将这些包裹从编号为 1 到 $2n$ 的格子移至编号为 -1 到 $2n - 2$ 的格子。

- 与通常的构造题不同，本题要求的是最短的操作序列，看起来难以下手。但经过一些尝试，或是对 n 较小的情况进行搜索，会发现它们的最短操作序列的长度都是 n 。
- 事实上，证明操作次数不少于 n 是容易的：考虑有多少对相邻的包裹的类型相同，设这个个数为 d 。初始时 $d=0$，而在结束局面中 $d=2n-2$ 。在一次操作过程中，取出包裹时不会 d 不会增加，而在放回包裹时，假设放的位置是 t 和 $t+1$，则只可能增加 $(t-1,t)$ 和 $(t+1,t+2)$ 这两对相邻的包裹。同时，容易发现第一次操作至多使得 d 增加 1 ，因此总的操作次数不少于 $1+\lceil (2n-3)/2 \rceil = n$ 。
- 接下来，我们就要尝试对所有 n 构造长度为 n 的操作序列。对于 n 较小 $(n \le 7)$ 的情况，我们可以直接利用搜索求出操作序列。经过观察发现，在 $n>3$ 的情形中，我们都是将这些包裹从编号为 1 到 $2n$ 的格子移至编号为 -1 到 $2n-2$ 的格子。
- 因此，我们可以定义函数 $\text{solve}(n,x)$ 表示将包裹从编号为 $x+1$ 到 $x+2n$ 的格子 (这些包裹形如 `B A B A ... B A`) 移至编号 $x-1$ 到 $x+2n-2$ 的格子，并排列成形如 `A A ... A B ... B B B`。

- 通过尝试, 或是对 n 稍大一些的情形的观察, 对于 $n \geq 8$ 的情况, 我们可以构造出如下的操作序列 (其中 _ 表示空格子):

- 通过尝试, 或是对 n 稍大一些的情形的观察, 对于 $n \geq 8$ 的情况, 我们可以构造出如下的操作序列 (其中 _ 表示空格子):
-

  `_ _ B A B A B A B A ... B A B A B A`

- 通过尝试, 或是对 n 稍大一些的情形的观察, 对于 $n \geq 8$ 的情况, 我们可以构造出如下的操作序列 (其中 _ 表示空格子):
- `_ _ B A B A B A B A ... B A B A B A`
- `A B B A B A B A B A ... B A B _ _ A`

- 通过尝试, 或是对 n 稍大一些的情形的观察, 对于 $n \geq 8$ 的情况, 我们可以构造出如下的操作序列 (其中 _ 表示空格子):
- `_ _ B A B A B A B A ... B A B A B A`
- `A B B A B A B A B A ... B A B _ _ A`
- `A B B A _ _ B A B A ... B A B B A A`

- 通过尝试, 或是对 n 稍大一些的情形的观察, 对于 $n \geq 8$ 的情况, 我们可以构造出如下的操作序列 (其中 _ 表示空格子):
- `_ _ B A B A B A B A ... B A B A B A`
- `A B B A B A B A B A ... B A B _ _ A`
- `A B B A _ _ B A B A ... B A B B A A`
- 在这里, 我们发现最后一行的红色部分正好符合 solve 函数的输入, 因此我们调用 solve$(n-4, x+4)$ 对其进行递归求解。

- 通过尝试, 或是对 n 稍大一些的情形的观察, 对于 $n \geq 8$ 的情况, 我们可以构造出如下的操作序列 (其中 _ 表示空格子):
- `_ _ B A B A B A B A ... B A B A B A`
- `A B B A B A B A B A ... B A B _ _ A`
- `A B B A _ _ B A B A ... B A B B A A`
- 在这里, 我们发现最后一行的红色部分正好符合 solve 函数的输入, 因此我们调用 $\text{solve}(n-4, x+4)$ 对其进行递归求解。
- `A B B A A A A ... B B B _ _ B B A A`

- 通过尝试, 或是对 n 稍大一些的情形的观察, 对于 $n \geq 8$ 的情况, 我们可以构造出如下的操作序列 (其中 _ 表示空格子):
- `_ _ B A B A B A B A ... B A B A B A`
- `A B B A B A B A B A ... B A B _ _ A`
- `A B B A _ _ B A B A ... B A B B A A`
- 在这里, 我们发现最后一行的红色部分正好符合 solve 函数的输入, 因此我们调用 solve$(n-4, x+4)$ 对其进行递归求解。
- `A B B A A A A ... B B B _ _ B B A A`
- `A _ _ A A A A ... B B B B B B B A A`

