Problem A. Mio visits ACGN Exhibition

简明题意

有一个 $n \times m$ 的网格，每个位置为 0 或 1 ，定义只能向右或向下走，问从左上到右下的所有路径中满足 0 的数量大于等于 p ， 1 的数量大于等于 q 的路径数，答案对 998244353 取模

数据范围

$1 \leq n, m \leq 500, 0 \leq p, q \leq 100000$

做法

如果直接暴力枚举每条路径并统计 0 , 1 的个数，复杂度 $O(2^{n+m})$, 显然不行

设 $dp[i][j][k]$ 代表在 (i, j) 处 0 的个数恰好为 k 的路径数

那么很明显有转移

$$\begin{cases} (i,j)=0, dp[i][j][k] = dp[i-1][j][k-1] + dp[i][j-1][k-1] \\ (i,j)=1, dp[i][j][k] = dp[i-1][j][k] + dp[i][j-1][k] \end{cases}$$

最后统计 $\sum_{k=p}^{n+m-1-q} dp[n][m][k]$ 即为答案，时间复杂度 $O(nm(n+m))$

然而这样空间复杂度 $O(nm(n+m))$ 会炸

我们发现每个位置的转移只与当前行与上一行有关，考虑用滚动数组去优化这个过程，每次只记录参与转移的两行

$dp[now][m][k]$，now 是当前行，now$\hat{1}$ 就是上一行，每转一行 now$\hat{=}$1

这样空间复杂度 $O(2m(n+m))$

或者可以开 $n \times m$ 个 vector，我们发现一个位置至多转移两次，那么在它转移两次后这个状态便不会被再次用到，直接释放这块的内存即可。理论上空间复杂度可以达到 $O(m(n+m))$ ，但由于多次释放和申请空间，时间上会稍微慢一点。

Tips: vector 在 clear 后还需进行 shrink_to_fit 否则相当于你释放了个假的空间

Problem B. Continued Fraction

考虑分数 $\frac{a}{b}$，如果这个分数是整数（$b|a$），那么它就已经是连分数了（单个整数算是连分数）。如果不是呢?

注意到:

$$\frac{a}{b} = \lfloor \frac{a}{b} \rfloor + \frac{a \bmod b}{b} = \lfloor \frac{a}{b} \rfloor + \frac{1}{\frac{b}{a \bmod b}}$$

然后我们再递归的把 $\frac{b}{a \bmod b}$ 展开成连分数堆上去就好。

可以注意到这个过程和欧几里得求 gcd 的算法几乎是完全一致的，所以时间复杂度是 $O(\log(\max(a,b)))$ 的。

Problem C. Crystal Caves

Tags : 贪心、DP

将曼哈顿距离拆成 x 和 y 上的分量，答案变成:

$$\frac{n(n-1)(n+1)}{6} + \max_{\forall t, a_t \in [l_t, r_t]} \{ \sum_{i=1}^{n} \sum_{j=i+1}^{n} |a_i - a_j| \}$$

后一部分不好计算，其式子记作 M, 考虑贪心。

定理一: M 的最大值可以当所有 a_t 在区间的端点取得。

考虑证明，设 $M = f(a_1, a_2, \cdots, a_n)$。

将 f 对 $a_i(1 \le i \le n)$ 求偏差分 (类似偏导) 得 $\Delta f_i(a_1, a_2, \cdots, a_n) = \sum_{j=1}^{n} [a_j < a_i] - [a_j > a_i]$。

很容易发现 Δf_i 是单调不降的，所以 f 的最大值只在端点取得。

这样，我们只要考虑在 a_i 取在端点的情况即可。

定理二: M 的最大者可以在 a_t 取左右端点尽可能均匀时取得。

考虑反证，若取得左端与取得右端的个数分别为 $a, b(a+b=n)$ 不均匀为，不妨设 n 为偶数，且 $a > b$。

则我们一定可以找到一个 i 取得左端，即 $a_i = l_i$ 使得 $\Delta f_i = \sum_{j=1}^{n} [a_j < a_i] - [a_j > a_i]$ 大于 0。

