杂题选讲

kradcigam

江苏省常州高级中学

2025.4.11

Contents

使用说明

- 难度分布（要求我的）：
  - 蓝：2 道；
  - 紫：5 道；
  - 黑：4 道。
- 题目都有超链接，可以直接点开；
- 希望大家能**认真思考**！

Contents

闲话

大家秒一下。

~~蓝题是最难找的。~~

Candies and Stones 4'

题目描述

有 n 个糖果和 m 块石头。Alice 和 Bob 轮流行动，Alice 先行动。

- Alice 行动时，如果 Bob 吃了 a 块糖和 b 块石头，Bob 就会得到 $f(a,b)$ 奖分。其中 $f(a,b)=(x_a+y_b) \bmod p$。
- Bob 行动时，他要么吃掉一块糖果，要么吃掉一块石头。

当 Bob 把除了一块糖和一块石头之外的糖果和石头都吃光时，他最后一次得分，游戏结束。Alice 不允许 Bob 吃所有的糖果，也不允许他吃所有的石头。求出 Bob 如何游戏才能获得最大的分数，并求出一组方案。

数据范围

$2 \le n, m \le 20000$，$1 \le p \le 10^9$，7.5s，45MB。

闲话

注意到直接开二维数组记录空间会被卡。

闲话

注意到直接开二维数组记录空间会被卡。
记录下输出方案 DP 卡空间的多种做法。

做法 1

我们发现直接用 bitset 优化空间就比题目要求的空间限制大了一点，所以我们就将 DP 拆成两半即可。
空间复杂度 $O(\frac{nm}{w})$。

做法 2

们考虑根号平衡。我们先完整地做一遍 DP，每 B 个记录一整行的 DP 值。
然后输出方案考虑从后往前对每个块再做一遍 DP，倒推出方案，记录下到这个块开头的决策位置。
直接做空间复杂度是有两部分：

1. 记录 DP 值：$O(\frac{nm}{B})$。
2. 第二遍 DP 记录方案：$O(Bm)$。

认为 n, m 同阶，所以 $B = \sqrt{n}$ 最优。
注意到第二部分可以用 bitset，于是空间复杂度就变成了 $O(\frac{nm}{B} + \frac{Bm}{w})$。
取 $B = \sqrt{nw}$，得到空间复杂度为 $O(n\sqrt{\frac{n}{w}})$。

做法 3

考虑分治，假设当前分支的区间为 $i \in [l_1, r1], j \in [l_2, r_2]$，表示当前知道 f_{l_1,r_1}，要求 f_{l_2,r_2}，并构造方案。我们先做一遍 DP，记录下 $f_{\lfloor \frac{l_1+r_1}{2} \rfloor, j}$ 的值，从而得到在第 $\lfloor \frac{l_1+r_1}{2} \rfloor$ 行的决策点位置 pos。
于是分治区间就拆成了 $[l1, \lfloor \frac{l_1+r_1}{2} \rfloor], [l_2, pos]$ 和 $[\lfloor \frac{l_1+r_1}{2} \rfloor + 1, r_1], [pos, r_2]$。
注意到如果我们把分治区间看成一个矩形，那时间复杂度就是矩形的面积。而我们一次分治会让矩形的面积除 2。
所以时间复杂度是
$O(nm + \frac{nm}{2} + \frac{nm}{4} + \cdots) = O(\sum_{i=0} \frac{nm}{2^i}) = O(nm)$。
空间复杂度 $O(n+m)$。

走亲访友 8'

题目描述

n 个点 m 条边的简单无向连通图，构造满足下面要求的路径：

- 起点为 s，终点不限。
- 对于走过的每条边 (u_i, v_i)，你要额外决定 $p_i \in \{0, 1\}$，满足：
  1. $p_i = 0$ 表示删除这条边，**且之后不能再次经过该边**；
     $p_i = 1$ 表示不删除这条边。
  2. 如果 $i > 1$，那么 $u_i = v_{i-1}$。
- 路径的长度不能超过 k。
- 最后未删除的边组成一棵 n 个结点的树。

数据范围

$1 \leq n \leq 10^3$，$1 \leq m \leq \frac{n(n-1)}{2}$，$k \geq n + m$。

Hint

可以证明在本题的限制条件下，一定存在合法的方案。

闲话

虽然我觉得这个题不是蓝题的难度)
赛时获得了这个题的一血)

算法 1：$k = 2m$

考虑 DFS 出一棵生成树，对于反向边，我们先往后走一次，再走回来，并在走回来的时候把这条边删掉。
这样一条边只会至多经过两次，路径长度至多为 $2m$。
~~不过这个做法似乎并没有前途。~~

算法 2：$k = n + m$

路径是陌生的，但是回路是熟悉的。
我们尝试将限制**变紧**：把路径改成回路，即要求起点终点相同。
考虑欧拉回路要求每个点的度数是偶数，于是我们可以先 DFS 出一棵生成树。然后自底向上，确定每个子树内的所有点度数都为偶数。
如果当前子树根的度数为奇数，那么就将这个点和它父亲的边复制一遍，即要求这条边走两遍。
对于该非生成树上的边，我们只会恰好经过一次，经过之后删去即可。
这样生成树的边只会至多经过两次，非生成树的变只会至多经过一次，路径长度至多为 $n + m - 1$。

