贪心博弈论乱讲

Lyrella

cdqz

2025 年 10 月 13 日

前言

这次的复习主要是通过回顾我们之前做过的题以及我分享自己找的一些不错的题目来展开。因为贪心博弈论的东西可讲的知识点不会太多，并且考虑到大家都已经有一定的基础，所以纯讲知识点的地方可能会过得有点快。

前言

这次的复习主要是通过回顾我们之前做过的题以及我分享自己找的一些不错的题目来展开。因为贪心博弈论的东西可讲的知识点不会太多，并且考虑到大家都已经有一定的基础，所以纯讲知识点的地方可能会过得有点快。

每个题我会留至少十分钟的时间给大家思考，具体的时间规划可灵活调整。如果在讲课的过程中你有地方没听懂可以立刻提出来大家一起解决，如果没跟上想再听一遍也可以提出来，但是如果神游于天地之间然后发现自己掉队了请暂时放弃没听到的内容立刻跟上，自己下来再去看。

前言

这次的复习主要是通过回顾我们之前做过的题以及我分享自己找的一些不错的题目来展开。因为贪心博弈论的东西可讲的知识点不会太多，并且考虑到大家都已经有一定的基础，所以纯讲知识点的地方可能会过得有点快。

每个题我会留至少十分钟的时间给大家思考，具体的时间规划可灵活调整。如果在讲课的过程中你有地方没听懂可以立刻提出来大家一起解决，如果没跟上想再听一遍也可以提出来，但是如果神游于天地之间然后发现自己掉队了请暂时放弃没听到的内容立刻跟上，自己下来再去看。

然后因为 lyr 水平不高，有小概率会紧张，讲课过程中可能会出现一些口误笔误，有时甚至还会出现一些逻辑的错误，请大家见谅并指出谢谢喵。

Contents

基础知识

先来说说贪心是什么？

基础知识

先来说说贪心是什么？贪心，顾名思义，就是现在进行什么决策最优就做什么。

基础知识

先来说说贪心是什么？贪心，顾名思义，就是现在进行什么决策最优就做什么。

但这样的思想往往不能保证最后最优，所以贪心它有一定的适用范围，也就是最优子结构。

基础知识

先来说说贪心是什么？贪心，顾名思义，就是现在进行什么决策最优就做什么。

但这样的思想往往不能保证最后最优，所以贪心它有一定的适用范围，也就是最优子结构。

最优子结构就是问题能够分解成子问题来解决，子问题的最优解能递推到最终问题的最优解。

基础知识

先来说说贪心是什么？贪心，顾名思义，就是现在进行什么决策最优就做什么。

但这样的思想往往不能保证最后最优，所以贪心它有一定的适用范围，也就是最优子结构。

最优子结构就是问题能够分解成子问题来解决，子问题的最优解能递推到最终问题的最优解。

最常见的贪心分为两种：分别是邻项交换法和反悔贪心。

基础知识

邻项交换法就是我们要决定一些元素的选择顺序的时候分别计算交换相邻两元素前后其对答案的贡献，然后选择更优的方案。

基础知识

邻项交换法就是我们要决定一些元素的选择顺序的时候分别计算交换相邻两元素前后其对答案的贡献，然后选择更优的方案。

反悔贪心就是当元素选择有限制的时候，我们还是每次选择最优的元素，显然这个操作不一定正确。但是我们可以添加新的元素代表不选这个元素去选其他元素的增加的收益，这样就能在后面某个时刻“纠正自己的错误”。

基础知识

接下来再说一下贪心的证明。

基础知识

接下来再说一下贪心的证明。

我们证明的方法通常也有两种：分别是反证法和归纳法。

基础知识

接下来再说一下贪心的证明。

我们证明的方法通常也有两种：分别是反证法和归纳法。

反证法就是假设我们当前不选最优的元素去证明答案一定不优。

基础知识

接下来再说一下贪心的证明。

我们证明的方法通常也有两种：分别是反证法和归纳法。

反证法就是假设我们当前不选最优的元素去证明答案一定不优。

归纳法就是我们从边界情况开始往回推理证明，考虑每一次都是去选最优的元素。

基础知识

接下来再说一下贪心的证明。

我们证明的方法通常也有两种：分别是反证法和归纳法。

反证法就是假设我们当前不选最优的元素去证明答案一定不优。

归纳法就是我们从边界情况开始往回推理证明，考虑每一次都是去选最优的元素。

其实还有一个对贪心来说比较重要的东西，那就是拟阵。

拟阵

拟阵是研究贪心算法的数学基础，如果一个问题可以转化为拟阵，那么它便一定可以使用贪心进行解决，且解决方法有迹可循。但不是所有可以贪心的问题都可以转化为拟阵。

拟阵

拟阵是研究贪心算法的数学基础，如果一个问题可以转化为拟阵，那么它便一定可以使用贪心进行解决，且解决方法有迹可循。但不是所有可以贪心的问题都可以转化为拟阵。

但考虑到拟阵这个东西目前只能帮助我们去更深刻地理解贪心算法，实际用处不太大，所以在思考良久后我最终还是不准备讲它。但是这里放一个关于拟阵的博客，这篇博客非常详细，有想学习的同学可以去看。

拟阵

拟阵是研究贪心算法的数学基础，如果一个问题可以转化为拟阵，那么它便一定可以使用贪心进行解决，且解决方法有迹可循。但不是所有可以贪心的问题都可以转化为拟阵。

但考虑到拟阵这个东西目前只能帮助我们去更深刻地理解贪心算法，实际用处不太大，所以在思考良久后我最终还是不准备讲它。但是这里放一个关于拟阵的博客，这篇博客非常详细，有想学习的同学可以去看。

https://www.luogu.com.cn/article/87d02q9f

过渡

讲上述东西，是为了让我们在考场上能判断一个题是否需要用贪心，如果连最基本的判断都不能做到，这次的复习也就没了意义。

过渡

讲上述东西，是为了让我们在考场上能判断一个题是否需要用贪心，如果连最基本的判断都不能做到，这次的复习也就没了意义。

有关贪心的东西也就说得差不多了，下面我们就看一些题吧。

过渡

讲上述东西，是为了让我们在考场上能判断一个题是否需要用贪心，如果连最基本的判断都不能做到，这次的复习也就没了意义。

有关贪心的东西也就说得差不多了，下面我们就看一些题吧。

注意题目是分了类的，每一类里面会按照主观难度排序，欢迎大家暴切水题 qwq。

例题

P1080

现在有 n 个人，第一个人位置固定，后面的人可以任意排序，对第 i 个人给定 a_i, b_i，第 i 个人的权值 $c_i = \frac{\Pi_{j=1}^{i-1} a_j}{b_i}$，求最大权值最小是多少。

例题

P1080

现在有 n 个人，第一个人位置固定，后面的人可以任意排序，对第 i 个人给定 a_i, b_i，第 i 个人的权值 $c_i = \frac{\Pi_{j=1}^{i-1} a_j}{b_i}$，求最大权值最小是多少。

考虑相邻的两个位置 i, j，我们分别写出交换前后它们的权值。

例题

P1080

现在有 n 个人，第一个人位置固定，后面的人可以任意排序，对第 i 个人给定 a_i, b_i，第 i 个人的权值 $c_i = \frac{\Pi_{j=1}^{i-1} a_j}{b_i}$，求最大权值最小是多少。

考虑相邻的两个位置 i, j，我们分别写出交换前后它们的权值。

假设 $s = \Pi_{k=1}^{i-1} a_k$。那么交换前 $ans = \max(\frac{s}{b_i}, \frac{s \times a_i}{b_j})$。交换后 $ans' = \max(\frac{s}{b_j}, \frac{s \times a_j}{b_i})$。

例题

P1080

现在有 n 个人，第一个人位置固定，后面的人可以任意排序，对第 i 个人给定 a_i, b_i，第 i 个人的权值 $c_i = \frac{\Pi_{j=1}^{i-1} a_j}{b_i}$，求最大权值最小是多少。

考虑相邻的两个位置 i, j，我们分别写出交换前后它们的权值。

假设 $s = \Pi_{k=1}^{i-1} a_k$。那么交换前 $ans = \max(\frac{s}{b_i}, \frac{s \times a_i}{b_j})$。交换后 $ans' = \max(\frac{s}{b_j}, \frac{s \times a_j}{b_i})$。

显然有 $\frac{s \times a_i}{b_j} > \frac{s}{b_j}, \frac{s \times a_j}{b_i} > \frac{s}{b_i}$。因为我们希望 $ans < ans'$，所以有 $\frac{s \times a_i}{b_j} < \frac{s \times a_j}{b_i}$。

例题

P1080

现在有 n 个人，第一个人位置固定，后面的人可以任意排序，对第 i 个人给定 a_i, b_i，第 i 个人的权值 $c_i = \frac{\Pi_{j=1}^{i-1} a_j}{b_i}$，求最大权值最小是多少。

考虑相邻的两个位置 i, j，我们分别写出交换前后它们的权值。

假设 $s = \Pi_{k=1}^{i-1} a_k$。那么交换前 $ans = \max(\frac{s}{b_i}, \frac{s \times a_i}{b_j})$。交换后 $ans' = \max(\frac{s}{b_j}, \frac{s \times a_j}{b_i})$。

显然有 $\frac{s \times a_i}{b_j} > \frac{s}{b_j}, \frac{s \times a_j}{b_i} > \frac{s}{b_i}$。因为我们希望 $ans < ans'$，所以有 $\frac{s \times a_i}{b_j} < \frac{s \times a_j}{b_i}$。

最后化简出来就是 $a_i \times b_i < a_j \times b_j$，于是据此排序即可。

例题

P2123

现在有 n 个人，全都可以任意排序，对第 i 个人给定 a_i, b_i，第 i 个人的权值 $c_i = \max(c_{i-1}, \sum_{j=1}^{i} a_j) + b_i$，求最大权值最小是多少。

例题

P2123

现在有 n 个人，全都可以任意排序，对第 i 个人给定 a_i, b_i，第 i 个人的权值 $c_i = \max(c_{i-1}, \sum_{j=1}^{i} a_j) + b_i$，求最大权值最小是多少。

还是可以仿照上题的方法，设 $s = \sum_{k=1}^{i-1} a_k$，然后去表示 c_i, c_j。

例题

P2123

现在有 n 个人，全都可以任意排序，对第 i 个人给定 a_i, b_i，第 i 个人的权值 $c_i = \max(c_{i-1}, \sum_{j=1}^{i} a_j) + b_i$，求最大权值最小是多少。

还是可以仿照上题的方法，设 $s = \sum_{k=1}^{i-1} a_k$，然后去表示 c_i, c_j。

$c_i = \max(c_{i-1}, s + a_i) + b_i$, $c_j = \max(c_i, s + a_i + a_j) + b_j$。因为 a_i, b_i 非负所以交换前 $ans = c_j$。同理我们可以得到交换后的 $ans' = \max(c'_j, s + a_i + a_j) + b_i$。

例题

P2123

现在有 n 个人，全都可以任意排序，对第 i 个人给定 a_i, b_i，第 i 个人的权值 $c_i = \max(c_{i-1}, \sum_{j=1}^{i} a_j) + b_i$，求最大权值最小是多少。

还是可以仿照上题的方法，设 $s = \sum_{k=1}^{i-1} a_k$，然后去表示 c_i, c_j。

$c_i = \max(c_{i-1}, s + a_i) + b_i, c_j = \max(c_i, s + a_i + a_j) + b_j$。因为 a_i, b_i 非负所以交换前 $ans = c_j$。同理我们可以得到交换后的 $ans' = \max(c'_j, s + a_i + a_j) + b_i$。

现在我们需要去比较 ans 和 ans' 了，但是这两个东西看起来非常大份，于是我们考虑处理一下式子。

例题

$$ans = \max(c_{i-1} + b_i + b_j, s + a_i + b_i + b_j, s + a_i + a_j + b_j)$$

$$ans' = \max(c_{i-1} + b_i + b_j, s + a_j + b_i + b_j, s + a_i + a_j + b_i)$$

例题

$$ans = \max(c_{i-1} + b_i + b_j, s + a_i + b_i + b_j, s + a_i + a_j + b_j)$$

$$ans' = \max(c_{i-1} + b_i + b_j, s + a_j + b_i + b_j, s + a_i + a_j + b_i)$$

这样形式是好看了但是式子太长了所以我们先换一下元，设 $x = c_{i-1} + b_i + b_j, y = \max(s + a_i + b_i + b_j, s + a_i + a_j + b_j), z = \max(s + a_j + b_i + b_j, s + a_i + a_j + b_i)$。

例题

$$ans = \max(c_{i-1} + b_i + b_j, s + a_i + b_i + b_j, s + a_i + a_j + b_j)$$

$$ans' = \max(c_{i-1} + b_i + b_j, s + a_j + b_i + b_j, s + a_i + a_j + b_i)$$

这样形式是好看了但是式子太长了所以我们先换一下元，设 $x = c_{i-1} + b_i + b_j, y = \max(s + a_i + b_i + b_j, s + a_i + a_j + b_j), z = \max(s + a_j + b_i + b_j, s + a_i + a_j + b_i)$。

于是这两个式子就变成了：

$$ans = \max(x, y)$$

$$ans' = \max(x, z)$$

例题

$$ans = \max(c_{i-1} + b_i + b_j, s + a_i + b_i + b_j, s + a_i + a_j + b_j)$$

$$ans' = \max(c_{i-1} + b_i + b_j, s + a_j + b_i + b_j, s + a_i + a_j + b_i)$$

这样形式是好看了但是式子太长了所以我们先换一下元，设 $x = c_{i-1} + b_i + b_j, y = \max(s + a_i + b_i + b_j, s + a_i + a_j + b_j), z = \max(s + a_j + b_i + b_j, s + a_i + a_j + b_i)$。

于是这两个式子就变成了：

$$ans = \max(x, y)$$

$$ans' = \max(x, z)$$

所以我们为了让 $ans < ans'$，也就需要让 $y < z$，原因显然。

例题

写出来就是让：

$$\max(s+a_i+b_i+b_j, s+a_i+a_j+b_j) < \max(s+a_j+b_i+b_j, s+a_i+a_j+b_i)$$

例题

写出来就是让：

$$\max(s+a_i+b_i+b_j, s+a_i+a_j+b_j) < \max(s+a_j+b_i+b_j, s+a_i+a_j+b_i)$$

化简一下就是：

$$\min(a_i, b_j) < \min(a_j, b_i)$$

例题

写出来就是让：

$$\max(s+a_i+b_i+b_j, s+a_i+a_j+b_j) < \max(s+a_j+b_i+b_j, s+a_i+a_j+b_i)$$

化简一下就是：

$$\min(a_i, b_j) < \min(a_j, b_i)$$

最后注意一下取等号的情况。如果取等就按 a_i 从小到大排序即可。

例题

P3574

给一棵树，配送员从根节点出发，需要遍历所有的点然后回到根节点。每第一次到一个点会给当前点的居民一台电脑，每个点的居民安装电脑需要 c_u 的时间。配送员走一条边需要 1 的时间，最后他会回到根节点再安装自己的电脑。求从他出发到所有人都安装好电脑的最短时间。

例题

P3574

给一棵树，配送员从根节点出发，需要遍历所有的点然后回到根节点。每第一次到一个点会给当前点的居民一台电脑，每个点的居民安装电脑需要 c_u 的时间。配送员走一条边需要 1 的时间，最后他会回到根节点再安装自己的电脑。求从他出发到所有人都安装好电脑的最短时间。

感觉上是比较贪心的，但是好像不太好直接做吧，于是去考虑 dp。

例题

P3574

给一棵树，配送员从根节点出发，需要遍历所有的点然后回到根节点。每第一次到一个点会给当前点的居民一台电脑，每个点的居民安装电脑需要 c_u 的时间。配送员走一条边需要 1 的时间，最后他会回到根节点再安装自己的电脑。求从他出发到所有人都安装好电脑的最短时间。

