CCPC Final 2023 题解

CCPC Final 2023 命题组

April 3, 2024

Statements

给定一个序列 a，初始有一个全零序列 b。

每次可以选择一个长度为 k 的前缀或者一个长度为 k 的后缀将其加一，代价是 k。

问最少需要多少代价能使得对于所有 i 都满足 $b_i \geq a_i$。

$1 \leq n \leq 10^6, 0 \leq a_i \leq 10^9$

Author: Lynkcat; Prepared by Lynkcat

Solution

考虑可能的序列 b 满足什么条件，可以考虑反向操作，将这个序列减为 0。

考虑差分，如果 $b_i > b_{i+1}$，那么至少要进行 $b_i - b_{i+1}$ 次前缀 i 减一的操作。反之亦然。

上述操作完之后，所有的元素都变成相同，如果这个时候 b 非负，那么满足条件。

为了方便，我们在序列开头和末尾添加两个足够大的元素 $b_0 = b_{n+1} = M$，上述条件就是 $b_0 \geq \sum_{i=0}^{n} \max(0, b_i - b_{i+1})$。

Solution

由于 $b_0 = b_{n+1}$ 以及它们足够大，我们有 $\sum_{i=0}^{n} \max(0, b_i - b_{i+1}) = \sum_{i=0}^{n} \max(0, b_{i+1} - b_i)$，把左右两边对应的限制相加，那么上述条件可以转化为

$$\sum_{i=0}^{n} |b_{i+1} - b_i| \le 2M$$

在原问题中，代价总和就等于 b_i 之和，所以问题可以转化为，可以花 1 的代价让 a_i 加上 1，问最少的代价满足上述条件。

Solution

令 $F=\sum_{i=0}^{n}|a_i-a_{i+1}|$。每次我们可以选择一段极长的相同的 $a_l=a_{l+1}=\cdots=a_r$，并且 $a_{l-1}>a_l, a_{r+1}>a_r$，将这一段加一就能将 F 整体地减少 2。这样操作的代价为这一段的长度。

我们贪心地找这样最短的极长段即可，可以对这一段加，然后加到这一段与两头的某个数字相同为止。一直做到 F 不超过 $2M$。

考虑段合并的过程，最后一定是所有数和最大的元素合并，往前就是左右两边独立合并。这个与笛卡尔树比较类似。

我们求出笛卡尔树之后，就能快速得到所有段合并的过程，并且从短到长贪心即可。时间复杂度 $O(n)$。

Statements

给定 n，对于 $i = 1, 2, \ldots, n$ 求出最长可能的周期字符串序列长度，满足序列中字符串的长度 $\leq i$。

一个字符串序列 $S_1, S_2, \ldots, S_\ell$ 是周期字符串序列，当且仅当对于每个 $1 \leq i < \ell$ 都满足 S_i 是 S_{i+1} 的周期，并且它们两两不同。

$1 \leq n \leq 2 \times 10^5$

Author: Kevin114514; Prepared by Kevin114514 & Kubic

Solution

首先考虑对于一个 n 如何求 $F(n)$。我们可以贪心地增量构造字符串序列，具体构造如下：

- 初始序列为 $\{\mathrm{a}\}$，对应了 $F(1) = 1$。
- 对于长度 l，假设我们已经有了一个所有串长度 $\leq l$ 的最优序列，那么我们将这个序列中的每个字符串后插入一个长度为 $l+1$ 的字符串，且在序列的开头插入由一个 a 和 l 个 b 构成的字符串 ab...b。显然，根据题目的性质，每个位置可能添加的字符串是唯一的，如果出现两个位置添加的字符串相同的情况，我们仅在靠前的位置处添加。

Solution

下面我们说明这样增量构造得到的序列是最优的，也就是这样构造得到的序列长度即为所求的 $F(n)$。

首先我们为 $F(n)$ 分析一个上界。设最优的字符串序列为 $\{S_1, S_2, \cdots, S_{F(n)}\}$。若 $|S_1| \neq n$，我们可以在开头插入一个长度为 n 的，满足 S_1 是其前缀的字符串，一定不会使答案变劣。若任何满足这些条件的字符串均在后面的某个位置 S_i 出现，我们允许开头元素存在重复，这不影响我们对 $F(n)$ 上界的分析。下设 $|S_1| = n$。