- 通过尝试, 或是对 n 稍大一些的情形的观察, 对于 $n \geq 8$ 的情况, 我们可以构造出如下的操作序列 (其中 _ 表示空格子):
- `_ _ B A B A B A B A ... B A B A B A`
- `A B B A B A B A B A ... B A B _ _ A`
- `A B B A _ _ B A B A ... B A B B A A`
- 在这里, 我们发现最后一行的红色部分正好符合 solve 函数的输入, 因此我们调用 solve$(n-4, x+4)$ 对其进行递归求解。
- `A B B A A A A ... B B B _ _ B B A A`
- `A _ _ A A A A ... B B B B B B B A A`
- `A A A A A A A ... B B B B B B B _ _`

- 通过尝试, 或是对 n 稍大一些的情形的观察, 对于 $n \geq 8$ 的情况, 我们可以构造出如下的操作序列 (其中 _ 表示空格子):
- `_ _ B A B A B A B A ... B A B A B A`
- `A B B A B A B A B A ... B A B _ _ A`
- `A B B A _ _ B A B A ... B A B B A A`
- 在这里, 我们发现最后一行的红色部分正好符合 solve 函数的输入, 因此我们调用 solve$(n-4, x+4)$ 对其进行递归求解。
- `A B B A A A A ... B B B _ _ B B A A`
- `A _ _ A A A A ... B B B B B B B A A`
- `A A A A A A A ... B B B B B B B _ _`
- 至此，我们便完成了长度为 n 的操作序列的构造，解决了此题。

- 给定一张 n 个点 m 条边的无向图, 你需要选择一个点集 S, 满足:
  - 1.一条边 (u,v) 是开启的当且仅当 $u \in S$ 或 $v \in S$, 则任意一对点都能只经过开启的边互相到达。
  - 2.不存在一条边 (u,v) 满足 $u \in S$ 且 $v \in S$ 。有可能无解。
- $2 \leq n \leq 3 \times 10^5, 0 \leq m \leq 3 \times 10^5$。

- 显然如果原图不连通则一定无解。我们不妨猜测当原图连通时一定有解，考虑归纳证明。

- 显然如果原图不连通则一定无解。我们不妨猜测当原图连通时一定有解，考虑归纳证明。
- 当 $n = 1$ 时，显然 $\varnothing$ 是合法的解。假设 $n = k - 1$ 时一定有解，考虑 $n = k$ 的情况。我们考虑删除一个非割点的点，容易发现这样的点是一定存在的。设这个点是 v，则 $G\backslash\{v\}$ 的点数为 $k - 1$，且是连通图，由归纳假设，我们可以为其找到一组解 S' 。接下来，考虑两种情况：

- 显然如果原图不连通则一定无解。我们不妨猜测当原图连通时一定有解，考虑归纳证明。
- 当 $n = 1$ 时，显然 $\varnothing$ 是合法的解。假设 $n = k - 1$ 时一定有解，考虑 $n = k$ 的情况。我们考虑删除一个非割点的点，容易发现这样的点是一定存在的。设这个点是 v，则 $G \backslash \{v\}$ 的点数为 $k - 1$，且是连通图，由归纳假设，我们可以为其找到一组解 S' 。接下来，考虑两种情况：
  - 若 G 中至少有一个与 v 相邻的点属于 S'，则令 $S = S'$ 即为一组合法的解。

- 显然如果原图不连通则一定无解。我们不妨猜测当原图连通时一定有解，考虑归纳证明。
- 当 $n=1$ 时，显然 $\varnothing$ 是合法的解。假设 $n=k-1$ 时一定有解，考虑 $n=k$ 的情况。我们考虑删除一个非割点的点，容易发现这样的点是一定存在的。设这个点是 v，则 $G\backslash\{v\}$ 的点数为 $k-1$，且是连通图，由归纳假设，我们可以为其找到一组解 S' 。接下来，考虑两种情况:
  - 若 G 中至少有一个与 v 相邻的点属于 S'，则令 $S=S'$ 即为一组合法的解。
  - 否则，G 中与 v 相邻的所有点都不属于 S'，则令 $S=S'\cup\{v\}$ 即为一组合法的解。这是因为 G 是连通图，因此至少有一个点与 v 相邻。

- 显然如果原图不连通则一定无解。我们不妨猜测当原图连通时一定有解，考虑归纳证明。
- 当 $n=1$ 时，显然 $\varnothing$ 是合法的解。假设 $n=k-1$ 时一定有解，考虑 $n=k$ 的情况。我们考虑删除一个非割点的点，容易发现这样的点是一定存在的。设这个点是 v，则 $G\backslash\{v\}$ 的点数为 $k-1$，且是连通图，由归纳假设，我们可以为其找到一组解 S' 。接下来，考虑两种情况:
  - 若 G 中至少有一个与 v 相邻的点属于 S'，则令 $S=S'$ 即为一组合法的解。
  - 否则，G 中与 v 相邻的所有点都不属于 S'，则令 $S=S'\cup\{v\}$ 即为一组合法的解。这是因为 G 是连通图，因此至少有一个点与 v 相邻。
- 至此，我们证明了有解当且仅当给定的图是连通图。然而，每次寻找一个非割点的复杂度太高，不能承受。