感觉上是比较贪心的，但是好像不太好直接做吧，于是去考虑 dp。

我们设 f_u 表示处理完以 u 为根的子树的时间，sz_u 表示走完 u 的子树然后回到 u 的时间。

例题

P3574

给一棵树，配送员从根节点出发，需要遍历所有的点然后回到根节点。每第一次到一个点会给当前点的居民一台电脑，每个点的居民安装电脑需要 c_u 的时间。配送员走一条边需要 1 的时间，最后他会回到根节点再安装自己的电脑。求从他出发到所有人都安装好电脑的最短时间。

感觉上是比较贪心的，但是好像不太好直接做吧，于是去考虑 dp。

我们设 f_u 表示处理完以 u 为根的子树的时间，sz_u 表示走完 u 的子树然后回到 u 的时间。

我们容易写出 dp 式子：

$$f_u \leftarrow \max(f_u, f_v + sz_u + 1), sz_u \leftarrow sz_v + 2$$

例题

因为 f_v 是已经求出来的定值，变化的是 sz_u，而 sz_u 的改变会影响 f_u 的取值，所以我们要找到一个合理的顺序。

例题

因为 f_v 是已经求出来的定值，变化的是 sz_u，而 sz_u 的改变会影响 f_u 的取值，所以我们要找到一个合理的顺序。

于是我们考虑两个儿子 x, y。把交换前后的答案写出来：

$$ans = \max(f_u, f_x + sz_u + 1, f_y + sz_u + sz_x + 1)$$

$$ans' = \max(f_u, f_y + sz_u + 1, f_x + sz_u + sz_y + 1)$$

例题

因为 f_v 是已经求出来的定值，变化的是 sz_u，而 sz_u 的改变会影响 f_u 的取值，所以我们要找到一个合理的顺序。

于是我们考虑两个儿子 x, y。把交换前后的答案写出来：

$$ans = \max(f_u, f_x + sz_u + 1, f_y + sz_u + sz_x + 1)$$

$$ans' = \max(f_u, f_y + sz_u + 1, f_x + sz_u + sz_y + 1)$$

不看第一项，然后把 $sz_u + 1$ 丢掉得到：

$$\max(f_x, f_y + sz_x) < \max(f_y, f_x + sz_y)$$

例题

因为 f_v 是已经求出来的定值，变化的是 sz_u，而 sz_u 的改变会影响 f_u 的取值，所以我们要找到一个合理的顺序。

于是我们考虑两个儿子 x, y。把交换前后的答案写出来：

$$ans = \max(f_u, f_x + sz_u + 1, f_y + sz_u + sz_x + 1)$$

$$ans' = \max(f_u, f_y + sz_u + 1, f_x + sz_u + sz_y + 1)$$

不看第一项，然后把 $sz_u + 1$ 丢掉得到：

$$\max(f_x, f_y + sz_x) < \max(f_y, f_x + sz_y)$$

这不就是国王游戏那道题了吗？直接根据 $f_x - sz_x > f_y - sz_y$ 排序即可。

例题

P9128

给一棵树，从根节点出发遍历所有点，设第一次到这个点的时间为 t_i，这个点的代价为 $t_i \times a_i$。要求必须/不必回到根节点，问遍历所有节点的最小时间和此时需要付出的最小的代价。

例题

P9128

给一棵树，从根节点出发遍历所有点，设第一次到这个点的时间为 t_i，这个点的代价为 $t_i \times a_i$。要求必须/不必回到根节点，问遍历所有节点的最小时间和此时需要付出的最小的代价。

和上一道题一样，考虑树 dp。我们设 f_u/g_u 表示第 $1/2$ 种问题的答案，s_u 表示 u 的子树 a 的和，sz_u 表示子树大小。

例题

P9128

给一棵树，从根节点出发遍历所有点，设第一次到这个点的时间为 t_i，这个点的代价为 $t_i \times a_i$。要求必须/不必回到根节点，问遍历所有节点的最小时间和此时需要付出的最小的代价。

和上一道题一样，考虑树 dp。我们设 f_u/g_u 表示第 $1/2$ 种问题的答案，s_u 表示 u 的子树 a 的和，sz_u 表示子树大小。

转移是简单的这里直接上式子：

$$f_u = \left(\sum_{v \in son_u} f_v + s_v \right) + 2 \left(\sum_{i=1}^{sonsz_u} s_{p_i} \sum_{j=1}^{i-1} sz_{p_j} \right)$$

例题

容易发现前面部分是定值，后面部分是国王游戏，于是就做完了

例题

容易发现前面部分是定值，后面部分是国王游戏，于是就做完了……了吗？

例题

容易发现前面部分是定值，后面部分是国王游戏，于是就做完了……了吗？

还有第二种不回根的要处理。其实这个也是简单的。

例题

容易发现前面部分是定值，后面部分是国王游戏，于是就做完了……了吗？

还有第二种不回根的要处理。其实这个也是简单的。

你考虑我们肯定选择一个最深的点停下来，所以我们可以预处理出深度以及需要考虑的点。对于其他的点我们是不是就正常做即可，现在考虑特殊的点。

例题

容易发现前面部分是定值，后面部分是国王游戏，于是就做完了……了吗？

还有第二种不回根的要处理。其实这个也是简单的。

你考虑我们肯定选择一个最深的点停下来，所以我们可以预处理出深度以及需要考虑的点。对于其他的点我们是不是就正常做即可，现在考虑特殊的点。

假设我们处理到 u 现在枚举到了一个 v 是特殊的点，假如我们用它那么它一定要被放在最后面。但是实际上它可能并没有，所以我们把原来的贡献减一下再加在最后面即可。

前言

在暴切水题之前我先讲讲我对反悔贪心的理解。就是说当我们遇到有限制的贪心的时候不能直接贪心，因为这样要么不满足限制，要么不是最优解。这个时候我们就需要加入一个“反悔”操作确保我们的正确性。

前言

在暴切水题之前我先讲讲我对反悔贪心的理解。就是说当我们遇到有限制的贪心的时候不能直接贪心，因为这样要么不满足限制，要么不是最优解。这个时候我们就需要加入一个“反悔”操作确保我们的正确性。

相当于我们在贪心的时候不仅考虑没选择的元素，还要考虑已选的元素。如果扔掉已选元素中最劣的然后限制变松可以选其他的元素让答案更优我们就“反悔”。

前言

在暴切水题之前我先讲讲我对反悔贪心的理解。就是说当我们遇到有限制的贪心的时候不能直接贪心，因为这样要么不满足限制，要么不是最优解。这个时候我们就需要加入一个“反悔”操作确保我们的正确性。

相当于我们在贪心的时候不仅考虑没选择的元素，还要考虑已选的元素。如果扔掉已选元素中最劣的然后限制变松可以选其他的元素让答案更优我们就“反悔”。

然后其实你可以类比网络流。网络流本质上也是贪心，然后里面对反边的操作也对应了这里的“反悔”操作，所以我们可以在两种类型的题之间转换。只不过有的题从反悔贪心的角度来做就很简洁，但是有的题又需要你从网络流的角度去思考。

前言

在暴切水题之前我先讲讲我对反悔贪心的理解。就是说当我们遇到有限制的贪心的时候不能直接贪心，因为这样要么不满足限制，要么不是最优解。这个时候我们就需要加入一个“反悔”操作确保我们的正确性。

相当于我们在贪心的时候不仅考虑没选择的元素，还要考虑已选的元素。如果扔掉已选元素中最劣的然后限制变松可以选其他的元素让答案更优我们就“反悔”。

然后其实你可以类比网络流。网络流本质上也是贪心，然后里面对反边的操作也对应了这里的“反悔”操作，所以我们可以在两种类型的题之间转换。只不过有的题从反悔贪心的角度来做就很简洁，但是有的题又需要你从网络流的角度去思考。

注意这里的“从网络流的角度”并不是让你死板的用网络流算法，只是说你可以借助网络流模型去理解思考题。当然如果网络流的时间复杂度已经对完了那不直接 rush?

前言

前面说了那么多，总而言之就是适当的结合两种算法在处理这类题的时候会有奇效。所以我并没有专门开一个网络流的标签，题就丢到反悔贪心里面了。

前言

前面说了那么多，总而言之就是适当的结合两种算法在处理这类题的时候会有奇效。所以我并没有专门开一个网络流的标签，题就丢到反悔贪心里面了。

那话不多说我们现在开始看题吧（()

例题

P2949

有 n 个工作，每个工作有截止时间和利润，完成一个工作需要一个单位时间，求最大利润。

例题

P2949

有 n 个工作，每个工作有截止时间和利润，完成一个工作需要一个单位时间，求最大利润。

按时间排序后用堆维护已选的工作的利润即可。

例题

P4053

有 n 个工作，每个工作有截止时间，完成每个工作会花费一定的时间。求最多能完成多少工作。

例题

P4053

有 n 个工作，每个工作有截止时间，完成每个工作会花费一定的时间。求最多能完成多少工作。

按截止时间排序后用堆维护已选的工作的花费时间即可。

例题

P1484

有 n 个位置需要种最多 k 棵树，每个位置种树可以获得一个利润。要求相邻位置不能种树，求最大获利。

例题

P1484

有 n 个位置需要种最多 k 棵树，每个位置种树可以获得一个利润。要求相邻位置不能种树，求最大获利。

假设我们选了位置 i，如果我们要反悔肯定是将 i 替换成它相邻两个的和，否则一定不优，这是显然的。

例题

P1484

有 n 个位置需要种最多 k 棵树，每个位置种树可以获得一个利润。要求相邻位置不能种树，求最大获利。

假设我们选了位置 i，如果我们要反悔肯定是将 i 替换成它相邻两个的和，否则一定不优，这是显然的。

考虑替换的代价是 $a_{i-1}+a_{i+1}-a_i$，于是我们考虑用堆维护未选的元素。当我们选择了 i 的时候我们将 a_i 删掉然后加入 $a_{i-1}+a_{i+1}-a_i$，这样就能保证之后可以“反悔”。然后删除的时候需要合并两边的信息所以还要一个链表。

例题

P3045

有 n 个物品，每个物品有两种价格，第一种是原价 a_i，还有一种是优惠价 b_i。你有 k 张优惠券和 m 的钱，求最多能买多少物品。

例题

P3045

有 n 个物品，每个物品有两种价格，第一种是原价 a_i，还有一种是优惠价 b_i。你有 k 张优惠券和 m 的钱，求最多能买多少物品。

假设已经用完了优惠券，已知有个 i 已经用了优惠券，现在考虑到了 j。如果 $b_i + a_j > a_i + b_j$ 那么我们就可以“反悔”。

例题

P3045

有 n 个物品，每个物品有两种价格，第一种是原价 a_i，还有一种是优惠价 b_i。你有 k 张优惠券和 m 的钱，求最多能买多少物品。

假设已经用完了优惠券，已知有个 i 已经用了优惠券，现在考虑到了 j。如果 $b_i + a_j > a_i + b_j$ 那么我们就可以“反悔”。

我们整理一下式子，也就是说当 $a_j - b_j > a_i - b_i$ 的时候我们需要“反悔”之前用给 i 的优惠券转而用给 j，于是用堆维护 Δ 即可。

例题

AGC018C

有 $x+y+z$ 个人，每个人都有不同数量的金币银币铜币，现在要求 x 个人保留金币，y 个人保留银币，z 个人保留铜币，问最多可以保留多少硬币。

例题

AGC018C

有 $x+y+z$ 个人，每个人都有不同数量的金币银币铜币，现在要求 x 个人保留金币，y 个人保留银币，z 个人保留铜币，问最多可以保留多少硬币。

我们首先考虑只有两种硬币的情况，我们可以让所有人全选一种，然后再将调整后收益更高的人给调整成另外的硬币。

例题

AGC018C

有 $x+y+z$ 个人，每个人都有不同数量的金币银币铜币，现在要求 x 个人保留金币，y 个人保留银币，z 个人保留铜币，问最多可以保留多少硬币。

我们首先考虑只有两种硬币的情况，我们可以让所有人全选一种，然后再将调整后收益更高的人给调整成另外的硬币。

但是现在有三种颜色似乎不太好做怎么办呢？假设原来每个元素是三元组 (a,b,c)，现在我们将其变成二元组 $(q=b-a,p=c-a)$ 表示我们先全部变成金币，然后再考虑转成其他硬币的收益。

例题

AGC018C

有 $x+y+z$ 个人，每个人都有不同数量的金币银币铜币，现在要求 x 个人保留金币，y 个人保留银币，z 个人保留铜币，问最多可以保留多少硬币。

我们首先考虑只有两种硬币的情况，我们可以让所有人全选一种，然后再将调整后收益更高的人给调整成另外的硬币。

但是现在有三种颜色似乎不太好做怎么办呢？假设原来每个元素是三元组 (a,b,c)，现在我们将其变成二元组 $(q=b-a,p=c-a)$ 表示我们先全部变成金币，然后再考虑转成其他硬币的收益。

于是题目就变成在 n 个元素中选 y 个获得 q 的收益，z 个获得 p 的收益，要求最大化总收益。

例题

最理想的情况就是我们直接找 y 个最大的 q 和 z 个最大的 p，但是这样大概率会选重。

例题

最理想的情况就是我们直接找 y 个最大的 q 和 z 个最大的 p，但是这样大概率会选重。

那么我们考虑在没有选中的元素中选择最大的，然后再考虑交换一些选 q 和选 p 的元素能不能让答案变大。

例题

最理想的情况就是我们直接找 y 个最大的 q 和 z 个最大的 p，但是这样大概率会选重。

那么我们考虑在没有选中的元素中选择最大的，然后再考虑交换一些选 q 和选 p 的元素能不能让答案变大。

我们尝试去写一下式子，假设 i 选 q j 选 p，当 $q_i + p_j < p_i + q_j$ 时交换更优。

例题

最理想的情况就是我们直接找 y 个最大的 q 和 z 个最大的 p，但是这样大概率会选重。

那么我们考虑在没有选中的元素中选择最大的，然后再考虑交换一些选 q 和选 p 的元素能不能让答案变大。

我们尝试去写一下式子，假设 i 选 q j 选 p，当 $q_i + p_j < p_i + q_j$ 时交换更优。

我们化简式子得到 $q_i - p_i < q_j - p_j$，所以如果我们按 $q_i - p_i$ 降序排序那么我们最优的选择方式一定是在一段前缀选 q 以及一段后缀选 p。

例题

最理想的情况就是我们直接找 y 个最大的 q 和 z 个最大的 p，但是这样大概率会选重。

那么我们考虑在没有选中的元素中选择最大的，然后再考虑交换一些选 q 和选 p 的元素能不能让答案变大。

我们尝试去写一下式子，假设 i 选 q j 选 p，当 $q_i + p_j < p_i + q_j$ 时交换更优。

我们化简式子得到 $q_i - p_i < q_j - p_j$，所以如果我们按 $q_i - p_i$ 降序排序那么我们最优的选择方式一定是在一段前缀选 q 以及一段后缀选 p。

所以我们处理出前缀选 q 和后缀选 p 的答案，然后枚举一下分界点即可。

例题

CF802O

有 n 道题，第 i 天可以花 a_i 准备一道，也可以花 b_i 打印一道，求准备并打印 k 道题的最少花费。

例题

CF802O

有 n 道题，第 i 天可以花 a_i 准备一道，也可以花 b_i 打印一道，求准备并打印 k 道题的最少花费。

看到恰好 k 是不是可以考虑 wqs 二分啊?

例题

CF802O

有 n 道题，第 i 天可以花 a_i 准备一道，也可以花 b_i 打印一道，求准备并打印 k 道题的最少花费。

看到恰好 k 是不是可以考虑 wqs 二分啊?