Solution

容易利用题目条件归纳得出，对于任意一个序列中的字符串 S_i，其可以被表示成若干个字符串 $T_1, T_2, \cdots, T_k$ 的拼接，且每个 T_j 均是 S_1 的前缀。更进一步地，我们要求这些字符串满足 $|T_1| = \max_{j=1}^{k} |T_j|$。同样可以归纳得出，对于任意一个序列中的字符串，均有满足上述两个条件的表示方法。

由序列中任意两个串不同可以得出，每个字符串的上述表示法中，一定要么 k 不同，要么存在一个对应位置的 T_j 满足在两个表示法中长度不同。也即，存在 $F(n)$ 的一个上界，为满足 $\sum_{i=1}^{k} x_i \le n, x_1 = \max_{i=1}^{k} x_i$ 的序列 $\{x_i\}(1 \le i \le k)$ 的个数。

Solution

取 $S_1 =$ ab...b 时，显然对于任意两个不同的满足上述限制的表示法，其对应的字符串都不相同。且容易归纳证明，增量构造得到的字符串序列中，所有可能满足上述限制的表示法对应的串均存在。于是上文给出的增量构造得到的字符串序列是最优的，同时我们可以利用对表示法序列的计数来计算 $F(n)$。

也就是

$$\sum_{i=1}^{k} x_i \leq n, x_1 = \max_{i=1}^{k} x_i$$

的序列个数。

Solution

可列出 $F(n)$ 对应的生成函数为:

$$\frac{1}{1-x}\sum_{k=1}^{\infty}\frac{x^k}{1-\dfrac{x-x^{k+1}}{1-x}}$$

将该式子化简，可以得到:

$$\sum_{k=1}^{\infty}\frac{x^k}{1-2x+x^{k+1}}$$

Solution

进行一些处理，可以得到：

$$\sum_{k=1}^{\infty} \frac{x^k}{(1-2x)\left(1+\dfrac{x^{k+1}}{1-2x}\right)}$$

- 对于 $k \le \sqrt{n}$，我们暴力计算贡献，时间复杂度 $O(n\sqrt{n})$。
- 对于 $k > \sqrt{n}$，我们考虑将上面的展开式展开成 $\sum_i G_i(x)/(1-2x)^i$ 的形式，每个 k 至多展开出根号项，且 $G_i(x)$ 仅在 $i \le \sqrt{n}$ 时有值，故这一部分同样可以做到时间复杂度 $O(n\sqrt{n})$。

  Bonus：可以使用多项式技巧做到 $O(n \log^2 n)$。

Statements

给你一个 n 个点的凸多边形，和内部若干条不相交的边，问能否将整个图二染色，使得不存在同色的环。

$n \le 2 \times 10^5$。

Author: apiad; Prepared by apiad

Solution

本题存在多种构造的方法。

介绍一个最简单的方法[1]，从 1 号点出发 BFS，然后根据距离的奇偶性黑红染色。证明如下：

- 首先在 BFS 树中，只有同层和相差一层的点会连边。如果相差一层，那么颜色一定不同，所以只需要考虑同层节点即可。
- 考虑同层的一个环，一定会把多边形切成若干个区域，考虑 1 所在的区域被 (u, v) 切开。那么 1 到环上其他所有点必须经过 u 或 v 中的一个，与距离相同矛盾。

