数字游戏

容易构造出以下dp状态：$dp[i][j]$，其中表示两人的分数差 $a-b$ 为 j，且进行到第 i 轮时有多少种不同的游戏，得到状态转移方程如下：

$$dp[i+1][j] = dp[i][j-2k] + 2dp[i][j-2k+1] + \cdots + (2k+1)dp[i][j] + \cdots + dp[i][j+2k]$$

直接暴力转移的话，复杂度为 $O((kt)^2)$，考虑进行优化。

容易发现，$dp[i+1][j]$ 的值仅与$dp[i][j-2k \dots j+2k]$ 这 $2k+1$ 个连续状态有关，可以考虑使用前缀和维护一个滑动窗口，这样单次转移就可以在常数时间内完成，复杂度为 $O(kt^2)$，便可以满分通过此题。

此外还有使用生成函数的做法，可以获得更优的效率，请各位选手自己研究。

班级对抗

现在将B班所有选手的 a_i 置为它的相反数，b_i 保持不变，把A班和B班的同学都放在一起，那么问题就转换成了从这个集合中取一些元素，使得它们 a_i 的和为0，且 b_i 的和最大，这就是一个经典的背包问题，时间复杂度为 $O((n+m)^2 \cdot \max\{a_i\})$。

在这道题目中，背包的容量最大可达 $n * \max\{a_i\}$，我们可以考虑从容量入手进行优化。如果我们以一种相对随机的方式安排物品的加入顺序，可以发现大多数最优状态都在0附近振荡，也就是说绝大多数有用的状态都不会过大或过小，因此可以设置一个上界，我们只考虑 $[-T,T]$ 的背包容量，对于超过这个容量的状态直接略过。此处我们并没有改变原本的dp状态以及转移方程，只是将背包的容量缩小了。

可以证明，当 T 取 $\sqrt{n+m} * \max\{a_i\}$ 时，超过上限的概率最多在 10^{-7} 左右。证明过程留给选手自行思考（提示：卡特兰数，斯特林公式）

序列

题目要求的是最长的最小字典序单峰子序列，以及最大字典序单峰子序列。

$f[i][0]$ 表示 $a[i]$ 一定取，序列 $a[1..i]$ 的最长上升子序列长度。

$f[i][1]$ 表示 $a[i]$ 一定取，序列 $a[1..i]$ 的最长单峰子序列长度。

$g[i][0]$ 表示 $a[i]$ 一定取，序列 $a[i..n]$ 的最长下降子序列长度。

$g[i][1]$ 表示 $a[i]$ 一定取，序列 $a[i..n]$ 的最长单峰子序列长度。

对于$f[i][0;1]$ 来说：

- $f[i][0] = max(f[j][0]) + 1$，其中 $a[j] < a[i]$
- $f[i][1] = max(f[i][0], f[j][1] + 1)$，其中 $a[j] > a[i]$

$g[i][0;1]$ 同理，只需反向做一遍即可。

通过枚举，可以很容易得到最长单峰子序列长度。考虑到字典序最小，我们可以从前往后遍历，然后贪心地选择，能选则选。在选择的过程中要维护当前是上升还是下降。字典序最大的也同理可得。

集合

- 问题：给定一个$1-n$的集合S，每次删除当前集合中最小的元素，再考虑**顺序的**随机删掉k个元素，直到$|S| \le k$ ，求每个元素最后被留下来的概率。
- $r = n\%(k+1)$，如果$r=0$，所有元素留下来的概率都是0
- 如果$n < k+1$，所有元素留下来的概率都是1

- 观察可知，如果最后留下的元素中，最小元素为x，那么$1, 2, 3 ... x-1$，一定全被删除了
- 将操作看成两种
  - 第一种，每次删掉一个最小的元素
  - 第二种，每次随机删除一个元素
- 设$dp[i][j]$ 为前i个元素全被删除后第二种操作还需要做j的方案数，转移如下
  - 做第一种操作，如果$j+k \le n-i-1$：
$$dp[i+1][j+k] = dp[i+1][j+k] + dp[i][j]$$
  - 做第二种操作，如果$j > 0$：
$$dp[i+1][j-1] = dp[i+1][j-1] + dp[i][j] \times j$$
- 考虑最后留下来的元素组成的集合T，从小到大为$x_1, x_2, ... x_r$，将所有方案按照集合T中最小元素x_1进行分类
  - 令$f[i]$ 表示集合T中最小元素为i的方案个数
$$j = n-i-1-r$$
$$f[i] = dp[i-1][j] \times j! \times \binom{n-i}{j}$$
  - 令$cnt[i]$ 表示集合T中最小元素为i时，元素$x > i$ 被留下来的方案数
$$j = n-i-1-r$$
$$cnt[i] = dp[i-1][j] \times j! \times \binom{n-i-1}{j}$$
  - 令前缀和$sum[i] = \sum_{j=1}^{i} cnt[i]$
- 那么首先总方案为$SUM = \sum_{i=1}^{n} f[i]$
- 元素i被留下来的方案数$ans[i] = f[i] + sum[i-1]$， 而概率$p[i] = \frac{ans[i]}{SUM}$
- 时间复杂度$O(n^2)$