然后 check 里面我们就只用管怎么选代价最小了。反悔贪心直接做即可。

例题

CF802O

有 n 道题，第 i 天可以花 a_i 准备一道，也可以花 b_i 打印一道，求准备并打印 k 道题的最少花费。

看到恰好 k 是不是可以考虑 wqs 二分啊？

然后 check 里面我们就只用管怎么选代价最小了。反悔贪心直接做即可。

当然这里会讲一种简洁的做法。我们对每个 b_i 考虑。如果有一个很小的 a 没有用我们就可以把它和 b_i 组成一道题，如果之前有一个 b 更大，那么我们可以用 b_i 替换掉。

例题

CF802O

有 n 道题，第 i 天可以花 a_i 准备一道，也可以花 b_i 打印一道，求准备并打印 k 道题的最少花费。

看到恰好 k 是不是可以考虑 wqs 二分啊？

然后 check 里面我们就只用管怎么选代价最小了。反悔贪心直接做即可。

当然这里会讲一种简洁的做法。我们对每个 b_i 考虑。如果有一个很小的 a 没有用我们就可以把它和 b_i 组成一道题，如果之前有一个 b 更大，那么我们可以用 b_i 替换掉。

为了方便你可以将 a 和 b 丢到一起用堆维护，复杂度 $\mathcal{O}(n \log^2 n)$。

例题

P5470

有两个序列，每个序列有 n 个元素，每个元素都有权值。对于每个序列你需要选择 m 个位置，满足至少有 k 个位置被同时选中。求最大权值。

例题

P5470

有两个序列，每个序列有 n 个元素，每个元素都有权值。对于每个序列你需要选择 m 个位置，满足至少有 k 个位置被同时选中。求最大权值。

貌似直接分析有点小困难（当然你秒了可以当我没说），所以我们可以网络流建模然后分析。

例题

P5470

有两个序列，每个序列有 n 个元素，每个元素都有权值。对于每个序列你需要选择 m 个位置，满足至少有 k 个位置被同时选中。求最大权值。

貌似直接分析有点小困难（当然你秒了可以当我没说），所以我们可以网络流建模然后分析。

考虑可以 s 连每个 a_i，流量为 1，边权为 a_i，a_i 连 b_i，流量为 1，边权为 0，b_i 连 t，流量为 1，边权为 b_i。因为还有一个至少选择 k 个相同位置的限制，所以新建两个虚点 u 和 v，每个 a_i 连 u，流量为 1，边权为 0，u 连 v，流量为 $m-k$，边权为 0，v 连每个 b_i，流量为 1，边权为 0。

例题

P5470

有两个序列，每个序列有 n 个元素，每个元素都有权值。对于每个序列你需要选择 m 个位置，满足至少有 k 个位置被同时选中。求最大权值。

貌似直接分析有点小困难（当然你秒了可以当我没说），所以我们可以网络流建模然后分析。

考虑可以 s 连每个 a_i，流量为 1，边权为 a_i，a_i 连 b_i，流量为 1，边权为 0，b_i 连 t，流量为 1，边权为 b_i。因为还有一个至少选择 k 个相同位置的限制，所以新建两个虚点 u 和 v，每个 a_i 连 u，流量为 1，边权为 0，u 连 v，流量为 $m-k$，边权为 0，v 连每个 b_i，流量为 1，边权为 0。

因为要求至少 k 个位置相同，这等价于至多 $m-k$ 个位置不同，所以 (u,v) 的流量为 $m-k$。然后我们设起始流量为 m 去跑最大费用最大流即可。

例题

考虑在图上分析可能的增广情况。因为每个点至多经过一次，所以增广路的形态不多，所以我们才能直接分讨做这道题。

例题

考虑在图上分析可能的增广情况。因为每个点至多经过一次，所以增广路的形态不多，所以我们才能直接分讨做这道题。

接下来我将这五种情况列出来，下面默认起点是 s 终点是 t:

例题

考虑在图上分析可能的增广情况。因为每个点至多经过一次，所以增广路的形态不多，所以我们才能直接分讨做这道题。

接下来我将这五种情况列出来，下面默认起点是 s 终点是 t:

1. $a_i \to b_i$

例题

考虑在图上分析可能的增广情况。因为每个点至多经过一次，所以增广路的形态不多，所以我们才能直接分讨做这道题。

接下来我将这五种情况列出来，下面默认起点是 s 终点是 t:

1. $a_i \to b_i$

2. $a_i \to b_j$

例题

考虑在图上分析可能的增广情况。因为每个点至多经过一次，所以增广路的形态不多，所以我们才能直接分讨做这道题。

接下来我将这五种情况列出来，下面默认起点是 s 终点是 t:

1. $a_i \to b_i$
2. $a_i \to b_j$
3. $a_i \to b_i \to a_j \to b_j$

例题

考虑在图上分析可能的增广情况。因为每个点至多经过一次，所以增广路的形态不多，所以我们才能直接分讨做这道题。

接下来我将这五种情况列出来，下面默认起点是 s 终点是 t:

1. $a_i \to b_i$
2. $a_i \to b_j$
3. $a_i \to b_i \to a_j \to b_j$
4. $a_i \to b_i \to a_j \to b_l$

例题

考虑在图上分析可能的增广情况。因为每个点至多经过一次，所以增广路的形态不多，所以我们才能直接分讨做这道题。

接下来我将这五种情况列出来，下面默认起点是 s 终点是 t:

1. $a_i \to b_i$
2. $a_i \to b_j$
3. $a_i \to b_i \to a_j \to b_j$
4. $a_i \to b_i \to a_j \to b_l$
5. $a_i \to b_j \to a_l \to b_l$

例题

这下就可以暴力分讨了。我们考虑这几种增广路的优先级。

例题

这下就可以暴力分讨了。我们考虑这几种增广路的优先级。

情况 1 一定是最优的，然后考虑情况 2 可行的时候需要 (u, v) 还有流量，因为它会消耗这条边的流量。对应的，有情况 3 可以增加 (u, v) 的流量，所以情况 3 的优先级一定是高于情况 2 的。

例题

这下就可以暴力分讨了。我们考虑这几种增广路的优先级。

情况 1 一定是最优的，然后考虑情况 2 可行的时候需要 (u, v) 还有流量，因为它会消耗这条边的流量。对应的，有情况 3 可以增加 (u, v) 的流量，所以情况 3 的优先级一定是高于情况 2 的。

剩下的情况 4、5 都只是在调整情况 2 的基础上做一次情况 1，所以优先级不重要。于是我们就可以钦定一个考虑的顺序了：1>3>4>5>2。

例题

这下就可以暴力分讨了。我们考虑这几种增广路的优先级。

情况 1 一定是最优的，然后考虑情况 2 可行的时候需要 (u, v) 还有流量，因为它会消耗这条边的流量。对应的，有情况 3 可以增加 (u, v) 的流量，所以情况 3 的优先级一定是高于情况 2 的。

剩下的情况 4、5 都只是在调整情况 2 的基础上做一次情况 1，所以优先级不重要。于是我们就可以钦定一个考虑的顺序了：$1>3>4>5>2$。

最后就是如何增广的问题，这个你直接对于每个部分都开一个堆维护即可，实际写起来没有太多细节但是有点麻烦，具体可以翻我代码。

例题

CF335F

有 n 个馅饼，对于你以原价购买的每一个馅饼，你可以免费选择一个价格严格低于它的馅饼。求买下所有馅饼的最小代价。

例题

CF335F

有 n 个馅饼，对于你以原价购买的每一个馅饼，你可以免费选择一个价格严格低于它的馅饼。求买下所有馅饼的最小代价。

考虑到相同价格的不能免费，于是对于一个价格的馅饼一起处理。

例题

CF335F

有 n 个馅饼，对于你以原价购买的每一个馅饼，你可以免费选择一个价格严格低于它的馅饼。求买下所有馅饼的最小代价。

考虑到相同价格的不能免费，于是对于一个价格的馅饼一起处理。并且免费的限制是对于价格严格低的有用，所以从大到小考虑。

例题

CF335F

有 n 个馅饼，对于你以原价购买的每一个馅饼，你可以免费选择一个价格严格低于它的馅饼。求买下所有馅饼的最小代价。

考虑到相同价格的不能免费，于是对于一个价格的馅饼一起处理。并且免费的限制是对于价格严格低的有用，所以从大到小考虑。

假设当前馅饼价格为 a_i，有 cnt_i 个，有 tot 次白嫖的机会。考虑用一个堆维护暂定被白嫖的馅饼。

例题

CF335F

有 n 个馅饼，对于你以原价购买的每一个馅饼，你可以免费选择一个价格严格低于它的馅饼。求买下所有馅饼的最小代价。

考虑到相同价格的不能免费，于是对于一个价格的馅饼一起处理。并且免费的限制是对于价格严格低的有用，所以从大到小考虑。

假设当前馅饼价格为 a_i，有 cnt_i 个，有 tot 次白嫖的机会。考虑用一个堆维护暂定被白嫖的馅饼。

每次取出最小的被白嫖的馅饼，假设价格为 x。如果 $x < a_i$ 一定会反悔，考虑反悔前代价多了 $2a_i$，反悔后代价多了 k。

例题

那如果是 $x \geq a_i$ 呢？如果当前只剩了一个馅饼倒还好，如果剩下至少两个，并且有 $2a_i > x$ 了那么还是需要反悔。这时候有一个巧妙的地方就是我们把 $2a_i - x$ 加入堆中，这样就能达到反悔的效果。

例题

那如果是 $x \geq a_i$ 呢？如果当前只剩了一个馅饼倒还好，如果剩下至少两个，并且有 $2a_i > x$ 了那么还是需要反悔。这时候有一个巧妙的地方就是我们把 $2a_i - x$ 加入堆中，这样就能达到反悔的效果。

最后注意一下就是每轮不要直接把新的白嫖的馅饼加进堆中，因为这样会影响后面的处理。正确的做法是先开一个数组存一下最后一起加进去。

前言

在结束之前，我们再看一点简单的杂题。因为这些题都是根据题目顺势而为得出的贪心，所以就没有具体的总结了。这里就给大家见一些我觉得还不错的贪心题或是经典的贪心。

例题

P1809

有 n 个人要过河，只有一条可同时坐两人的船。过河时间以较慢的人为准。现在给出每个人的过河时间，求最短多久所有人都能过河。

例题

P1809

有 n 个人要过河，只有一条可同时坐两人的船。过河时间以较慢的人为准。现在给出每个人的过河时间，求最短多久所有人都能过河。

可以归纳做。$n < 4$ 是平凡的。考虑 $n = 4$。

例题

P1809

有 n 个人要过河，只有一条可同时坐两人的船。过河时间以较慢的人为准。现在给出每个人的过河时间，求最短多久所有人都能过河。

可以归纳做。$n < 4$ 是平凡的。考虑 $n = 4$。

因为我尽职尽责所以还是模拟一下。第一种策略：$t_1 + t_2 + t_3 + t_1 + t_4$，第二种策略：$t_2 + t_1 + t_2 + t_2 + t_4$。

例题

P1809

有 n 个人要过河，只有一条可同时坐两人的船。过河时间以较慢的人为准。现在给出每个人的过河时间，求最短多久所有人都能过河。

可以归纳做。$n < 4$ 是平凡的。考虑 $n = 4$。

因为我尽职尽责所以还是模拟一下。第一种策略：$t_1 + t_2 + t_3 + t_1 + t_4$，第二种策略：$t_2 + t_1 + t_2 + t_2 + t_4$。这个显然两者取 $\min$ 即可。

例题

P1809

有 n 个人要过河，只有一条可同时坐两人的船。过河时间以较慢的人为准。现在给出每个人的过河时间，求最短多久所有人都能过河。

可以归纳做。$n < 4$ 是平凡的。考虑 $n = 4$。

因为我尽职尽责所以还是模拟一下。第一种策略：$t_1 + t_2 + t_3 + t_1 + t_4$，第二种策略：$t_2 + t_1 + t_2 + t_2 + t_4$。这个显然两者取 $\min$ 即可。

$n = 5$ 的情况也可以去手玩一下，在此不罗列了。其实这个问题我们每次可以只考虑最慢的两个人渡河的时间。设最慢的需要 x 时间渡河，次慢的需要 y 时间渡河。我们是不是也可以得到两种解决办法：

例题

1. $2a_1 + x + y$

2. $a_1 + 2a_2 + x$

例题

1. $2a_1 + x + y$

2. $a_1 + 2a_2 + x$

于是比较哪种更优选哪种。

例题

P10609

题面有点长于是懒惰的林月如在此省略了 1 万字 awa

例题

P10609

题面有点长于是懒惰的林月如在此省略了 1 万字 awa

游戏是这样的，先手只需要留住一张牌，而后手需要考虑的事情就多了。

例题

P10609

题面有点长于是懒惰的林月如在此省略了 1 万字 awa

游戏是这样的，先手只需要留住一张牌，而后手需要考虑的事情就多了。

首先每类牌实际上只用留一张就好了，所以可以先把那些有多张的扔掉肯定不劣。

例题

P10609

题面有点长于是懒惰的林月如在此省略了 1 万字 awa

游戏是这样的，先手只需要留住一张牌，而后手需要考虑的事情就多了。

首先每类牌实际上只用留一张就好了，所以可以先把那些有多张的扔掉肯定不劣。

为了兜底，先手一定会把最小值最小的那一行删掉。因为万一没有删掉，后手就可以把那一列留到最后就完蛋了。

例题

对于后手来说，如果某一列某个位置是某一行的最小值，那后手一定不会删这一列，因为这会抬高先手的最低分数，这显然不是后手想看到的。

例题

对于后手来说，如果某一列某个位置是某一行的最小值，那后手一定不会删这一列，因为这会抬高先手的最低分数，这显然不是后手想看到的。

注意到先手删了一行之后矩阵变成了 $(n-1) \times n$，因此一定存在一列没有任何一个数是某一行最小值，删掉这一列就可以了。

例题

P9965

小 E 有 n 种颜色的球，其中第 i 种有 a_i 个。有两类工具，第一类可以把一个指定颜色的球变成一个任意颜色的球；第二类可以把一个指定颜色的球变成两个这种颜色的球。一个变化之后的球也可以通过工具产生新的变化。关于第 i 种颜色的第一类工具有 b_i 个，第二类工具有 c_i 个。小 E 想知道，如果每一个工具最多只能使用一次，那么对于每种颜色 i，第 i 种颜色的球最后最多能有多少个。以及，小 E 最后最多能有多少个球。

例题

P9965

小 E 有 n 种颜色的球，其中第 i 种有 a_i 个。有两类工具，第一类可以把一个指定颜色的球变成一个任意颜色的球；第二类可以把一个指定颜色的球变成两个这种颜色的球。一个变化之后的球也可以通过工具产生新的变化。关于第 i 种颜色的第一类工具有 b_i 个，第二类工具有 c_i 个。小 E 想知道，如果每一个工具最多只能使用一次，那么对于每种颜色 i，第 i 种颜色的球最后最多能有多少个。以及，小 E 最后最多能有多少个球。

所有 $a_i \neq 0$ 的先把 c_i 用了肯定是好的。

例题

P9965

小 E 有 n 种颜色的球，其中第 i 种有 a_i 个。有两类工具，第一类可以把一个指定颜色的球变成一个任意颜色的球；第二类可以把一个指定颜色的球变成两个这种颜色的球。一个变化之后的球也可以通过工具产生新的变化。关于第 i 种颜色的第一类工具有 b_i 个，第二类工具有 c_i 个。小 E 想知道，如果每一个工具最多只能使用一次，那么对于每种颜色 i，第 i 种颜色的球最后最多能有多少个。以及，小 E 最后最多能有多少个球。

所有 $a_i \neq 0$ 的先把 c_i 用了肯定是好的。

对于 $a_i = 0 \wedge b_i > 0$ 的我拿一个球过去，然后把他的 c_i 用掉肯定也是不劣的，因为他反正一定还可以把这个球拿出来。

例题

P9965

小 E 有 n 种颜色的球，其中第 i 种有 a_i 个。有两类工具，第一类可以把一个指定颜色的球变成一个任意颜色的球；第二类可以把一个指定颜色的球变成两个这种颜色的球。一个变化之后的球也可以通过工具产生新的变化。关于第 i 种颜色的第一类工具有 b_i 个，第二类工具有 c_i 个。小 E 想知道，如果每一个工具最多只能使用一次，那么对于每种颜色 i，第 i 种颜色的球最后最多能有多少个。以及，小 E 最后最多能有多少个球。