[1]这个做法来自验题队伍 Shanghai JTU: Nemesis。

Solution

也有一些相对正常的做法。考虑这个图，在图中一定存在一个不超过二度的点，那么我们可以将这个点从图中删除，剩下的图仍然可以看成凸多边形加内部若干条弦的结构。

我们可以通过这个方法得到一个删除序列，然后按照这个序列逆序构造。

每次加入一个点，它至多只有两个邻居已经加入，那么至少存在一种颜色只出现了不超过一次，那么就把这个点设成出现不超过一次的颜色即可。

考虑每种颜色，相当于每次只加入了一个叶子，所以一定不会成环。

时间复杂度是 $O(n)$。这个题还存在分治等其他构造方法。

Statements

给你一个序列，每次你可以把相邻两项合并成为他们的 max 或者 min，问能否把 a 变成 b。

$|a|, |b| \leq 10^5$。

Author: Kubic; Prepared by Kubic

Solution

b 中的每一个元素一定是由 a 中一段区间合并而来。

而 $a_l, a_{l+1}, \ldots, a_r$ 能合并成 b_i 当且仅当 b_i 在 $a_l, a_{l+1}, \ldots a_r$ 中出现过。你可以根据最后保留的数的大小，考虑用 max 或者 min 和周围的数合并即可。

因此 a 能够变为 b 当且仅当 b 为 a 的子序列。$O(n)$ 判断即可。

Statements

给两个 1 到 n 的排列 a,b。

初始有个二元组 $(x,y)=(a_1,b_1)$，对于每个 $2\le i\le n$，每次可以选择

- 把 (x,y) 变成 $(\max(a_i,x),\max(b_i,y))$。
- 把 (x,y) 变成 $(\min(a_i,x),\min(b_i,y))$。

问最终二元组 (x,y) 所有的可能性。

$1\le n\le 5\times 10^5$。

Author: Kubic; Prepared by Kubic

Solution

令最终的二元组为 (a_i, b_j)。考虑有哪些 (i, j) 是可能取到的。

不妨令 $i \le j$。先枚举 i 处进行的操作类型，则 $i \sim n$ 中的操作类型均已确定。

若 $2 \sim i-1$ 中的操作结束之后的二元组为 (x, y)。则 $i \sim n$ 中的操作对 y 的影响一定形如：$y \leftarrow \min(\max(y, l), r)$。其中 $l \le r$ 是仅由 $i \sim n$ 中的操作决定的参数。

因此 j 只有两种可能的取值：$b_j = l$ 或 $b_j = r$。

Solution

l, r 可以在扫描 a_i 的过程中用线段树维护相应信息 $O(n \log n)$ 求出。

只需要再进行 $O(1)$ 次判断是否存在一种对 $2 \sim i-1$ 的操作方式使得 $[x < a_i] = p, [y < b_j] = q$。其中 $p, q \in \{0, 1\}$。

同样可以在扫描 a_i 的过程中用线段树维护 $[x < a_i] = p$ 的前提下 y 的最小、最大值。总时间复杂度 $O(n \log n)$。

Solution

再介绍一个更加简单的做法[2]。

考虑判断 (x, y) 能否作为最终结果，注意到每个数我们只关心它和 x, y 的相对大小关系，所以可以视作 $a_i, b_i \in \{-1, 0, 1\}$，只需要判断最后能否从任意初始状态得到 $(0, 0)$。

于是使用线段树维护转移矩阵，状态为区间 $[l, r]$ 能否让 (x_1, y_1) 变成 (x_2, y_2)，注意这里的转移矩阵只有 $0/1$ 因此矩阵乘法可以使用位运算加速。

[2]做法来自 Haitang0520 Fan Club(dXqwq)。

Solution

通过归纳（或者打表找规律）可以发现将所有合法的 (x_i, y_i) 按照 $x_i + y_i$ 为关键字进行排序之后，(x, y) 的下一个合法二元组一定是 $(x+1, y), (x, y+1), (x+1, y+1)$ 中的一个。

而将 x, y 修改为下个可能的合法二元组，在线段树上的 a_i, b_i 修改只有 $O(1)$ 个，因此我们只会进行 $O(n)$ 次线段树操作，时间复杂度 $O(n \log n)$。