- 显然如果原图不连通则一定无解。我们不妨猜测当原图连通时一定有解，考虑归纳证明。
- 当 $n=1$ 时，显然 $\varnothing$ 是合法的解。假设 $n=k-1$ 时一定有解，考虑 $n=k$ 的情况。我们考虑删除一个非割点的点，容易发现这样的点是一定存在的。设这个点是 v，则 $G\backslash\{v\}$ 的点数为 $k-1$，且是连通图，由归纳假设，我们可以为其找到一组解 S' 。接下来，考虑两种情况:
  - 若 G 中至少有一个与 v 相邻的点属于 S'，则令 $S=S'$ 即为一组合法的解。
  - 否则，G 中与 v 相邻的所有点都不属于 S'，则令 $S=S'\cup\{v\}$ 即为一组合法的解。这是因为 G 是连通图，因此至少有一个点与 v 相邻。
- 至此，我们证明了有解当且仅当给定的图是连通图。然而，每次寻找一个非割点的复杂度太高，不能承受。
- 事实上，观察归纳的过程，我们只需按照任意一个 DFS 序依次加入点即可。时间复杂度为 $O(n+m)$ 。

杂题

- 给定正整数 k，要求构造一个 $n \times n$ 的网格图，满足 $n \leq 18$，每个格子中有一个 $[0, 131071]$ 中的数字，并且从左上角出发，只往下或往右走，走到右下角，并且经过的所有数异或和为 0 的方案数为 k。
- $k \leq 131071$。

- 这种构造满足条件的路径条数的方法，一个经典的想法是拆二进制位。

- 这种构造满足条件的路径条数的方法，一个经典的想法是拆二进制位。
- 具体地，我们构造一个如下的网格（空白位置默认为 0）：

- 这种构造满足条件的路径条数的方法，一个经典的想法是拆二进制位。
- 具体地，我们构造一个如下的网格（空白位置默认为 0）：
-

		1						
			2					
1				4				
	2				$\ddots$			
		4				2^{n-4}		
			$\ddots$				2^{n-3}	
				2^{n-4}				2^{n-2}
					2^{n-3}			
						2^{n-2}		

如小雪落下海岸线

Solution

- 这种构造满足条件的路径条数的方法，一个经典的想法是拆二进制位。
- 具体地，我们构造一个如下的网格（空白位置默认为 0）：
-

		1						
			2					
1				4				
	2				$\ddots$			
		4				2^{n-4}		
			$\ddots$				2^{n-3}	
				2^{n-4}				2^{n-2}
					2^{n-3}			
						2^{n-2}		

- 可以看出，这样从左上角走到右下角的方案书就是 2^{n-1} 了。

- 这种构造满足条件的路径条数的方法，一个经典的想法是拆二进制位。
- 具体地，我们构造一个如下的网格（空白位置默认为 0）：

		1						
			2					
1				4				
	2				$\ddots$			
		4				2^{n-4}		
			$\ddots$				2^{n-3}	
				2^{n-4}				2^{n-2}
					2^{n-3}			
						2^{n-2}		

- 可以看出，这样从左上角走到右下角的方案书就是 2^{n-1} 了。
- 然后考虑加二进制位。如果只需要加一个二进制位，例如加 4，那就在第 3 行最后一列放上一个 $4 \oplus 2^{n-2}$，那就有恰好 4 种方案从第 3 行“脱离”出来，走到最后一列，然后再向下走到右下角。

如小雪落下海岸线

Solution

- 这种构造满足条件的路径条数的方法，一个经典的想法是拆二进制位。
- 具体地，我们构造一个如下的网格（空白位置默认为 0）：

		1						
			2					
1				4				
	2				$\ddots$			
		4				2^{n-4}		
			$\ddots$				2^{n-3}	
				2^{n-4}				2^{n-2}
					2^{n-3}			
						2^{n-2}		

- 可以看出，这样从左上角走到右下角的方案书就是 2^{n-1} 了。
- 然后考虑加二进制位。如果只需要加一个二进制位，例如加 4，那就在第 3 行最后一列放上一个 $4 \oplus 2^{n-2}$，那就有恰好 4 种方案从第 3 行“脱离”出来，走到最后一列，然后再向下走到右下角。
- 对于加多个二进制位，只需要把它们都放到最后一列，然后在最后一列做一个后缀异或和即可。

如小雪落下海岸线

Solution

- 这种构造满足条件的路径条数的方法，一个经典的想法是拆二进制位。
- 具体地，我们构造一个如下的网格（空白位置默认为 0）：
-

		1						
			2					
1				4				
	2				$\ddots$			
		4				2^{n-4}		
			$\ddots$				2^{n-3}	
				2^{n-4}				2^{n-2}
					2^{n-3}			
						2^{n-2}		