所有 $a_i \neq 0$ 的先把 c_i 用了肯定是好的。

对于 $a_i = 0 \wedge b_i > 0$ 的我拿一个球过去，然后把他的 c_i 用掉肯定也是不劣的，因为他反正一定还可以把这个球拿出来。

所以在经过初步处理后就只剩下两种类型的球：$(a_i, b_i, 0), (0, 0, c_i)$。

例题

考虑第一问，直接把所有球能挪就挪到 i 上，然后如果 i 是第二类，那么再把 c_i 用掉就可以了。

例题

考虑第一问，直接把所有球能挪就挪到 i 上，然后如果 i 是第二类，那么再把 c_i 用掉就可以了。

考虑第二问，事实上第二类球也能够产生用途，我拿一个球给他，他虽然拿不出去，但是可以用掉他的 c 来使得球的总数增多。因为拿不出来了，所以我们可能只能给有限个第二类球各分配一个球。那么显然我们应该优先分配给 c 更大的。

例题

考虑第一问，直接把所有球能挪就挪到 i 上，然后如果 i 是第二类，那么再把 c_i 用掉就可以了。

考虑第二问，事实上第二类球也能够产生用途，我拿一个球给他，他虽然拿不出去，但是可以用掉他的 c 来使得球的总数增多。因为拿不出来了，所以我们可能只能给有限个第二类球各分配一个球。那么显然我们应该优先分配给 c 更大的。

于是排个序直接贪就做完了，时间复杂度 $\mathcal{O}(n \log n)$。

例题

P5290

给一棵树，每个点都有权值。要求把点划分成若干集合，要求集合内的点不存在祖先后代的关系。一个集合的贡献是集合中的最大值，求整棵树最小贡献。

例题

P5290

给一棵树，每个点都有权值。要求把点划分成若干集合，要求集合内的点不存在祖先后代的关系。一个集合的贡献是集合中的最大值，求整棵树最小贡献。

我们考虑用堆去维护每个点为根的子树中每个集合的最大值。现在我们需要考虑的是合并子树信息。

例题

P5290

给一棵树，每个点都有权值。要求把点划分成若干集合，要求集合内的点不存在祖先后代的关系。一个集合的贡献是集合中的最大值，求整棵树最小贡献。

我们考虑用堆去维护每个点为根的子树中每个集合的最大值。现在我们需要考虑的是合并子树信息。

考虑一定是贡献大的与贡献大的合并，贡献小的与贡献小的合并。证明考虑反证法。

例题

P5290

给一棵树，每个点都有权值。要求把点划分成若干集合，要求集合内的点不存在祖先后代的关系。一个集合的贡献是集合中的最大值，求整棵树最小贡献。

我们考虑用堆去维护每个点为根的子树中每个集合的最大值。现在我们需要考虑的是合并子树信息。

考虑一定是贡献大的与贡献大的合并，贡献小的与贡献小的合并。证明考虑反证法。

具体就是说如果两个大的集合不合并到一起那么这两个大的贡献就都会被算到，否则就只会算其中更大的贡献。

例题

P5290

给一棵树，每个点都有权值。要求把点划分成若干集合，要求集合内的点不存在祖先后代的关系。一个集合的贡献是集合中的最大值，求整棵树最小贡献。

我们考虑用堆去维护每个点为根的子树中每个集合的最大值。现在我们需要考虑的是合并子树信息。

考虑一定是贡献大的与贡献大的合并，贡献小的与贡献小的合并。证明考虑反证法。

具体就是说如果两个大的集合不合并到一起那么这两个大的贡献就都会被算到，否则就只会算其中更大的贡献。

于是直接启发式合并即可，时间复杂度两只 $\log$。

例题

P6631

有一个非负整数序列，每一步你可以选择一个区间 $[l, r]$ 执行以下三种操作中的一种：

1. 将下标在这个区间里的所有数都减 1。

2. 将下标在这个区间里且下标为奇数的所有数都减 1。

3. 将下标在这个区间里且下标为偶数的所有数都减 1。

求最少多少步能将所有数变成 0。

例题

解决这个问题我们可以从最终的形态入手。我们称第一种操作的区间为一类区间，第二种操作的区间为二类区间。

例题

解决这个问题我们可以从最终的形态入手。我们称第一种操作的区间为一类区间，第二种操作的区间为二类区间。

首先我们考虑 a_1 是怎么变成 0 的？我们肯定会操作若干左端点为 1 的第一类和第二类区间。

例题

解决这个问题我们可以从最终的形态入手。我们称第一种操作的区间为一类区间，第二种操作的区间为二类区间。

首先我们考虑 a_1 是怎么变成 0 的？我们肯定会操作若干左端点为 1 的第一类和第二类区间。

考虑处理完了 a_1 后最左边是不是肯定不会再有区间覆盖了，那么我们的 a_2 是不是就变成了新的左端点？

例题

解决这个问题我们可以从最终的形态入手。我们称第一种操作的区间为一类区间，第二种操作的区间为二类区间。

首先我们考虑 a_1 是怎么变成 0 的？我们肯定会操作若干左端点为 1 的第一类和第二类区间。

考虑处理完了 a_1 后最左边是不是肯定不会再有区间覆盖了，那么我们的 a_2 是不是就变成了新的左端点？

这就启发我们从左往右依次处理 a_i。现在我们应当考虑处理到了 i，前面有 x 个一类区间还没有分配右端点，y 个二类区间还没有分配右端点，要求将 a_i 消成 0 该怎么分配。

例题

解决这个问题我们可以从最终的形态入手。我们称第一种操作的区间为一类区间，第二种操作的区间为二类区间。

首先我们考虑 a_1 是怎么变成 0 的？我们肯定会操作若干左端点为 1 的第一类和第二类区间。

考虑处理完了 a_1 后最左边是不是肯定不会再有区间覆盖了，那么我们的 a_2 是不是就变成了新的左端点？

这就启发我们从左往右依次处理 a_i。现在我们应当考虑处理到了 i，前面有 x 个一类区间还没有分配右端点，y 个二类区间还没有分配右端点，要求将 a_i 消成 0 该怎么分配。

注意到奇数和偶数下标的二类区间独立所以需要分开记录。

例题

凭我们的直觉肯定是能用一类区间就先用，但是这样真对吗？

例题

凭我们的直觉肯定是能用一类区间就先用，但是这样真对吗？唉其实还真是，但是为什么呢？我们来简单证明一下。

例题

凭我们的直觉肯定是能用一类区间就先用，但是这样真对吗？唉其实还真是，但是为什么呢？我们来简单证明一下。

考虑当前用二类区间，这样下一个位置就不会被覆盖，于是考虑下一个位置会用哪种区间。

例题

凭我们的直觉肯定是能用一类区间就先用，但是这样真对吗？唉其实还真是，但是为什么呢？我们来简单证明一下。

考虑当前用二类区间，这样下一个位置就不会被覆盖，于是考虑下一个位置会用哪种区间。

如果下一个位置用一类区间，我们考虑可以等价为在当前用一类区间然后在下一个位置用二类区间。

例题

凭我们的直觉肯定是能用一类区间就先用，但是这样真对吗？唉其实还真是，但是为什么呢？我们来简单证明一下。

考虑当前用二类区间，这样下一个位置就不会被覆盖，于是考虑下一个位置会用哪种区间。

如果下一个位置用一类区间，我们考虑可以等价为在当前用一类区间然后在下一个位置用二类区间。

如果下一个位置还用二类区间，我们可以考虑把这两个区间的交替换成一个一类区间。于是得证。

例题

凭我们的直觉肯定是能用一类区间就先用，但是这样真对吗？唉其实还真是，但是为什么呢？我们来简单证明一下。

考虑当前用二类区间，这样下一个位置就不会被覆盖，于是考虑下一个位置会用哪种区间。

如果下一个位置用一类区间，我们考虑可以等价为在当前用一类区间然后在下一个位置用二类区间。

如果下一个位置还用二类区间，我们可以考虑把这两个区间的交替换成一个一类区间。于是得证。

于是我们继续考虑之前提到的简化后的问题。考虑如果 $x+y \leq a_i$，我们直接考虑还要多加入多少一类区间即可。

例题

具体就是贪心地考虑能放多少一类区间，显然数量是 $\min(a_i, a_{i+1})$。然后剩下如果还没消完就放二类区间。

例题

具体就是贪心地考虑能放多少一类区间，显然数量是 $\min(a_i, a_{i+1})$。然后剩下如果还没消完就放二类区间。

最后考虑 $x+y>a_i$ 的情况。首先肯定有 $x+y-a_i$ 个区间不能要了，我们令 $k=x+y-a_i$。考虑对一类区间和二类区间数量都减去 k，这样我们实际少了 k 个区间，对于丢失的区间我们可以看成额外可以从 i 开始的区间。

例题

具体就是贪心地考虑能放多少一类区间，显然数量是 $\min(a_i, a_{i+1})$。然后剩下如果还没消完就放二类区间。

最后考虑 $x + y > a_i$ 的情况。首先肯定有 $x + y - a_i$ 个区间不能要了，我们令 $k = x + y - a_i$。考虑对一类区间和二类区间数量都减去 k，这样我们实际少了 k 个区间，对于丢失的区间我们可以看成额外可以从 i 开始的区间。

于是你就先不管那 k 个区间，正常贪心，最后再加回去即可。

例题

P6646

有 n 个商品，每个商品都有种类和价格。对于一个种类 i，必须买 x 个商品（$x \in [l_i, r_i]$）。你需要求出前 k 便宜的方案所需价钱。如果没有输出 -1。注意如果有相同钱数，但是具体方案不相同的，算作两种方案。

例题

P6646

有 n 个商品，每个商品都有种类和价格。对于一个种类 i，必须买 x 个商品（$x \in [l_i, r_i]$）。你需要求出前 k 便宜的方案所需价钱。如果没有输出 -1。注意如果有相同钱数，但是具体方案不相同的，算作两种方案。

首先对于这种求前 k 优的东西我们有一个通用的思路，就是先找到最优解，然后考虑从最优解拓展到稍微劣一点的解，把它扔进堆中，以此类推。

例题

P6646

有 n 个商品，每个商品都有种类和价格。对于一个种类 i，必须买 x 个商品（$x \in [l_i, r_i]$）。你需要求出前 k 便宜的方案所需价钱。如果没有输出 -1。注意如果有相同钱数，但是具体方案不相同的，算作两种方案。

首先对于这种求前 k 优的东西我们有一个通用的思路，就是先找到最优解，然后考虑从最优解拓展到稍微劣一点的解，把它扔进堆中，以此类推。

于是我们把每种方案用一种状态描述出来，然后把每个状态看成一个点。这样我们就能建出一棵外向树。

例题

P6646

有 n 个商品，每个商品都有种类和价格。对于一个种类 i，必须买 x 个商品（$x \in [l_i, r_i]$）。你需要求出前 k 便宜的方案所需价钱。如果没有输出 -1。注意如果有相同钱数，但是具体方案不相同的，算作两种方案。

首先对于这种求前 k 优的东西我们有一个通用的思路，就是先找到最优解，然后考虑从最优解拓展到稍微劣一点的解，把它扔进堆中，以此类推。

于是我们把每种方案用一种状态描述出来，然后把每个状态看成一个点。这样我们就能建出一棵外向树。

但是考虑从一个状态可能拓展出很多不同的状态，如果贸然更新可能会爆炸。为了保证时间复杂度，我们需要找到一种合理的拓展方法。

例题

也就是说原始的树是一个多叉树，但是我们尝试用某种构造方法让其“退化“成接近链的形态。

例题

也就是说原始的树是一个多叉树，但是我们尝试用某种构造方法让其“退化“成接近链的形态。

考虑到原始题面是种类数 $m > 0$，每种商品个数限制是 $[l_i, r_i]$。我们先尝试解决弱化后的问题，然后再尝试拼成这个问题。下文的序列全是对每个种类排序后的序列。

例题

也就是说原始的树是一个多叉树，但是我们尝试用某种构造方法让其“退化“成接近链的形态。

考虑到原始题面是种类数 $m > 0$，每种商品个数限制是 $[l_i, r_i]$。我们先尝试解决弱化后的问题，然后再尝试拼成这个问题。下文的序列全是对每个种类排序后的序列。

我们先来考虑 $m = 1, l = r$ 的情况。

例题

也就是说原始的树是一个多叉树，但是我们尝试用某种构造方法让其“退化“成接近链的形态。

考虑到原始题面是种类数 $m > 0$，每种商品个数限制是 $[l_i, r_i]$。我们先尝试解决弱化后的问题，然后再尝试拼成这个问题。下文的序列全是对每个种类排序后的序列。

我们先来考虑 $m = 1, l = r$ 的情况。

最开始我们肯定是选最小的 l 个商品，拓展肯定是最后一个商品的选择往后挪一个位置。那我们现在来考虑一个更普通的情况。

例题

也就是说原始的树是一个多叉树，但是我们尝试用某种构造方法让其“退化“成接近链的形态。

考虑到原始题面是种类数 $m > 0$，每种商品个数限制是 $[l_i, r_i]$。我们先尝试解决弱化后的问题，然后再尝试拼成这个问题。下文的序列全是对每个种类排序后的序列。

我们先来考虑 $m = 1, l = r$ 的情况。

最开始我们肯定是选最小的 l 个商品，拓展肯定是最后一个商品的选择往后挪一个位置。那我们现在来考虑一个更普通的情况。

考虑一个选择方案是选择了一段前缀，然后后面又有一些的位置被选择了。这时我们应该如何拓展才能做到不重不漏？

例题

为了不重不漏我们肯定要去关注某个东西，通过控制这个东西的形态达到不重不漏的效果。

例题

为了不重不漏我们肯定要去关注某个东西，通过控制这个东西的形态达到不重不漏的效果。考虑到我们要关注的东西肯定得有一点特征，这样才能引起我们的关注对吧？对于上述一种选择方法哪个地方比较特殊呢？

例题

为了不重不漏我们肯定要去关注某个东西，通过控制这个东西的形态达到不重不漏的效果。考虑到我们要关注的东西肯定得有一点特征，这样才能引起我们的关注对吧？对于上述一种选择方法哪个地方比较特殊呢？

显然一段前缀一定比较特殊而且还比较好控制，因为我们只需要关注前缀的结尾即可。

例题

为了不重不漏我们肯定要去关注某个东西，通过控制这个东西的形态达到不重不漏的效果。考虑到我们要关注的东西肯定得有一点特征，这样才能引起我们的关注对吧？对于上述一种选择方法哪个地方比较特殊呢？

显然一段前缀一定比较特殊而且还比较好控制，因为我们只需要关注前缀的结尾即可。

接下来我们就通过前缀做文章。设计状态 (x, y, z) 表示前面选择了连续的 x 个，当前考虑的第 $x+1$ 个选择了 y，(y, z) 是一个没有被选择的极长段，这里注意是开区间。初始状态为 $(l-1, l, +\infty)$。

例题

为了不重不漏我们肯定要去关注某个东西，通过控制这个东西的形态达到不重不漏的效果。考虑到我们要关注的东西肯定得有一点特征，这样才能引起我们的关注对吧？对于上述一种选择方法哪个地方比较特殊呢？

显然一段前缀一定比较特殊而且还比较好控制，因为我们只需要关注前缀的结尾即可。

接下来我们就通过前缀做文章。设计状态 (x, y, z) 表示前面选择了连续的 x 个，当前考虑的第 $x+1$ 个选择了 y，(y, z) 是一个没有被选择的极长段，这里注意是开区间。初始状态为 $(l-1, l, +\infty)$。

转移考虑一步一步地来，实际上可以分为两种。第一种有 $(x, y, z) \to (x, y+1, z)$，也就是考虑让当前考虑的被选择的位置后移一位。

例题

第二种考虑改变前面连续段的结尾，有 $(x, y, z) \to (x-1, x+1, y-1)$，也就是当前考虑的点前移，并把这个点往后移一位。

例题

第二种考虑改变前面连续段的结尾，有 $(x, y, z) \to (x-1, x+1, y-1)$，也就是当前考虑的点前移，并把这个点往后移一位。

现在考虑 $m = 1$ 的情况。其实和上面情况的本质是相同的，只不过现在可以加入更多被选择的点。于是我们修改一下边界的转移，有 $(x, x+1, +\infty) \to (x+1, x+2, +\infty)$。