Statements

给你一棵树，一个 1 到 n 的排列 p 和一个常数 K。

q 组询问，每次问排列中的一个区间 $p_l, p_{l+1}, \ldots, p_r$，这些点 K-邻域的并的大小。

$nK \le 2 \times 10^6, q \le 5 \times 10^5$

Author: Lynkcat & apiad, Prepared by Qingyu

Solution

考虑离线，我们从左到右扫描询问的右端点 r，维护左端点的答案。

对于每个点 u，令 lst_u 表示最大的 $x(x \le r)$，满足 u 在 p_x 的 K-邻域中。

那么统计答案的时候，只需要统计 $lst_u \ge l$ 的点个数即可。

Solution

考虑如何维护 lst。以 1 号点为根，求出 BFS 序。

容易发现一个点 p_r 的 K-邻域，在 BFS 序里每一层都是连续的一段，并且只会在其中的 $2K+1$ 层内。

所以每次只需要对 $2K+1$ 个区间做区间赋值即可，并且更新对应 lst。

同时用树状数组之类的数据结构，维护后缀和，统计答案。

时间复杂度 $O((nK+q)\log n)$。

Statements

有一场 n 道题目的比赛，一共 m 秒。

已知对手会在比赛开始后的第 $a_1, a_2, \cdots, a_n$ 秒通过一道题目，而自己做每道题目的用时为 $b_1, b_2, \cdots, b_n$ 秒。

安排自己做题的策略（即，解决题目的顺序和提交题目的时间），最大化 Last Success 是自己的时间。

$n \le 10^5, m \le 10^9$。

Author: znstz; Prepared by znstz

Solution

考虑时间轴，对手通过题目的时间点把这个时间轴分成了若干段，对于一段时间 $[l, r)$，令 x 为这个时间段中自己通过第一道题目的时间，则这一段时间对答案的贡献为 $r - x$。

显然，做题一定按照 b_i 升序排列。下文中称自己过题的时间点为“时间点”。

确定做题顺序后，我们可以把一些时间点往后挪。

Solution

从右往左扫每个段，令当前段为 $[l, r)$，所有时间点按递增排序为 $t_1, t_2, \cdots, t_k$，考虑反悔贪心。维护一个大根堆，如果：

- 遇到一段空段：将 $r - l$ 加入堆。
- 否则：对于 $t_2, \cdots, t_k$，取出堆顶并加到答案里（对应的操作是把这个时间点挪到对应段的段头）；对于第一个时间点，有两种情况：
  - 如果把这个时间点挪到堆顶对应的段带来的贡献为正，就挪过去并加一下贡献，然后把 $r - l$ 加到堆中。
  - 否则，由于可能前面过来一个时间点把当前段占掉，这个时间点去占后面的段。这种情况可以把当前段的贡献 $t_1 - l$，和当前堆顶的贡献合并，直接把 $t_1 - l$ 加到堆顶即可。

时间复杂度 $O(n \log n)$。

Statements

Alice 和 Bob 玩游戏。有一个长度为 $2n$ 的序列，两个人轮流操作，Alice 先手。

他们每次可以删除序列中的一个元素，当只剩一个元素的时候，游戏结束。

如果过程中序列变成回文的，Bob 获胜，否则 Alice 获胜。问最后的胜者。

$1 \le n \le 10^6$

Author: Kevin114514; Prepared by Kevin114514 & apiad

Solution

如果序列一旦变成回文，那么 Bob 就可以模仿 Alice 的操作，让整个序列一直都是回文的。

所以只需要考虑最后的时刻两个元素会不会相同即可，也就是希望最后时刻只有一种不同的元素。

所以 Alice 希望操作过程中不同的元素尽量多，Bob 希望操作过程中不同的元素尽量少。两个人可以贪心地选取出现次数最多和最少的元素。

事实上，Bob 获胜当且仅当一开始不同的元素个数不超过 n。

Solution

如果长度为 $2n$ 的序列中，只有不超过 n 种不同的元素。Bob 一定能保证两轮过后不同的元素不超过 $n-1$ 个。

- 如果 Alice 操作后还有不超过 $n-1$ 种，那么成立。
- 否则至少存在一种元素，出现次数为 1，那么 Bob 可以取这种元素。

反之亦然。如果不同元素个数超过 n，那么 Alice 一定能保证两轮后不同的元素个数超过 $n-1$。

如果现在不同元素个数超过 $n+1$，那么一定成立，否则至少存在一种元素出现次数大于 1，Alice 只要选取这种元素就能保证 Bob 操作时候还有至少 $n+1$ 种不同的元素。