- 可以看出，这样从左上角走到右下角的方案书就是 2^{n-1} 了。
- 然后考虑加二进制位。如果只需要加一个二进制位，例如加 4，那就在第 3 行最后一列放上一个 $4 \oplus 2^{n-2}$，那就有恰好 4 种方案从第 3 行“脱离”出来，走到最后一列，然后再向下走到右下角。
- 对于加多个二进制位，只需要把它们都放到最后一列，然后在最后一列做一个后缀异或和即可。
- 时间复杂度主要在 checker 上面。

- 给定正整数 n, 要求将 $\{2, 3, \ldots, 3n+1\}$ 划分成 n 个三元组，使得每个三元组都形成一个钝角三角形。
- $n \leq 10^5$。

- 有很多种构造方法，这里先讲原题解的方法。

- 有很多种构造方法，这里先讲原题解的方法。
- 首先我们注意到一点，如果 (a, b, c) 是钝角三角形，那么 $(2a, 2b, 2c)$ 也是钝角三角形，$(2a+1, 2b+1, 2c+1)$ 大概率是钝角三角形。这启发我们进行递归构造。

- 有很多种构造方法，这里先讲原题解的方法。
- 首先我们注意到一点，如果 (a,b,c) 是钝角三角形，那么 $(2a,2b,2c)$ 也是钝角三角形，$(2a+1,2b+1,2c+1)$ 大概率是钝角三角形。这启发我们进行递归构造。
- 一个简单的想法是，如果 $2 \nmid n$，那先拎出一个 $(2,3n,3n+1)$，然后将剩下的 $3n-3$ 个数按奇偶分组，划分成 $\{3,5,7,\ldots,3n-2\}$ 和 $\{4,6,8,\ldots,3n-1\}$ 递归去做；

- 有很多种构造方法，这里先讲原题解的方法。
- 首先我们注意到一点，如果 (a, b, c) 是钝角三角形，那么 $(2a, 2b, 2c)$ 也是钝角三角形，$(2a+1, 2b+1, 2c+1)$ 大概率是钝角三角形。这启发我们进行递归构造。
- 一个简单的想法是，如果 $2 \nmid n$，那先拎出一个 $(2, 3n, 3n+1)$，然后将剩下的 $3n-3$ 个数按奇偶分组，划分成 $\{3, 5, 7, \dots, 3n-2\}$ 和 $\{4, 6, 8, \dots, 3n-1\}$ 递归去做；
- 如果 $2 \mid n$，那先拎出 $(2, 3n-2, 3n-1), (3, 3n, 3n+1)$，然后将剩下的 $3n-6$ 个数按奇偶分组，划分成 $\{4, 6, 8, \dots, 3n-4\}$ 和 $\{5, 7, 9, \dots, 3n-3\}$ 递归去做。

- 有很多种构造方法，这里先讲原题解的方法。
- 首先我们注意到一点，如果 (a, b, c) 是钝角三角形，那么 $(2a, 2b, 2c)$ 也是钝角三角形，$(2a+1, 2b+1, 2c+1)$ 大概率是钝角三角形。这启发我们进行递归构造。
- 一个简单的想法是，如果 $2 \nmid n$，那先拎出一个 $(2, 3n, 3n+1)$，然后将剩下的 $3n-3$ 个数按奇偶分组，划分成 $\{3, 5, 7, \ldots, 3n-2\}$ 和 $\{4, 6, 8, \ldots, 3n-1\}$ 递归去做；
- 如果 $2 \mid n$，那先拎出 $(2, 3n-2, 3n-1), (3, 3n, 3n+1)$，然后将剩下的 $3n-6$ 个数按奇偶分组，划分成 $\{4, 6, 8, \ldots, 3n-4\}$ 和 $\{5, 7, 9, \ldots, 3n-3\}$ 递归去做。
- 考虑这样做的正确性。将 $2, 3, \ldots, 3n+1$ 从 2 **开始编号**，则在第一种情况中，一个数 x 的编号将变为 $2x-1$ 或 $2x$；在第二种情况中，一个数 x 的编号将变为 $2x$ 或 $2x+1$。