例题

第二种考虑改变前面连续段的结尾，有 $(x, y, z) \to (x-1, x+1, y-1)$，也就是当前考虑的点前移，并把这个点往后移一位。

现在考虑 $m = 1$ 的情况。其实和上面情况的本质是相同的，只不过现在可以加入更多被选择的点。于是我们修改一下边界的转移，有 $(x, x+1, +\infty) \to (x+1, x+2, +\infty)$。

接下来我们考虑 $m > 1, l = r = 1$ 的情况，这个部分我们重点关注如何处理多颜色的情况，因此我们需要让选择范围特殊一点。

例题

第二种考虑改变前面连续段的结尾，有 $(x, y, z) \to (x-1, x+1, y-1)$，也就是当前考虑的点前移，并把这个点往后移一位。

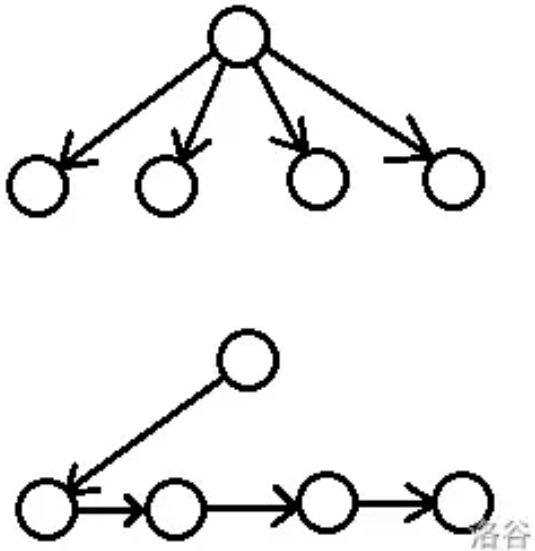
现在考虑 $m = 1$ 的情况。其实和上面情况的本质是相同的，只不过现在可以加入更多被选择的点。于是我们修改一下边界的转移，有 $(x, x+1, +\infty) \to (x+1, x+2, +\infty)$。

接下来我们考虑 $m > 1, l = r = 1$ 的情况，这个部分我们重点关注如何处理多颜色的情况，因此我们需要让选择范围特殊一点。

考虑最开始所有种类都选的第一个点，然后拓展就该是选一个点往后移动一个位置。但是这样每次要拓展的情况显然很多，也就是之前说的状态的外向树度数太多，需要换一种拓展方式。

例题

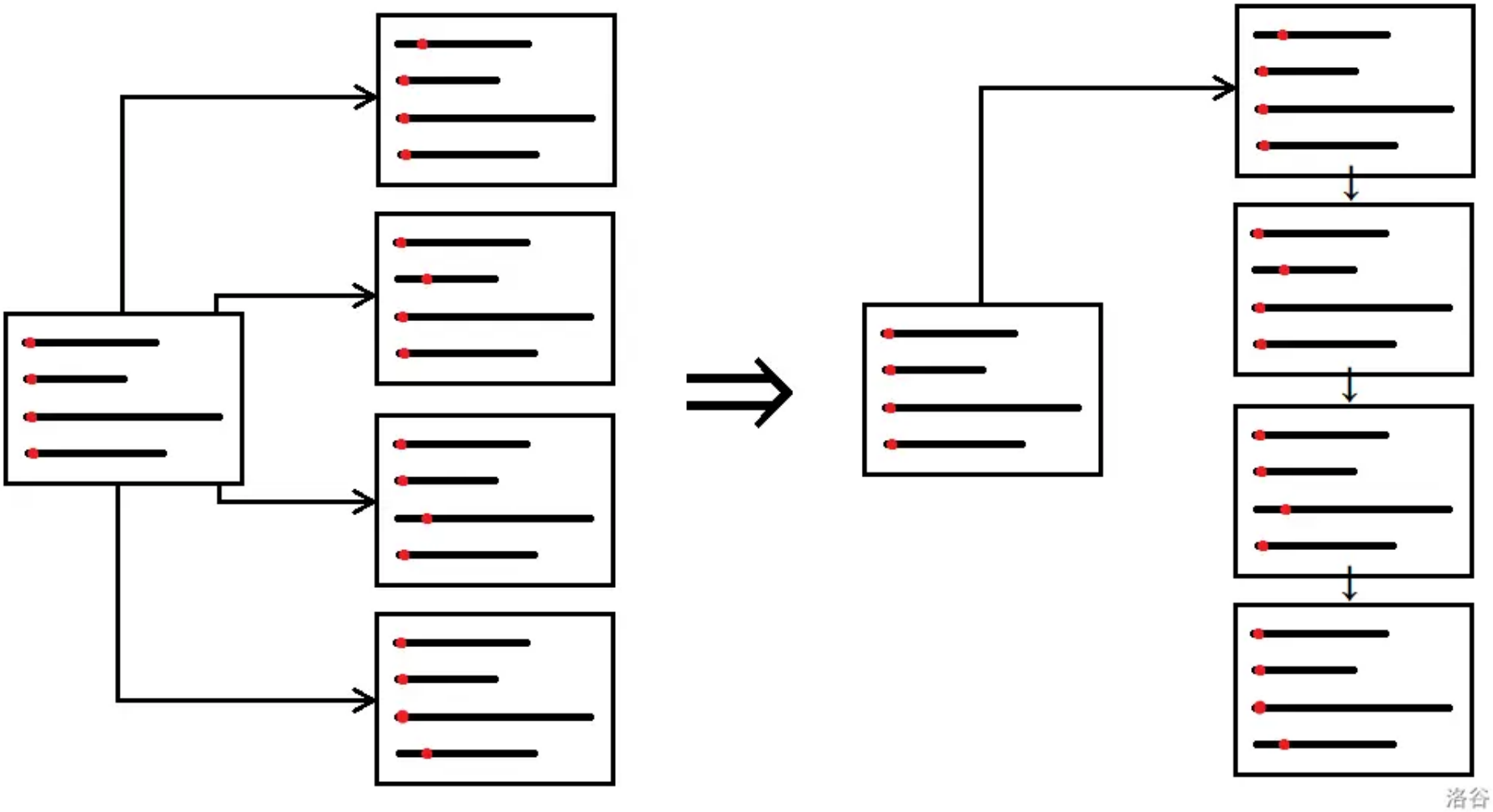
如下图，我们的目标就是把多个同时转移的状态变成一步步拓展的形式。



例题

具体的，我们可以考虑将哪种颜色的点后移造成的影响最小，将其操作后我们一定会得到次优的解。于是我们考虑对 $a_2 - a_1$ 排序，然后分别考虑往后移动一步的情况。

例题



例题

考虑一个新的状态，设 (p, k) 表示当前考虑到第 p 种颜色的第 k 个位置。首先我们可以有 $(p, k) \to (p, k+1)$，也就是往后移当前的选择位置。

例题

考虑一个新的状态，设 (p,k) 表示当前考虑到第 p 种颜色的第 k 个位置。首先我们可以有 $(p,k)\to(p,k+1)$，也就是往后移当前的选择位置。

然后考虑前面的图中所表示的转移，$(p,k)\to(p+1,1)$。就是说我们接着去考虑后面的颜色，但是呢我们需要把当前的状态恢复再往后。还有一种状态就是当前的颜色不恢复。

例题

考虑一个新的状态，设 (p,k) 表示当前考虑到第 p 种颜色的第 k 个位置。首先我们可以有 $(p,k)\to(p,k+1)$，也就是往后移当前的选择位置。

然后考虑前面的图中所表示的转移，$(p,k)\to(p+1,1)$。就是说我们接着去考虑后面的颜色，但是呢我们需要把当前的状态恢复再往后。还有一种状态就是当前的颜色不恢复。

最后我们再来简单说一下这道题怎么搞？考虑多种颜色就是上面的做法，然后考虑到上面的第一种转移需要考虑一个颜色的后继状态于是我们对于每个颜色又去用 $m=1$ 的情况拓展。也就是把 $m=1$ 和 $m>1$ 的情况拼起来即可。

例题

考虑一个新的状态，设 (p,k) 表示当前考虑到第 p 种颜色的第 k 个位置。首先我们可以有 $(p,k)\to(p,k+1)$，也就是往后移当前的选择位置。

然后考虑前面的图中所表示的转移，$(p,k)\to(p+1,1)$。就是说我们接着去考虑后面的颜色，但是呢我们需要把当前的状态恢复再往后。还有一种状态就是当前的颜色不恢复。

最后我们再来简单说一下这道题怎么搞？考虑多种颜色就是上面的做法，然后考虑到上面的第一种转移需要考虑一个颜色的后继状态于是我们对于每个颜色又去用 $m=1$ 的情况拓展。也就是把 $m=1$ 和 $m>1$ 的情况拼起来即可。

代码似乎有些困难，不会写可以找我 qwq。

总结

总的来说我认为贪心首先考察选手的意识。就是说选手能不能根据题目去判断一道题是否是贪心，这就要熟悉贪心的一些基本知识、会证明贪心。然后就是一些比较经典的 tricks 比如邻向交换、反悔贪心之类的。

总结

总的来说我认为贪心首先考察选手的意识。就是说选手能不能根据题目去判断一道题是否是贪心，这就要熟悉贪心的一些基本知识、会证明贪心。然后就是一些比较经典的 tricks 比如邻向交换、反悔贪心之类的。

但我觉得其实最主要的还是一个不断弱化问题的思想，只有我们明确了做这道题需要干什么，我们的思考才有方向、有目标，我们的思维才能更集中、才更有可能想到点子上。

总结

总的来说我认为贪心首先考察选手的意识。就是说选手能不能根据题目去判断一道题是否是贪心，这就要熟悉贪心的一些基本知识、会证明贪心。然后就是一些比较经典的 tricks 比如邻向交换、反悔贪心之类的。

但我觉得其实最主要的还是一个不断弱化问题的思想，只有我们明确了做这道题需要干什么，我们的思考才有方向、有目标，我们的思维才能更集中、才更有可能想到点子上。

最后留一个贪心的题单，里面有困难题，如果你已经把今天的题全部切完了可以去做 awa。

https://www.luogu.com.cn/training/473519

Contents

组合游戏

对于在信息学竞赛中的博弈论，我们研究的是组合博弈问题。而组合博弈又分为两种类型：公平组合游戏和非公平组合游戏。这里介绍两种类型游戏的概念。

组合游戏

对于在信息学竞赛中的博弈论，我们研究的是组合博弈问题。而组合博弈又分为两种类型：公平组合游戏和非公平组合游戏。这里介绍两种类型游戏的概念。

在公平组合游戏中，两名玩家交替行动，并且操作和当前局面状态不区分操作者，最后就是一个状态只能到达一次。游戏结束当且仅当其中一方无法操作。

组合游戏

对于在信息学竞赛中的博弈论，我们研究的是组合博弈问题。而组合博弈又分为两种类型：公平组合游戏和非公平组合游戏。这里介绍两种类型游戏的概念。

在公平组合游戏中，两名玩家交替行动，并且操作和当前局面状态不区分操作者，最后就是一个状态只能到达一次。游戏结束当且仅当其中一方无法操作。

而非公平游戏中操作和当前局面状态跟操作者会有关系。比如在 DAG 上移动，A 想让路径尽量长，B 想要路径尽量短。

先后手与必胜必败

接下来我们再定义一些基础的东西。

先后手与必胜必败

接下来我们再定义一些基础的东西。

一局游戏中先行动的为先手，反之为后手。定义一个局面如果无论如何操作都会输那么这个局面就是必败局面，如果当前局面存在一种操作使得操作后的局面是必败局面那么当前局面就是必胜局面。

先后手与必胜必败

接下来我们再定义一些基础的东西。

一局游戏中先行动的为先手，反之为后手。定义一个局面如果无论如何操作都会输那么这个局面就是必败局面，如果当前局面存在一种操作使得操作后的局面是必败局面那么当前局面就是必胜局面。

然后就有先手必胜和后手必胜的说法，其定义就是把两个定义拼起来，如果初始局面是必胜局面，因为当前轮到先手操作，所以是先手必胜；后手必胜同理。

一点小结论

根据上面的定义我们有一些小小的结论。

一点小结论

根据上面的定义我们有一些小小的结论。

1. 没有后继状态的状态是必败状态。

一点小结论

根据上面的定义我们有一些小小的结论。

1. 没有后继状态的状态是必败状态。

2. 从任何必胜点操作，至少存在一种方案可以进入必败点。

一点小结论

根据上面的定义我们有一些小小的结论。

1. 没有后继状态的状态是必败状态。

2. 从任何必胜点操作，至少存在一种方案可以进入必败点。

3. 一个状态是必败状态当且仅当它的所有后继状态均为必胜状态。

一点小结论

根据上面的定义我们有一些小小的结论。

1. 没有后继状态的状态是必败状态。

2. 从任何必胜点操作，至少存在一种方案可以进入必败点。

3. 一个状态是必败状态当且仅当它的所有后继状态均为必胜状态。

由此你能发现一个局面的状态由后继局面的状态决定，这其实就是一个倒推。

一个转换

为了方便，我们考虑把一个局面抽象成一个点，把一个局面与后继局面连一条有向边，于是我们就能得到一张 DAG。之后我们通常就会在 DAG 上面做题。然后你可以把每个点分为必胜点和必败点。并且我们可以翻译一下上面的小结论。

一个转换

为了方便，我们考虑把一个局面抽象成一个点，把一个局面与后继局面连一条有向边，于是我们就能得到一张 DAG。之后我们通常就会在 DAG 上面做题。然后你可以把每个点分为必胜点和必败点。并且我们可以翻译一下上面的小结论。

1. 所有出度为零的点都是必败点。

一个转换

为了方便，我们考虑把一个局面抽象成一个点，把一个局面与后继局面连一条有向边，于是我们就能得到一张 DAG。之后我们通常就会在 DAG 上面做题。然后你可以把每个点分为必胜点和必败点。并且我们可以翻译一下上面的小结论。

1. 所有出度为零的点都是必败点。

2. 从任何必胜点操作，至少存在一种方案可以进入必败点。

一个转换

为了方便，我们考虑把一个局面抽象成一个点，把一个局面与后继局面连一条有向边，于是我们就能得到一张 DAG。之后我们通常就会在 DAG 上面做题。然后你可以把每个点分为必胜点和必败点。并且我们可以翻译一下上面的小结论。

1. 所有出度为零的点都是必败点。

2. 从任何必胜点操作，至少存在一种方案可以进入必败点。

3. 无论如何操作，必败点只能进入必胜点。

结语

好了，到这里博弈的基本概念也就说得差不多了。后面会讲一些博弈的模型，但是并不会讲 SG 函数。我估计会等到 NOIP 后再来讲 qwq。

结语

好了，到这里博弈的基本概念也就说得差不多了。后面会讲一些博弈的模型，但是并不会讲 SG 函数。我估计会等到 NOIP 后再来讲 qwq。

然后考虑题目依然很简单，欢迎大家继续爆切。

Bash Game

让我们从最简单的巴什博弈开始吧（诶)！

Bash Game

让我们从最简单的巴什博弈开始吧（诶）！

Bash Game

有一堆石子个数为 n，两名玩家轮流在石子堆中拿石子，每次至少取 1 个，至多取 k 个，最后一个取走石子的玩家为胜者。

Bash Game

让我们从最简单的巴什博弈开始吧（诶）！

Bash Game

有一堆石子个数为 n，两名玩家轮流在石子堆中拿石子，每次至少取 1 个，至多取 k 个，最后一个取走石子的玩家为胜者。

结论是：若 $(k+1) \mid n$ 先手必败；否则先手必胜。

Bash Game

让我们从最简单的巴什博弈开始吧（诶）！

Bash Game

有一堆石子个数为 n，两名玩家轮流在石子堆中拿石子，每次至少取 1 个，至多取 k 个，最后一个取走石子的玩家为胜者。

结论是：若 $(k+1) \mid n$ 先手必败；否则先手必胜。

证明考虑最简单的情况。如果有 $n \leq k$ 那么先手必胜，如果 $n = k+1$ 那么无论先手取多少个都会寄，因为后手总能取完。如果 $(k+1) \mid n$，那么若先手取了 x 个，后手一定会取 $k+1-x$ 个使得轮到先手时 $(k+1) \mid n$ 依然成立，所以先手必败；反之先手可以将 $(k+1) \mid n$ 的局面留给后手然后先手就必胜了。