Statements

给定平面上的 n 个点 $A_i=(x_i,y_i)$，你需要求出满足如下条件的排列 $p_1,p_2,\ldots,p_n$ 数量：

- 路径 $p_1,p_2,\ldots,p_n$ 不存在右转，只能左转或直行。形式化地，对于 $1\le i\le n-2$，要满足 $\overrightarrow{A_iA_{i+1}}$ 和 $\overrightarrow{A_{i+1}A_{i+2}}$ 的叉积非负。
- 路径不自交。

$1\le n\le 2000$

Author: 5ab; Prepared by 5ab

Solution

首先我们可以观察到合法的路径满足如下性质：路径一旦经过了凸包上一个点 X，就会连续经过整个凸包，直到 X 的上一个点 Y。

考虑凸包上的一个点 P，若其路径的后继不是 P 的下一个点 Q 的话，则这条路径就不可能再向前走到那个点。为了经过 Q，这条路径必须向后走到 Q。容易证明这样的例外恰好发生两次，且这两个点在凸包上互为前驱后继，设这两个点为 P, Q。

接下来考察凸包内部的情况，可以观察到如下性质：所有与 P 连接的点和所有与 Q 的点可以用一条与 PQ 相交（不包括端点）的直线分割。进一步观察可得，这样的分割也对应了一种连接方案。

Solution

枚举凸包上的边和内部的分割方案，判断是否合法即可。由于本质不同的半平面只有 $O(n^2)$ 个，所以这个做法是 $O(n^3)$ 的，难以通过。

注意到一组分割方案对应合法的边是两段区间，二分出四个端点即可。复杂度 $O(n^2 \log n)$。

Solution

另一种方法是考虑拆贡献，每个分割方案的贡献都可以拆成 $2 + \{$分割方案包含的凸包点数$\}$。考虑对凸包上每个点计算贡献，即经过这个点的直线有多少种划分内部点的方案，这个显然可以直接统计辐角。

复杂度 $O(n^2 \log n)$，可以用哈希表做到 $O(n^2)$。

Statements

给你一棵 n 个点的有根树。每次询问 $a_1, a_2, \ldots, a_k$ 问是否可能为 DFS 序中连续的一段。

$n \leq 10^5, \sum k \leq 10^6$

Author: apiad; Prepared by apiad

Solution

考虑 DFS 序中两个相邻点 x, y，y 一定是 x 的一个儿子，或者从 x 出发往祖先回溯若干步之后的一个新儿子。

需要注意几点：

- 如果是儿子，那么这个子树之前一定不能访问过。
- 如果回溯，那么需要保证回去路上的所有点，它的所有子树都已经被访问。

Solution

回到原题，可以根据上述两点，进行判定[3]。

当前访问的点是 x，访问到的下个点是 y，如果这个点是 x 的儿子，那么需要保证这个点子树内部没有被访问过的点。

否则首先需要保证 y 的父亲为 x 的祖先。然后从 x 往上回溯，如果这个点在之前的 DFS 序里，那么需要保证子树内部的所有点都已经被访问。一直回溯到一个未在之前的 DFS 序里面出现的点，或者到达 y 的父亲祖先即可。

[3]这个做法来自验题队伍 JohnVictor，MagicSpark，huzhaoyang。

Solution

我们可以用 DFS 序 + 树状数组来查询一个子树里面的节点个数。

因为每次往上回溯，一旦碰到一个未在之前 DFS 序出现过的点就会结束。所以每次询问一共访问 $O(k)$ 个点，总的时间复杂度为 $O(k \log n)$，可以通过此题。

Solution

事实上，我们还能将这个做法优化到线性[4]，具体来说：

考虑边遍历边维护一个 dfs 栈。当前访问的点是 x，访问到的下个点是 y，如果这个点是 x 的儿子，那么需要保证这个点之前没有被访问过。判定成功后将 y 压到栈顶。