Contents

与自动辅助驾驶畅游世界 8'

题目描述

给定一张 n 个点，m 条边的有向图，有一个起点 s，终点 t。初始时，小明在起点 s，每次他可以选择：

- 随波逐流：随机移动到一条出边；
- 矢志不渝：移动到一条指定的出边，花费 1 的金币（若再次到达该点需重新支付费用）。

小明想要知道，他最少需要准备多少枚金币，才能保证在抵达终点 t 前的任何时刻都存在一条从他的所在地抵达终点 t 的路径。

数据范围

$1 \le n \le 3000$，$1 \le m \le 30000$。

Hint

假设所有结点都可以到达结点 t，那么就可以一直执行随波逐流的操作，就可以到达结点 t。

解法

令当前在结点 x 至少需要准备的金币为 f_x。
考虑类似 01BFS 的过程。
假设当前在更新的点为 cur，令已经确定 f_x 的点为关键点。
首先对于结点 cur 的入边 (y, cur)，可以更新：$f_y \leftarrow f_{cur} + 1$。
考虑推广之前的 Hint，如果一个结点不经过关键点能到达的点集全部都能到达 t，那么就可以更新：$f_y \leftarrow f_{cur}$。
每次需要从不能到达 t 的点集反向 BFS，时间复杂度 $O(nm)$。

Flip the Cards 9'

题目描述

你有 n 张牌，第 i 张牌的正面 a_i，背面 b_i，且形成 $2n$ 的排列。我们认为这 n 张牌是“排好序”的，当且仅当 $\forall i \in [1, n)$ 有 $a_i < a_{i+1}, b_i > b_{i+1}$。

你可以进行进行下面的操作任意次：

- 选择一张牌 i 并把它翻过来，即交换 a_i, b_i。
- 重新排序这 n 张牌，顺序随意。

求如果要使这 n 张牌变成“排好序”的，你最少需要翻几次牌，或告知无解。

数据范围

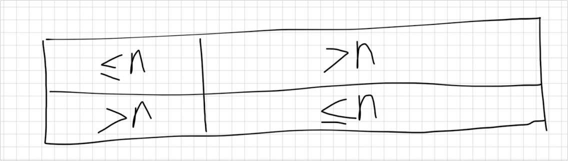
$1 \le n \le 2 \times 10^5$，$1 \le a_i, b_i \le 2n$。

闲话

我觉得这个题很难。

做法

首先考虑如果有一张牌正反面都 $\leq n$ 就无解了。因为有一张牌正反面都 $\leq n$，就一定有牌正反面 $> n$，这两张牌无论怎么放都会冲突。
于是现在所以牌就是一面 $\leq n$，一面 $> n$，我们分析一下最终局面，一定形如这样：



- 第一行的一段前缀 $\leq n$，后缀 $> n$；
- 第二行的对应的那段前缀 $> n$，对应的后缀 $\leq n$。

做法

我们把 $>n$ 的数映射到 $\leq n$ 上，就变成我们定义 f_i 表示一面为 i 的另一面对应的数。
然后就合法方案就等价于将 f 分成两个互补的下降子序列。

做法

结论

对于前缀最小值 $>$ 后缀最大值，则前面对后面没有影响。将这样的段分开，则最终每段划分方案要么不存在，要么唯一。

证明

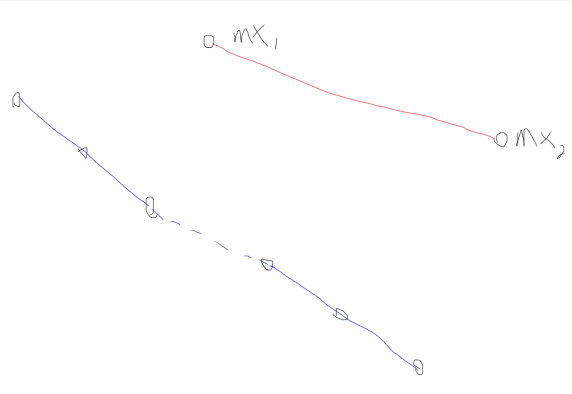
根据 Dilworth 定理，有解即是要求对于一个段的 LIS ≤ 2。

- 我们考虑 $[1, n]$ 最大值的位置假设在 mx_1，注意到我们已经按照前缀最小值 $>$ 后缀最大值分过段，所以要求 $mx_1 \neq 1$。并且，由于 LIS ≤ 2，所以下标在 $[1, mx_1)$ 的数递减。
- 我们考虑 $[mx_1 + 1, n]$ 最大值的位置假设在 mx_2，注意到我们已经按照前缀最小值 $>$ 后缀最大值分过段，所以要求 $mx_1 + 1 \neq mx_2$。
  并且，由于 LIS ≤ 2，所以下标在 $[mx1_1, mx_2)$ 的数递减。

做法

证明

不难发现，此过程可以继续归纳下去，并且方案唯一。



做法

那么，我们只需要对于每个段进行贪心地分配，如果有解，则对两个子序列的分配方案求出较小值。
时间复杂度 $O(n)$。

健身房 9’