所以 $f(\cdots, r_i, \cdots) - f(\cdots, l_i, \cdots) \ge \Delta f_i \times (r_i - l_i) > 0$。

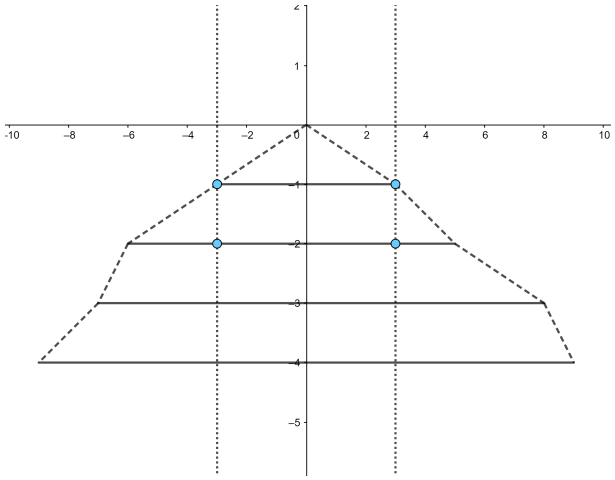
则 a_i 取右端更优，矛盾。所以原命题成立。

算法一

根据上述结论，考虑将 n 为奇数转化为 n 为偶数的情况。

其实只要 a_1 在 $[l_1, r_1]$ 任取，然后去掉第一层，按偶数处理情况即可。

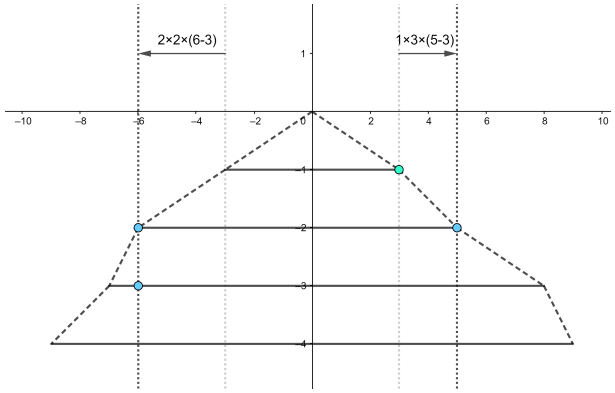
考虑 DP，初始时，左右两边的点都放置在 l_1 与 r_1 上，如图。



初始

然后设 $f_{i,j}$ 表示前 i 层，左边固定了 j 个点的最大值。

考虑转移，我们固定左右中的一个点，然后将其它点推入下一层的端点，并计算贡献，如图。



转移

然后递推即可。

时间复杂度 $O(n^2)$。

Problem D. Character Distance

我们先把整个序列排好序。

首先我们可以发现当我们选定了一个数 x 之后只有两种可能（保证字典序最小）：

1. 把比 x 小的数都按大小放在前面，然后放一个 x，之后每隔 d 放一个，其余从小到大插空。
2. 由于情况 1 没法放完 x，x 从后往前，每隔 d 放一个 x，其余从小到大插空。

然后我们会发现，对于情况 1 的来说，x 越大字典序越小，对于情况 2 来说，x 放置的第一个数位置越靠后，字典序越小。

所以我们就可以遍历所有的 x，然后对于每个 x 就 $O(1)$ 判断属于情况 1 还是情况 2 然后对两个分别存其最小的，最后 $O(n)$ 遍历即可。

当然如果你比较细心也会发现，两种情况之间，假设情况 1 中最大 x 第一个放置的位置为 $p1$，情况 2 中最大 x 第一个放置的位置为 $p2$，那么当 $p1 \geq p2 - 1$ 的时候情况 1 的更优，否则情况 2 更优。