- 有很多种构造方法，这里先讲原题解的方法。
- 首先我们注意到一点，如果 (a,b,c) 是钝角三角形，那么 $(2a,2b,2c)$ 也是钝角三角形，$(2a+1,2b+1,2c+1)$ 大概率是钝角三角形。这启发我们进行递归构造。
- 一个简单的想法是，如果 $2\nmid n$，那先拎出一个 $(2,3n,3n+1)$，然后将剩下的 $3n-3$ 个数按奇偶分组，划分成 $\{3,5,7,\ldots,3n-2\}$ 和 $\{4,6,8,\ldots,3n-1\}$ 递归去做；
- 如果 $2\mid n$，那先拎出 $(2,3n-2,3n-1),(3,3n,3n+1)$，然后将剩下的 $3n-6$ 个数按奇偶分组，划分成 $\{4,6,8,\ldots,3n-4\}$ 和 $\{5,7,9,\ldots,3n-3\}$ 递归去做。
- 考虑这样做的正确性。将 $2,3,\ldots,3n+1$ 从 2 **开始编号**，则在第一种情况中，一个数 x 的编号将变为 $2x-1$ 或 $2x$；在第二种情况中，一个数 x 的编号将变为 $2x$ 或 $2x+1$。
- 可以归纳证明，所有三角形将形如 $(ka+r,kb+r,kc+r)$，其中 $k=2^t,t\in\mathbb{Z},|r|<k$，且 $(a,b,c)=(2,x,x+1)(x\geq 3)$ 或 $(3,x,x+1)(x\geq 6)$。

你说空瓶适合许愿

Solution

- 有很多种构造方法，这里先讲原题解的方法。
- 首先我们注意到一点，如果 (a,b,c) 是钝角三角形，那么 $(2a,2b,2c)$ 也是钝角三角形，$(2a+1,2b+1,2c+1)$ 大概率是钝角三角形。这启发我们进行递归构造。
- 一个简单的想法是，如果 $2\nmid n$，那先拎出一个 $(2,3n,3n+1)$，然后将剩下的 $3n-3$ 个数按奇偶分组，划分成 $\{3,5,7,\dots,3n-2\}$ 和 $\{4,6,8,\dots,3n-1\}$ 递归去做；
- 如果 $2\mid n$，那先拎出 $(2,3n-2,3n-1),(3,3n,3n+1)$，然后将剩下的 $3n-6$ 个数按奇偶分组，划分成 $\{4,6,8,\dots,3n-4\}$ 和 $\{5,7,9,\dots,3n-3\}$ 递归去做。
- 考虑这样做的正确性。将 $2,3,\dots,3n+1$ **从 2 开始编号**，则在第一种情况中，一个数 x 的编号将变为 $2x-1$ 或 $2x$；在第二种情况中，一个数 x 的编号将变为 $2x$ 或 $2x+1$。
- 可以归纳证明，所有三角形将形如 $(ka+r,kb+r,kc+r)$，其中 $k=2^t,t\in\mathbb{Z},|r|<k$，且 $(a,b,c)=(2,x,x+1)(x\ge 3)$ 或 $(3,x,x+1)(x\ge 6)$。
- 列式子可得 $(kc+r)^2-(ka+r)^2-(kb+r)^2=k^2((c+1)^2-(a+1)^2-(b+1)^2)+2k(a+b-c+1)-1$，则右式 >0 当且仅当 $(c+1)^2-(a+1)^2-(b+1)^2\ge 0$，即 $(a+1,b+1,c+1)$ 不为锐角三角形时恒成立。
- 但是悲催地发现 $(3+1,6+1,7+1)=(4,7,8)$ 真是锐角三角形。寄！

- 当然没完全寄。以下称一个三角形 (a, b, c) 合法，当且仅当 (a, b, c) 是钝角三角形，且 $(a+1, b+1, c+1)$ 不是锐角三角形。

- 当然没完全寄。以下称一个三角形 (a,b,c) 合法，当且仅当 (a,b,c) 是钝角三角形，且 $(a+1,b+1,c+1)$ 不是锐角三角形。
- 首先当 $x>6$ 的时候 $(3,x,x+1)$ 确实合法，并且 $(2,x,x+1)$ 始终合法，所以只用调整递归边界即可。

- 当然没完全寄。以下称一个三角形 (a,b,c) 合法，当且仅当 (a,b,c) 是钝角三角形，且 $(a+1,b+1,c+1)$ 不是锐角三角形。
- 首先当 $x>6$ 的时候 $(3,x,x+1)$ 确实合法，并且 $(2,x,x+1)$ 始终合法，所以只用调整递归边界即可。
- 原先我们失败的原因是递归边界之一是当 $n=2$ 时取 $(2,4,5),(3,6,7)$ 作为解，现在我们改为以 $n=1,4,5,6$ 作为边界，爆搜出一组合法解。不到一秒就能搜出来，这道题就做完了。