题目描述

健身房里有 k 个器材。
有 n 个人预约了健身。第 i 个人预约给定了 l_i, r_i, p_i，意思是要分配给他 $l_i, l_i+1, \cdots, r_i$ 中的一个（记为 x），他在第 x 个小时中用器材 p_i 健身。
同一时间不能有两个人用同一个健身器材。此外，老板还希望让健身房里没人的时刻尽量多，这样可以节约电费。
构造一组最优解。

数据范围

$1 \le n \le 10^6$，$1 \le k \le 10^9$，$1 \le l_i \le r_i \le 10^9$，$1 \le p_i \le k$。

算法 1（我赛时的唐龙做法）

在不影响合法性的情况下，我们让第一次的时间尽量晚，显然是不劣的，因为这样可以使得更多的器材得到匹配。
同时，对于一个健身房开门的时刻，我们一定会对每个器材，分配给能选择中的 r_i 最小的人。

证明

假设存在一组时间方案 $\{p_i\}_{i=1}^k$ 合法，并且在保证合法性的情况下，第一次操作最晚可以为 t。
那么，一定存在一组方案为 $\{\max\{p_i, t+i-1\}\}_{i=1}^k$。
首先因为我们保证了合法性，所以这段被重新分配的前缀一定能满足 l 的限制，另外由于它们时刻更大，所以能够覆盖的器材集合更多。
所以一定也是一组合法方案。

解法

我们深入剖析下不影响合法性会限制什么。现在只考虑**有解**的情况。

对于第 x 种器材，我们令 $cnt_{x,j}$ 表示满足 $p_i = x \land r_i \leq j$ 的数量，那么限制即为:

要求第一次健身房开发时刻必须不晚于 $\min_{j, cnt_{x,j}>0}\{j - cnt_{x,j} + 1\}$。

证明

令 $lim = \min_{j, cnt_{x,j}>0}\{j - cnt_{x,j} + 1\}$。

那么相当于将所有区间的 $l_i \leftarrow \max\{l_i, lim\}$。

解法

证明

判断合法考虑 Hall 定理，简单分析可得不合法当且仅当存在：

- 选择区间 $[L, R]$，令满足 $L \le l_i \wedge r_i \le R$ 的 i 的数量为 cnt；
- $cnt \le R - L + 1$。

假设选择的区间 $L > lim$，那么我们发现将左端点 chkmax 的操作是不影响的。因为原本就合法，所以现在仍然合法。
若选择的区间 $L = lim$，那么根据我们上面对 lim 的求值可得：

$$\forall j \ge lim, j - cnt_{x,j} + 1 \ge lim$$

$$\forall j \ge lim, cnt_{x,j} \le j - lim + 1$$

所以是合法的。

解法

于是，我们对每个器材建线段树求出 lim，然后用一个小根堆维护。
由于要求哪些器材当前时刻可以分配，于是我们再在外面套个按时间轴扫描线即可。
时间复杂度 $O(n\log n)$。

算法 2（高妙做法）

首先“在不影响合法性的情况下，我们让第一次的时间尽量晚，显然是不劣的，因为这样可以使得更多的器材得到匹配。”这个结论仍然需要被用到。

我们先考虑一个特殊性质下的情况：

- 同种健身器材的 r_i 互不相同。

这时候我们发现算法 1 中的 lim 可以直接用一个堆，找最小值即可。

算法 2（高妙做法）

对于一般情况，假设存在 i, j 满足 $p_i = p_j$，并且 $r_i = r_j$。
不妨假设 $l_i \le l_j$，那么我们可以直接将 $r_i \leftarrow r_i - 1$。

正确性证明

假设最后位置 i 取了 r_i，位置 j 取了位置 pos。由于有 $pos \ge l_j \wedge l_i \le l_j$，所以 $l_i \le pos$，所以必然可以调整。

后面的做法一致。
时间复杂度 $O(n \log n)$，根本想不到这一步啊⋯⋯

信号 9'

题目描述

给定长度为 n 的非负整数序列 $a_1, a_2, \ldots, a_n$ 和正整数 k。
你需要构造一个长度为 n 的非负整数序列 $b_1, b_2, \ldots, b_n$，满足：

- $\forall 1 \le i \le n$，$b_i \in [0, 2^k)$；
- $\forall 1 \le i \le n$，$\operatorname{popcount}(b_i \oplus b_{i \bmod n+1}) = a_i$。

或者报告不存在合法解。
这里，$\oplus$ 代表按位异或运算，$\operatorname{popcount}(n)$ 表示非负整数 n 二进制表示下 1 的个数。

数据范围

$n \ge 2$，$k \ge 1$，$nk \le 5 \times 10^6$，$0 \le a_i \le k$。

Hint

等价于有 k 次操作：

- 每次操作选择偶数个位置，将这些位置 $b_i \leftarrow b_i - 1$。

要求最后 $\forall i, b_i = 0$。

根据这个问题的解，构造原问题的解是一个前缀异或和的过程。

Hint

等价于有 k 次操作:

- 每次操作选择偶数个位置，将这些位置 $b_i \leftarrow b_i - 1$。

要求最后 $\forall i, b_i = 0$。
根据这个问题的解，构造原问题的解是一个前缀异或和的过程。
贪心是没有前途的（~~其实只是我不会也不知道正确的贪心~~），思考 DP！

解法

考虑 DP，令 $f_{i,j}$ 表示当前在第 i 个数，其中有 j 次操作当前选择了奇数个位置，是否存在这样的方式。
于是我们需要求解的就是 $f_{0,0}$ 到 $f_{n,0}$ 的路径。
直接暴力转移时间复杂度 $O(nk^2)$。
考虑对于 $f_{i,j}$ 能转移到的是个区间，所以可以使用前缀和处理，时间复杂度 $O(nk)$。
构造方案的时候，找到一条路径即可。

更进一步

对于这个子问题，注意到对于 f_i，合法的 j 构成一段区间。

证明

我们归纳地说明这件事，首先 f_0 显然成立。
我们深入剖析一下转移的结构：
假设当前考虑 $f_{i-1,j}$ 向 $f_{i,j'}$ 的转移，假设第 i 个数为 x，令这 j 次操作，和这 x 次修改的交大小为 t：

- 上界：$\min\{i,x\}$；
- 下界：$\max\{0,i+x-k\}$。

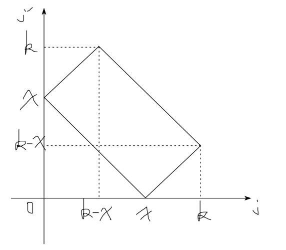
所以 j'：

- 上界：$i+x-2\max\{0,i+x-k\}$；
- 下界：$|x-i|$。

更进一步

证明

可行的 j'：



不难发现，一个区间的 j 可转移到 j' 的并集依旧是一个区间。

于是对于这个子问题可以做到时间复杂度 $O(n)$。

Squid Game 10'

题目描述

给定一个 n 个点的树，以及 m 条路径。
你需要选取尽可能少的点，使得对于每条路径 (x_i, y_i)，都存在一个被选的 z 点，使得链上到 z 距离最短的点既不是 x_i 也不是 y_i。

数据范围

$1 \le n, m \le 3 \times 10^5$。

解法

以 1 为根，用 T_u 表示 u 子树。
分类讨论，对于一条路径 u, v，考虑是不是祖先-后代链。

- 如果是祖先后代链，不妨设 u 是 v 的祖先，且 v 在 w 子树中，其中 $w \in \mathrm{son}(u)$，那么说明在 $T_w \setminus T_v$ 中至少选一个点。我们称这个 w 为关键点。
- 如果不是祖先后代链，说明在 $T_1 \setminus (T_u \cup T_v)$ 中至少选一个点。

解法

设 f_x 表示仅考虑关键点在 x 子树内的前提下，x 子树内至少选多少点，那么转移就是：

- 对于一个点 x，初值为 $f_x = \sum_{y \in \mathrm{son}(x)} f_y$；
- 但此时可能出现对于一条祖先后代链为 (u, v)，并且满足 $w = x$，此时这条链并不合法的情况，那么说明 f_x 选的点全都在 v 子树内，于是只需要如下更新即可：

$$f_x \leftarrow f_v + 1$$

解法

对于一条非祖先-后代链 (u, v)，如果出现不合法的情况，那么说明 f_1 选的点全都在 u 或 v 子树内，于是只需要如下更新即可：

$$f_1 \leftarrow f_u + f_v + 1$$

时间复杂度可以为 $O(n+m)$，求 LCA 可以求 $O(n) - O(1)$，求 w 可以离线 DFS。

Contents

Assigning Fares 12’

题目描述

给出一棵有 n 个节点的树和 m 条树上的路径。
你要给每个点分配一个可重复的正整数标号，使得这 m 条路径上点的标号都是单调的（增/减）。
如果有解，你还要使得标号的最大值最小并输出方案。

数据范围

$2 \le n \le 5 \times 10^5$，$1 \le m \le 5 \times 10^5$。

解法

令 a_i 表示最后第 i 个点的点权，b_i 表示 $[a_i < a_{fa_i}]$。
于是，限制就可以表示成 b_a 之间的相等或不等关系，使用树上差分和带权并查集即可对边集得到等价类。
注意到答案显然具有可二分性，我们考虑二分 mid，并判断是否合法。
我们考虑希望 a_x 取值的限制尽可能少，所以：

- 对于 $b_i = 0$，我们希望 a_i 尽量大；
- 对于 $b_i = 1$，我们希望 a_i 尽量小。

解法

注意到对于一组合法解，我们将所有权值都变成 $a_i \leftarrow k - a_i + 1$，则仍然是一组合法解。
于是，我们可以令 f_i 表示假设 $b_i = 1$ 时，a_i 的最小值。那么当 $b_i = 0$ 时，a_i 的最大值是 $k - f_i + 1$。