Problem E. The Legend of God Flukehn in Eastern

标记：冯神（flukehn）金将坐标 (FC_x, FC_y)

简易题解

首先很容易发现，最多只有一个棋子会从冯神的手中逃脱。考虑冯神金将的策略，很容易发现在大局上肯定是从 y 坐标小的棋子开始吃起，然后按顺序向上吃。反过来考虑对手策略，对手可以从下向上枚举某个棋子是否可以逃脱。

有一个特殊情况，就是当对手两个棋子走到 x 轴相邻的位置的时候，冯神就必须要在下方左右横跳等对手先手送子。此时则需要考虑剩下没吃的棋子的位置。

详细题解

定理 1（最多一子逃逸定理）

最多只会有一个棋子逃逸。

证明

对于任意两个棋子 $A, B(A_y \leq B_y)$，冯神可选择追棋子 A，若对方选择持续逃逸棋子 A，则冯神可以追至足够远处，然后回头吃 B。若对方选择放弃逃逸棋子 A，则冯神可以直接吃 A。

定理 2.1（非相邻三角区外逃逸定理）

某回合开始之前，若存在棋子 A, B 有 $A_y \leq B_y$ 且 $|A_x - B_x| > 1$，满足 $B_y - A_y < |A_x - B_x|$。则此时存在某种策略使得 A, B 中至少有一个棋子可以逃逸。

证明

策略就是在不送死的情况下先走 B 再走 A，如果冯神选择先吃 B 的话，那么 A 肯定会到冯神下面就可以逃逸了，如果冯神选择先吃 A 的话那么吃掉 A 的回合 B 就可以直接逃逸了。

定理 2.2（相邻逃逸定理）

某回合开始之前，若存在棋子 A, B, C 有 $A_y = B_y \leq C_y$ 满足 $|A_x - B_x| = 1$ 且 $C_x \notin \{A_x, B_x\}$。则此时存在某种策略使得 A, B, C 中至少有一个棋子可以逃逸。

证明

A, B 不动 C 往下走到与 A, B 其中一个人形成定理 2.1 的局面，期间如果冯神选择先吃 C，那么 A, B 就会有一个棋子在冯神下方直接逃逸，否则冯神只能走到 A, B 的正下方，如果冯神先手吃 A, B 其中一个，那么另一个就可以逃跑，所以冯神只能在 A, B 下方左右横跳，直到 C 与其中一个形成定理 2.1 的棋局。

定理 2.3（双列 AK 定理）

某回合开始之前，若存在 x_1, x_2，对于所有的棋子 A，满足 $A_y > FC_y, FC_x \in \{x_1, x_2\}$ 且 $A_x \in \{x_1, x_2\}$，则存在某种策略所使得有棋子均无法逃逸。

证明

同理只需要冯神在下方反复横跳，棋子就只能上来送。

定理 3（有序吃子定理）

如果一直不达成定理 2.1 和某两子相邻的情况，冯神如果想吃所有子，必须按照 y 坐标从小到大吃子，且一定能吃完。

证明

如果冯神不按照 y 坐标吃的话，那么在吃到某一个子的时候，存在另一个子在冯神下方或水平，则可以逃逸。如果按照最优的路径吃的话，那么吃完这个子的时候，因为其他子都不与这个子构成定理 2.1 的情况，所以一定能吃掉下一个子。

定理 4（逃逸瓶颈定理）

在不达成定理 2.1 或定理 2.2 的形式的情况下，没有棋子可以逃逸。

证明

反证法：考虑某种棋局即无法达成定理 2.1，又无法达成定理 2.2，存在棋子有可能逃逸。

先考虑无法达成定理 2.1 的情况，那么在这种情况下对于任意两个棋子 A, B 满足 $|A_x - B_x| > 1, A_y \le B_y$，需要满足 $|A_x - B_x| \le B_y - 2A_y$（因为冯神需要花费 A_y 步吃到棋子 A）。很明显当 $A_y = B_y$ 的时候，必定满足 $|A_x - B_x| = 1$，否则就达成定理 2.1。