- 当然没完全寄。以下称一个三角形 (a,b,c) 合法，当且仅当 (a,b,c) 是钝角三角形，且 $(a+1,b+1,c+1)$ 不是锐角三角形。
- 首先当 $x>6$ 的时候 $(3,x,x+1)$ 确实合法，并且 $(2,x,x+1)$ 始终合法，所以只用调整递归边界即可。
- 原先我们失败的原因是递归边界之一是当 $n=2$ 时取 $(2,4,5),(3,6,7)$ 作为解，现在我们改为以 $n=1,4,5,6$ 作为边界，爆搜出一组合法解。不到一秒就能搜出来，这道题就做完了。
- 还有一种非常简单的做法，就是将 $[2,3n+1]$ 分成 $[2,n+1]$，$[n+2,2n+1]$，$[2n+2,3n+1]$ 三组，然后对于后两组，每次取出正中间两个数和第一组的一个数，保证每个三角形较小两条边之和最多比第三条边大 2。感性理解这玩意就是对的。

- 当然没完全寄。以下称一个三角形 (a,b,c) 合法，当且仅当 (a,b,c) 是钝角三角形，且 $(a+1,b+1,c+1)$ 不是锐角三角形。
- 首先当 $x>6$ 的时候 $(3,x,x+1)$ 确实合法，并且 $(2,x,x+1)$ 始终合法，所以只用调整递归边界即可。
- 原先我们失败的原因是递归边界之一是当 $n=2$ 时取 $(2,4,5),(3,6,7)$ 作为解，现在我们改为以 $n=1,4,5,6$ 作为边界，爆搜出一组合法解。不到一秒就能搜出来，这道题就做完了。
- 还有一种非常简单的做法，就是将 $[2,3n+1]$ 分成 $[2,n+1]$，$[n+2,2n+1]$，$[2n+2,3n+1]$ 三组，然后对于后两组，每次取出正中间两个数和第一组的一个数，保证每个三角形较小两条边之和最多比第三条边大 2。感性理解这玩意就是对的。
- 还有一万种构造方法，就不展开讲了，感兴趣的话可以自行思考/zhq

- 给定长度为 n $(1 \le n \le 1\,000)$ 的数列 $a_1, a_2, \ldots, a_n$，找到其所有逆序对的一个排列 (u_i, v_i)，使得依次交换 u_1 和 v_1 位置上的数、u_2 和 v_2 位置上的数、……最后得到的数列不减。

- 不妨设 a 是一个排列。将排列转化成成逆排列考虑是一个常用技巧。

- 不妨设 a 是一个排列。将排列转化成成逆排列考虑是一个常用技巧。
- 注意到交换原排列的两个位置等同于交换逆排列的两个值，对逆排列冒泡排序即可。

- 给定一张 n $(2 \le n \le 5 \cdot 10^5)$ 个点 m $(1 \le n \le 10^6)$ 条边的简单连通图，在下面两项中选择一项完成：
  - 找到一条至少 $\lceil \frac{n}{2} \rceil$ 个点的简单路径。
  - 找到一些点对，满足
    - 所有点互不相同；
    - 包含至少 $\lceil \frac{n}{2} \rceil$ 个点；
    - 对于任意一对点对 (a, b) 和 (c, d)，点集 $\{a, b, c, d\}$ 的导出子图包含至多 2 条边。
- 注意这里一对点不需要直接相连。

- 我们需要解决图上的问题，则必然需要解决树上的问题。

- 我们需要解决图上的问题，则必然需要解决树上的问题。
- 令根节点的深度为 1，如果有一个点的深度至少为 $\lceil \frac{n}{2} \rceil$，则路径已经找到了。

- 我们需要解决图上的问题，则必然需要解决树上的问题。
- 令根节点的深度为 1，如果有一个点的深度至少为 $\lceil \frac{n}{2} \rceil$，则路径已经找到了。
- 否则，树的深度至多为 $\lfloor \frac{n}{2} \rfloor$，那么考虑对于每个深度 i $(1 \le i \le \lfloor \frac{n}{2} \rfloor)$，将该深度的点两两配对直至至多剩下一个点为止。已配对点数至少为 $n - \lfloor \frac{n}{2} \rfloor = \lceil \frac{n}{2} \rceil$。

- 我们需要解决图上的问题，则必然需要解决树上的问题。
- 令根节点的深度为 1，如果有一个点的深度至少为 $\lceil \frac{n}{2} \rceil$，则路径已经找到了。
- 否则，树的深度至多为 $\lfloor \frac{n}{2} \rfloor$，那么考虑对于每个深度 i $(1 \le i \le \lfloor \frac{n}{2} \rfloor)$，将该深度的点两两配对直至至多剩下一个点为止。已配对点数至少为 $n - \lfloor \frac{n}{2} \rfloor = \lceil \frac{n}{2} \rceil$。
- 对于两对点 (a, b) 和 (c, d)，不妨设 $dep_a = dep_b < dep_c = dep_d$，则边 (a, b) 和 (c, d) 都不存在，且 c, d 各至多有一个父亲，所以 $(c, a), (c, b)$ 中至多有一条边存在，d 同理。因此条件满足。