Nim 游戏

P2197

有 n 堆石子第 i 堆有 a_i 个，每次操作可以任选一堆，取出任意多个物品，但不能不取。取走最后一个石子的人胜利。

Nim 游戏

P2197

有 n 堆石子第 i 堆有 a_i 个，每次操作可以任选一堆，取出任意多个物品，但不能不取。取走最后一个石子的人胜利。

结论是：若 $\oplus_{i=1}^{n} a_i = 0$ 后手必胜；否则先手必胜。

Nim 游戏

P2197

有 n 堆石子第 i 堆有 a_i 个，每次操作可以任选一堆，取出任意多个物品，但不能不取。取走最后一个石子的人胜利。

结论是：若 $\oplus_{i=1}^{n} a_i = 0$ 后手必胜；否则先手必胜。

还是考虑最简单的状态，如果全为空那么后手必胜。如果一个局面满足 $\oplus_{i=1}^{n} a_i \neq 0$，假设等于 x，那么我们会找一个 a_i 使得 $a_i > a_i \oplus x$ 并把 a_i 改成 $a_i \oplus x$。设 x 最高位是第 k 位，那么一定有奇数堆石子数的第 k 位为 1，我们选择一堆去异或就能满足条件。

Nim 游戏

P2197

有 n 堆石子第 i 堆有 a_i 个，每次操作可以任选一堆，取出任意多个物品，但不能不取。取走最后一个石子的人胜利。

结论是：若 $\oplus_{i=1}^{n} a_i = 0$ 后手必胜；否则先手必胜。

还是考虑最简单的状态，如果全为空那么后手必胜。如果一个局面满足 $\oplus_{i=1}^{n} a_i \neq 0$，假设等于 x，那么我们会找一个 a_i 使得 $a_i > a_i \oplus x$ 并把 a_i 改成 $a_i \oplus x$。设 x 最高位是第 k 位，那么一定有奇数堆石子数的第 k 位为 1，我们选择一堆去异或就能满足条件。

如果异或和为零，那么无论怎么操作，局面都会变成不等于零的情况，于是得证。

Nim 游戏

接下来看一道简单题。

Nim 游戏

接下来看一道简单题。

P4301

相对于 Nim 游戏有些不同，就是每个人第一次可以拿走若干堆但不能全拿走，然后问第一轮先手最少拿走多少石子才能让先手获胜。

Nim 游戏

接下来看一道简单题。

P4301

相对于 Nim 游戏有些不同，就是每个人第一次可以拿走若干堆但不能全拿走，然后问第一轮先手最少拿走多少石子才能让先手获胜。

考虑先手必胜必须要第一轮后让后手无论拿掉哪几堆都弄不出异或和为 0 的情况。于是我们贪心然后线性基做完了。

NimK

NimK

还是 Nim 游戏，但是每次能够从不超过 k 堆中取石子，获胜条件与之前相同。

NimK

NimK

还是 Nim 游戏，但是每次能够从不超过 k 堆中取石子，获胜条件与之前相同。

结论：若将所有石子堆用二进制表示，考虑每一位上 1 的个数，若每一位 1 的个数都是 $k+1$ 的倍数那么先手必败；反之先手必胜。

NimK

NimK

还是 Nim 游戏，但是每次能够从不超过 k 堆中取石子，获胜条件与之前相同。

结论：若将所有石子堆用二进制表示，考虑每一位上 1 的个数，若每一位 1 的个数都是 $k+1$ 的倍数那么先手必败；反之先手必胜。

这个东西的证明可以类比 Nim 游戏。如果全为空那么后手必胜；如果所有二进制位上 1 的个数都是 $k+1$ 的倍数那么我们无论怎么操作都不能继续满足条件。

NimK

NimK

还是 Nim 游戏，但是每次能够从不超过 k 堆中取石子，获胜条件与之前相同。

结论：若将所有石子堆用二进制表示，考虑每一位上 1 的个数，若每一位 1 的个数都是 $k+1$ 的倍数那么先手必败；反之先手必胜。

这个东西的证明可以类比 Nim 游戏。如果全为空那么后手必胜；如果所有二进制位上 1 的个数都是 $k+1$ 的倍数那么我们无论怎么操作都不能继续满足条件。

否则，如果存在某些二进制位上 1 的个数不为 $k+1$ 的倍数，假设满足条件的最高位有 m 个 1，我们任取 $t \equiv m \pmod{k+1}$ 个 1 变成 0，考虑下一个满足条件的数位。

NimK

假设现在又有 m' 个 1，设原来改变的 t 堆中这一位有 a 个 1 和 b 个 0。

NimK

假设现在又有 m' 个 1，设原来改变的 t 堆中这一位有 a 个 1 和 b 个 0。

若 $a \geq m' \bmod (k+1)$ 那么我们直接把 $m' \bmod (k+1)$ 个 1 变成 0 即可。

NimK

假设现在又有 m' 个 1，设原来改变的 t 堆中这一位有 a 个 1 和 b 个 0。

若 $a \geq m' \bmod (k+1)$ 那么我们直接把 $m' \bmod (k+1)$ 个 1 变成 0 即可。

若 $b \geq k+1-m' \bmod (k+1)$ 同理。

NimK

假设现在又有 m' 个 1，设原来改变的 t 堆中这一位有 a 个 1 和 b 个 0。

若 $a \geq m' \bmod (k+1)$ 那么我们直接把 $m' \bmod (k+1)$ 个 1 变成 0 即可。

若 $b \geq k+1-m' \bmod (k+1)$ 同理。

否则我们就改变原来 t 堆之外的 $m' \bmod (k+1)-a$ 堆，将其从 1 变成 0，然后再将这 t 堆也从 1 变成 0。考虑一共操作的堆数为：$a+b+m' \bmod (k+1)-a=b+m' \bmod (k+1)<k+1$。于是操作者一定能从必败到必胜态，得证。

NimK

P2490

有一个 $1 \times n$ 的棋盘，其上有 k 个棋子，从左到右棋子颜色白黑交替出现。小 A 能移动白棋，小 B 能移动黑棋，白棋不能往左，黑棋不能往右，且每次操作能移动 1 到 d 颗棋子。每当移动某一个棋子时，这个棋子不能跨越两边的棋子，当然也不可以出界。当谁不可以操作时，谁就失败了。小 A 先手，求有多少种初始局面下小 A 有必胜方案。

NimK

P2490

有一个 $1 \times n$ 的棋盘，其上有 k 个棋子，从左到右棋子颜色白黑交替出现。小 A 能移动白棋，小 B 能移动黑棋，白棋不能往左，黑棋不能往右，且每次操作能移动 1 到 d 颗棋子。每当移动某一个棋子时，这个棋子不能跨越两边的棋子，当然也不可以出界。当谁不可以操作时，谁就失败了。小 A 先手，求有多少种初始局面下小 A 有必胜方案。

这个其实就是 NimK 模型，我们把每对白黑之间的间隙看成石子堆即可。考虑必胜条件是石子堆二进制表示下存在一位 1 的个数不是 $d+1$ 的倍数。这个比较难处理于是正难则反，考虑用总方案减去必败方案。现在考虑必败方案。

NimK

P2490

有一个 $1 \times n$ 的棋盘，其上有 k 个棋子，从左到右棋子颜色白黑交替出现。小 A 能移动白棋，小 B 能移动黑棋，白棋不能往左，黑棋不能往右，且每次操作能移动 1 到 d 颗棋子。每当移动某一个棋子时，这个棋子不能跨越两边的棋子，当然也不可以出界。当谁不可以操作时，谁就失败了。小 A 先手，求有多少种初始局面下小 A 有必胜方案。

这个其实就是 NimK 模型，我们把每对白黑之间的间隙看成石子堆即可。考虑必胜条件是石子堆二进制表示下存在一位 1 的个数不是 $d+1$ 的倍数。这个比较难处理于是正难则反，考虑用总方案减去必败方案。现在考虑必败方案。

考虑必败要求石子堆二进制表示下每一位都是 $d+1$ 的倍数，对于这个东西我们考虑 dp。

NimK

我们设 $f_{i,j}$ 表示从高到低考虑到二进制下的第 i 位，$\sum_{s=1}^{k} a'_s = j$ 的合法方案数。

NimK

我们设 $f_{i,j}$ 表示从高到低考虑到二进制下的第 i 位，$\sum_{s=1}^{k} a'_s = j$ 的合法方案数。

转移就是普通背包转移，只不过要乘上一个组合数即可。还有一个比较特殊的地方就是因为我们要满足必败的条件所以我们只考虑在 $d+1$ 的倍数处的转移即可。

NimK

我们设 $f_{i,j}$ 表示从高到低考虑到二进制下的第 i 位，$\sum_{s=1}^{k} a'_s = j$ 的合法方案数。

转移就是普通背包转移，只不过要乘上一个组合数即可。还有一个比较特殊的地方就是因为我们要满足必败的条件所以我们只考虑在 $d+1$ 的倍数处的转移即可。

最后的答案即为 $\binom{n}{k} - \sum f_{0,i} \times \binom{n-i-\frac{k}{2}}{\frac{k}{2}}$。简单解释一下最后这个地方的组合数。考虑我们一共有 $\frac{k}{2}$ 个石子堆，对于每一堆我们在其右端点处计数即可。

阶梯 Nim 游戏

P3480

n 堆石子，保证了初始石子数单调不降。轮流取石子，每次从任意堆拿走任意个要求取完后每一堆剩余石子个数单调不降，不能操作的人输。

阶梯 Nim 游戏

P3480

n 堆石子，保证了初始石子数单调不降。轮流取石子，每次从任意堆拿走任意个要求取完后每一堆剩余石子个数单调不降，不能操作的人输。

首先就是我们可以差分一下，然后每次操作就相当于是一个位置 i 减去 x，然后位置 $i-1$ 加上 x。

阶梯 Nim 游戏

P3480

n 堆石子，保证了初始石子数单调不降。轮流取石子，每次从任意堆拿走任意个要求取完后每一堆剩余石子个数单调不降，不能操作的人输。

首先就是我们可以差分一下，然后每次操作就相当于是一个位置 i 减去 x，然后位置 $i-1$ 加上 x。

结论是：阶梯 Nim 游戏等价于编号为奇数堆的石子堆做普通 Nim 游戏。

阶梯 Nim 游戏

P3480

n 堆石子，保证了初始石子数单调不降。轮流取石子，每次从任意堆拿走任意个要求取完后每一堆剩余石子个数单调不降，不能操作的人输。

首先就是我们可以差分一下，然后每次操作就相当于是一个位置 i 减去 x，然后位置 $i-1$ 加上 x。

结论是：阶梯 Nim 游戏等价于编号为奇数堆的石子堆做普通 Nim 游戏。

考虑证明。拿走某一堆石子的一部分，相当于将某个位置的石子移动到它左边的位置上。如果大家都只动奇位置的石子，那么这等价于两人在玩 Nim 游戏。

阶梯 Nim 游戏

如果当前操作者必胜那么他就会正常在奇数编号上进行 Nim 游戏，否则他就会去操作偶数编号的石子堆。但是如果他将偶数堆的若干石子拿到了奇数堆，下一个人就会把这若干石子又拿到偶数堆，并且此时先后手还不会改变。所以得证。

阶梯 Nim 游戏

P5363

小 A 小 B 进行如下的一场游戏。二人轮流操作，且小 A 先手。当轮到一个玩家的时候，他可以选择一枚金币，并将其向左移动任意多格，且至少移动一格。金币不能被移出棋盘，不能重叠，也不能越过其它金币。

棋盘大小为 $1 \times n$，最初摆有 m 个金币。求小 A 必胜的初始状态数。

阶梯 Nim 游戏

P5363

小 A 小 B 进行如下的一场游戏。二人轮流操作，且小 A 先手。当轮到一个玩家的时候，他可以选择一枚金币，并将其向左移动任意多格，且至少移动一格。金币不能被移出棋盘，不能重叠，也不能越过其它金币。

棋盘大小为 $1 \times n$，最初摆有 m 个金币。求小 A 必胜的初始状态数。

首先不难发现这是一个阶梯博弈的模型，把一枚金币向左移动，相当于把这枚金币和它左边的金币的距离减小，同时它和右边的金币增加等量的距离。

阶梯 Nim 游戏

P5363

小 A 小 B 进行如下的一场游戏。二人轮流操作，且小 A 先手。当轮到一个玩家的时候，他可以选择一枚金币，并将其向左移动任意多格，且至少移动一格。金币不能被移出棋盘，不能重叠，也不能越过其它金币。

棋盘大小为 $1 \times n$，最初摆有 m 个金币。求小 A 必胜的初始状态数。

首先不难发现这是一个阶梯博弈的模型，把一枚金币向左移动，相当于把这枚金币和它左边的金币的距离减小，同时它和右边的金币增加等量的距离。

于是我们把问题转化成 $n-m$ 个石子放到 $m+1$ 个台阶上，求有多少种初始局面使得奇数台阶的石子异或和非 0。

阶梯 Nim 游戏

解释一下转化。$n-m$ 是 n 个位置放入 m 个金币还剩 $n-m$ 个位置代表石子。$m+1$ 表示 m 个金币将棋盘分成 $m+1$ 个阶梯。

阶梯 Nim 游戏

解释一下转化。$n-m$ 是 n 个位置放入 m 个金币还剩 $n-m$ 个位置代表石子。$m+1$ 表示 m 个金币将棋盘分成 $m+1$ 个阶梯。

计数和前面的那个 dp 题很像。先容斥，对异或和为 0 的状态计数，考虑按位 dp。设 $f_{i,j}$ 表示从高到底考虑到第 i 位，用了 j 个石子的合法方案数。转移与前面同理。

阶梯 Nim 游戏

解释一下转化。$n-m$ 是 n 个位置放入 m 个金币还剩 $n-m$ 个位置代表石子。$m+1$ 表示 m 个金币将棋盘分成 $m+1$ 个阶梯。

计数和前面的那个 dp 题很像。先容斥，对异或和为 0 的状态计数，考虑按位 dp。设 $f_{i,j}$ 表示从高到底考虑到第 i 位，用了 j 个石子的合法方案数。转移与前面同理。

枚举做完所有二进制位用掉的石子个数，剩下的随便丢到偶数编号台阶上，插板即可。

Anti Nim

P4279

和 Nim 游戏一样，但是获胜条件相反。

Anti Nim

P4279

和 Nim 游戏一样，但是获胜条件相反。

给出结论：先手必胜当且仅当每堆的物品数都为 1 并且有偶数堆，或者有些堆的物品数大于 1 且异或和不为 0。

Anti Nim

P4279

和 Nim 游戏一样，但是获胜条件相反。

给出结论：先手必胜当且仅当每堆的物品数都为 1 并且有偶数堆，或者有些堆的物品数大于 1 且异或和不为 0。

第一种情况显然，这里只证明第二个结论。证明考虑分为异或和是否为 0 分讨。先考虑异或和不为 0 的情况。若至少还有两堆物品数大于 1 那先手一定可以将局势变为至少有两堆物品数大于 1 且异或和为 0；若只有一堆物品数大于 1，则先手一定可以将局势变为有奇数个 1。

Anti Nim

P4279

和 Nim 游戏一样，但是获胜条件相反。

给出结论：先手必胜当且仅当每堆的物品数都为 1 并且有偶数堆，或者有些堆的物品数大于 1 且异或和不为 0。

第一种情况显然，这里只证明第二个结论。证明考虑分为异或和是否为 0 分讨。先考虑异或和不为 0 的情况。若至少还有两堆物品数大于 1 那先手一定可以将局势变为至少有两堆物品数大于 1 且异或和为 0；若只有一堆物品数大于 1，则先手一定可以将局势变为有奇数个 1。