我们先考虑不存在某个回合开始的时候有 $A_y = B_y$ 的情况，那么很明显，所有的棋子唯一占据一个纵坐标，当冯神的金将按照 y 坐标大小从小到大吃的话，若他即将吃的子向下移动而且两者 x 坐标不相同，金将横向移动，否则斜上方或正上方移动。易证，如此策略可以保证当金将在吃掉棋子 A 和其一下的棋子，并走到 A 的坐标的时候，其余的棋子总共最多走了 A_y 步。这也说明了第一段中括号内的那句话。而在满足了第一段的要求下，很明显本情况没有棋子可以逃逸。

我们不妨接着考虑某回合开始前存在棋子 A, B 满足 $A_y = B_y$ 且 $|A_x - B_x| = 1$ 且 A, B 为满足前两个条件中 A_y 最小的那个。由于不能达成定理 2.2，对于所有棋子 $C, C_y \ge A_y$ 满足 $C_x \in A_x, B_x$。由于无法达成定理 2.1，所以 A, B 无论如何往下走都无法与其他棋子形成定理 2.1 的情况。即如果 A, B 不与其之下的任意棋子 C 形成新的相同形态的局面，则无论 A, B 或上面棋子的如何移动，冯神总可以把下面所有棋子解决之后移动到届时 A, B 中 y 较小的棋子的位置或其下方，此时根据定理 2.3 可证明所有棋子无法逃逸。

若在冯神没解决其下所有棋子的期间 A, B 中某个棋子与其之下的棋子 C 形成了相同类型的局面，若 $C_x \notin A_x, B_x$，易证另一个棋子一定可以与 C 形成定理 2.1 的局面，矛盾。所以 $C_x \in A_x, B_x$，则回到了上一段开头的局面，可进行完全相同的分析。

综上两种情况都无法逃逸，与题设矛盾，所以本定理成立。

至此证明如果无法达成定理 2.1 和定理 2.2 的局面，则必定所有棋子无法逃逸。

定理 5（单子移动定理）

如果可逃逸一个棋子，那么从游戏开始到达成定理 2.1 或 2.2 的情况之前，只移动一个棋子一定是解集中的一个。

证明

如果想要使棋子 $A, B(A_y \le B_y)$ 构成定理 2.1 或者定理 2.2，那么往下走 A_y 只会让冯神更早的吃到 A 棋子而增大 B_y 构成任意两个定理的步数，是完全无益的。

做法补充

我们把棋子按照 y 排好序之后，从下往上枚举每一个棋子是否可以与上一个形成定理 2.1、定理 2.2 或者定理 2.3 即可。

Problem F. Four Column Hanoi Tower

记 $g(n)$ 为有三根柱子 (A, B, C) 时，按规则把 A 柱上的 n 个盘子移动到 C 柱上去所需要的最少步数。

记 $f(n)$ 为有四根柱子 (A, B, C, D) 时，按规则把 A 柱上的 n 个盘子移动到 D 柱上去所需要的最少步数。

对于 $g(n)$，采用如下算法移动盘子（记为 ThreePegsHanoi）：

1)、将 A 柱上方 $n-1$ 个盘子使用 ThreePegsHanoi 算法经过 C 柱移至 B 柱。2)、将 A 柱上最后一个盘子直接移至 C 柱。3)、将 B 柱上的 $n-1$ 个盘子使用 ThreePegsHanoi 算法经过 A 柱移至 C 柱。