解法

考虑转移，我们对于所有连通块分开处理：

1. 对于和 x 在同一个连通块的儿子 i，那么 b_i 的值就根据 b_x 确定了，所以就可以确定 a_x 的限制；
2. 否则，对于同一个连通块，我们对其中要求 b_i 取值相反的两种部分，分别求出 f 的最大值，假设为 mx_1, mx_2 不妨假设 $mx_1 \le mx_2$。
   于是，可能的取值区间就是 $[mx_1+1, k-mx_2] \cup [mx_2+1, k-mx_1]$。
   注意到最优解一定可以形如全取 $[mx_1+1, k-mx_2]$ 或全取 $[mx_2+1, k-mx_1]$。
   并且由于有一第一类限制的存在，所以我们可能出现全取 $[mx_1+1, k-mx_2]$ 无解的情况。
   于是，我们维护 $[l_1, r_1], [l_2, r_2]$ 表示两种情况即可。

解法

对于构造方案，我们维护 rev_i 表示是否需要将子树内的权值变成 $k - a_i + 1$，然后根据 DP 的转移去更新 rev_i。
时间复杂度 $O(n \log n)$。

Fast as Fast as Ryser 13'

题目描述

有 n 个点，第 i 个点和第 j 个点之间有 $a_{i,j}$ 条边。
求大小为 k 的匹配的方案数，对 2^{64} 取模。
对 $k = 1, 2, 3, \cdots, \lfloor \frac{n}{2} \rfloor$ 求出答案。

数据范围

$1 \le n \le 40$，6s。

Hint

对 2^{64} 取模并没有性质，只是单纯为了减少常数。
别想偏了。

解法

一个神奇的想法是在匹配里加入 $(1,2),(3,4),(5,6),\cdots$ 这些边。于是 S 是一组匹配当且仅当新图构成了一个仅有环和链构成的图。

由于这样子连边后只有 $\frac{n}{2}$ 个连通块，那么可以设：

- f_S 为 S 内的点构成一个环的方案数；
- g_S 为 S 内的点构成一条链的方案数。

解法

对于环，我们可以钦定从最小点出发计数；
对于链，我们可以使用 unsigned __int128 算 2 遍，也可以通过环删去一条边来处理。
注意到匹配数其实只关心选的链的条数。
我们再用子集卷积 exp 计算即可。

解法

我们令 $m=\lceil \frac{n}{2} \rceil$。
这部分是这样的，我们令 DP：$f_{i,S}$ 表示当前选了 i 条链，覆盖点集的状态为 S。
每次我们对 S 最高位的 0 分讨是选链还是环。转移时使用子集卷积。
由于选了 i 条链时，最高位的 0 一定不超过第 $m-1-i$ 位。
所以复杂度为：

$$\sum_{i=0}^{m}\sum_{j=0}^{m-i}2^j j^2=\sum_{i=0}^{m}O(2^{m-i}(m-i)^2)=O(2^m m^2)$$

Bomb 15'

题目描述

Peter 在一条直线上有 n 个炸弹，第 i 个炸弹位于位置 x_i。每个炸弹都有一个爆炸半径 r_i（r_i 是一个整数）。当一个炸弹爆炸时，所有不超过爆炸半径的炸弹也会爆炸。一个爆炸半径为 r 的炸弹将花费 r^2 美元。Peter 希望为每个炸弹选择爆炸半径 r_i，以便无论最初引爆哪个炸弹，最终所有炸弹都会爆炸。帮助 Peter 最小化 n 个炸弹的总成本。

数据范围

$1 \le n \le 3000$，$1 \le x_i \le 10^6$。

解法

我们考虑对于前 i 个点观察其导出子图的形态。当前会形成若干强连通分量，假设总共有 k 个。

就一定需要满足其中第 i 个强连通分量能够到达第 $i+1$ 个强连通分量，否则不管后面如何加点都于事无补。因此为了保证强连通分量的极大性，第 i 个强连通分量不能够到达地 $i-1$ 个强连通分量。

考虑当前加入一个点 i，对强连通分量产生的变化。

解法

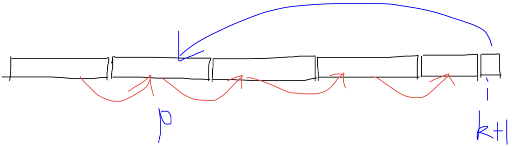
首先，需要满足**限制**：第 k 个强连通分量要能到达点 i，不然必然不合法。

此外，加入点 i 可以看成是先将点 i 作为第 $k+1$ 个连通块。然后我们找到当前 i 向左能够到达最远的强连通分量，假设是第 p 个，那么就可以将的 p 个强连通分量到第 $k+1$ 个连通块合并。

由于我们需要满足上面提到的**限制**，所以一个暴力的想法是，我们记录每个强连通分量向右到达最远位置。此时，一个性质是，对于新连通块向右到达最远位置，一定是第 p 个强连通分量向右到达最远位置和点 i 向右到达最远位置的较大值。

解法

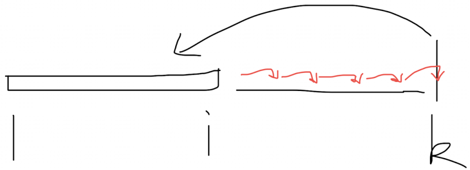
这是因为，如果存在第 $q(q > p)$ 个强连通分量向右到达最远位置比 i 远，那么说明存在一个点的半径比 i 大，由于它的坐标更左，所以理应在此之前就将第 p 个强连通分量合并了。