至少有两堆物品数大于 1 那么先手决策完之后，必定至少有一堆物品数大于 1 且异或和不为 0，所以先手必败。于是得证。

Anti 游戏

说到了 Anti Nim 那就顺便说说 Anti 游戏。考虑到这次不讲 SG 函数于是这部分就大力口，胡即可。

Anti 游戏

说到了 Anti Nim 那就顺便说说 Anti 游戏。考虑到这次不讲 SG 函数于是这部分就大力口，胡即可。

考虑到 Anti 游戏通常是需要用 SG 函数处理，比如可以自己画一个 DAG 实践一下看看。但是在某些特殊的情况下我们也可以对题目进行一些神秘的小转化去掉 Anti，后面会有题的，尽情期 die。

威佐夫博弈

P2252

有两堆石子，数量任意，可以不同。游戏开始由两个人轮流取石子。游戏规定，每次有两种不同的取法：一是可以在任意的一堆中取走任意多的石子；二是可以在两堆中同时取走相同数量的石子。最后把石子全部取完者为胜者。现在给出初始的两堆石子的数目，你先取，假设双方都采取最好的策略，问最后你是胜者还是败者。

威佐夫博弈

P2252

有两堆石子，数量任意，可以不同。游戏开始由两个人轮流取石子。游戏规定，每次有两种不同的取法：一是可以在任意的一堆中取走任意多的石子；二是可以在两堆中同时取走相同数量的石子。最后把石子全部取完者为胜者。现在给出初始的两堆石子的数目，你先取，假设双方都采取最好的策略，问最后你是胜者还是败者。

这个博弈论模型的结论有点抽象，需要一些铺垫，我们先来了解一下 Beatty 序列。

威佐夫博弈

一个正无理数 r 生成的 Beatty 序列是指这样的一个序列：$\mathfrak{B}_r = (\lfloor ri \rfloor)_{i>0}$。

威佐夫博弈

一个正无理数 r 生成的 Beatty 序列是指这样的一个序列：$\mathfrak{B}_r = (\lfloor ri \rfloor)_{i>0}$。

然后有一个 Rayleigh 定理（Beatty 定理），假设正无理数 $r > 1$，有 s 满足 $\frac{1}{r} + \frac{1}{s} = 1$，则 $\mathfrak{B}_r$ 与 $\mathfrak{B}_s$ 是全体正整数的一个分割。

威佐夫博弈

一个正无理数 r 生成的 Beatty 序列是指这样的一个序列：$\mathfrak{B}_r = (\lfloor ri \rfloor)_{i>0}$。

然后有一个 Rayleigh 定理（Beatty 定理），假设正无理数 $r > 1$，有 s 满足 $\frac{1}{r} + \frac{1}{s} = 1$，则 $\mathfrak{B}_r$ 与 $\mathfrak{B}_s$ 是全体正整数的一个分割。

接下来我们要证明这个定理。

威佐夫博弈

构造法大牛逼！

威佐夫博弈

构造法大牛逼!

我们考虑所有 $\frac{j}{r}(j>0)$ 和 $\frac{k}{s}(k>0)$ 排成的非递减序列。

威佐夫博弈

构造法大牛逼！

我们考虑所有 $\frac{j}{r}(j>0)$ 和 $\frac{k}{s}(k>0)$ 排成的非递减序列。

首先该序列中的数两两不同，否则有 $\frac{j}{r}=\frac{k}{s}$，得到 $\frac{r}{s}=r-\frac{r}{r}=r-1=\frac{j}{k}$，左边是无理数，右边是有理数，矛盾。

威佐夫博弈

构造法大牛逼！

我们考虑所有 $\frac{j}{r}(j>0)$ 和 $\frac{k}{s}(k>0)$ 排成的非递减序列。

首先该序列中的数两两不同，否则有 $\frac{j}{r}=\frac{k}{s}$，得到 $\frac{r}{s}=r-\frac{r}{r}=r-1=\frac{j}{k}$，左边是无理数，右边是有理数，矛盾。

知道每个数互不相同后我们可以考察一个数在序列中的位置。对于 $\frac{j}{r}$，首先有 $\frac{i}{r}(i\in[1,j))$ 在其前面，然后考虑 $\frac{i}{s}$，这里面又有 $\left\lfloor\frac{js}{r}\right\rfloor$ 个小于 $\frac{j}{r}$，所以我们知道了 $\frac{j}{r}$ 的位置：$j+\left\lfloor\frac{js}{r}\right\rfloor=j+\lfloor j(s-1)\rfloor=\lfloor js\rfloor$。

威佐夫博弈

对于 $\frac{k}{s}$ 同理。然后你会发现我们通过 $\frac{j}{r}(j>0)$ 和 $\frac{k}{s}(k>0)$ 构造出来的序列其数值与下标的映射，其定义域就是两个 Beatty 序列，值域就是 $\mathbb{Z}+$，于是我们就巧妙地证明了这个定理。

威佐夫博弈

对于 $\frac{k}{s}$ 同理。然后你会发现我们通过 $\frac{j}{r}(j>0)$ 和 $\frac{k}{s}(k>0)$ 构造出来的序列其数值与下标的映射，其定义域就是两个 Beatty 序列，值域就是 $\mathbb{Z}+$，于是我们就巧妙地证明了这个定理。

现在定义「奇异局势」表示对于先手必败的局势。现在我们可以打表寻找前面若干项「奇异局势」，这里列出来：
$(0,0),(1,2),(3,5),(4,7),(6,10)\ldots$

威佐夫博弈

对于 $\frac{k}{s}$ 同理。然后你会发现我们通过 $\frac{j}{r}(j>0)$ 和 $\frac{k}{s}(k>0)$ 构造出来的序列其数值与下标的映射，其定义域就是两个 Beatty 序列，值域就是 $\mathbb{Z}+$，于是我们就巧妙地证明了这个定理。

现在定义「奇异局势」表示对于先手必败的局势。现在我们可以打表寻找前面若干项「奇异局势」，这里列出来：
$(0,0),(1,2),(3,5),(4,7),(6,10)\ldots$

令 S_i 表示前 i 项出现的所有数的集合，可以发现一个性质：对于「奇异局势」的第 i 项 (n,m) 满足 $i=n-m, n=\operatorname{mex}(S_{i-1})$。

威佐夫博弈

对于 $\frac{k}{s}$ 同理。然后你会发现我们通过 $\frac{j}{r}(j>0)$ 和 $\frac{k}{s}(k>0)$ 构造出来的序列其数值与下标的映射，其定义域就是两个 Beatty 序列，值域就是 $\mathbb{Z}+$，于是我们就巧妙地证明了这个定理。

现在定义「奇异局势」表示对于先手必败的局势。现在我们可以打表寻找前面若干项「奇异局势」，这里列出来：
$(0,0),(1,2),(3,5),(4,7),(6,10)\ldots$

令 S_i 表示前 i 项出现的所有数的集合，可以发现一个性质：对于「奇异局势」的第 i 项 (n,m) 满足 $i=n-m, n=\operatorname{mex}(S_{i-1})$。

并且奇异局势只能一步走到非奇异局势，非奇异局势可以一步走到奇异局势，证明显然。

威佐夫博弈

于是我们考虑将一个奇异局势的第一个元素用 $\lfloor cx \rfloor$ 表示，那么第二个元素则可以表示为 $\lfloor (c+1)x \rfloor$。其中 c 为正无理数，x 为正整数。

威佐夫博弈

于是我们考虑将一个奇异局势的第一个元素用 $\lfloor cx \rfloor$ 表示，那么第二个元素则可以表示为 $\lfloor (c+1)x \rfloor$。其中 c 为正无理数，x 为正整数。

那么我们是不是就可以用前面的 Beatty 定理，列出方程 $\frac{1}{c} + \frac{1}{c+1} = 1$，结合 c 为正无理数，得到 $c = \frac{1+\sqrt{5}}{2}$。于是我们就能刻画出奇异局势的一般形式了！

威佐夫博弈

于是我们考虑将一个奇异局势的第一个元素用 $\lfloor cx \rfloor$ 表示，那么第二个元素则可以表示为 $\lfloor (c+1)x \rfloor$。其中 c 为正无理数，x 为正整数。

那么我们是不是就可以用前面的 Beatty 定理，列出方程 $\frac{1}{c} + \frac{1}{c+1} = 1$，结合 c 为正无理数，得到 $c = \frac{1+\sqrt{5}}{2}$。于是我们就能刻画出奇异局势的一般形式了！

最后总结一下其结论：钦定任意一个局势 (x, y) 中 $x \leq y$，令 $k = y - x$，若满足 $y = \left\lfloor \frac{1+\sqrt{5}}{2} k \right\rfloor$ 则先手必败；否则先手必胜。

斐波那契博弈

P6487

有一堆石子，个数为 n，两个人轮流取石子，最开始先手可以取任意数量石子但不能取完，之后每个人每次取石子的数量必须满足最少为 1 最多为对手所取的石子数的 2 倍。

斐波那契博弈

在开始讲博弈之前我们先来了解一下齐肯多夫定理。

斐波那契博弈

在开始讲博弈之前我们先来了解一下齐肯多夫定理。

齐肯多夫定理说了个什么事呢？它告诉我们任何正整数都可以表示成若干个不连续的斐波那契数之和。听着很神秘但是其实很简单。

斐波那契博弈

在开始讲博弈之前我们先来了解一下齐肯多夫定理。

齐肯多夫定理说了个什么事呢？它告诉我们任何正整数都可以表示成若干个不连续的斐波那契数之和。听着很神秘但是其实很简单。

我们尝试证明。

斐波那契博弈

在开始讲博弈之前我们先来了解一下齐肯多夫定理。

齐肯多夫定理说了个什么事呢？它告诉我们任何正整数都可以表示成若干个不连续的斐波那契数之和。听着很神秘但是其实很简单。

我们尝试证明。首先如果一个正整数 n 已经是斐波那契数那么一定是对的。考虑 n 不为斐波那契数。

斐波那契博弈

在开始讲博弈之前我们先来了解一下齐肯多夫定理。

齐肯多夫定理说了个什么事呢？它告诉我们任何正整数都可以表示成若干个不连续的斐波那契数之和。听着很神秘但是其实很简单。

我们尝试证明。首先如果一个正整数 n 已经是斐波那契数那么一定是对的。考虑 n 不为斐波那契数。

我们先找到一个 i 满足 $\mathrm{fib}_i < n < \mathrm{fib}_{i+1}$，然后我们贪心地将 n 变成 $n - \mathrm{fib}_i$ 即可。现在我们需要证明我们相邻两次取的 i 不相邻。

斐波那契博弈

在开始讲博弈之前我们先来了解一下齐肯多夫定理。

齐肯多夫定理说了个什么事呢？它告诉我们任何正整数都可以表示成若干个不连续的斐波那契数之和。听着很神秘但是其实很简单。

我们尝试证明。首先如果一个正整数 n 已经是斐波那契数那么一定是对的。考虑 n 不为斐波那契数。

我们先找到一个 i 满足 $\mathrm{fib}_i < n < \mathrm{fib}_{i+1}$，然后我们贪心地将 n 变成 $n - \mathrm{fib}_i$ 即可。现在我们需要证明我们相邻两次取的 i 不相邻。

若相邻则有 $\mathrm{fib}_{i-1} + \mathrm{fib}_i = \mathrm{fib}_{i+1} > n$，矛盾，所以得证。

斐波那契博弈

现在可以给出斐波那契博弈的结论了：如果石子数 n 为斐波那契数则先手必败。

斐波那契博弈

现在可以给出斐波那契博弈的结论了：如果石子数 n 为斐波那契数则先手必败。

令 $n = \mathrm{fib}_i$，如果先手取了 $x \geq \mathrm{fib}_{i-2}$ 个石子，那么下一次后手一定能取完。我们现在考虑把 n 分成 fib_{i-1} 和 fib_{i-2} 两堆，因为先手第一次取的石子数没有超过两堆中的任意堆，于是就递归到了两个子问题中。考虑基本情况是 $i = 2$ 的时候先手必败，于是倒推回去其实就是数学归纳法。

斐波那契博弈

现在可以给出斐波那契博弈的结论了：如果石子数 n 为斐波那契数则先手必败。

令 $n = \mathrm{fib}_i$，如果先手取了 $x \geq \mathrm{fib}_{i-2}$ 个石子，那么下一次后手一定能取完。我们现在考虑把 n 分成 fib_{i-1} 和 fib_{i-2} 两堆，因为先手第一次取的石子数没有超过两堆中的任意堆，于是就递归到了两个子问题中。考虑基本情况是 $i = 2$ 的时候先手必败，于是倒推回去其实就是数学归纳法。

对于 n 不为斐波那契数的情况考虑用齐肯多夫表示法将石子看成若干堆数目为斐波那契数的石子堆做即可。因为每一堆做完后都不能去消除更大的堆，于是先手第一次必须拿掉最小的那堆才行。

斐波那契博弈

P6791

两个人取石子，规则和上题一样，但是最后取完的人输，还有一个要求就是先手第一次最多只能取 k 个，求对于 $n \in [1, m]$ 中有多少 n 满足先手必胜。

斐波那契博弈

P6791

两个人取石子，规则和上题一样，但是最后取完的人输，还有一个要求就是先手第一次最多只能取 k 个，求对于 $n \in [1, m]$ 中有多少 n 满足先手必胜。

发现其实这题是 Anti 游戏，考虑去掉。我们可以把 n 转化成 $n-1$，也就是考虑给对手留最后一个，然后就可以正常做了。

斐波那契博弈

P6791

两个人取石子，规则和上题一样，但是最后取完的人输，还有一个要求就是先手第一次最多只能取 k 个，求对于 $n \in [1, m]$ 中有多少 n 满足先手必胜。

发现其实这题是 Anti 游戏，考虑去掉。我们可以把 n 转化成 $n-1$，也就是考虑给对手留最后一个，然后就可以正常做了。

考虑获胜的条件是用齐肯多夫表示法表示出的第一项小于等于 k，考虑数位 dp 板子即可。

例题

P4363

有一个 $n \times m$ 的棋盘，每个格子都有 $a_{i,j}, b_{i,j}$。先手如果选择一个格子则获得 $a_{i,j}$，后手可类比。一个格子能被选择当且仅当其没被选且这个格子的左侧及上方的所有格子都被选。两个人都希望自己的得分减去对方的得分得到的结果最大。求这个值。

例题

P4363

有一个 $n \times m$ 的棋盘，每个格子都有 $a_{i,j}, b_{i,j}$。先手如果选择一个格子则获得 $a_{i,j}$，后手可类比。一个格子能被选择当且仅当其没被选且这个格子的左侧及上方的所有格子都被选。两个人都希望自己的得分减去对方的得分得到的结果最大。求这个值。

额。

例题

P4363

有一个 $n \times m$ 的棋盘，每个格子都有 $a_{i,j}, b_{i,j}$。先手如果选择一个格子则获得 $a_{i,j}$，后手可类比。一个格子能被选择当且仅当其没被选且这个格子的左侧及上方的所有格子都被选。两个人都希望自己的得分减去对方的得分得到的结果最大。求这个值。

额。其实这个题就纯套了个博弈的壳子，你考虑暴力的话我们可以去搜每种状态下的答案。具体就是记录状压选择的状态，存每一行铺到了哪个位置但是这个是 $\mathcal{O}(n^m)$ 的。

例题

P4363

有一个 $n \times m$ 的棋盘，每个格子都有 $a_{i,j}, b_{i,j}$。先手如果选择一个格子则获得 $a_{i,j}$，后手可类比。一个格子能被选择当且仅当其没被选且这个格子的左侧及上方的所有格子都被选。两个人都希望自己的得分减去对方的得分得到的结果最大。求这个值。

额。其实这个题就纯套了个博弈的壳子，你考虑暴力的话我们可以去搜每种状态下的答案。具体就是记录状压选择的状态，存每一行铺到了哪个位置但是这个是 $\mathcal{O}(n^m)$ 的。