否则，设序列中上一个满足父亲没出现在序列里的点为 p，我们要判断以下几个条件：

- 保证 y 的父亲为 p 的祖先并且 p 子树内的点均被访问过。
- x 是叶子。
- y 子树中没有点被遍历过。

[4] 这个做法来自 Lynkcat。

Solution

其中，第一个条件可以判断栈内每个点的儿子是否均被访问过，然后将栈清空，第二个条件可以通过预处理每个点的儿子个数来判断，第三个条件，我们可以发现只需要判断序列中上一个父亲与 y 的父亲不同的点是否在 y 子树内即可。

综上所述，时间复杂度 $O(n + \sum k_i)$。

Statements

给你一个 $n \times n$ 的矩阵，每行的最左边和每列的最上面有一根小木棍。令 $x_1, x_2, \ldots, x_n, y_1, y_2, \ldots, y_n$ 分别为它们的长度。

我们要求这些木棍的长度为 0 到 n 之间的整数，并且这些木棍不交。

我们定义一个 01 矩阵 A，其中 01 表示每格是否被木棍占据。

现在给你一个由 $01?$ 构成的矩阵，问有多少种方法把 $?$ 换成 01，满足它是合法的。也就是存在一些木棍，满足它们得到的矩阵与这个 01 矩阵相同。

$n \leq 3000$

Author: Kubic; Prepared by Kubic

Solution

先考虑如何判断一个矩阵 A 是否合法。

假设存在一组 $x_1, x_2, \dots, x_n, y_1, y_2, \dots y_n$ 满足条件。显然我们一定可以找到一条从**格点** $(0,0)$ 到 **格点** (n,n)，每次往右或往下走一格的路径 P，使得所有行木棍都在 P 左下，所有列木棍都在 P 右上。

那么 A 合法当且仅当存在至少一个 P 合法。我们考虑 P 需要满足哪些性质。

Solution

我们求出 a_i 表示第 i 行最多有多少个在 P 左下，b_i 表示第 i 列最多有多少个在 P 右上。

具体地，对于第 i 行，我们只需要找到最小的 $j(1 \le j < n)$ 满足 $A_{i,j} = 0, A_{i,j+1} = 1$，那么就有 $a_i = j$。对于列 b_i 的计算方法类似。

设 $X_i(P)$ 表示 P 中行坐标从 $i-1$ 到 i 时的经过的边对应的列坐标，$X_i(P)$ 表示 P 中列坐标从 $i-1$ 到 i 时的经过的边对应的行坐标。那么 P 合法当且仅当对于所有 $1 \le i \le n$，满足 $X_i(P) \le a_i$，$Y_i(P) \le b_i$。

因此我们可以根据 $a_1, a_2, \dots a_n$ 计算出一条路径 P_1，根据 $b_1, b_2, \dots, b_n$ 计算出一条路径 P_2。这样 P 合法当且仅当 P 在 P_1 左下且在 P_2 右上。

Solution

再考虑如何进行计数。

可以发现，如果存在至少一个 P 合法，那么 P_1 一定是合法的。因此我们可以把 A 放在它对应的 P_1 处统计贡献。

只需要对 P_1 进行 dp，过程中统计有多少个 A 对应当前的 P_1 即可。时间复杂度 $O(n^2)$。

Statements

交互题。给你一个数组 $a_1, a_2, \ldots, a_n$，你可以询问其中一些区间的最大子段和。

请用不超过 $2n$ 次询问得到一个新数组 b，满足 b 的所有区间的最大子段和都和 a 相同。

$n \le 2000, |a_i| \le 10^9$。你构造的 b 只需要满足 $|b_i| \le 10^{15}$ 即可。

Author: Kubic; Prepared by Kubic

Solution

有一个简明做法[5]。

首先可以通过 n 次询问确定每个数的正负，然后可以把相同符号的数字合并，那么可以得到一个正负相间的序列，并且知道所有正的数的值。

考虑查询相邻三项 +-+，如果得到的结果与两个正数都不同，那么我们能知道这个负数的值，然后把这三项合并。否则我们不能合并这三项，但可以得到这个负数的绝对值不小于这两个正数中较小的。