解法

我们关注第一个连通块的信息，假设当前第一个连通块为前 i 个结点，且向右到达最远位置为 j（并且对于 $i \neq n$，有 $j > i$），令前 i 个点半径平方和的最小值为 $dp_{i,j}$。

假设下一次第一个连通块拓展位置为 k，那么我们先声称 $p \in [i+1 \sim k-1]$ 这部分的点，半径一定为 $a_{p+1} - a_p$。



解法

现在，我们来说明一下这部分的最优性。令点 p 向右到达最远位置为 nxt_p，那么为了保证连通性，一定需要有 $[p, nxt_p]$ 能将 $[i+1, k-1]$ 覆盖满。

而我们将第 p 个点的半径为 $a_{p+1} - a_p$ 不仅保证了总和是下界，同时还保证了已经分配得尽可能均匀，所以是最优的。

解法

于是我们考虑转移，枚举点 k 的向右到达最远位置为 l。
那么可以得到：

$$dp_{i,j} + \sum_{p=i+1}^{k-1} (a_{p+1} - a_p)^2 + \max\{a_l - a_k, a_k - a_i\}^2 \to dp_{k,\max\{j,l\}}$$

此时，我们就会意识到 j 的作用是为了限制时刻都满足过程中 $j > i$，直到 $i = n$。
现在，我们考虑优化转移，首先这个 $\sum_{p=i+1}^{k-1}(a_{p+1} - a_p)^2$ 可以用前缀和轻松处理。我们对两个 $\max$ 的取值分类讨论。

解法

1. $\max\{j,l\}$ 的取值为 l:
   这个时候，我们发现我们不关心 j 的取值，于是可以看成直接从 $dp_{i,i+1}$ 转移得到。
   1. $\max\{a_l-a_k,a_k-a_i\}$ 的取值为 a_l-a_k $a_l-a_k\geq a_k-a_i$ 得到 $a_i\geq 2a_k-a_l$。
      我们可以枚举 k，再枚举 l，双指针可行找到 i 的最大可行值为 pos，预处理 $g_p=\min_{t=p}^{i-1} dp_{t,t+1}$ 即可。
   2. $\max\{a_l-a_k,a_k-a_i\}$ 的取值为 a_k-a_i
      那么我们可以枚举 k，再枚举 i，双指针找到 l 的最大值，最后求完之后对 dp_i 做一遍后缀 chkmin 即可；
2. $\max\{j,l\}$ 的取值为 j:
   朴素的想法是，我们可以使用斜率优化。
   我们分析一下性质，发现由于过程中保持 j 不变，所以此时每次一定是从 i 转移到 $i+1$。

时间复杂度 $O(n^2)$。

这个做法的代码

代码非常简洁：

```cpp
const int N = 5010;
const ll INF = 1e18;
int n, a[N];
ll dp[N][N], s[N], g[N];
signed main() {
    while (cin >> n) {
        F(i, 1, n) cin >> a[i];
        F(i, 2, n) s[i] = s[i - 1] + (ll) (a[i] - a[i - 1]) * (a[i] - a[i - 1]);
        F(i, 1, n)
            F(j, i, n)
                dp[i][j] = INF;
        F(i, 2, n) dp[1][i] = (ll) (a[i] - a[1]) * (a[i] - a[1]);
        F(i, 2, n) {
            g[i] = INF;
            DF(j, i - 1, 1) g[j] = min(g[j + 1], dp[j][j + 1] - s[j + 1]);
            int pos = 1;
            DF(j, n, i) {
                while (a[i] - a[pos] > a[j] - a[i]) pos++;
                chkmin(dp[i][j], s[i] + g[pos] + (ll) (a[j] - a[i]) * (a[j] - a[i]));
                chkmin(dp[i][j], dp[i - 1][j] + (ll) (a[i] - a[i - 1]) * (a[i] - a[i - 1]));
            }
            pos = n;
            F(j, 1, i - 1) {
                while (a[pos] - a[i] > a[i] - a[j]) pos--;
                chkmin(dp[i][pos], s[i] - s[j + 1] + dp[j][j + 1] + (ll) (a[i] - a[j]) * (a[i] - a[j]));
            }
            DF(j, n - 1, i) chkmin(dp[i][j], dp[i][j + 1]);
        }
        cout << dp[n][n] << '\n';
    }
    return 0;
}
```

但对比 std 仍显得无比冗长。

更进一步的代码

```cpp
#include <bits/stdc++.h>
using namespace std;
typedef long long LL;

const int MAXN = 3000 + 10;
LL dp[MAXN];
int x[MAXN], n;

inline LL sqr(LL x) {return x * x;}

int main() {
  while (scanf("%d", &n) == 1) {
    for (int i = 0; i < n; ++ i) scanf("%d", x + i);
    if (n == 1) {puts("0\n0"); continue;}
    dp[n - 2] = 2 * sqr(x[n - 1] - x[n - 2]);
    for (int i = n - 3; i >= 0; -- i) {
      dp[i] = 1ll << 60;
      LL sum = sqr(x[i + 1] - x[i]);
      for (int j = i + 2, k(i); j < n; ++ j) {
        while (x[k] - x[i] < x[j] - x[k]) ++ k;
        LL cost = sum + dp[j - 1] - sqr(x[j] - x[j - 1]);
        dp[i] = min(dp[i], cost + sqr(min(x[k] - x[i], x[j] - x[k - 1])));
        sum += sqr(x[j] - x[j - 1]);
      }
    }
    printf("%lld\n", dp[0]);
  }
  return 0;
}
```