- 我们需要解决图上的问题，则必然需要解决树上的问题。
- 令根节点的深度为 1，如果有一个点的深度至少为 $\lceil \frac{n}{2} \rceil$，则路径已经找到了。
- 否则，树的深度至多为 $\lfloor \frac{n}{2} \rfloor$，那么考虑对于每个深度 i $(1 \le i \le \lfloor \frac{n}{2} \rfloor)$，将该深度的点两两配对直至至多剩下一个点为止。已配对点数至少为 $n - \lfloor \frac{n}{2} \rfloor = \lceil \frac{n}{2} \rceil$。
- 对于两对点 (a, b) 和 (c, d)，不妨设 $dep_a = dep_b < dep_c = dep_d$，则边 (a, b) 和 (c, d) 都不存在，且 c, d 各至多有一个父亲，所以 $(c, a), (c, b)$ 中至多有一条边存在，d 同理。因此条件满足。
- 对于图上的情况，根据套路，我们拎出一棵 DFS 树，那么由于只有返祖边，(a, b) 和 (c, d) 都不存在仍然满足，并且若 $(c, a), (c, b)$ 都存在，那么 a, b 有祖孙关系，矛盾。这样这题就做完了。

- 给定长度为 n $(1 \le n \le 10^6)$ 的数列 $a_1, a_2, \ldots, a_n$ $(i - n \le a_i \le i - 1)$，找到这些数的一个和为 0 的非空子集。

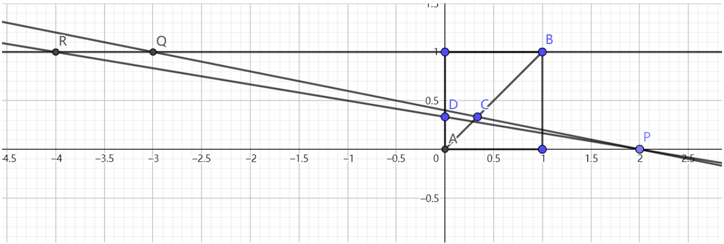
- 给定一个序列，还是可以想一想怎么变成一张图的。

- 给定一个序列，还是可以想一想怎么变成一张图的。
- 注意到 $1 \le i - a_i \le n$，从 i 连向 $i - a_i$ 得到一个基环树森林，其中任意一个环满足 $\sum i = \sum (i - a_i)$，即 $\sum a_i = 0$。做完了。

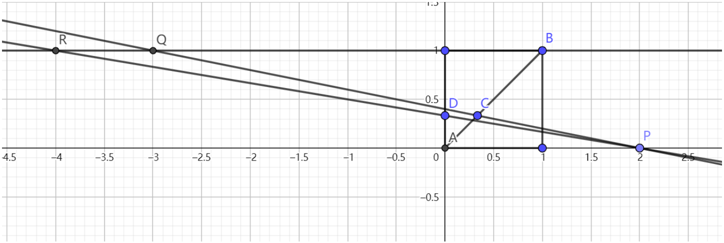
- 平面上有 6 个初始已知点 $(0,0),(0,\frac{1}{2}),(0,1),(1,0),(1,\frac{1}{2}),(1,1)$，你需要进行若干次操作，每次操作形如：
  1. 过两已知点作一条直线；
  2. 将两条直线的交点变成已知点。
- 现在给定一个点 $(X_a/X_b, Y_a/Y_b)$，满足 $0 \le X_a < X_b \le 10^9, 0 \le Y_a < Y_b \le 10^9$（注意这个点横纵坐标均在 $[0,1]$ 内），要求使用不超过 150 次 `1` 操作将这个点变为已知点。
- 部分分：$X_a = 0$。

- 首先考虑部分分，即求 y 轴上一点的坐标。进一步，可以转化成连接 x 轴上一个整点和直线 $x = 1$ 上一个整点与 x 轴的交点。这两者是对称的，于是只用找到 x 轴上一个值域 10^9 以内的整点。

- 首先考虑部分分，即求 y 轴上一点的坐标。进一步，可以转化成连接 x 轴上一个整点和直线 $x = 1$ 上一个整点与 x 轴的交点。这两者是对称的，于是只用找到 x 轴上一个值域 10^9 以内的整点。
- 思考一段时间后发现倍增是一个不错的选择。令 $A(0,0)$，$B(1,1)$，$C(\frac{1}{3}, \frac{1}{3})$，$D(0, \frac{1}{3})$。设点 P 和 A 或 B 的距离为 d，则每次选择用 C 还是 D 翻过去可以选择让 $d \leftarrow 2d$ 还是 $d \leftarrow 2d + 1$，即 $BQ = 2AP, BR = 2AP + 1$：