[5] 做法来自 apiad

Solution

可以考虑查询最小的正数，以及周围的两个 +-+，也就是连续五项 +-+-+，并且中间的是最小的正数。

如果两次查询都没有合并，那么得到这个数周围的两个负数的绝对值都不小于它。那么在最大子段和中，单独选这个数一定不优，所以这个数可以和周围的两个负数合并，并且把其中一个负数设成这个数的相反数。

那么每两次查询一定能使得负数段减少 1，这部分的查询不会 n 次，总的查询次数不会超过 $2n$ 次。

Solution

以下为原做法。

令 $p(l,r)$ 表示 $[l,p(l,r)]$ 为 $[l,r]$ 的最大前缀和。根据最大子段和的经典算法，显然有 $p(l,r)=p(l+1,r)$ 或 $p(l,r)=l-1$。考虑所有 $mss(l,r)>mss(l+1,r)$ 的位置，一定有 $mss(l,r)=sum(l,p(l,r))$。

从左往右依次加入每个元素。设已经加入了 $a_1\dots a_{t-1}$，即将加入 a_t。

我们按照 $p(i,t)$ 的相等关系将所有 i 分段。令当前的分段从左到右依次为 $[l_1,r_1]\dots[l_m,r_m]$。再令 p_i 表示满足 $mss(l_i,r_i)=sum(p_i,r_i-1)$ 的点。我们希望维护所有的 l_i,r_i,p_i。

显然一定有 $p(r_i,t-1)=r_i-1$。因此我们要求 $sum(r_i,r_{i+1}-1)\le 0$，否则 $p(r_i,t-1)$ 取 $r_{i+1}-1$ 是更优的。

Solution

依次考虑每个 $i = m \sim 1$，若 $mss(p_i, t) > mss(p_i, t-1)$，则我们不妨认为 $p(p_i, t) = t$，因此有 $sum(r_i, t) = mss(p_i, t) - mss(p_i, t-1)$，据此可确定 a_{r_i} 的值。否则一定有 $sum(r_i, t) \leq 0$，一旦出现这种情况就终止。考虑证明其正确性：

- 我们可以通过 mss 的四边形不等式得到 $mss(p_i, t) - mss(p_i, t-1) \leq mss(p_{i+1}, t) - mss(p_{i+1}, t)$。而在我们的构造中 $mss(p_i, t) - mss(p_i, t-1) = sum(r_i, t)$。因此 $sum(r_i, t) \leq sum(r_{i+1}, t)$，即 $sum(r_i, r_{i+1} - 1) \leq 0$。

因此我们证明了上述构造中对 a_{r_i} 进行的赋值不会影响 $[1, t-1]$ 内部的正确性。

Solution

对于上面访问到的所有 i，将它们对应的 $[l_i, r_i]$ 合并，并计算出新的最大子段和决策点 p 即可得到加入 a_t 后的分段。

依次加入每个元素，并在加入 a_n 后将所有 a_{r_i} 均赋值为 $-\infty$ 即可得到完整构造。

Statements

一条数轴上有 $n+1$ 个洞，每两个洞之间有一个球。

你可以按任意顺序推球，每个球会有一个限制：只能往左，只能往右，或者可以朝两个方向。球会一直移动，直到碰到一个空的洞。

问最多会有多少个球，不会掉到相邻的洞里。

$1 \leq n \leq 5000$

Author: Yakumo_Ran; Prepared by Kubic & Qingyu

Solution

先枚举那个没有球的洞在什么位置，然后分别考虑左右两边，相当于每次可以删除一个球，如果这个球指向的球和它方向相反就把它指向的球涂黑，最终就是要最大化涂黑的球的数量。

假设考虑的是右边，那么删除的过程中第一个球必须要朝右，除此之外没有额外限制条件。

对于右边的情况，这个题就变成从右往左依次扫，因为一定恰好有一个球会贡献，所以每次遇到 $><$ 而且两个都没涂黑的时候就要决策一下涂黑哪个。

如果剩下来的球是 $>$ 就不会对后续做出贡献直接删掉，如果是 $<$ 就可以观望。

dp 记一下第一个 $<$ 是否被涂黑，容易做到 $O(n^2)$。