有没有大神教教我 std 在写啥。。。

倾诉 18'

题目描述

小 I 的小圈子里有 n 个人，第 i 个人初始**有正整数** a_i 的烦恼。
小 I 可以在活动中组织**不超过** k 次倾诉。每次倾诉中，某个倾诉者 p $(1 \le p \le n-1)$ 向右手边的人 $p+1$ 倾诉，这首先导致 $a_{p+1} \leftarrow a_{p+1} + \frac{1}{2} a_p$，然后 $a_p \leftarrow 0$。小 I 可以任意选择每次倾诉的倾诉者。注意编号为 n 的人不会向其他人倾诉。
小 I 希望大家的烦恼尽可能少，于是他想知道：在活动过后，所有人最终烦恼的最大值最小是多少。
你需要输出答案的精确值。具体地，答案总能写成 $\frac{S}{2^n}$ 的形式，你需要输出 S 的二进制表示。

数据范围

$1 \le n, \sum n \le 2 \times 10^4$，$1 \le k \le 10^9$，$\forall 1 \le i \le n, 1 \le a_i \le 10^6$。

闲话

思考了一下讲我的互测题还是讲这个题，觉得讲我的互测题有点抽象，而且不会有人补)，不过这个题好像更抽象)))

赛时凭借~~暴力~~唯一通过了这道题，并且分享一下我赛时是怎么想到~~、并卡常~~的。

官方题解 Link:

https://qoj.ac/download.php?type=solution&id=9684。

性质分析

以下定义 $lim = \lceil \log_2 a \rceil$。
首先结构一定是形如将整个序列划分成若干子段，形如 (l, r, p) 的结构，其中 $p \geq r$，并且 p 严格单调递增，那么这一段的操作次数就是 $p - l$，不妨令 $f(l, r, p)$ 表示对应的权值。
由于 $a_n \geq 1$，所以说明最后一段的权值一定 ≥ 1。所以我们可以说明一定存在一组解，满足 $p - r \leq lim$，因为否则我们让 p 减小 1，则这一个段的权值依旧 < 1，同时还节省了 1 次操作次数，并且仍然满足 p 单调递增的限制，于是 (p, r) 只有 $O(n \log n)$ 种。

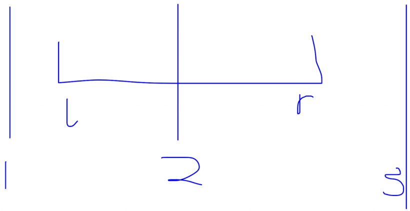
随机二分 trick

考虑对 $O(n^2\log n)$ 组 $f(l,r,p)$ 进行随机二分，每一次可以扔掉期望一半，假设当前随机取的中点为 $f(L,R,P)$。我们考虑 DP，令 $f_{i,j}$ 表示当前最后一段满足 $p=i,r=j$，最小的操作次数，那么限制就变成了判断 $f_{n,n}\le k$ 即合法。
我们考虑从小到大枚举 p，将第一维滚动，然后在枚举 $O(n\log a)$ 种 r，注意到 (l,r,p) 的权值随着 l 递减而递增，所以存在一个分界点 $lpos$，满足 $l\ge lpos$，$f(l,r,p)\le f(L,R,P)$。那么，我们就可以树状数组维护后缀 dp_j-j 的 max，单点 chkmax，维护 DP。

解法

现在问题转化成如何求出 $lpos$。
我们先在外层二分，那么现在我们只需要判断 $f(l,r,p)$ 和 $f(L,R,P)$ 的大小关系。我们考虑二分 + 哈希求出 LCP，考虑计算 $\lfloor f(l,r,p)2^t \rfloor \pmod{10^9+7}$。
我们令 $h=p-t$，这里是 0 次项的位置，然后分类讨论。

计算哈希值



不难发现，我们只需要求出 $a_i 2^i$ 的前缀和，以及一个区间的进位即可。

计算哈希值

这里说的一个区间的进位是指，对于 (l,r)，求出 $\lfloor \sum_{i=l}^{r} \frac{a_i}{2^{r-i+1}} \rfloor$。
我们考虑对于 $l \ge r - lim$，预处理即可。对于 $l < r - lim$，我们发现由于这部分的贡献和 < 1，所以最多只会使得下取整的值增加 1。
注意到关于 l 减小是单调上升的，所以我们维护 pos_i 表示 $l \le pos_i$ 下取整的值会再额外增加 1 即可。这里求出 pos_i 可以从左往右递推。
于是就可以做到 $O(n \log V)$ 预处理，$O(1)$ 查询哈希值。
求出 LCP 之后考虑具体计算 2^{-t} 那一位的值，只需要 $\lfloor f(l,r,p)2^t \rfloor - 2\lfloor f(l,r,p)2^{t+1} \rfloor \pmod{10^9+7}$ 即可。
PS：赛后听 qiuzx 说这部分是一道 UCup 题，我赛时还在这部分卡了很久。