- 首先考虑部分分，即求 y 轴上一点的坐标。进一步，可以转化成连接 x 轴上一个整点和直线 $x=1$ 上一个整点与 x 轴的交点。这两者是对称的，于是只用找到 x 轴上一个值域 10^9 以内的整点。
- 思考一段时间后发现倍增是一个不错的选择。令 $A(0,0)$，$B(1,1)$，$C(\frac{1}{3},\frac{1}{3})$，$D(0,\frac{1}{3})$。设点 P 和 A 或 B 的距离为 d，则每次选择用 C 还是 D 翻过去可以选择让 $d \leftarrow 2d$ 还是 $d \leftarrow 2d+1$，即
$BQ=2AP, BR=2AP+1$：



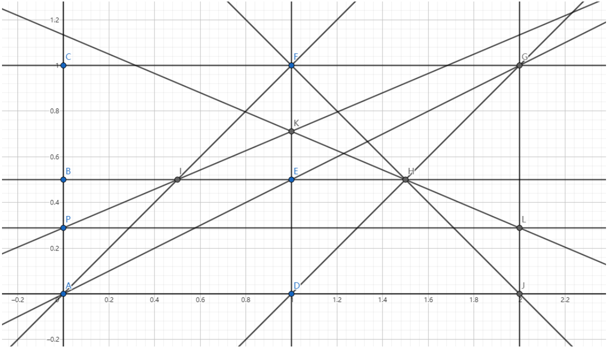
- 每次可以做到 $\times 2$ 或者 $\times 2+1$，所以可以在 $\log V$ 步内找到这个整点，其中 $V=10^9$。那么找到 y 轴上一个分数点的步数即为 $2\log V$。

- 那么对于题意中任意一个点 (X, Y)，我们找出点 $(X, 0)$ 和 $(0, Y)$，然后尝试作出直线 $y = Y$ 和 $x = X$。这两者是对称的，只考虑前者。

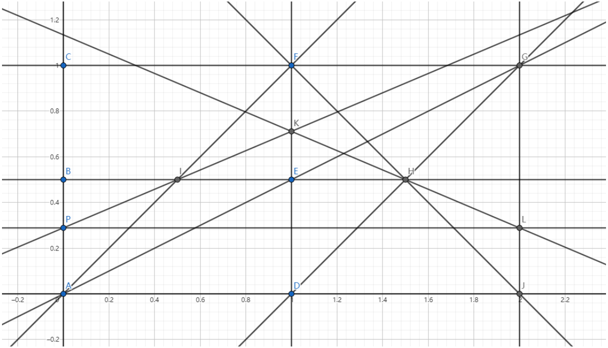
我们有相遇的时间

Solution

- 那么对于题意中任意一个点 (X, Y)，我们找出点 $(X, 0)$ 和 $(0, Y)$，然后尝试作出直线 $y = Y$ 和 $x = X$。这两者是对称的，只考虑前者。
- 随便手玩玩即可。下图的思路是把 P 对称一次到点 K，再把 K 对称一次到点 L，连接 PL：



- 那么对于题意中任意一个点 (X, Y)，我们找出点 $(X, 0)$ 和 $(0, Y)$，然后尝试作出直线 $y = Y$ 和 $x = X$。这两者是对称的，只考虑前者。
- 随便手玩玩即可。下图的思路是把 P 对称一次到点 K，再把 K 对称一次到点 L，连接 PL：

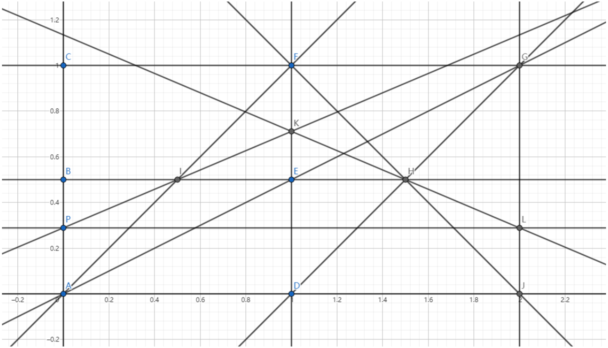


- 然后就在 $4 \log V$ 步内做完了，带点小常数。150 步很充裕。

我们有相遇的时间

Solution

- 那么对于题意中任意一个点 (X,Y)，我们找出点 $(X,0)$ 和 $(0,Y)$，然后尝试作出直线 $y=Y$ 和 $x=X$。这两者是对称的，只考虑前者。
- 随便手玩玩即可。下图的思路是把 P 对称一次到点 K，再把 K 对称一次到点 L，连接 PL：



- 然后就在 $4\log V$ 步内做完了，带点小常数。150 步很充裕。
- 其实仔细想想会发现可以通过一些更精细的预处理把 $4\log_2 V$ 变成 $4\log_3 V$，代价是多点常数。肯定能压到更小。

感谢聆听！