因此求解 $g(n)$ 只需要两次 $g(n-1)$ 和一步移动，即 $g(n)=2g(n-1)+1=2^n-1$。

对于 $f(n)$，采用如下算法移动盘子（记为 FourPegsHanoi）：

1)、将 A 柱上 n 个盘子划分为上下两部分，下方部分共有 k $(1\le k\le n)$ 个盘子，上方部分共有 $n-k$ 个盘子。2)、将 A 柱上面部分 $n-k$ 个盘子使用 FourPegsHanoi 算法经过 C、D 柱移至 B 柱（想想为什么不能同时放到 B、C 柱上）。3)、将 A 柱剩余的 k 个盘子使用 ThreePegsHanoi 算法经过 C 柱移至 D 柱。4)、将 B 柱上的 $n-k$ 个盘子使用 FourPegsHanoi 算法经过 A、C 柱移至 D 柱。

因此求解 $f(n)$ 只需要两次 $f(n-k)$ 和一次 $g(k)$ 移动，即

$$f(n)=\begin{cases}1, & n=1\\ \min\limits_{1\le k\le n-1}\{2f(n-k)+g(k)\}, & n>1\end{cases}$$

到这里就可以得到一个时间复杂度 $O(n^2)$ 的 DP 算法，常数会有点大不太能通过本题。

实际上 $g(n)=2^n-1$，增长的速度非常快，设最优转移点为 k，对于任意 i $(1\le i<k)$,

有 $2f(k)+g(n-k)<2f(i)+g(n-i)$, 当 n 增大时，f 不变，$g(n-i)>g(n-k)$，转移点的贡献始终比前面的点要优，因此前面的 $k-1$ 个点永远都不会再成为最优转移点了，即 最优转移点具有决策单调性。因此第二层枚举可以从上一个最优转移点开始枚举，经过这个剪枝就已经可以通过本题。

事实上在 2014 年已经有论文证明出来最优中间点就是 $n-\lfloor\sqrt{2n+1}\rceil+1$，因此上一段的" 简陋优化" 从最优转移点枚举实际上复杂度是 $O(n\sqrt{n})$，当然如果用决策单调性可以做到 $O(n\log n)$，用论文的结论的话单词查询甚至不用 $O(n)$。

由于数很大，所以需要使用高精度。

Problem G. Magic Number Group

考虑离线莫队。

题目相当于找出每段区间里每个数字的所有因子，找到其出现次数最多的因子的个数。

但是 a_i 最大可达到 10^6, 因子数量太多。仔细想想会发现我们并不需要求出所有因子，只需要求出所有质因子即可，又因为一个数的质因子个数非常少（最多不超过 7 个，记为 k），因此我们可以预处理所有数的质因子后把问题近似为求区间众数。维护每个质因子出现的次数和出现 i 次的质因子的数量，即可离线后使用莫队求解本题。

由于 n 和 q 为同一数量级，总时间复杂度为 $O(kn\sqrt{n})$。

注意：本题时限较为严格，如果将每个数拆成 kn 个质因数然后利用分块求区间众数，或者采用不当的筛素数方法，或者使用回滚莫队等，都可能造成常数过大而 TLE，需要一些卡常技巧才能通过。

Problem H. Hearthstone So Easy

结论为：

1. 如果 A 第一回合被扣血死，那么 B 赢；
2. 如果 A 第一回合能打死 B，或者使 B 只剩一点血，那么 A 赢；
3. 其他情况 B 赢。

证明：

首先情况 1 和情况 2 显然。

由题目我们可以知道一条重要的信息：每回合开始时，A 和 B 的**血量必定是相同的**。

我们考虑在第 $i+1,(i>0)$ 回合 A **刚好**可以击杀 B，即在第 i 回合 A 无法击杀 B。那么在第 i 回合 B 一定可以击杀 A。所以只要 A 第一回合无法击杀 B，那么 B 一定赢。

注意特判初始血量为 1 的情况。

Problem I. Homework

n 个节点的一棵无根树, 边权非负, 每个点独立完成作业用时 a_i (非负), u 去抄另一个在 t_v 时刻已经完成作业的 v 的作业, 用时为 $dist(u,v)+t_v$, 其中 $dist(u,v)$ 是两点间的距离. 带修改点权和边权, 询问每个点最早完成作业的时间 (的异或和). 询问至多 200. $1\le n,q\le 10^5, 0\le a_i,w_i\le 10^9$

先考虑查询, 暴力做法, 对每个点来说, 去找已经完成作业的点, 看看能不能抄他的作业使得时间变短. 或者这样考虑, 已完成作业的点, 去给其他点抄作业, 能不能让其他点的时间变短. 复杂度 $O(n^3)$(直接求两点间距离), 或者 $O(n^2\log n)$(用倍增或树链剖分求两点间距离).