但是你仔细一想，发现后面的选择跟前面的状态不全有关。事实上他只会和选择的最右和最下的格子有关，然后这不就是一个裸出来的轮廓线 dp 吗？直接维护轮廓线状态就是 $\mathcal{O}\binom{n+m}{m}$，然后搞一个 $\mathcal{O}(n)$ 左右的转移即可。

例题

魔女之夜

为了锻炼大家的语文能力所以这道题我建议去读题。

例题

魔女之夜

为了锻炼大家的语文能力所以这道题我建议去读题。

这个题就是把若干 Nim 游戏放到一起，但是最后是一个 Anti Nim。我们先去考虑获胜条件然后再加入计数。

例题

魔女之夜

为了锻炼大家的语文能力所以这道题我建议去读题。

这个题就是把若干 Nim 游戏放到一起，但是最后是一个 Anti Nim。我们先去考虑获胜条件然后再加入计数。

容易发现如果所有堆石子数不全为 1 且异或和不为 0 那么先手一定获胜；反之则后手一定获胜。如果一个人在这个局面能够获胜那么他一定能控制下一个局面的先后手，所以两人会在第一个不全为 1 的游戏中分出胜负。

例题

魔女之夜

为了锻炼大家的语文能力所以这道题我建议去读题。

这个题就是把若干 Nim 游戏放到一起，但是最后是一个 Anti Nim。我们先去考虑获胜条件然后再加入计数。

容易发现如果所有堆石子数不全为 1 且异或和不为 0 那么先手一定获胜；反之则后手一定获胜。如果一个人在这个局面能够获胜那么他一定能控制下一个局面的先后手，所以两人会在第一个不全为 1 的游戏中分出胜负。

然后我们现在先考虑全为 1 的影响。如果有奇数个 1 那么会改变一次先后手，否则不变。

例题

于是我们得到了先手的必胜条件：前缀中交换先后手奇数次，且下一个非全为 1 的游戏异或和为 0；或者前缀中交换先后手偶数次，且下一个非全为 1 的游戏异或和不为 0。

例题

于是我们得到了先手的必胜条件：前缀中交换先后手奇数次，且下一个非全为 1 的游戏异或和为 0；或者前缀中交换先后手偶数次，且下一个非全为 1 的游戏异或和不为 0。

对此计数，考虑枚举前缀中全为 1 且有奇数个 1 的游戏个数，然后枚举下一个游戏。对于后面的游戏我们不关心于是随便排即可。最后我们再把对答案没影响的那种全为 1 且有偶数个 1 的游戏用插板法狠狠插一下即可。

例题

P3210

有 n 个石子堆，每次只能取空堆旁边的石子。注意 0 号位置和 $n+1$ 号位置不存在，也就是说如果 2 号堆不为空堆那么不能直接取 1 号堆，右侧同理。有两个人轮流取石子，谁最后取得的总石子数最多，谁就获得了这场游戏的胜利。

例题

P3210

有 n 个石子堆，每次只能取空堆旁边的石子。注意 0 号位置和 $n+1$ 号位置不存在，也就是说如果 2 号堆不为空堆那么不能直接取 1 号堆，右侧同理。有两个人轮流取石子，谁最后取得的总石子数最多，谁就获得了这场游戏的胜利。

由于得分之和是定值，且双方都想让自己分数最大，我们令 d 为先手得分与后手得分的差。类似于对抗搜索，那么先手要让其值尽可能大，而后手相反。

例题

P3210

有 n 个石子堆，每次只能取空堆旁边的石子。注意 0 号位置和 $n+1$ 号位置不存在，也就是说如果 2 号堆不为空堆那么不能直接取 1 号堆，右侧同理。有两个人轮流取石子，谁最后取得的总石子数最多，谁就获得了这场游戏的胜利。

由于得分之和是定值，且双方都想让自己分数最大，我们令 d 为先手得分与后手得分的差。类似于对抗搜索，那么先手要让其值尽可能大，而后手相反。

对于取石子，可以转化为两端分别有一个栈，可以从栈顶取石子，中间有若干个双端队列，可以从其两端取石子。

例题

如果取一个位置后，接下来的位置比刚才取的那个位置权值小，也就是从选择方向开始权值是递减的，每次决策肯定都是取当前局面权值最大的位置。如果不保证递减，就有可能取完一个位置后，使得一个权值更大的位置可以取，这时按最大值决策就有可能不是最优。

例题

如果取一个位置后，接下来的位置比刚才取的那个位置权值小，也就是从选择方向开始权值是递减的，每次决策肯定都是取当前局面权值最大的位置。如果不保证递减，就有可能取完一个位置后，使得一个权值更大的位置可以取，这时按最大值决策就有可能不是最优。

对于 a_{i-1}, a_i, a_{i+1}，若有 $a_i \geq a_{i-1}, a_i \geq a_{i+1}$，当第一次选 a_{i-1} 最优时一个人选了 a_{i-1} 另一个人一定会选 a_i，然后第一个人再选 a_{i+1}。考虑到这样的过程中这三个石子的选择一定是连续的，所以我们可以把他们当成一个石子来考虑，也就是把它们合并成一个权值为 $a_{i-1} + a_{i+1} - a_i$ 的石子。

例题

如果取一个位置后，接下来的位置比刚才取的那个位置权值小，也就是从选择方向开始权值是递减的，每次决策肯定都是取当前局面权值最大的位置。如果不保证递减，就有可能取完一个位置后，使得一个权值更大的位置可以取，这时按最大值决策就有可能不是最优。

对于 a_{i-1}, a_i, a_{i+1}，若有 $a_i \geq a_{i-1}, a_i \geq a_{i+1}$，当第一次选 a_{i-1} 最优时一个人选了 a_{i-1} 另一个人一定会选 a_i，然后第一个人再选 a_{i+1}。考虑到这样的过程中这三个石子的选择一定是连续的，所以我们可以把他们当成一个石子来考虑，也就是把它们合并成一个权值为 $a_{i-1} + a_{i+1} - a_i$ 的石子。

合并完后所有的权值情况只存在递增，递减和下凹的情况了，这三个情况对于双端队列都是可以单调的从大到小选，对于从栈顶方向开始递减的栈的部分位置，也是可以单调的从大到小选，这些位置就可以直接排序后先后手一个一个选了。

例题

而对于从栈顶方向开始递增的栈的部分位置，一定是最后才开始选的，因为这些决策一定是劣于其他决策，提前处理出其选择的结果，最后根据先后手的情况分配即可。

例题

前言：本来准备回应某人的要求放 Pocky Game 的，但是我在某天随机找博弈论的题的时候做到了这道题，然后花费了大量时间，于是就想给大家分享一下。

例题

前言：本来准备回应某人的要求放 Pocky Game 的，但是我在某天随机找博弈论的题的时候做到了这道题，然后花费了大量时间，于是就想给大家分享一下。

P2599

有 n 堆石子，游戏由两个人进行，两人轮流操作，每次操作者都可以从最左或最右的一堆中取出若干颗石子，可以将那一堆全部取掉，但不能不取，不能操作的人就输了。问先手是否有必胜策略。

例题

前言：本来准备回应某人的要求放 Pocky Game 的，但是我在某天随机找博弈论的题的时候做到了这道题，然后花费了大量时间，于是就想给大家分享一下。

P2599

有 n 堆石子，游戏由两个人进行，两人轮流操作，每次操作者都可以从最左或最右的一堆中取出若干颗石子，可以将那一堆全部取掉，但不能不取，不能操作的人就输了。问先手是否有必胜策略。

我声称其是 Pocky Game 的加强版。

例题

好了提 Pocky Game 并不是彩蛋，其实是在告诉你这道题也可以区间 dp。

例题

好了提 Pocky Game 并不是彩蛋，其实是在告诉你这道题也可以区间 dp。事实上对于这种只能在两端取石子的博弈游戏我们通常可以考虑区间 dp，这是一个神秘的 trick 可以记一下。

例题

好了提 Pocky Game 并不是彩蛋，其实是在告诉你这道题也可以区间 dp。事实上对于这种只能在两端取石子的博弈游戏我们通常可以考虑区间 dp，这是一个神秘的 trick 可以记一下。

我们设 $f_{i,j}$ 表示在原来的区间 $[i,j]$ 左边添加 $f_{i,j}$ 会使先手必败，$g_{i,j}$ 表示在区间 $[i,j]$ 右边添加 $g_{i,j}$ 会使先手必败。显然两个函数对称，所以我们只对 $f_{i,j}$ 进行考察，同理可得到 $g_{i,j}$。

例题

好了提 Pocky Game 并不是彩蛋，其实是在告诉你这道题也可以区间 dp。事实上对于这种只能在两端取石子的博弈游戏我们通常可以考虑区间 dp，这是一个神秘的 trick 可以记一下。

我们设 $f_{i,j}$ 表示在原来的区间 $[i,j]$ 左边添加 $f_{i,j}$ 会使先手必败，$g_{i,j}$ 表示在区间 $[i,j]$ 右边添加 $g_{i,j}$ 会使先手必败。显然两个函数对称，所以我们只对 $f_{i,j}$ 进行考察，同理可得到 $g_{i,j}$。

首先我们可以证明 $f_{i,j}$ 唯一。考虑反证法，如果存在至少两个必败态，那么先手一定会操作一次使得新的状态取到最小的那个 $f_{i,j}$ 使后手必败，于是矛盾。因此得证。证明存在性类似，这道题不是重点。

例题

现在考虑如何求解 f。先考虑边界，有 $f_{i,i} = a_i$，因为只剩下两堆一模一样的情况的时候，后手只需要模仿先手的行动对称执行就好了，这样子一定不会输，即先手必败。

例题

现在考虑如何求解 f。先考虑边界，有 $f_{i,i}=a_i$，因为只剩下两堆一模一样的情况的时候，后手只需要模仿先手的行动对称执行就好了，这样子一定不会输，即先手必败。

接下来考虑 $f_{i,j}$，我们可以大力分讨取值得到转移。因为我们要考虑新的左端点的必败情况，于是我们会根据右端点和之前局面的情况来确定。接下来这个分讨特别神秘，我研究了很久，请做好准备 ()

例题

现在考虑如何求解 f。先考虑边界，有 $f_{i,i}=a_i$，因为只剩下两堆一模一样的情况的时候，后手只需要模仿先手的行动对称执行就好了，这样子一定不会输，即先手必败。

接下来考虑 $f_{i,j}$，我们可以大力分讨取值得到转移。因为我们要考虑新的左端点的必败情况，于是我们会根据右端点和之前局面的情况来确定。接下来这个分讨特别神秘，我研究了很久，请做好准备 ()

第一种情况，$a_j=g_{i,j-1}$，因为原来的区间已经是必败了所以左端点不填数就行了。

例题

第二种情况，$a_j < f_{i,j-1}, a_j < g_{i,j-1}$。这种情况下当我们拿完了 a_{i-1} 和 a_j 中的一堆后当时的先手就会获胜，所以谁去拿完一堆谁就输了。考虑要求现在的先手要拿完两堆中的一堆怎么办？这就是一个 Nim 游戏，于是有 $f_{i,j} = a_j$。

例题

第二种情况，$a_j < f_{i,j-1}, a_j < g_{i,j-1}$。这种情况下当我们拿完了 a_{i-1} 和 a_j 中的一堆后当时的先手就会获胜，所以谁去拿完一堆谁就输了。考虑要求现在的先手要拿完两堆中的一堆怎么办？这就是一个 Nim 游戏，于是有 $f_{i,j} = a_j$。

第三种情况，$f_{i,j-1} \le a_j < g_{i,j-1}$。考虑在此情况基础上继续分讨，若 $f_{i,j} \le f_{i,j-1}$，那么先手可以将右边的 a_j 取到 $f_{i,j}$，然后就变成了情况二，这时先后手发生变化了，所以原来的先手获胜了，因此不行。

例题

第二种情况，$a_j < f_{i,j-1}, a_j < g_{i,j-1}$。这种情况下当我们拿完了 a_{i-1} 和 a_j 中的一堆后当时的先手就会获胜，所以谁去拿完一堆谁就输了。考虑要求现在的先手要拿完两堆中的一堆怎么办？这就是一个 Nim 游戏，于是有 $f_{i,j} = a_j$。

第三种情况，$f_{i,j-1} \le a_j < g_{i,j-1}$。考虑在此情况基础上继续分讨，若 $f_{i,j} \le f_{i,j-1}$，那么先手可以将右边的 a_j 取到 $f_{i,j}$，然后就变成了情况二，这时先后手发生变化了，所以原来的先手获胜了，因此不行。

那么 $f_{i,j} > f_{i,j-1}$。现在分析局面，如果先后手轮流操作，当两端有石子小于 $f_{i,j-1}$ 的时候就会变成情况二，所以大家都想避免这种情况。于是有 $a_{i-1} \ge f_{i,j-1}, a_j \ge f_{i,j-1}$，但是因为当 $a_{i-1} = f_{i,j-1}$ 的时候操作者可以直接拿掉 a_j 然后进入获胜态，于是我们得到最后的条件：$a_{i-1} > f_{i,j-1}, a_j \ge f_{i,j-1}$。

例题

于是问题就变成了一个隐秘的 Nim 游戏。是的你没听错也没看错，就是 Nim 游戏。我们考虑当前操作者一定会让 a_{i-1} 变成 a_j+1，这样我们就能在后面的过程中模仿对方的操作从而获胜，于是 $f_{i,j}=a_j+1$。

例题

于是问题就变成了一个隐秘的 Nim 游戏。是的你没听错也没看错，就是 Nim 游戏。我们考虑当前操作者一定会让 a_{i-1} 变成 a_j+1，这样我们就能在后面的过程中模仿对方的操作从而获胜，于是 $f_{i,j}=a_j+1$。

第四种情况，$g_{i,j-1}<a_j\le f_{i,j-1}$，这个与上面对称于是同理有 $f_{i,j}=a_j-1$。

例题

于是问题就变成了一个隐秘的 Nim 游戏。是的你没听错也没看错，就是 Nim 游戏。我们考虑当前操作者一定会让 a_{i-1} 变成 a_j+1，这样我们就能在后面的过程中模仿对方的操作从而获胜，于是 $f_{i,j}=a_j+1$。

第四种情况，$g_{i,j-1}<a_j\le f_{i,j-1}$，这个与上面对称于是同理有 $f_{i,j}=a_j-1$。

最后一种情况，$a_j>g_{i,j-1},a_j>f_{i,j-1}$。讨论和后半段的分析同上，可得 $f_{i,j}=a_j$。

例题

于是问题就变成了一个隐秘的 Nim 游戏。是的你没听错也没看错，就是 Nim 游戏。我们考虑当前操作者一定会让 a_{i-1} 变成 a_j+1，这样我们就能在后面的过程中模仿对方的操作从而获胜，于是 $f_{i,j}=a_j+1$。

第四种情况，$g_{i,j-1}<a_j\le f_{i,j-1}$，这个与上面对称于是同理有 $f_{i,j}=a_j-1$。

最后一种情况，$a_j>g_{i,j-1},a_j>f_{i,j-1}$。讨论和后半段的分析同上，可得 $f_{i,j}=a_j$。

于是你正常区间 dp 然后大力分讨两个函数的转移即可，代码好写，上面分析已经嚼碎喂给你了，代码就按照上面讲的写即可，还是不会可以下来找我。

Contents

瞎扯

到这里这次讲课的内容已经全部讲完了，SG 函数的内容我会等到 NOIP 后的复习时再讲，拟阵也会有概率讲的，请敬请期待！

瞎扯

到这里这次讲课的内容已经全部讲完了，SG 函数的内容我会等到 NOIP 后的复习时再讲，拟阵也会有概率讲的，请敬请期待！

希望这次的讲课能让大家在贪心和博弈论这两个方面的水平有一定提升，最后就是有不懂的地方不管是思路还是代码都可以来问我。

瞎扯

到这里这次讲课的内容已经全部讲完了，SG 函数的内容我会等到 NOIP 后的复习时再讲，拟阵也会有概率讲的，请敬请期待！

希望这次的讲课能让大家在贪心和博弈论这两个方面的水平有一定提升，最后就是有不懂的地方不管是思路还是代码都可以来问我。

最后感谢大家的倾听！（本来想夹带一点私货的但是算了）