当前解法的时间复杂度

这样，我们分析一下时间复杂度。

- 随机二分；
- (r, p)；
- 二分 $lpos$；
- 二分 LCP。

总共 4 个 log，时间复杂度 $O(n\log^3 n\log V)$。

卡常

CTS 的评测机很慢（比 QOJ 慢不少），赛时进行了一些卡常：

- 二分 $lpos$ 时，上下边界可以设成随机二分的上下边界，正确性显然。虽然这样不能改变时间复杂度，但是可以优化常数。具体来讲，这部分的开销变成了 $\log n + (\log n - 1) + (\log n - 2) + \cdots + 1$，带了 $\frac{1}{2}$ 的常数；
- 二分 LCP 的时候，由于大概率 LCP 较小，所以可以使用倍增。具体来讲，先从 $2^0, 2^1, \cdots, 2^t$，再从 $2^{t-1}, 2^{t-2}, \cdots, 2^0$，实测本地随机数据能快超过一倍。

这样就可以通过了！

优化两次二分

首先我们把 $f(l,r,p)$ 看成是一个最低位为 2^{l-p} 的二进制字符串，这样我们可以更好地定义 LCP。
我们先优化二分 $lpos$ 再二分 LCP 的部分，我们改成直接二分 LCP，假设当前二分判断 x。
那么就是判断否存在 l 满足 $\lfloor f(L,R,P)2^x \rfloor = \lfloor f(l,r,p)2^x \rfloor$。
由于上面我们更改了 LCP 的定义，于是首先有要求 $l \le p-x$，注意到 $[p-x,r]$ 这部分的贡献是确定的，我们直接利用哈希值就可以求出，于是我们就确定了 $[l,p-x)$ 这部分的进位的值。
由于进位只有 $\log V$ 种，并且有单调性，所以直接在上面二分就是 $\log\log V$ 的。
考虑求出 LCP 后，$2^{-(x+1)}$ 位就会存在一个大小关系的变化点，利用进位的 $\log V$ 种值就可以求出变化点。

优化 $\log n \log\log V$

首先可以用哈希表做到 $O(1)$，但这样显然只会更慢)
考虑由于进位对于 $l < r - lim$ 只有 2 种取值的性质。
假设取到 LCP 的最小的左端点为 l'，那么对于判断 x 满足 $x < p - l' - lim$，一定有 $l \le x - lim$。
是我们可以先二分出一个决策点 x'，并只用 $r - l \ge lim$ 的两种取值进行判断，现在我就求出了 LCP'。
于是真实的 LCP 在 $[\mathrm{LCP}' - lim, \mathrm{LCP}]$ 之间，我们再在这个长度为 $O(\log V)$ 的区间里二分。
这样就变成了时间复杂度 $O(\log n + (\log\log V)^2)$。
实际上我们可以通过在这个长度为 $O(\log V)$ 的区间里双指针变成时间复杂度 $O(\log n + \log V)$。
这样现在总时间复杂度变成了 $O(n\log n\log V(\log n + \log V))$。

优化二分次数

我们对于同一个 r 的所有 t 一起求出分界点。
假设当前的 l 满足 $l \geq r - lim$，那么由于有单调性，所以我们可以双指针求出，这部分的比较我稍后再说。
假设 $l < r - lim$，那么我们进行一次二分，则对于 $t' > t$，一定有 $lpos = 1$，这是因为 $[1, l-1]$ 这部分的值 $< [l, r]$ 这部分的值的一半。而 $[l, r]$ 这部分的值 $/2$ 加上 $[1, l-1]$ 这部分的值，显然只会更小。

比较一个长数和一个 lim 长度的数

由于 $x \leq lim$，所以我们可以计算 $\lfloor f(L,R,P)2^x \rfloor$ 和 $\lfloor f(l,r,p)2^x \rfloor$ 的**精确值**。
如果不相等，则已经比较出大小，否则还需要判断 $f(L,R,P)$ 在低于 2^{-x} 位里有没有值。
直接判断 $f(L,R,P)2^n$ 的哈希值是否和 $\lfloor f(L,R,P)2^x \rfloor 2^{n-x}$ 的哈希值是否一样即可。

优化 DS 部分

我们发现一次随机二分，我们调用了 $O(n\log V)$ 次树状数组查询，现在这部分已经成为了复杂度的瓶颈。
而由于我们得到了对于 $lpos$ 的性质，即不满足 $t-lpos\le lim$ 和 $lpos=1$ 的，只有 $O(n)$ 个。
对于 $t-lpos\le lim$ 的，我们可以双指针求，对于 $t=1$ 的，则是一个全局 $\min$。
于是，我们只需要进行 $O(n)$ 次树状数组查询。
现在总时间复杂度变成了 $O(n\log n(\log n+\log V))$。

Contents

谢谢大家

Thanks!

祝大家在 NOI2025 中取得好成绩！