仔细思考上述暴力, 发现暴力的过程和 Dijkstra 算法几乎一模一样. 注意到两点之间的路径唯一. 假设从 u 到 v 需要经过 p, 不妨设 u 需要去松弛 v, 那么 u 一定会先去松弛 p, 然后 p 再去松弛 v, 这样和 u 去松弛 v 的效果等价. 所以只需要考虑真实存在的边即可. 用堆优化的 Dijkstra 可以做到 $O(n\log n)$.

存在 $m\le 200$ 次查询, 这样总复杂度会达到 $O(mn\log n)$, 无法通过本题, 需要一个 $O(n)$ 的做法.

根据上述分析, 一个点只会被他的儿子或者父亲更新. 所以考虑换根 dp. 第一次 dfs 求 $dp(u)$ 为仅考虑以 u 为根的子树, u 最早完成作业的时间. 转移为 $dp(u)=\min_{v\in SON(u)}\{dp(v)+w(u,v)\}$ ($SON(u)$ 表示 u 的儿子的集合, $w(u,v)$ 表示 u,v 之间的边的权值)

然后第二次 dfs 换根求答案. 设当前点为 u, 他的父亲为 f, 如果 $dp(f)\leftarrow dp(u)$, 那么 $dp(u)\ge dp(f)+w(f,u)$(因为边权非负), 也就是说不会出现重复走一条边的情况. 所以再令 $dp(u)=\min\{dp(u),dp(f)+w(f,u)\}$ 即可.

修改可以在 $O(1)$ 时间内完成, 总复杂度为 $O(mn+q)$, m 是询问数量.

Problem J. LRU

显然, 这个问题是单调的, 即大的 cache 表现的一定不比小 cache 更差, 二分 cache 的大小就行了.

主要是 check 函数的写法, 可以用一个优先队列和一个 map, 优先队列按照最近访问时间排序, 修改一个块的最近访问时间直接在 map 里修改, 然后每次从优先队列取出堆顶元素的时候判断优先队列里记录的时间和 map 里记录的时间是否一致, 不一致就把 map 里记录的最近访问时间加上去再 push 进优先队列, 然后看下一个元素, 这样的 check 是 $O(nlogn)$ 的, 总时间复杂度 $O(nlog^2n)$.

或者把 cache 里的元素按照最近访问时间顺序组织成一个链表, 然后用 unordered_map 记录 cache 里每个块的链表节点的位置, 这样链表头就是下一个要被替换的块, 加入块在链表尾加入, 访问 cache 里已有的元素就直接把他从链表中间删掉然后在链表尾加进去. 这样总的复杂度只需要 $O(nlogn)$, 但是这题定位是 easy, 就把 $O(nlog^2n)$ 也放过了.

Problem K. Many Littles Make a Mickle

签到中的签到题，求 $\sum_{i=1}^{n} mi^2$，因为 n 很小直接暴力也行，然后利用公式 $\frac{n(n+1)(2n-1)}{6}$ 也行。

Problem L. It Rains Again

挡雨板相交的情况可能很复杂

考虑如果在地面某个位置垂直向上看，可以看到挡雨板的话，这块地面就不会淋到雨

那么答案就是所有线段 $\{(x1,y1),(x2,y2)\}$ 的水平区间 $[x1,x2]$ 的并的长度和

数据范围不大，可以通过差分、前缀和、排序求区间并等各种办法通过